Data Integrator™

Technical Manuals

Version 6.5

Document Number DI-65-0100-001

January 30, 2003

Master Contents

Getting Started Guide

Designer Guide

Administrator Guide

Reference Guide

Performance Optimization Guide

Advanced Development and Migration Guide

LiveLoad User's Guide

Supplement for J.D. Edwards

Supplement for Oracle Applications

Supplement for PeopleSoft

Supplement for SAP

Supplement for Siebel

Data Integrator™

Getting Started Guide

Version 6.5

Document Number DI-65-0103-001

January 30, 2004

Contents

1 Welcome

This guide provides an overview of the BusinessObjects™ Data Integrator Platform and architecture. It also describes how to upgrade and install Data Integrator.

With Data Integrator you can:

- Easily build and execute batch processing applications that create and update a data warehouse.
- Combine batch capabilities with request-response processing logic and message handling to support e-commerce and its data integration requirements.

This chapter covers the following topics:

Audience and assumptions

This and other Data Integrator product documentation assumes that:

- You are an application developer, consultant, or database administrator working on data extraction, data warehousing, or data integration.
- You understand your source data systems, RDBMS, business intelligence, and e-business messaging concepts.
- You understand your organization's data needs.
- You are familiar with SQL (Structured Query Language).
- You are familiar enough with Microsoft Windows or UNIX platforms to effectively install Data Integrator.

Data Integrator product documentation

Data Integrator documentation is provided in PDF format on the Data Integrator CD. You can read the PDF files with the latest version of Adobe Acrobat Reader (download instructions at the Adobe website).

If you install the Data Integrator documentation (recommended), you can open it in two ways:

- Select **Start** > **Programs** > **BusinessObjects Data Integrator *version*> Data Integrator Documentation**.
- Go to the **Help** menu in the Data Integrator Designer or Administrator.

  Using either method, you can open any of the following:

  - Release Notes
  - Release Summary
  - Technical Manuals

The following technical documentation is currently available:

- *Data Integrator Release Summary* and *Data Integrator Release Notes* — Provide the latest information about the product including information that was not available when the other documents were published.
- *Data Integrator Technical Manuals* include:
  - *Data Integrator Getting Started Guide* — Introduces the features and benefits of using Data Integrator. This guide explains Data Integrator architecture and how to install the product including system requirements and pre-installation preparation that will help you understand dependencies between Data Integrator and your computing environment.

- *Data Integrator Designer Guide* — Provides conceptual and procedural information for designing, developing, and implementing applications using the Data Integrator Designer.
- *Data Integrator Administrator Guide* — Provides conceptual and procedural information for scheduling and monitoring batch jobs as well as configuring and administering real-time jobs. Explains the Data Integrator Administrator features.
- *Data Integrator Reference Guide* — Provides detailed information about Data Integrator Designer objects, data types and expressions, transforms, functions, and user exits.
- *Data Integrator Advanced Development and Migration Guide* — Discusses how to migrate Data Integrator applications through development phases and includes multi-user development concepts and procedures.
- *Data Integrator Performance Optimization Guide* — Provides information about improving the performance of your Data Integrator environment and the Designer options you can use to reduce job run-times by measuring and tuning performance. Includes documentation on bulk loading and parallel execution.
- *Data Integrator LiveLoad User's Guide* — Provides information about using Data Integrator Liveload to provide a continuously available data warehouse.
- *Data Integrator Supplement for J.D. Edwards* — Provides information about using Data Integrator to extract data from J.D. Edwards World and J.D. Edwards OneWorld.
- *Data Integrator Supplement for Oracle Applications* — Provides information about using Data Integrator to extract data from Oracle Applications.
- *Data Integrator Supplement for PeopleSoft* — Provides information about using Data Integrator to extract data from PeopleSoft ERP and HRMS software.

- *Data Integrator Supplement for SAP* — Provides information about using Data Integrator to extract data from SAP R/3 and SAP BW. This supplement includes conceptual, procedural, and reference information.
- *Data Integrator Supplement for Siebel* — Provides information about using Data Integrator to extract data from Siebel applications.

If you install the documentation with Data Integrator, you can also go to **Start > Programs > BusinessObjects Data Integrator 6.1> Data Integrator Documentation > Tutorial** to view:

- *Data Integrator Core Tutorial* — Provides exercises to help you learn how to extract, transform, and load data with the Data Integrator Designer. Exercises include using design logic to extract data from SAP R/3 and non-SAP sources.

The latest versions of these manuals are also available on the documentation page at Business Objects Customer Support Online. To access the documentation page, you must have a valid user name and password. To obtain your user name and password, register at Business Objects Customer Support Online.

Suggested reading path

Begin by reading the *Data Integrator Getting Started Guide* which includes an introduction to functionality and architecture as well as installation instructions.

Understand how to use Data Integrator Designer by reading the *Data Integrator Designer Guide* and doing the exercises in the *Data Integrator Core Tutorial*. The Designer also provides on-line help and context-sensitive help:

- To access on-line help, select **Help** > **Contents**.
- To access context-sensitive help, click an object and press F1.

Use the *Data Integrator Reference Guide* to answer specific questions about objects and window options in the Designer.

The *Data Integrator Administrator Guide* provides information about how to use the Administrator which is a browser-based graphical user interface used to schedule and monitor batch jobs and to configure real-time jobs and adapter interfaces available with Data Integrator.

Use the *Data Integrator Performance Optimization Guide* for tips about using system and Designer options related to performance optimization. These include measuring and tuning techniques, bulk loading, and parallel processing objects within a data flow.

BusinessObjects Rapid Marts are documented in individual guides that are included on the Rapid Mart product CDs. Use these guides to answer specific questions about each Rapid Mart.

To install and configure Data Integrator's pre-packaged adapters, read the individual adapter guides on the adapter product CDs.

2 Introducing the Data Integration Platform

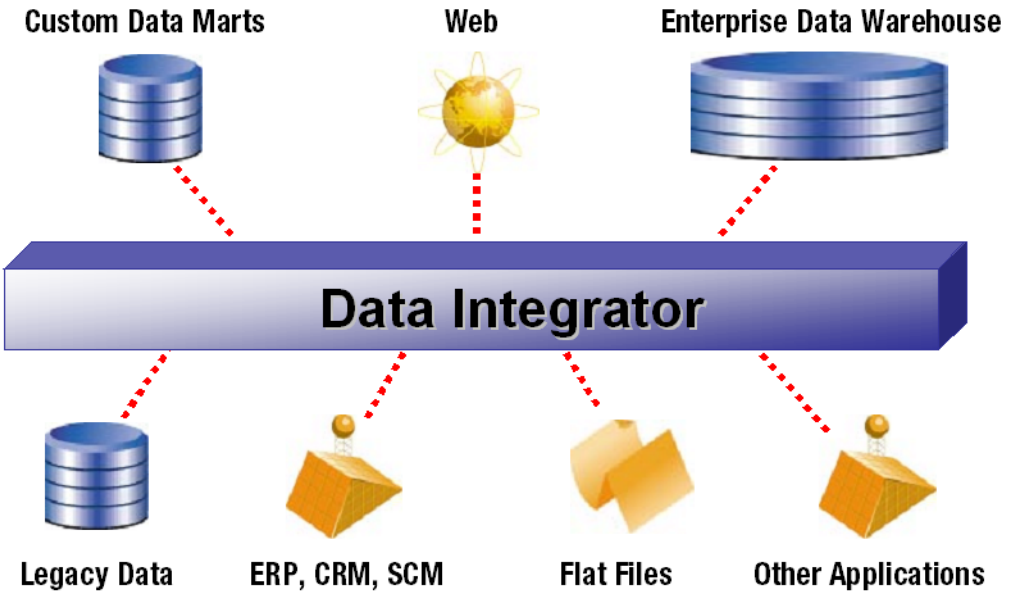
This chapter introduces the benefits and functionality of the Business Objects Data Integration Platform. Sections include:

Business Objects Data Integration Platform

True data integration requires intelligent data caching that blends batch extraction, transformation, and loading (ETL) technology with real-time bi-directional data flow across multiple applications for the extended enterprise. By building a relational data store and intelligently blending direct real-time and batch data-access methods for accessing data from enterprise resource planning (ERP) systems, Business Objects has created a powerful, high-performance data integration platform that allows businesses to fully leverage their ERP and enterprise application infrastructure for multiple uses.

Business Objects provides the industry's first and only real-time data integration platform to drive today's new generation of real-time analytic, supply-chain management and sell-side e-business applications. With Business Objects' highly scalable data platform, enterprises can maintain a real-time, on-line dialogue with customers, suppliers, employees, and partners, providing them with the critical information they need for transactions and business analysis.

The Data Integration Platform consists of BusinessObjects™ Data Integrator, an award-winning data movement-and-management server, and BusinessObjects™ Rapid Marts, a suite of packaged applications that speed the delivery and integration of back-office data in the areas of sales and marketing, finance and operations, inventory and purchasing, as well as manufacturing and distribution.



Data Integrator product benefits

The Business Objects Data Integration Platform enables you to develop enterprise data integration for *batch and real-time* uses.

- Create a *single infrastructure* for batch and real-time data movement to enable faster and lower cost implementation.
- Your enterprise can *manage data* as a corporate asset independent of any single system. Integrate data across many systems and reuse that data for many purposes.
- Perform *intelligent caching* of critical data and business logic. Get higher performance, 24x7 availability, and reduce the burden on enterprise systems.
- Use *pre-packaged data solutions* for fast deployment and quick ROI. These solutions extract historical and daily data from operational systems and cache this data in open relational databases.

Data Integrator and Rapid Marts combine to create a data integration platform for traditional data warehouse applications and for integrating across a variety of enterprise applications. Data Integrator customizes and manages data access and uniquely combines industry-leading, patent-pending technologies to deliver data to analytic, supply-chain management, customer relationship management, and Web applications. Rapid Marts provide pre-packaged data content, balancing batch and real-time data movement to deliver the high-performance, scalable information flow required across the extended enterprise.

Single point of integration

Data Integrator combines both batch and real-time data movement and management with intelligent caching to provide a single data integration platform for information management from any information source, for any information use.

This unique combination allows you to:

- Stage data in an operational data store, data warehouse, or data mart.
- Update staged data in batch or real-time modes.
- Direct XML-based requests (queries or updates) to the staged data or directly to back-office systems. User-specified rules control request routing.
- Create a single graphical development environment for developing, testing, and deploying the entire data integration platform.
- Manage a single metadata repository to capture the relationships between different extraction and access methods and provide integrated lineage and impact analysis.

High availability and performance

Data Integrator's high-performance engine and proven data movement and management capabilities provide:

- A high-performance, scalable, multi-instance data-movement engine for fast execution
- 24x7 data availability to ensure the data loads do not affect the performance and correctness of the active data warehouse

Packaged data access through Rapid Marts

Business Objects' expertise in providing critical data to information users has resulted in the creation of pre-packaged data applications with functional content (data and business logic) for many specific areas within an enterprise, enabling:

- Flexible access to historical data and real-time information
- Packaged data staging for internal and external users
- Quick deployment and implementation for fast ROI
- Easy customization for specific business or information requirements

Data Integrator interfaces

Data Integrator provides many interfaces. These interfaces are:

- Data-level read/write interfaces using SQL:
  - Oracle
  - DB2/UDB
  - DB2/UDB via DBConnect to MVS
  - DB2/UDB via DBConnect to AS/400
  - MS SQL Server
  - Informix IDS
  - ODBC to generic databases
  - Sybase
  - Teradata
- Data-level read interfaces using DETAIL:
  - DB2/AS400
  - DB2/MVS
  - IMS
  - VSAM
  - ADABAS
  - Flat files on AS/400
  - Flat files on Windows
- Application-level interfaces (read/write and batch/real-time unless otherwise specified):
  - SAP R/3 ABAP (batch, read only)
  - SAP R/3 IDoc
  - SAP R/3 RFC/BAPI
  - SAP BW (batch)
  - J.D. Edwards One World or World (batch, read only)

PeopleSoft interface (batch)

Siebel Application (batch)

Oracle Application (batch)

- Technology read/write real-time interfaces:

  HTTP/HTTPS

  IBM MQSeries

  Java Message Service (JMS)

  Oracle Applications

  SNMP

  Trilium

  Web Services

- You can use the Data Integrator Interface Development Kit to develop adapters that read from and/or write to other applications.

In addition to the interfaces listed above, the Data Integrator Nested Relational Data Model (NRDM) allows you to apply the full power of SQL transforms to manipulate, process, and enrich hierarchical business documents. For more information, see Chapter 9, "Nested Data," in the *Data Integrator Designer Guide*.

Data Integrator functional summary

Data Integrator performs three key functions that can be combined to create a scalable, high-performance data platform:

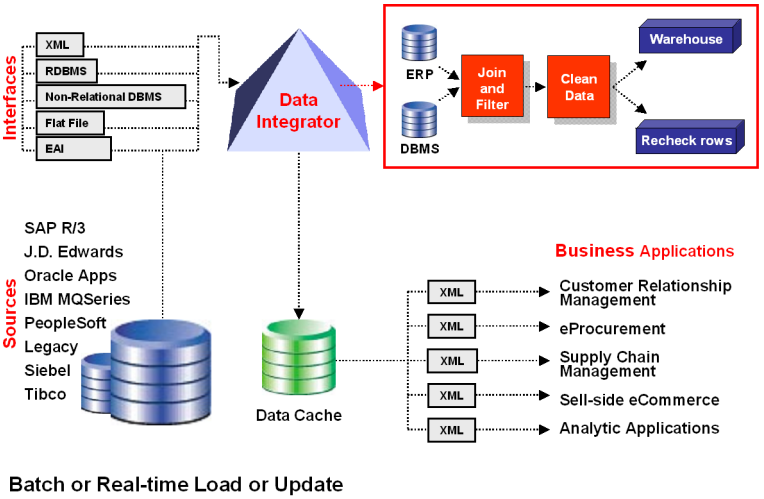
- Loading data — Populate ERP or enterprise application data into an operational data store (ODS) or analytical data warehouse, and update in batch and/or real-time.
- Routing requests — Intelligently create information requests to a data warehouse or ERP system using complex rules.
- Applying transactions — Apply transactions against ERP systems.

Loading data

Data mappings and transformations can be specified using the Data Integrator Designer graphical user interface. Data Integrator automatically generates the appropriate interface calls to access the data in the source system.

For most ERP applications, Data Integrator generates SQL optimized for the specific target database (Oracle, DB2, SQL Server, Informix, and so on). To perform batch data extraction against SAP R/3, Data Integrator generates optimized ABAP, eliminating the need to develop and maintain customized ABAP code.

Automatically generated optimized code reduces the cost of maintaining data warehouses and enables you to build data solutions quickly, meeting user requirements faster than other methods (for example custom-coding, direct-connect calls, or PL/SQL).

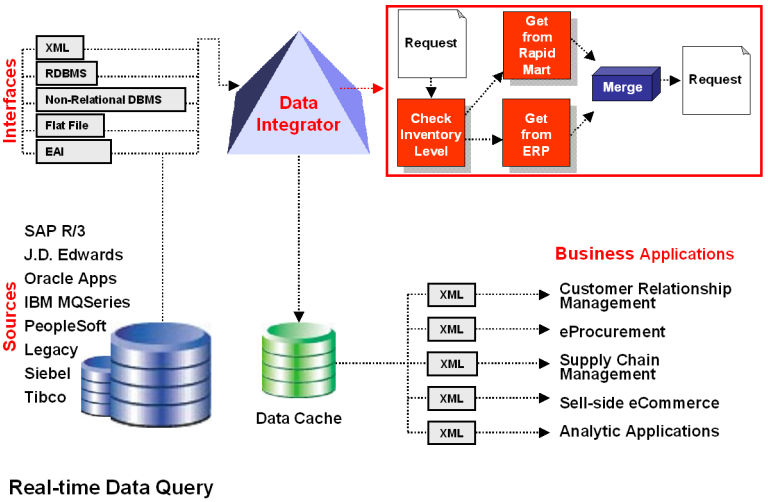


Data Integrator can process a variety of real-time data interactions: XML messages, XML files, EAI messages, flat files, SQL updates, HTTP, and SAP real-time interfaces (for example outbound IDocs). These data formats can be used to update a target data warehouse and a variety of source systems in sequence. These transactions can be enriched with other source system data as specified in a real-time job design.

Having a single data integration platform that can manage the breadth of systems found in today's heterogeneous IT environments enables IT departments to easily manage corporate data as an enterprise-wide asset. The automatic code generation reduces ongoing maintenance costs and resource requirements, allowing these resources to be deployed against other critical IT projects.

Routing requests

You can create complex data flows in Data Integrator that check or use values in a data warehouse before querying the ERP system directly. Specify data mappings, transformations, and access rule logic in a data flow using the Data Integrator Designer graphical user interface.

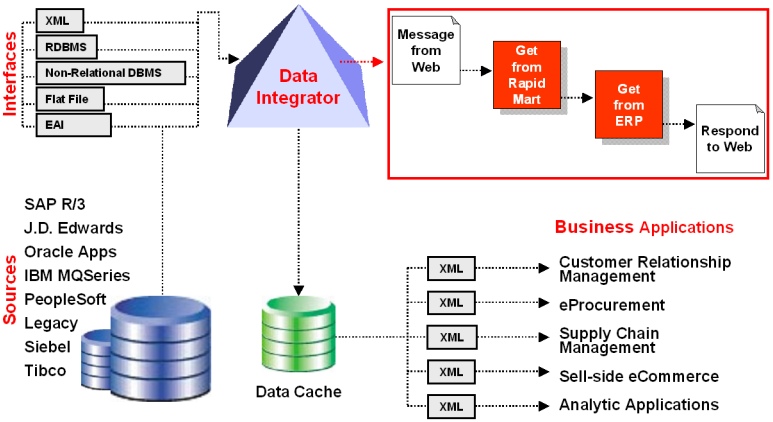


Data Integrator can generate data queries in a variety of data formats (see "Data Integrator interfaces" on page 13) and any custom format using a Data Integrator adapter. By generating these native calls, Data Integrator makes it unnecessary to develop and maintain customized code to manage the process.

Applying transactions

Data Integrator can apply data changes in a variety of data formats (see "Data Integrator interfaces" on page 13) and any custom format using a Data Integrator adapter. Enterprise customers can apply data changes against multiple back-office systems singularly or in sequence. By generating calls native to the system in question, Data Integrator makes it unnecessary to

develop and maintain customized code to manage the process. You can design access intelligence into each transaction by adding flow logic that checks values in a data warehouse or in the transaction itself before running a transaction directly against the target ERP system.



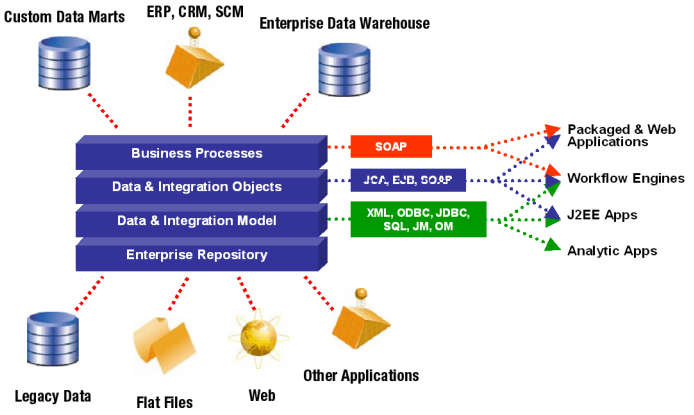
Apply Real-time Transaction

3 Data Integrator Architecture

This chapter describes Data Integrator components and their distribution on your network.

This chapter contains the following sections:

The Data Integrator architecture is layered to allow data integration to occur over a variety of open, industry-standard APIs for optimal data and metadata management.



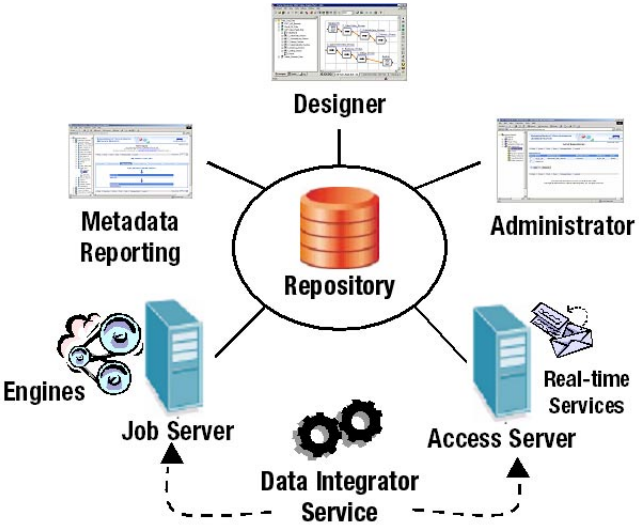
Data Integration Architecture

Standard Data Integrator components

Data Integrator includes the following standard components:

- Data Integrator Job Server
- Data Integrator engine
- Data Integrator Designer
- Data Integrator repository
- Data Integrator Access Server
- Data Integrator Administrator
- Data Integrator Metadata Reporting tool
- Data Integrator Web Server
- Data Integrator Service
- Data Integrator SNMP Agent

The following diagram summarizes the relationships between these components.



Data Integrator Job Server

The Data Integrator Job Server starts the data movement engine that integrates data from multiple heterogeneous sources, performs complex data transformations, and manages extractions and transactions from ERP systems. The Data Integrator Job Server can move data in either batch or real-time mode and uses distributed query optimization, multi-threading, in-memory caching, in-memory data transformations, and parallel pipelining to deliver high data throughput and scalability.

Data Integrator engine

When Data Integrator jobs are executed, the Job Server starts Data Integrator engine processes to perform data extraction, transformation, and movement. Data Integrator engine processes use parallel pipelining and in-memory data transformations to deliver high data throughput and scalability.

Data Integrator Designer

The Designer is a development tool with a unique graphical user interface. It enables developers to define data management applications that consist of data mappings, transformations, and control logic.

Use the Designer to create applications containing work flows (job execution definitions) and data flows (data transformation definitions).

To use the Designer, create objects, then drag, drop, and configure them by selecting icons in flow diagrams, table layouts, and nested workspace pages. The objects in the Designer represent metadata. The Designer interface allows you to manage metadata stored in a Data Integrator repository. From the Designer, you can also trigger the Job Server to run your jobs for initial application testing.

Data Integrator repository

The Data Integrator repository is a set of tables that hold user-created and predefined system objects, source and target metadata, and transformation rules. It is set up on an open client/server platform to facilitate sharing metadata with other enterprise tools. Each repository is stored on an existing RDBMS.

Each repository is associated with one or more Data Integrator Job Servers. Job Servers run jobs. When you are designing a job, you can run it from the Designer. The Designer tells the Job Server to run the job. The Job Server gets the job from its associated repository and then starts an engine to process the job. During production, the Job Server runs jobs triggered by a scheduler or by a real-time service managed by an Access Server. In production environments, you can balance loads appropriately because you can link multiple repositories to a single Job Server.

There are two types of repositories:

- A local repository is used by an application designer to store definitions of Data Integrator objects (like projects, jobs, work flows, and data flows) and source/target metadata.
- A central repository is an optional component that can be used to support multi-user development. The central repository provides a shared object library allowing developers to check objects in and out of their local repositories.

Data Integrator Access Server

The Access Server is a real-time, request-reply message broker that collects message requests, routes them to a real-time service, and delivers a message reply within a user-specified time frame. The Access Server queues messages and sends them to the next available real-time service across any number of computing resources. This approach provides automatic scalability because the Access Server can initiate additional real-time services on additional computing resources if traffic for a given real-time service is high. You can configure multiple Access Servers.

Data Integrator Administrator

The Administrator provides browser-based administration of Data Integrator resources including:

- Scheduling, monitoring, and executing batch jobs
- Configuring, starting, and stopping real-time services
- Configuring Job Server, Access Server, and repository usage
- Configuring and managing adapters
- Managing users
- Publishing batch jobs and real-time services via Web services

Data Integrator Metadata Reporting tool

This provides browser-based reports on Data Integrator metadata, which is stored in the repository. Reports are provided for:

- Repository summary — View reports on all projects, jobs, work flows, data flows, datastores, custom transforms, file formats, and custom functions in your repository
- Datastore analysis — For each datastore connection, view overview, table, function, and hierarchy reports.

- Operational statistics — Investigate job and dataflow execution statistics. Includes graphical reports.
- Business Objects Universe analysis — For every configured Universe connection, view Universe, class, and object lineage. For Business Objects Auditor databases, you can view document table and document column lineage reports.
- Dependency analysis — Search for specific objects in your repository and understand how those objects impact or are impacted by other Data Integrator or Business Objects Universe objects. Metadata search results provide links back into associated reports.

Use Metadata Reporting Tools to configure your Business Objects connections as well as refresh repository information by calculating dependencies and column mappings.

Data Integrator Web Server

The Data Integrator Web Server supports browser access to the Administrator and the Metadata Reporting tool. The Windows service name for this server is Data Integrator Web Server. The UNIX equivalent is the AL_JobService. Both use a Tomcat servlet engine to support browser access.

Data Integrator Service

The Data Integrator Service is installed when Data Integrator Job and Access Servers are installed. The Data Integrator Service starts Job Servers and Access Servers when you restart your system. The Windows service name is Data Integrator Service. The UNIX equivalent is a daemon named AL_JobService.

Data Integrator SNMP Agent

Data Integrator error events can be communicated using SNMP-supported applications for better error monitoring. Install a Data Integrator SNMP (simple network management protocol) agent on any computer running a Job Server. The Data Integrator SNMP agent monitors and records information about the Job Servers and jobs running on the computer where the agent is installed. You can configure NMS (network management software) applications to communicate with the Data Integrator SNMP agent. Thus, you can use your NMS application to monitor the status of Data Integrator jobs.

Optional Data Integrator components

Optional extra components include:

- Data Integrator Multi-user
- Data Integrator LiveLoad
- Data Integrator Adapter SDK

Data Integrator Multi-user

Data Integrator Multi-user is an advanced optional component that enables your development team to work together on interdependent parts of an application through all phases of development. While each user works on applications in a unique local repository, the team uses a central repository to store the master copy of the entire project. The central repository preserves all versions of an application's objects, so you can revert to a previous version if needed.

Multi-user development includes other advanced features such as labeling and filtering to provide you with more flexibility and control in managing application objects.

See the *Data Integrator Advanced Development and Migration Guide* for more details.

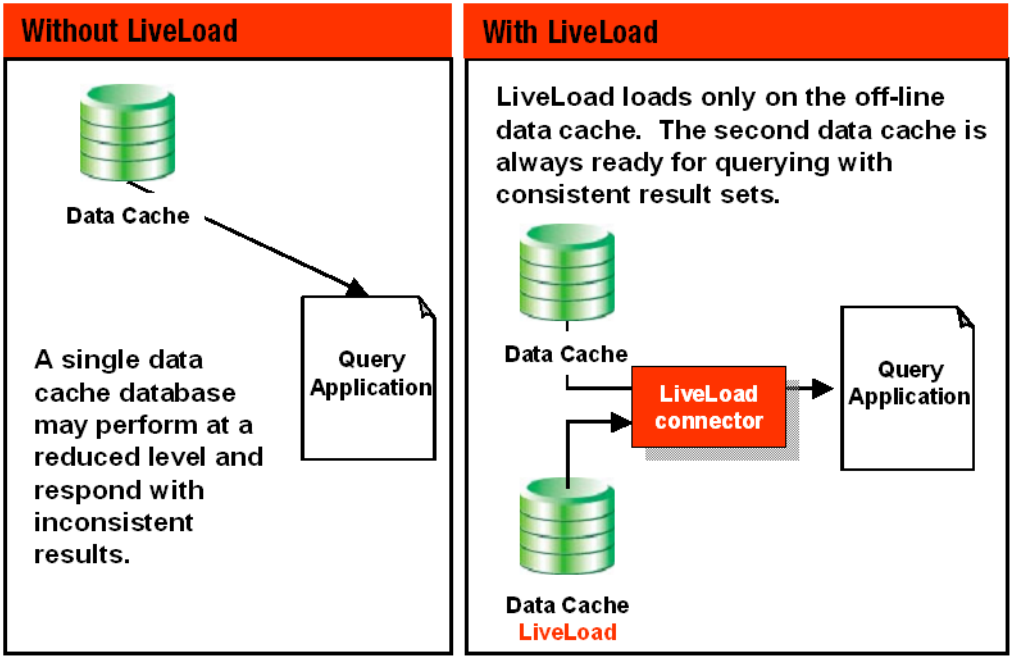
Data Integrator LiveLoad

Data Integrator LiveLoad is an optional component that uses database-mirroring techniques to ensure that your data warehouse is unaffected during batch loading. The use of LiveLoad can solve typical data management problems such as:

- Unacceptably slow data warehouse performance during a typical load.
- Having to take a warehouse off-line during a batch load to avoid offering compromised data from partial loads.

Data Integrator LiveLoad patent-pending technology solves these problems by maintaining two identical databases—a master and a mirror—through a single access point, thereby isolating target applications from the effects of a batch load. With LiveLoad, one database is always available for queries (live) while the other is being loaded with the most recent data (load). LiveLoad guarantees the data integrity of the live database at all times.

Changes executed in the load database are buffered in staging files for execution in the currently live database after the databases have been switched.



For detailed information, see the *Data Integrator LiveLoad User's Guide*.

Data Integrator Adapter SDK

The Data Integrator Adapter SDK provides a Java platform for rapid development of adapters to other applications and middleware products such as EAI systems. The adapters use industry-standard XML and Java technology to eliminate the learning curve. The adapters provide all necessary styles of interaction including:

- reading, writing, and request-reply from Data Integrator to the other systems.
- request-reply from the other systems to Data Integrator.

For detailed information, see the *Data Integrator Adapter SDK User's Guide* in your Data Integrator installation directory/Adapters/SDK/Doc.

Data Integrator management tools

Data Integrator has several management tools that assist in managing your Data Integrator components.

License Manager

The License Manager displays the Data Integrator components for which you currently have a license.

Repository Manager

The Repository Manager allows you to create, upgrade, and check the versions of local and central repositories.

Server Manager

The Server Manager allows you to add, delete, or edit the properties of Job Servers and Access Servers. It is automatically installed on each computer on which you install a Job Server or Access Server.

Use the Server Manager to define links between Job Servers and repositories. You can link multiple Job Servers on different machines to a single repository (for load balancing) or each Job Server to multiple repositories (with one default) to support individual repositories (separating test from production, for example).

You can also specify a Job Server as SNMP-enabled. For more information see "SNMP support" on page 541 of the *Data Integrator Designer Guide*.

Data Integrator operating system platforms

Data Integrator Designer runs on the following Windows platforms:

- NT
- 2000 Professional
- 2000 Server
- 2000 Advanced Server
- 2000 Datacenter Server
- XP
- 2003

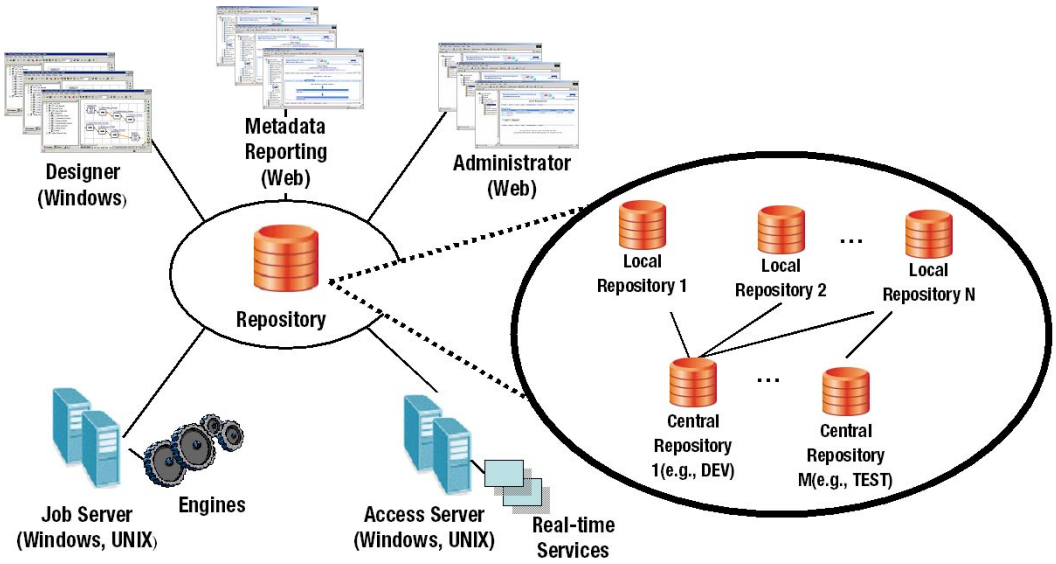
All other Data Integrator components run on the above Windows platforms and the following UNIX platforms:

- Solaris 2.8, and 2.9 (Sun OS releases 5.7, 5.8, and 5.9)
- HP-UX version 11.00 (PA_RISC 2.0), and 11.1
- IBM AIX 5.1, and AIX 5.2

Refer to your *Data Integrator Release Notes* for additional versions supported in this release. If you have a login to the Business Objects Customer Support Web site (go to Business Objects Customer Support Online to register), you can log in to http://www.techsupport.businessobjects.com and view the latest Product Compatibility Matrix for all currently supported versions.

Data Integrator distributed architecture

Data Integrator has a distributed architecture. An Access Server can serve multiple Job Servers and repositories. The multi-user licensed extension allows multiple Designers to work from a central repository. The following diagram illustrates both of these features.



You can distribute Data Integrator components across multiple computers, subject to the following rules:

- Engine processes run on the same computer as the Job Server that spawns them
- Adapters require a local Job Server

Distribute Data Integrator components across a number of computers to best support the traffic and connectivity requirements of your network. You can create a minimally distributed system, designed for developing and testing or a highly distributed system designed to scale with the demands of a production environment.

Resource distribution

Minimal implementation of a Data Integrator application requires several computers, some of which are devoted to supporting single applications. Here is an example of a distribution in a development environment:

Example: Development distribution model

Computer	Supported system components
Back-office application server	Back-office application
Back-office database server	Database underlying back-office application
Data Integrator server	Designer Repository Job Server Access Server Administrator Data Integrator Web Server Data warehouse database
Web application server	Web server (Apache, IIS, Personal Web Server) Application server (Cold Fusion, for example) Message Client library

A production implementation distributes some of the high-load applications to dedicated computers:

Example: Production distribution model

Computer	Supported system components
Back-office application server	Back-office application
Back-office database server	Database underlying back-office application
Database server	Data warehouse database
Data Integrator server #1	Designer Repository Job Server Access Server
Data Integrator server #2	Administrator Data Integrator Web Server
Data Integrator server #3	Repository Job Server
Data Integrator server #4	Repository Job Server
Web application server	Application server (Cold Fusion, for example) Access Server
Web server	Web server application (IIS, Personal Web Server) Message Client library

In the previous configuration, server #1 might be set up to support design, system #2 supports system administration, while servers #3 and #4 might be supporting only real-time messaging. Repositories on all three servers might be needed to support Oracle applications.

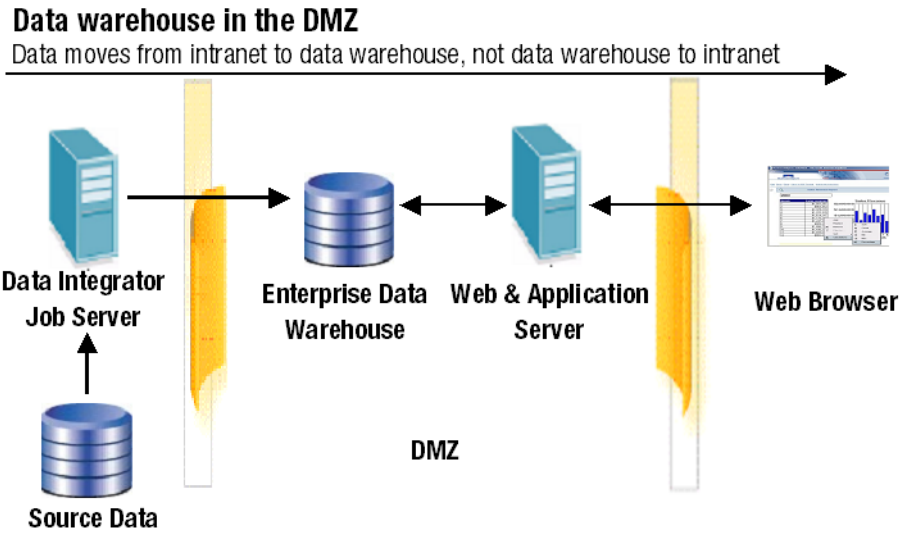
Network security options

You can distribute Data Integrator applications in a variety of locations on your network based on the access each component provides and the security systems your network employs. The following sections describe some network architecture configurations to help you determine where your data warehouse can reside.

Data warehouse with external Web server in DMZ

One configuration of network architecture places data accessed by a Web application on the same network as the Web application but isolated by a firewall from a company intranet. The intermediate network, referred to as a demilitarized zone (DMZ), is protected from public access by its own firewall. Security allows traffic only from the intranet out to the DMZ. If a computer hacker manages to infiltrate the DMZ, only the data in the data warehouse itself is at risk.

Placing the data warehouse in the DMZ works if the Web application only reads data from the data warehouse.



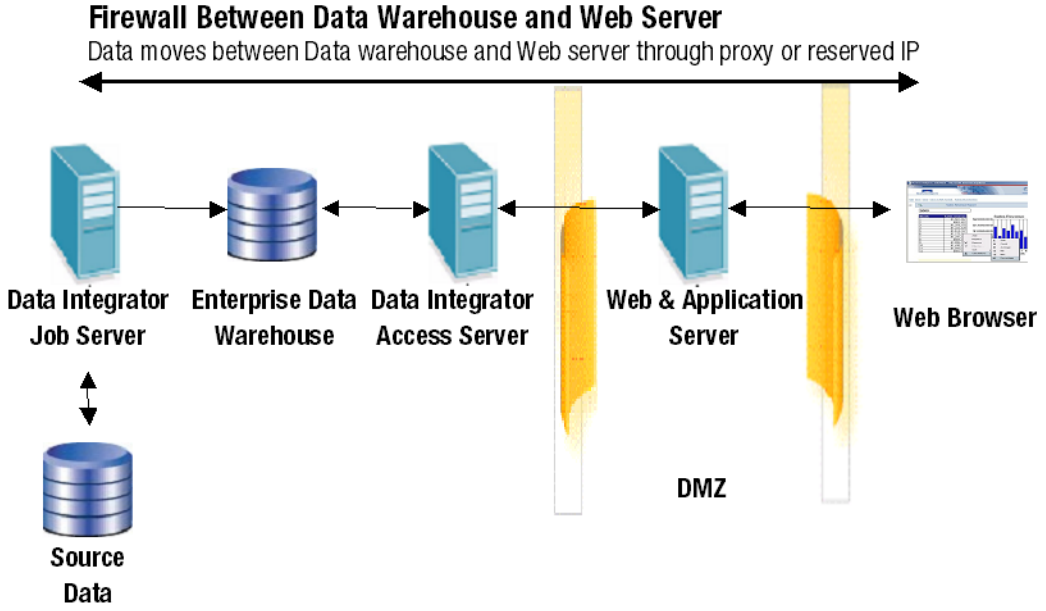
Firewall between data warehouse and external Web server

Another network configuration separates a data warehouse from the Web server using a firewall. There are several ways to provide access across the firewall:

- Configure communication to include IP address verification when receiving and sending messages
- Implement a proxy server to handle communication across the firewall

Assess your network security to determine if this configuration provides appropriate protection.

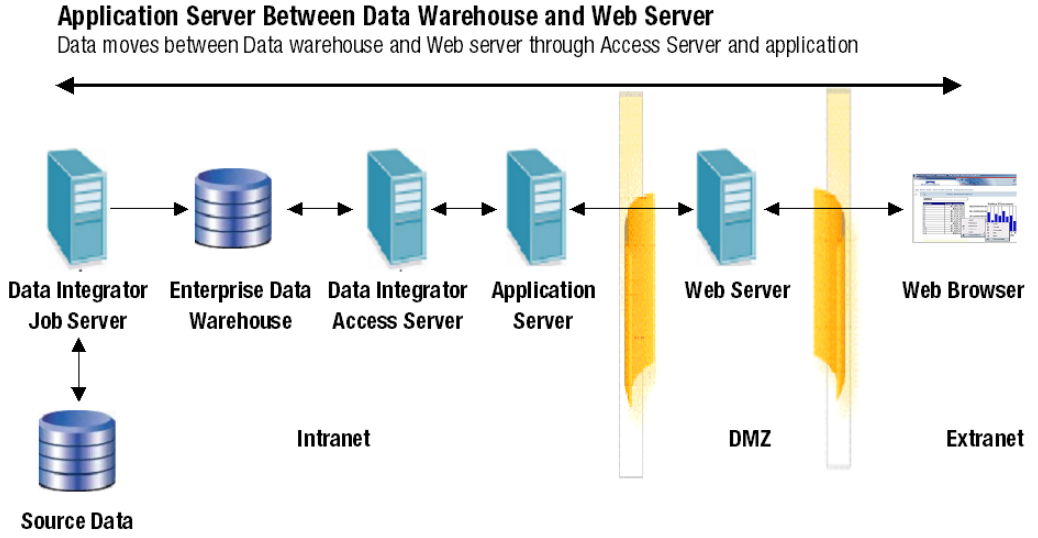
With proper security, placing the data warehouse behind a firewall separated from the DMZ works for both read-only and read/write applications.



Application server behind a firewall

Another configuration places an application server between the external Web server and a data warehouse. Typically, e-commerce applications designed for higher volume and both read and write transactions include an application server separate from the Web server. The application server generally manages resources, security across a firewall to the Web server, and connectivity between databases and the Web server.

The data warehouse resides on the intranet with the application server.



Access validation

One typical goal of Data Integrator applications is to provide the best data available, by request, from one application (such as an ERP system or data warehouse) to another (such as a Web server or real-time processing environment). With this in mind, there are many handshakes between applications to produce a complete data path. Actual connections vary among implementations.

Connection	Format	Security Infrastructure	Configuration
Web application to data warehouse	SQL	Direct calls (both in DMZ)	Application calls
		IP address validation	
		Proxy server	
Web application to LiveLoad Connector		IP address validation	Application calls (COM, JDBC)
Web application to Data Integrator Access Server	XML	IP address validation	Application calls (COM, C++, Java)
Application server to Data Integrator Access Server	XML	IP address validation	Application calls (COM, C++, Java)
Data Integrator Access Server to real-time services (via Job Server)	XML	IP address validation	Data Integrator Administrator
Real-time services to Data Integrator Access Server	XML	IP address validation	Data Integrator Administrator
Real-time services to data warehouse	SQL	Database login	Designer metadata
Real-time services to ERP system	SQL function calls	Security profiles, application login	Designer metadata

Host names and port numbers

Communication between a Web application, the Data Integrator Access Server, the Data Integrator Job Server, and real-time services occurs through TCP/IP connections specified by IP addresses (or host names) and port numbers.

If your network does not use static addresses, use the name of the computer as the host name. If connecting to a computer that uses a static IP address, use that number as the host name for Access Server and Job Server configurations.

To allow for a highly scalable system, each component maintains its own list of connections. You define these connections through the Server Manager, the Data Integrator Administrator, Repository Manager, and the Message Client library calls (from Web client).

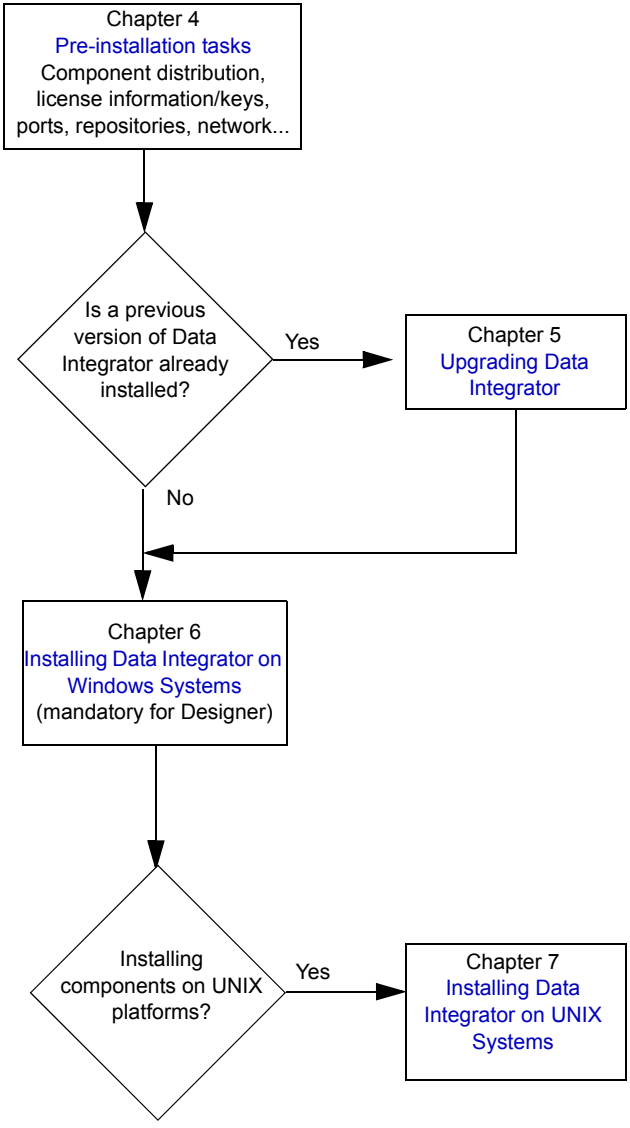
4 Preparing to Install Data Integrator

This chapter discusses information to consider before installing Data Integrator including:

- System installation overview
- System requirements
- Pre-installation tasks
- Data Integrator licenses
- Repository database information

System installation overview

There are several installation process stages. The following flowchart outlines steps and associated chapters in this manual:



➢ To prepare for Data Integrator installation

1. Decide which Data Integrator components you need to upgrade or install.

Component	Role
Designer	Use to develop and test batch, SAP R/3, and real-time jobs
Job Server	Processes jobs and real-time messages
Administrator	Use to monitor jobs, Access Servers, and adapters
Access Server	Provides real-time access to Data Integrator server components

2. Determine the computers on which to upgrade or install these components.

   You may choose to upgrade or install components on one or more computers based on available resources and amount of system traffic.

   You must install the Designer on a supported Windows platform. Install the Administrator, Job Server, and Access on Windows or UNIX platforms (see "Data Integrator operating system platforms" on page 31).

3. Determine the locale for each Job Server installation.

   Locales support the processing of data stored in different human languages such as rules for capitalization, time and date formats, and basic character sets.

   The Data Integrator installation program prompts you to specify a locale in terms of language, territory, and code page for each Job Server installation. Data Integrator obtains the default information from the host computer's operating system. When you install the Job Server, you are also installing Data Integrator's engine; the processing that occurs within Data Integrator will use this locale.

Data Integrator supports both single and multi-byte code pages. Many Asian scripts require multi-byte code pages. Data Integrator also supports `UTF-8`, a unicode, multi-byte code page that includes most of the world's languages.

In a production environment, carefully selecting code pages is especially important. Data Integrator supports and will transcode unique code pages when data passes from sources, through the engine, to targets. However for best performance, avoid transcoding by using the same code pages or the `UTF8` unicode for objects in a job.

For more information, see Chapter 9, "Locales and Multi-Byte Functionality," in the *Data Integrator Reference Guide*.

4. Update or install the Data Integrator Designer, Administrator, Job Server, and Access Server.

   See detailed installation steps for updating or installing Data Integrator in the remaining chapters of this guide.

5. Test connectivity:

   a. If you are using BusinessObjects Rapid Marts, install your Rapid Mart and run the initial extraction job to test connectivity.

      NOTE: Install pre-packaged BusinessObjects Rapid Marts or adapters from separate product CDs. For those products, follow the installation documentation included with the product CD.

   b. If you are not using a Rapid Mart, populate a data warehouse by creating and testing a batch job in Data Integrator.

   c. If you are updating or installing Data Integrator and you are not installing a Rapid Mart, see "Verifying connectivity" on page 105 for step-by-step instructions and the location of sample files for testing your Data Integrator installation.

6. Determine the business operations that you want to process.

   If you are creating a data warehouse for an analytic business application such as Actuate, Brio, Hyperion, or Supply Point, use SQL from the DBMS on which the data warehouse runs to query the data warehouse.

   If you are creating a data warehouse for a Web application, use SQL to query the data warehouse directly from your Web client application. When satisfied that you have completed your business operation design, specify the real-time data requirements for each operation.

   Data Integrator processes real-time business operations in the form of XML messages. You must translate the data requirements for each business operation into a Document Type Definition (DTD) or XML Schema format that describes the XML of each message that Data Integrator will process. See the *Data Integrator Designer Guide* for more information.

7. Connect to source or target applications through either of these methods:

   - The Message Client library. See "Installing Message Client libraries" on page 115 and "Using the Message Client library" on page 117.
   - A Data Integrator adapter. Find installation and configuration instructions for individual adapters and the Adapter SDK in the Adapters subdirectory of the Data Integrator package you install.

8. Implement and test data flows to support analytic or Web application needs.

   - For analytic applications, schedule batch jobs using the Data Integrator Administrator as described in the *Data Integrator Administrator Guide*.
   - For Web applications, configure real-time jobs as services in the Data Integrator Administrator as described in the *Data Integrator Administrator Guide*.

System requirements

Before installing Data Integrator, ensure that your system has compatible hardware and software. This section discusses:

- Data Integrator repository requirements
- Database requirements
- Data Integrator Designer requirements
- Data Integrator Job Server requirements
- Data Integrator Access Server requirements
- Data Integrator Administrator requirements
- Web applications communicating with Data Integrator

Data Integrator repository requirements

Data Integrator stores your design metadata for source and target tables as well as database functions and definitions of built-in Data Integrator objects in database tables. This set of database tables is called the Data Integrator **repository**. You must create space for local and/or central repositories on an existing database before you run the Data Integrator installation program. The Data Integrator installation program connects to the repository and populates it.

To open the Data Integrator Designer, launch the Designer application from the computer on which it is installed and log in to the repository. When you execute Data Integrator jobs from the Designer, the Data Integrator Job Server connects to this local repository to read application information.

A Data Integrator repository requires a minimum of 20 MB of free disk space on the database server. Built-in Data Integrator objects require less space. However, user-defined or purchased Rapid Mart applications might require more space. Sizing requirements depend on the scale of your planned operations.

Database requirements

Data Integrator supports the following database connections for repositories, data sources, and data targets. Database client and server software versions must be compatible.

O/S platforms	Database connection	Minimum version[1]	Repository	Source	Target
Windows and UNIX	Oracle[2]	8.0.5	Yes	Yes	Yes
	DB2/UDB[3]	7.2	Yes	Yes	Yes
	ODBC[4]		No	Yes	Yes
	DETAIL	5.1	No	Yes	No
	Sybase[5]	12.0	Yes	Yes	Yes
	Teradata	2.5	No	Yes	Yes
Windows	Microsoft SQL Server	7.0	Yes	Yes	Yes
	Informix[6]	7.3	Yes	Yes	Yes

NOTES:

1. Plus higher OS binary application compatible versions supported by the vendor. Refer to *Data Integrator Release Notes* for additional versions supported in this release. If you have a login to the Business Objects Customer Support Online documentation page, you can also view the Product Compatibility Matrix for all currently supported versions.

   To access this Web site, you must have a valid user name and password. To obtain your user name and password, go to Business Objects Customer Support Online and register.

2. When using an Oracle 64-bit client, the 32-bit client libraries must be installed. The default installation of the Oracle 64-bit client includes installation of the 32-bit client. For non-standard installations, these libraries must be specified as part of the installation. After installation, the libraries should be in the $ORACLE_HOME\lib32 directory.

   If these 32-bit libraries are not present, you will receive an error message stating that Data Integrator cannot load the client library. Examples of error messages:

- (5.2) 02-05-02 14:55:23 (E) (26797:0001) CON-120103: System call <dlopen> to load and initialize functions failed for <libclntsh.a>. Make sure the SHARED LIBRARY is installed and resides in the correct location.
- (5.2) 02-05-02 14:55:23 (E) (26797:0001) REP-100108: Cannot perform operation on Repository<Default Repository> because a connection to the repository was never opened. A connection must be opened to the repository before performing any operation.

3. Data Integrator supports a native DB2 interface. You can connect to MVS or AS/400 through DBConnect with the DB2 Client Enterprise Edition. If you plan to use template tables on DB2 on MVS, ensure that the database "create schema implicitly" setting is on.
4. You can connect to any ODBC-compliant database through an ODBC driver, for example DataDirect Technologies' 32-BIT ODBC driver. (For supported versions, see the Product Compatibility Matrix at Business Objects Customer Support Online.)
5. If you are using both Sybase and SQL Server on a Windows platform, ensure that the Sybase path precedes the SQL Server path in the environment variables %PATH% statement.
6. Install the Informix client software and DataDirect Technologies' 32-BIT ODBC driver. Create an ODBC connection to the Data Integrator repository database.

Data Integrator Designer requirements

The Data Integrator Designer requires specific Windows software and hardware.

Software

- Operating systems: see "Data Integrator operating system platforms" on page 31.
- Desktop Development Kit (DDK) option for extraction from SAP R/3 sources
- Database client library software for the database serving as your Data Integrator repository

Hardware

- Pentium processor with at least 128 MB RAM and 100 MB free disk space (memory-intensive jobs require more free disk space)
- Screen resolution of 1024 x 768 pixels with 16-bit or 65536 colors recommended (minimum 256 colors).
- Paging file with a minimum setting of 256 MB and a maximum setting of 512 MB

Data Integrator Job Server requirements

You can install the Data Integrator Job Server on the same computer as the Data Integrator Designer or on a different computer. If you choose to install the Designer and Job Server on the same computer, you must add the RAM and free disk space for the second component to the requirements of the first.

See also: "Additional system requirements for UNIX" on page 132.

Software

- Operating systems: see "Data Integrator operating system platforms" on page 31.
- If you plan to run scheduled jobs in a Windows environment, enable the Windows Task Scheduler on the Job Server computer.
- Install database connectivity software for the database(s) serving as your repository, sources, and targets.

  NOTES:

  - For Oracle bulk loading, install the Oracle bulk loading utility `sqlldr` on the Job Server computer.
  - For DB2 bulk loading, Data Integrator installs a proprietary executable, `db2bulkload`, in the Business Objects/Data Integrator/bin directory. To use DB2 bulk loading utility, the DB2 user specified in the datastore for the job must have system privileges. DB2 database or DB2 client application enabler software installations create a default user called `db2user`. Before you install a Job Server, you must know the password for `db2user`. The Job Server requires path information for the client application enabler software included in the alias for this user. For more information, see your DB2 database documentation.

Hardware

- Pentium processor with at least 256 MB RAM and 100 MB free disk space
- 1 GB virtual memory
- Recommended for best performance: dual processors (minimum 500 Mhz) with at least 512 MB physical memory

Data Integrator Access Server requirements

You can install the Access Server independently of other Data Integrator components.

You can configure a single Access Server to serve one or more Web applications, regardless of the load-balancing configuration. However, if request processing requires more resources, you can configure multiple Access Servers.

Software

- Operating systems: see "Data Integrator operating system platforms" on page 31

Hardware

- Pentium processor with at least 256 MB RAM and 100 MB free disk space
- 512 MB virtual memory
- Recommended for best performance: dual processors (minimum 500 Mhz) with at least 512 MB physical memory
- See also: "Additional system requirements for UNIX" on page 132.

Data Integrator Administrator requirements

You can install the Administrator independently of other Data Integrator components. You install the Web Server when you install the Administrator. The Web Server supports web browser capabilities for both the Administrator and the Metadata Reporting tool.

You can configure a single Administrator to serve one or more Access Servers, repositories and the Job Servers connected to them. You can also configure multiple Administrators.

See also: "Additional system requirements for UNIX" on page 132.

Software

- Operating systems: see "Data Integrator operating system platforms" on page 31
- Microsoft Internet Explorer

Hardware

- Pentium processor with at least 128 MB RAM and 100 MB free disk space
- Screen resolution of 1024 x 768 pixels with 16-bit or 65536 colors recommended (minimum 256 colors).

Web applications communicating with Data Integrator

To send messages to the Access Server for processing, a Web application must:

- Be able to generate XML-formatted strings
- Connect to the Access Server using Message Client library calls in C++, Java, or COM

Pre-installation tasks

Before installing Data Integrator components you must:

- Determine component distribution
- Obtain license keys
- Check port assignments
- Create a database for each repository
- Check network connections

Determine component distribution

The installation program can install multiple Data Integrator components:

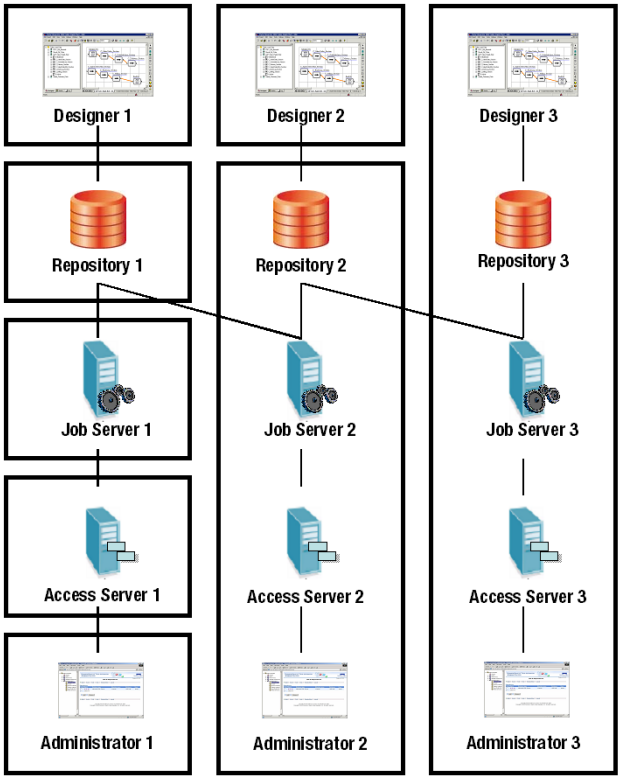
Component	Function	Number installed per computer	Always installed with the following components
Designer	Develop and test data flows	One (Windows only)	Repository Manager and the License Manager tools
Job Server	Process batch jobs and real-time messages	One (Windows or UNIX)	Server Manager tool, Data Integrator Service, HTTP adapter, SNMP Agent and Web Services adapter
Administrator	Monitor: jobs, Access Servers, and adapters	One (Windows or UNIX)	Data Integrator Web Server and Data Integrator Web Server service
Access Server	Provide network access to Data Integrator server components	One (Windows or UNIX)	Server Manager Tool and Data Integrator Service

NOTE: You can configure several Job Server and Access Server *instances* from one Job Server or Access Server installation. Configure Job Server and Access Server instances using the Server Manager.

You can install Data Integrator components on one or more computers based on available resources and the amount of traffic the system processes. Before installing Data Integrator components, consider the following restrictions:

- If you plan to use Data Integrator in a multi-user development environment, the computers containing the local repositories for each user must be able to connect to the computer containing the central repository.
- A local Data Integrator repository connects a Data Integrator Designer with a Data Integrator Job Server. You can install a Job Server on the same computer as the repository, on a stand-alone computer (with repository database client software), or on the same computer as a Data Integrator Designer.
- Any Data Integrator Designer can access any local repository and Data Integrator Job Server. Business Objects does not recommend that multiple Data Integrator Designers simultaneously access the same local repository and Job Server. Using simultaneous access risks metadata corruption in the repository. Also, you cannot link Data Integrator Designer to more than one local repository at any given time.
- Install database client software on the computer containing the Job Server to support the repository associated with that Job Server. For example, if the repository associated with a Job Server named “oradev” is an Oracle database, then Oracle client software must be installed on the computer containing the “oradev” Job Server.

The following diagram depicts a simple distribution of Data Integrator components among computers in a network. Note that you can install each Data Integrator on one or more computers. Also note that there are many possible connections between Data Integrator components. Each of the boxes in the diagram represents a different computer.



You can install the Administrator on any computer with or without other Data Integrator components. No license is required. It is Java-based. You also install the Data Integrator Web Server and Data Integrator Web Server *service* when you install the Administrator. The Data Integrator Web Server starts automatically on restart. See also the discussion under "Data Integrator distributed architecture" on page 32.

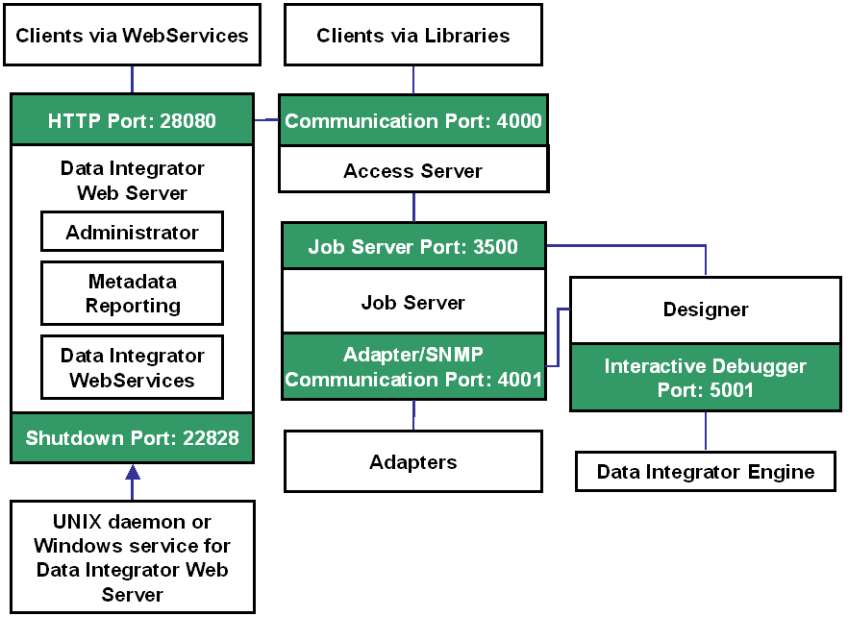
Obtain license keys

You must obtain the appropriate license key for each Data Integrator component. To do this, send your host ID information and a list of components to licensekey@businessobjects.com. Business Objects will generate a license file and contact you when it is available for access at webkey.businessobjects.com. For more information on licenses, see "Data Integrator licenses" on page 60.

Check port assignments

Verify that default ports for Data Integrator components are available and not in use by other programs on each computer.

For a development system, you can install many components on the same computer. This simplifies many connections between components (the host name is always the same), but you must still define connections based on TCP/IP protocol. The following example configuration diagram shows default port numbers.



This table details Data Integrator default ports:

Component Port	Description	Default	To set
Job Server			
Job Server port or TCP/IP port	Receives commands from Designer, Access Server, and schedulers	3500	Use Server Manager
Adapter and SNMP communication port	Receives commands and sends data to adapters	4001	Use Server Manager
Data Integrator Web Server			
HTTP port	Supports communication between Access Servers, the Administrator, and the Metadata Reporting tool	28080	In UNIX, use Server Manager In Windows, edit configuration files manually (see "To reset the ports for the Data Integrator Web server" on page 96)
Shutdown port	Connects to Data Integrator Web Server service. The Web Server service uses this port to start and shutdown the Web Server which supports the Administrator and the Metadata Reporting tool.	22828	In UNIX, use Server Manager In Windows, edit configuration files manually (see "To reset the ports for the Data Integrator Web server" on page 96)
Access Server			
Communication port	Communicates with the Administrator and the Metadata Reporting tool. Receives and sends messages from clients and services	4000	Use Server Manager
Designer			
Interactive Debugger port	Supports communication for the debugging feature	5001	Contact Business Objects Technical Support

Create a database for each repository

There are two types of repositories: local and central. Local repositories are working repositories where users create and modify applications. Central repositories serve as historical libraries, storing version-controlled master copies of applications you create and modify in local repositories.

You associate each local repository with one or more Data Integrator Job Servers. If you choose to install Job Servers on more than one computer, you must create a local repository to support each Job Server. There is no relationship between central repositories and Job Servers.

Create local repositories before installing each Job Server. For more information, see "Repository database information" on page 68.

- To set up a single-user development environment, create a database for the local repository.
- To set up a multi-user development environment, create databases for each local repository (one per user) and a database for the central repository. See Chapter 5, "Multi-user development," in the *Data Integrator Advanced Development and Migration Guide* for more information about using the multi-user development features.

Check network connections

Before installing Data Integrator components, prepare the computers involved and verify network connections by executing the `Ping` command on each computer.

For example, if the Designer will be on a Windows computer and your Job Server will be on a UNIX computer, from the Windows computer open the DOS prompt and enter:

```
C:\> ping hostname
```

where *`hostname`* is the host name of the UNIX computer.

Then from the UNIX computer, log in as root and from the prompt, enter:

```
$ ping hostname
```

where *hostname* is the host name of the Windows computer.

Data Integrator licenses

This section explains:

- License types
- Optional license-controlled features
- Managing your licenses
- Obtaining your license files
- Viewing previously-generated license files

License types

Data Integrator licenses can be:

- Unrestricted or restricted
- Evaluation, emergency, or permanent

Unrestricted

Business Objects generates unrestricted licenses for components not tied to a specific computer. Data Integrator verifies that the appropriate license exists each time a licensed component starts. The Data Integrator Designer component has an unrestricted license.

Restricted

Business Objects generates and uses restricted licenses for components tied to a specific computer. Restricted licenses use the ethernet/MAC address from the computer on which you installed the component. This value appears in the Data Integrator installation as “Host ID.”

When a Data Integrator component starts, it searches for the license and validates it against the Host ID.

NOTE: Data Integrator components using a restricted license require a new license file if:

- You transfer the installation to another computer, or
- You install a new network card on the computer

In either case, contact licensekey@businessobjects.com to submit your old and new host ID information along with a list of affected components. Business Objects will generate a new file and contact you when it is available for you to access at webkey.businessobjects.com.

NOTE: You cannot reset your license key to a NULL value.

Evaluation

An evaluation license allows you to run Data Integrator for a specific period of time. You can use this license on any computer until the license expires. An evaluation license is unrestricted for all Data Integrator components.

Emergency

Similar to evaluation licenses, the emergency license allows you to run Data Integrator for a limited period of time. Emergency licenses differ from evaluation licenses only in that the limited period of use is significantly shorter. Contact Business Objects Customer Support Online should you require an emergency license.

Permanent

A permanent license allows you to run Data Integrator components indefinitely. You purchase unrestricted permanent licenses for Data Integrator Designers and restricted permanent licenses for Data Integrator Job Servers. No license is required for the remaining core Data Integrator components.

Data Integrator component licenses:

Data Integrator component	Evaluation license	Emergency license	Permanent license
Designer	Unrestricted	Unrestricted	Unrestricted
Job Server	Restricted	Restricted	Restricted

Business Objects may bundle several features with Data Integrator depending on what you purchase. These features include:

HTTP adapter	Allows Data Integrator to use the HTTP and HTTPS protocols for outbound calls from its jobs to external applications. Requires Job Server installation.
Multi-byte capability	Supports the use of multi-byte data types (such as those used in China and Japan), and setting locales (languages, territories, and code pages) for sources, the Data Integrator Job Server and engine, and targets. Included with Designer or Job Server installation.
Real-time jobs	Allow: • Web-based transactions • XML message handling • On-demand processing • Real-time adapter connections • SAP R/3 IDoc support (purchased separately) Included with Designer installation.
SNMP agent	Supports SNMP protocol to communicate between (network management software) applications and Data Integrator Job Servers. With SNMP, you use your NMS application to monitor the status of Data Integrator jobs. Included with Job Server installation.
Web Services adapter	Supports outbound calls from Data Integrator jobs to external applications. Acts as a client to external applications that use Web Services. Requires Job Server installation.

Optional license-controlled features

Extend Data Integrator functionality with the following licensed-controlled components. The license for each component is tied to your Data Integrator Designer license. Purchase this additional functionality through your Business Objects Sales Representative.

Optional license-controlled features

Name	License Extension to Data Integrator that:
JD Edwards	Supports a batch interface to JD Edwards World and OneWorld sources and targets.
LiveLoad	Supports a mirrored database architecture for your data warehouse so data updates do not impact service.
Multi-user development	Supports version control of jobs. Developers work in local repositories then check work into a central repository. Store jobs for development, test and production servers.
Oracle Applications	Supports an interface to Oracle Applications eBusiness Suite software for batch data integration.
PeopleSoft	Supports an interface to PeopleSoft ERP and HRMS software.
Siebel	Supports an interface between Data Integrator and a third-party database running Siebel Applications. MS SQL Server is supported.
SAP BW	Supports an interface between Data Integrator and SAP Business Information Warehouse by using SAP BW's Staging BAPIs.
SAP R/3 ABAP	Supports R/3 data flow and data transport objects which allow Data Integrator to process batch data warehouse updates from SAP R/3.
SAP R/3 BAPI	Supports BAPI function calls from Data Integrator to SAP R/3 for transactions and queries. Includes BAPIs with table parameters.
SAP R/3 IDoc	Supports translation of IDoc messages sent from SAP R/3 to Data Integrator.

Managing your licenses

Easily manage your Data Integrator licenses using local license control. Simply obtain licenses from the licensing Web site and save them to the local host where you will install your licensed components.

Obtaining your license files

The Business Objects licensing Web site at webkey.businessobjects.com generates your license files and displays your license information.

➢ To obtain your license file(s) prior to installation

1. Decide which Data Integrator components you will install on which computers.
2. Create a list of all involved host IDs. To obtain a host ID, insert the Data Integrator CD into a computer and click the **Get Host ID** button.

   The **lmhostid** program is included on your CD. You can also run it from the DOS or UNIX command line to determine the host ID that Business Objects licensing will use.
3. From your browser window, go to the licensing Web site at webkey.businessobjects.com, enter the code provided in your License Authorization Code e-mail letter, then click **Submit**.
4. At the first License Fulfillment page, confirm the line items of your Data Integrator order.

   NOTE: That page does not specify quantities.
5. Click to select a line item for which you want to generate a license file, then click **Generate License**.
6. At the second License Fulfillment page, enter the host ID for the computer on which you will be installing the Data Integrator component(s) associated with your chosen line item(s). Click **Generate License**.

   NOTE: You can select multiple items for the same host ID.

Business Objects generates a License Certificate page. This page contains product component name(s), effective date, license type, license key information, and host ID information. When saved, this page becomes a license file.

7. Save the License Certificate page.
   - Method 1:
     a. Click the **Save** button on the license file page.
     b. Browse to the location where you want to save this file.

        Business Objects recommends you save all license files to your Data Integrator/License directory; however, you can initially save license files anywhere you choose. During installation, Data Integrator automatically copies license files to the Data Integrator/License directory. The copied files become the official license keys for your product components.
     c. Enter a unique, descriptive name for this license file with ".lic" as the file extension. (Example: JScomputer1.lic)
   - Method 2:
     a. In the browser window's menu, click **File > Save As**. A warning message may appear that says, "This Web page may not save correctly, save it anyway?" Click **OK** to get to the Save Web Page window.
     b. Browse to the location where you want to save this file.

Business Objects recommends you save all license files to your Data Integrator/License directory; however, you can initially save license files anywhere you choose. During installation, Data Integrator automatically copies license files to the Data Integrator/License directory. The copied files become the official license keys for your product components.

c. From the **Save as type** list, choose any file extension.

d. Later, you must manually change this extension to ".lic" for Data Integrator to recognize the file.

e. In the **File name** text box, enter a unique, descriptive name for this license file.

8. Generate a license file for each host ID by repeating steps 4 through 7.

NOTE: When saving license files on an AIX platform using either method, you must name the file al_license.lic.

You can generate license files at your convenience. For example, your Data Integrator product may be associated with five license files. However, you may generate only two of these licenses now and return to the Web site later to view your earlier licenses or generate more.

Viewing current license information

On Windows, go to **Start** > **Programs** > **Data Integrator** > **License Manager** to view license information for your Data Integrator components and licensed extensions.

Viewing previously-generated license files

View previously-generated license files to find key information needed for new licenses such as host IDs.

➢ To view previously generated license files

1. From your browser window, go to Business Objects' licensing Web site at webkey.businessobjects.com, reenter your License Authorization code, and click **Submit**.
2. At the first License Fulfillment page, confirm the line items of your Data Integrator order.

   NOTE: Quantities are not specified here.
3. Click **View License**.

   The Select Fulfillment Record page appears, showing all previously-generated licenses against the order(s) you placed. Previously-generated licenses are shown in a table associated with product number, option information, the number of days the license is valid, host ID number, and effective date.
4. At the Select Fulfillment Record page, click to select a license, and click **View License**.

   The License Certificate page appears, containing product component name(s), effective date, license type, license key information, and host ID information.

Repository database information

This section describes information necessary for you to create either local or central repositories in various types of databases.

For DB2

A local or central Data Integrator Repository requires that you create a dedicated database and define a user with the right to:

- Create a table
- Start a session
- Create a sequence

Required Data

User name/password:_____
DB2 Data source:_____
DB2 version:_____

Install the DB2 application enabler software and use DB2 Control Center and DB2 Script Center to verify the connection between the Designer computer and the Data Integrator Repository computer.

For Informix

A local or central Data Integrator Repository requires that you create a dedicated database and define a user with the right to:

- Create a table
- Start a session
- Create a sequence

- Unlimited tablespace

Required Data

User name/password:_____
Informix Data source:_____
Informix version:_____

Install the Informix client software and a supported version of DataDirect Technologies' Connect for 32-BIT ODBC 32-BIT driver. Create an ODBC connection to the Data Integrator Repository database.

For Oracle

A local or central Data Integrator Repository requires that you create a dedicated database and define a user with the right to:

- Create a table
- Start a session
- Create a sequence

Grant the user the *connect* and *resource* roles.

Required Data

User name/password:_____
Database connection name:_____
Version:_____

Install the Oracle client software and use SQL*Plus to verify the connection between the Designer computer and the Data Integrator repository.

For Microsoft SQL Server

A local or central Data Integrator Repository requires that you create a dedicated database and define a user aliased to **dbo**.

Required Data

User name/password:_____

Database server name:_____

Database name:_____

SQL Server version:_____

Install Microsoft SQL Server client software and use Microsoft SQL Server's SQL Query Tool to verify the connection between the Designer computer and the Data Integrator Repository computer.

For Sybase

A local or central Data Integrator Repository requires that you create a dedicated database and define a user.

Required Data

User name/password:_____

Database server name:_____

Database name:_____

Sybase version:_____

Install Sybase and verify the connection between the Designer computer and the Data Integrator Repository computer.

5 Upgrading Data Integrator

This chapter provides information required to upgrade from previous product versions. This chapter includes:

- Upgrade paths
- Upgrading your Data Integrator system
- Before upgrading repositories with the Repository Manager:
- After upgrading Data Integrator

Upgrade paths

You upgrade Data Integrator by installing the new Data Integrator components then upgrading your repositories. This section discusses:

- Data Integrator component upgrades
- Repository upgrades
- Upgrading to a multi-user development environment

Data Integrator component upgrades

Other than repositories, all Data Integrator components must be upgraded to the same version.

Business Objects recommends that after you back up existing repositories using your database's backup utility, you upgrade components in the following order:

1. Data Integrator Designer
2. Data Integrator repository
3. Data Integrator Job Server
4. Data Integrator Access Server
5. Data Integrator Message Client and LiveLoad Connector libraries
6. If you are using the SAP R/3 ABAP interface, re-install the Data Integrator functions for SAP R/3. For more information, see "Installing Data Integrator functions on SAP R/3" on page 16 of the *Data Integrator Supplement for SAP*.

Repository upgrades

Minor releases do not require a repository upgrade. For example, if 6.0 is a major release and 6.1 is a minor release, then when upgrading from 6.0 to 6.1, it would not be necessary to upgrade your repository. However, product features introduced in the latest product version might not be available without the repository upgrade. To see whether you need to upgrade your repository for a particular release, see the Release Summary document.

Using database backup utilities, always make a copy of your existing repository before upgrading. Note that you will upgrade using the original repository (not your backup copy).

The following repository upgrade paths are available:

To upgrade from	Procedure
Version 3.1, 4.x, 5.x, or 6.0	Install this release using one of the following methods: • Upgrade your repository during the installation process • Upgrade your repository after installing the Designer by running the Repository Manager utility and choosing the **Upgrade** option. See "Before upgrading repositories with the Repository Manager:" on page 81 if you choose this second option.

Upgrading to a multi-user development environment

To use Data Integrator for controlling changes in a multi-user environment, you must create a central repository in addition to local repositories for each Data Integrator Designer user. However, you do not need to create or link these repositories during the installation process. You can create and link central and local repositories at any time using the Repository Manager and Data Integrator Designer.

For more information on central repository connection restrictions, see "Determine component distribution" on page 53. For information about the multi-user development environment and how local and central repositories are used, see Chapter 5, "Multi-user development," in the *Data Integrator Advanced Development and Migration Guide*.

Upgrading your Data Integrator system

When moving to a new product version, upgrade your test or development environment before upgrading your production environment.

To install a new product version on a computer hosting an existing version, first uninstall the old version. Note that uninstalling does not remove your custom files (configuration files, job logs, and job launch files), so after you install the new version, you have two options for handling the old directory and custom files:

- To create new configurations in your new version, manually delete the old directory and its files
- To use old configurations in the new version, transfer specific files from the old directory to your new Data Integrator directory

  See "Using configurations from a previous version" on page 76 for details.

NOTE: The procedure in the following section directs you to follow instructions in "Running the installation program" on page 86. These instructions explain how to use the Data Integrator installation program (Setup.exe).

Business Objects recommends that while upgrading Data Integrator you do not use the installation program to reconfigure an Access Server. Instead, after installing the Designer and required repositories, Job Servers, and Access Servers, return to this chapter and continue. Use the explicit instructions in this chapter as your guide for upgrading Data Integrator.

Using configurations from a previous version

If you want to use custom configurations from a prior version of the product:

1. Move the DSConfig.txt file, from the `bin` subdirectory, into a newly created "dummy" installation directory structure before installing the new version.

   Example "dummy" directory structure:

   D:/Program Files/Business Objects/Data Integrator 6.1/bin/DSConfig.txt

   During the installation process, Data Integrator will overwrite the "dummy" directory structure with the new installation structure, merging custom settings (such as repository and Access Server connections added to the Administrator) from your old DSConfig.txt file into the new DSConfig.txt file.

2. Also, before installing the new version, manually move the following subdirectories out of your installation directory.
   - Conf - Contains Administrator settings
   - Log - Contains old logs

3. If your current installation includes support for real-time message handling, then when you configured an Access Server (using the Data Integrator Server Manager) you specified a location for each Access Server configuration directory. If those directories are in the Data Integrator installation directory make a copy of each Access Server configuration directory.

   Save your current copy until after you test the upgrade and confirm that your new system is working as expected. When you merge the old DSConfig.txt file with the new one, Data Integrator should find the Access Server configuration directories. If not, use the Server Manager to point to the backup Access Server directories.

4. Uninstall the existing product.
   - In Windows, use the Add/Remove Programs utility.
   - In UNIX see "Uninstalling Data Integrator" on page 161.
5. Install the new product then move the content of the old `Conf` and `Log` subdirectories into the new subdirectories.

Uninstall existing product version(s)

Before installing Data Integrator on a system, make sure all previously installed versions of the product are removed from that computer. If this is a new installation, skip this section and go to "Install a new version" on page 78.

➢ Before installing a new version

1. Use your database backup utilities program to back up all existing repositories.
2. Open the Server Manager tool.
3. Make note of the port number used by each existing Access Server and Job Server.

   Because these port numbers are part of the server configuration, use the same port numbers to configure Job Servers and Access Servers after the upgrade. If running more than one Job Server or Access Server on a single machine, assign different port numbers to each. Port numbers for each installed component must be unique.

Server instance	Default communication port	Example configuration
Access Server #1	4000	4000
Access Server #2	4000	4001
Administrator	28080	28080
Job Server #1	3500	3500
Job Server #2	3500	3501
Job Server #3	3500	3502

4. Close the Server Manager tool.
5. Open your Windows Services panel and stop all Data Integrator Services (Data Integrator Service and the Data Integrator Web Server).

   You can use the Task Manager to verify that no Data Integrator–related processes (processes beginning with "al_") are running.
6. Uninstall the existing version of the product.

   To uninstall Data Integrator from a Windows operating system, go to **Start > Settings > Control Panel > Add/Remove Programs (or Change/Remove Programs)**, select the product application, and click **Remove**.
7. Log off of your computer then log back on to refresh the environmental variable.
8. Follow the instructions in the next section, "Install a new version".

Install a new version

Log off of your computer then log back on (or restart the computer) to refresh the environmental variable before installing this product.

➢ To install a new version of Data Integrator

1. Use the Data Integrator installation program to install the new Data Integrator version, but do not configure an Access Server at this time. Follow the instructions in "Installing Designer, Job Servers, and Access Servers" on page 86. When finished, return to this page and proceed with step 2.
2. Restart the computer when prompted by the installation program.

3. Configure Access Servers.
   a. From the **Start** menu, choose **Programs** > **Data Integrator** > **Server Manager**.
   b. In the Data Integrator Server Manager Utility window, click **Edit Access Server Config**.
   c. In the Access Server Configuration Editor window, click **Add**.
   d. In the Access Server Properties window, enter the name of the Access Server directory from step 4; refer to the port numbers mentioned in step 3. No parameters are required. Click **OK**.
   e. To configure more than one Access Server on this computer, repeat the previous step. When you are finished adding Access Servers, click **OK**.
   f. In the Data Integrator Server Manager Utility window, click **Restart**.
4. Verify that Data Integrator processes are running.

   In the Windows Task Manager, click the **Processes** tab, and look for:

   - `al_jobserver.ex`
   - `al_jobservice.e`
   - `AL_AccessServer`
5. If you are using the LiveLoad option, install the new version of LiveLoad Connector on computers where they are used.

   See "Installing LiveLoad Connectors" on page 7 of the *Data Integrator LiveLoad User's Guide*.
6. Install the new version of Business Objects Client libraries on the computers where they are used.

   See "Installing Message Client libraries" on page 115.

If you are upgrading your Message Client library, you must stop your Web application, install the Message Client library files, then restart your Web application computer.

7. Upgrade or create a repository in the Repository Manager.

   Go to **Start > Programs > Business Objects Data Integrator 6.0 > Repository Manager**.

   See "Upgrading repositories" on page 81 for details.

Upgrading repositories

After installing a new version of Data Integrator, you can create new repositories or upgrade existing repositories. To associate an existing repository with your new installation of Data Integrator, you must upgrade that repository.

Before upgrading repositories with the Repository Manager:

- Back up each repository that you intend to upgrade using an external database backup utility. For more information, see your database documentation.
- Install the new version of Data Integrator Designer

➢ To upgrade a repository

1. From the **Start** menu on the computer where you installed the Designer, choose **Programs > Business Objects Data Integrator > Repository Manager**.
2. Enter the repository connection information for your original repository in the Repository Manager window.

   NOTE: Do not attempt to upgrade the backup copy of your repository or you will experience upgrade errors.

3. Click **Get Version**.

   The Repository Manager displays the version number recorded in the repository.

4. Click **Upgrade**.

   The Repository Manager converts the pre-existing repository to the format required by the new version of Data Integrator and adds metadata for new and changed objects.

5. Repeat steps 2 through 4 for each repository you want to upgrade.

   NOTE: Corresponding local and central repositories must have the same Data Integrator version.

6. Click **Close**.

NOTES:

- If you are upgrading from version 5.0 or earlier, after upgrading your repositories, reimport all source and target database tables through the Designer to ensure good data in the repository's primary/foreign key table.
- If you need to create or upgrade multiple repositories, use the RepoManBatch.exe utility. For more information, see "Creating or upgrading repositories in batch mode" on page 100.

After upgrading Data Integrator

Upgrading requires you to re-establish the overflow directory for your bulk loader options. Overflow directories may have been specified as target table options or as part of the datastore definitions.

In addition, run tests of your production jobs to ensure that your existing designs continue to run as expected in the new version of Data Integrator. When you are confident that the new version of Data Integrator performs as well as or better than the previous version, consider upgrading your production environment.

For tips, tricks, and solutions to any recent upgrade issues, go Business Objects Customer Support Online, and see the *Data Integrator Migration Guide*.

Unsuccessful upgrade

Errors that prevent installation progress (for example an incorrect server name, user name, or password) yield immediate error messages.

For more information about troubleshooting installation problems, see "Recovering from installation errors" on page 103.

Successful upgrade

A "successful completion" message verifies successful Data Integrator installation.

The `LINK_DIR` system variable is defined during the Data Integrator installation. Its value is set to the location of the directory in which you installed Data Integrator.

6 Installing Data Integrator on Windows Systems

The Windows installation program:

- Copies the files from the installation CD to their designated locations
- Installs Data Integrator Service as a Windows service
- Prompts you to configure repositories, Job Servers, and Access Servers (although these can also be configured after installation)

This chapter contains the following sections:

Running the installation program

You can install the following components on Windows systems:

- Data Integrator Designer
- Data Integrator Job Server
- Data Integrator Access Server
- Data Integrator Administrator

Run the installation program to install all components or a subset. What you choose to install will depend on how you distribute the components across computers. See "Determine component distribution" on page 53. You can run the program for Installing Designer, Job Servers, and Access Servers.

Errors that prevent installation progress (for example, an incorrect server name, user name, or password) yield immediate error messages. For information about solving installation problems, see "Recovering from installation errors" on page 103.

Installing Designer, Job Servers, and Access Servers

Before installing Data Integrator components:

- If using local license management, save your license file to an appropriate directory.
- On Job Server hosts, ensure you can connect from an RDBMS SQL tool (such as Oracle SQL*Plus or DB2 Script Center) to the repositories that the Job Servers on that host will access.
- On Access Server hosts, identify a free port number for the Access Server Administrator browser interface. By default, the installation program uses port number 28080.

➢ To select installation options and copy files

1. Log on to your computer using an account with local Windows administrator privileges and access to your network.
2. Insert the Data Integrator Installation CD into your CD-ROM drive.

   The main installation window appears with several options.

   NOTE 1: If the installer does not automatically appear, run `Autorun.exe` from the top level of the CD contents.

3. Click **Install Data Integrator**.

   If a previous version of the product exists on your computer, a question message appears reminding you to stop all Data Integrator components and stop the Data Integrator service (from the Windows Services dialog box) before continuing.

4. If existing Data Integrator components and/or services are running, stop them and click **Yes** to continue the installation.

   The User Information window appears showing your computer's host ID. You may want to note your host ID.

5. Enter your information in the **Name** and **Company** boxes. Click **Next**.
6. In the License Agreement window, read the software license agreement.
7. Click **Yes** if you agree to the terms. You can also **Print** the document for your records.

8. In the Data Integrator Location window, indicate a directory into which Data Integrator will be installed.

   The default is C:\Business Objects\Data Integrator. Accept the default or browse to choose a different directory. Click **Next**.

   NOTE: This documentation assumes the Data Integrator Directory is C:\Business Objects\Data Integrator.

9. In the Data Integrator License Information window, enter or **Browse** to the location of your Data Integrator license file. (See "Data Integrator licenses" on page 60 for information on obtaining your Data Integrator license file.)

   Click **Next**.

10. In the Select Components window, select one or more of the following Data Integrator components to install on your computer:

    - Data Integrator Designer (includes the license extensions that you purchased)
    - Data Integrator Job Server
    - Data Integrator Administrator (includes Access Server, metadata reporting tool, and Data Integrator Web Server)
    - Data Integrator Multi-byte (only selectable if you have selected one of the above components)
    - Data Integrator Documentation (includes the Tutorial)
    - Data Integrator Context-sensitive Help

    Click **Next**.

11. In the Select Program Folder window, choose or enter the program directory, and click **Next**.

12. In the Start Copying Files window, review your configuration settings and if correct, click **Next**.

    The Setup Status window displays progress in copying files. The installer creates a system variable, LINK_DIR, that defines the path used by the Data Integrator components.

13. When copying is complete, you are asked if you want to create or upgrade a repository.
    - If you want to defer creating or upgrading repositories, click the second option and continue the installation at "To enter locale information" on page 90.
    - Otherwise click the first option and proceed to "To create or upgrade repositories" on page 89.

➢ To create or upgrade repositories

This process creates or updates the Data Integrator repository schema on one or more databases. You must have already created the database on a supported RDBMS.

1. In the Repository Manager window, enter your repository database information from "Repository database information" on page 68.
2. Select your repository database type (**Local** or **Central**).
3. (Optional) You can also select the **Show Details** check box. It allows you to view the SQL code used to create or upgrade a repository from the text box in the Repository Manager.
4. Select either **Create** or **Upgrade**.

   The installation program creates or upgrades the repository in the specified database.
5. If you want to create or upgrade more repositories, repeat steps 1 through 4. Otherwise, click **Close**.

The installation program continues with locale configuration.

➢ To enter locale information

1. In the Locale window, accept the default language, territory, and code page settings (same as the host's computer) or select others and click **OK**.

   NOTE: Here the locale (language, territory, and code page) determines how the Data Integrator Job Server and engine process data. Data Integrator uses international standards for language and territory codes. A code page specifies the encoding for a particular character set. Asian scripts require multi-byte code pages (typically two-byte characters). For more information, see Chapter 9, "Locales and Multi-Byte Functionality," in the *Data Integrator Reference Guide*.

2. The installation program displays the Data Integrator Server Manager Utility window:
   - If you did *not* select Data Integrator Job Server in the Product Components window or do *not* want to configure Job Servers now, skip to "To configure Access Servers" on page 94.
   - If you selected Data Integrator Job Server in the Product Components window and want to configure Job Servers now, proceed to the next section, "To configure Job Servers".

➢ To configure Job Servers

1. In the Data Integrator Server Manager window, click **Edit Job Server Config**.
2. In the Job Server Configuration Editor window, click **Add** to add a Job Server.

3. In the Job Server Properties window, enter information about the Job Server.

Property	Description
Job Server name	Enter a name that uniquely identifies the Job Server. The Job Server name cannot be changed.
Job Server port	Enter the TCP/IP port that the Job Server uses to receive commands from the Designer and the Access Server. If a computer hosts multiple Jobs Servers, each Job Server must have a unique port number. Choose a port number for a server that is not used by another process on the computer. If you are unsure of which port number to use, use the default port number and increment it for each additional Job Server you configure.
Support Adapter and SNMP communication	Select this check box if the Job Server manages adapters or if this Job Server will be the one to communicate with an SNMP agent. Each computer that hosts adapters and an SNMP agent must have exactly one Job Server designated to manage them.
Communication port	Enter the port number that the Job Server uses for communicating with adapters or an SNMP agent. The default is 4001.
Enable SNMP	Enables this Job Server to send events to the SNMP agent. For more information, see "SNMP support" on page 541 of the *Data Integrator Designer Guide*.

4. Under **Associated Repositories**, enter the repositories associated with this Job Server. Every Job Server must be associated with at least one local repository.
   - Click **Add** to associate a new local repository with this Job Server.
   - Under **Repository Information**, enter the required connection information for the repository.

     NOTE: Do not enter a blank or invalid password; this may prohibit you from logging on to Data Integrator.

- Select the **Default repository** check box if this is the default repository for this Job Server. You must specify exactly one default repository.
- Click **Revert** to undo these entries.
- Click **Apply** to save your entries and associate the repository with the Job Server.

  The associated repository updates with the Job Server's computer name and port number.
- Repeat these steps until you are done adding all associated repositories.

5. Edit and delete associated repositories as necessary.
   - Select a repository and click **Edit** to change the repository's default setting.
     - Under **Repository Information**, enter the password.
     - Select or clear the **Default repository** check box, indicating whether this is the default repository for the Job Server.
     - Click **Apply** to save the change, or click **Revert** to make no changes.
   - Select a repository and click **Delete** to remove an associated repository.
     - Under **Repository Information**, enter the password.
     - Click **Apply** to remove this associated repository, or click **Revert** to make no changes.
6. When you are finished adding all associated repositories, including one default, click **OK**.
7. From the Job Server Configuration Editor window, complete any other necessary steps. You can:
   - Add another Job Server
     - Click **Add**.
     - Repeat steps 3 through 6.

- Change a Job Server's configuration
  - Select the Job Server.
  - Click **Edit**.
  - Repeat steps 3 through 6.
- Remove a Job Server
  - Select the Job Server.
  - Click **Delete**.

  NOTE: If the Job Server has associated repositories, you must first delete those and then click **OK** before you can delete the Job Server.
- Resynchronize your Job Server configuration with a repository
  - Select the Job Server.
  - Click **Resync with Repository**.
  - In the Job Server Properties window, select an associated repository.
  - Click **Resync**.
  - Under **Repository Information**, enter the repository password.
  - Click **Apply**.

  Cases when you must resynchronize the Job Server and repository include:
  - You have uninstalled Data Integrator and are reinstalling the same Data Integrator version without creating a new repository.
  - You created a new repository using the Repository Manager after installing Data Integrator.

  If you resynchronize your Job Server configuration with a repository, you must reassociate this repository with the Administrator and the metadata reporting tool. See the *Data Integrator Administrator Guide*.

8. When you have configured all the Job Servers you want, click **OK**.
9. You return to the Data Integrator Server Manager window:
   - If you did *not* select Data Integrator Administrator in the Select Components window or do *not* want to configure Access Servers now, skip to "To restart servers" on page 96.
   - If you selected Data Integrator Administrator in the Select Components window and want to configure Access Servers now, proceed to the next section "To configure Access Servers".

➢ To configure Access Servers

When you configure the location for an Access Server installation, Data Integrator creates space for the Access Server log files.

1. In the Data Integrator Server Manager window, click **Edit Access Server Config**.
2. In the Access Server Configuration Editor window, click **Add**.

3. In the Access Server Properties window, enter the Access Server configuration information.

Field	Description
Directory	The location of the log files for this instance of the Access Server. Do not change this value after the initial configuration.
Communication Port	The port on this computer that the Access Server uses to communicate with the Administrator. Make sure that this port number is unused and is unique for each Access Server.
Parameters	You can leave this box blank. Access Server parameters are described in "Configuring Access Server output" on page 86 of the *Data Integrator Administrator Guide*.
Enable Access Server	An option to control the automatic start of the Access Server when the Data Integrator Service starts.

Click **OK**. You return to the Access Server Configuration Editor window:

4. Optionally, perform further actions:
   - To change an Access Server's configuration:

     In the Access Server Configuration Editor window, highlight the Access Server, click **Edit,** and repeat step 3.
   - To remove an Access Server:

     In the Access Server Configuration Editor window, highlight the Access Server and click **Delete**.
5. After you configure your Access Servers, click **OK** to return to the Data Integrator Server Manager window.

   Continue with the next section, "To restart servers".

➢ To restart servers

1. In the Data Integrator Server Manager window, identify the account information that the Data Integrator Service uses to access system services. Click **Restart**. The Data Integrator Service starts.

   NOTE: You must specify a Windows user account and password to start the Data Integrator Service and access other Data Integrator-related applications on your network. The installation process automatically uses the current Windows user account to install the service. However, Business Objects recommends that you not use your personal Windows user account to run the Data Integrator Service. You must configure this account with the right to "log on as a service."

2. In the Information window, click **OK**.

   The Data Integrator Web Server starts. When the Installation window informs you that installation is complete, click **OK**.

3. If the Data Integrator CD is still in the CD-ROM drive, remove it.

4. In the Setup Complete window, click **Yes** to restart your computer, or click **No** to restart your computer later. Click **Finish**.

➢ To reset the ports for the Data Integrator Web server

NOTE: The Data Integrator Web Server uses a Tomcat server as a servlet engine. When you install the Administrator, you also automatically install the Data Integrator Web Server and its service. The Web Server service uses port 22828 to communicate with the Web Server. The Web Server uses port 28080 as its HTTP port. If either port is used by another application, you must change them.

1. Go to the Data Integrator Directory\ext\WebServer\conf directory and open the server-di.xml file.

2. Locate the tag:

   ```
   <Server port="22828", shutdown =
   "shutdown" debug = "0">
   ```

   Change the value of this port.

3. Locate the tag:

   ```
   <Connector className="org.apache.catalina
   connector.http.Http.service.Connector"
   port ="28080"/>
   ```

   Change the value of this port.

4. Restart the Data Integrator Service and Data Integrator Web Server.

Verifying that Job and Access Servers are running

To verify that Job Servers are running, check in the Windows Task Manager **Processes** tab for:

- `al_jobservice.e` (represents the Data Integrator service)
- `al_jobserver.ex` (one per Job Server)
- `aL_AccessServer` (one per Access Server)

If you do not see all the processes expected, check for error messages in these log files:

- Job Server logs are in
  `Data Integrator Directory``\log\`
  `JobServerName``\server_eventlog.txt`
- Access Server logs are in
  `AccessServerPathName``\error_mm_dd_yyyy.log`

To perform a connectivity test for the Access Server, see "Verifying connectivity" on page 105.

Working with multiple network cards

If you have multiple network cards on the Designer computer and you are trying to connect to a remote Job Server, you may get a notification error when you open the Designer. To fix this, specify the IP address for the network card you want the Designer to use. For more information, see your Microsoft Windows documentation.

When there are multiple network cards in a remote Job Server computer, the Designer may not find the one you want it to use. You may need to:

1. Determine which cards are available by running the `ipconfig` command in DOS.
2. Use a SQL editor to update the value for MACHINE_NAME to the IP address you want the Job Server to use to connect to the Designer. This column is in the AL_MANCHINE_INFO repository table. A row is created in this table for each configured Job Server.

Starting Data Integrator services automatically

The Data Integrator Service and Data Integrator Web Server service start automatically when the computer restarts. The Data Integrator Service, in turn, starts Job Servers and Access Servers on the restarted computer.

You can change service startup to **Manual** in the Windows Services window.

Logging in to the Administrator

After an Access Server is running, you can configure it using the Data Integrator Administrator. From the **Start** menu, select **Business Objects** > **Data Integrator** > **Web Administrator**.

Log in to Data Integrator Administrator using the default name (`admin`) and password (`admin`). See the *Data Integrator Administrator Guide* for more information.

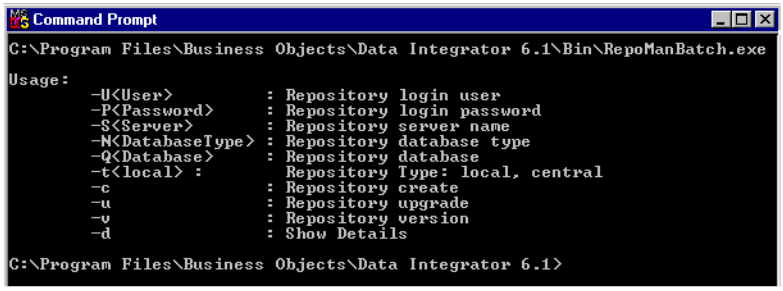
Configuring repositories after installation

To check the version, to upgrade, or to create a repository after installation:

1. From the **Start** > **Programs** > **Data Integrator** menu, choose **Repository Manager**.
2. To check the version of an existing repository, select **Get version**.
3. To create or upgrade repositories, follow the procedure in "To create or upgrade repositories" on page 89.

Creating or upgrading repositories in batch mode

You can use the executable called `RepoManBatch.exe` (stored in the Data Integrator installation/bin directory) to create or update repositories in batch mode. Specify parameters using a command prompt.



For example:

```
RepoManBatch -Usa -P -NMicroSoft_SQL_Server
-SServer -QJake -c -tcentral -d
```

or

```
RepoManBatch -UJake -PJake -NOracle -Sdbsvr -v
```

Usage:

Flag	Description
-U	Repository login user
-P	Repository login password
-S	Repository server name: • Microsoft SQL Server: database server name • Oracle: database connection name • DB2: data source • Sybase: server • Informix: data source
-N	Repository database type: • Microsoft_SQL_Server • DB2 • Oracle • Informix • Sybase
-Q	Repository database • Microsoft_SQL_Server • Sybase
-t	Repository Type: local, central
-c	Repository create
-u	Repository upgrade
-v	Repository version
-d	Show Details

Configuring Job Servers or Access Servers after installation

To create, edit, or delete Job Servers and Access Servers after installation:

1. From the **Start** > **Program** > **Data Integrator** menu, choose **Server Manager**.

   The Data Integrator Server Manager Utility window opens. This window shows the Job Servers and Access Servers currently configured to run on your computer.

2. Follow the procedures in "To configure Job Servers" on page 90 and/or "To configure Access Servers" on page 94.

3. Follow the procedure in "To restart servers" on page 96.

Recovering from installation errors

Errors that prevent installation progress (for example an incorrect server name, user name, or password) result in an immediate error message describing the problem.

Repository problems

Common installation errors include:

- The DBMS connection was lost while building the repository tables. If this happens, run the installation program again.
- There is not enough space available in the database for the repository tables. If this happens, use your DBMS administration tools to allocate more space for the repository and run the installation program again.

If you experience problems when upgrading or creating a repository, you can select the **Show Details** check box before you click **Update** or **Create** again. This option allows you to view the SQL code that Data Integrator uses to perform these operations. The **Show Details** option can help you diagnose problems.

Administrator problems

If you cannot log in to the Administrator using the default user name and password, change the Web Server startup properties to a user account instead of a local system account or verify that the local system account has sufficient privileges to run executable files. Then stop and restart the Web Server's service and try again.

➢ To locate service startup properties for the Web Server

On Windows NT, go to **Start** > **Settings** > **Control Panel** > **Services** and double-click **Data Integrator Web Server**.

On Windows 2000 or XP, go to **Start** > **Settings** > **Administrative Tools** > **Services**, double-click **Data Integrator Web Server**, and select the **Log On** tab.

Verifying connectivity

This section describes specific steps that you can follow to test Data Integrator's real-time features after installation. These step-by-step procedures use sample files available in the Data Integrator installation.

Distributing the test files

The Data Integrator installation and the installation CD include test files to exercise connectivity between each component of the Data Integrator system. To perform the connectivity test, place test files in the appropriate locations on the various computers used in your application.

In this list, the Data Integrator directory represents the location where you installed Data Integrator.

Test file	Copy from	Copy to
ClientTest.exe	Data Integrator\Bin	Web client computer, C:\temp
mny2412d.dll	Data Integrator\Bin	Web client computer, C:\temp
ClientTest.txt	Data Integrator\ConnectivityTest	Web client computer, C:\temp
TestIn.xml	Data Integrator\ConnectivityTest	Web client computer, C:\temp
TestIn.xml	Data Integrator\ConnectivityTest	Designer computer, C:\temp
TestIn.dtd	Data Integrator\ConnectivityTest	Designer computer, C:\temp
TestOut.dtd	Data Integrator\ConnectivityTest	Designer computer, C:\temp

NOTE: You must keep `mny2412d.dll` with `ClientTest.exe`. To move `ClientTest.exe`, re-run the Message Client Setup and point it to the directory you want. For more information, see "Installing Message Client libraries" on page 115.

You need not move the `TestConnectivity.atl` file from its location on the Designer computer.

Testing a job

When you execute a real-time job from the Designer, it always executes in "test mode" using a file as input and producing a file as output.

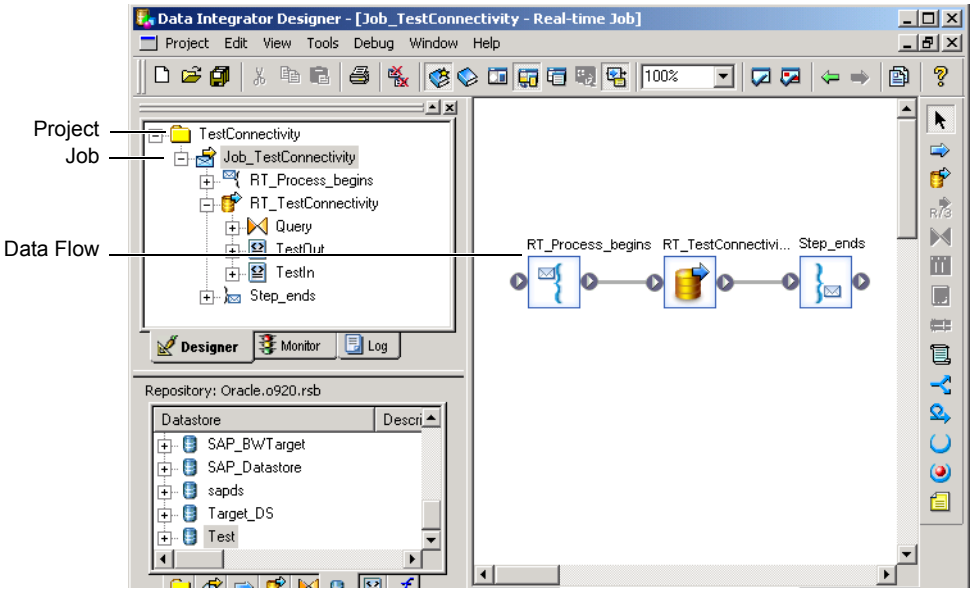
The test files include a sample job and data flow that you can use to verify that a real-time job can successfully execute from the Designer. Test files include the XML test input for the flow (the string `Hello World`) and the corresponding DTD for flow input and output.

This procedure describes how to import, display, execute, and show output for the test flow.

➢ To test a job

1. Start Data Integrator and log into your repository.

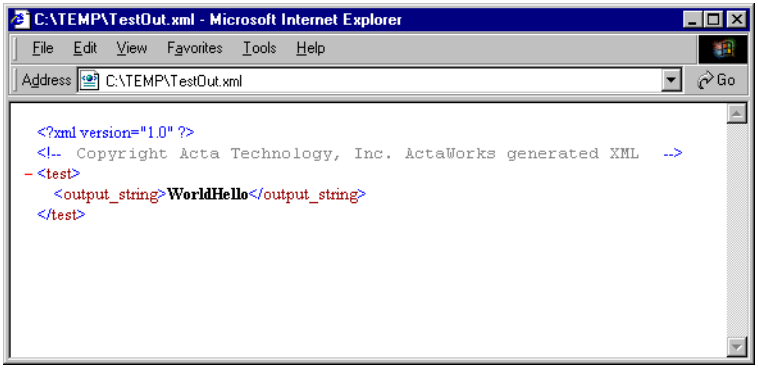
   From the **Start** menu, choose **Data Integrator Designer** from the program group created by the Data Integrator installation. The Designer prompts you for login information associated with your repository database.
2. From the **Tools** menu, select **Import from File**.
3. Navigate in the Data Integrator install directory to \ConnectivityTest.
4. Select the test ATL file `TestConnectivity.atl` and click **Open**.
5. In the **Project menu**, select **New** > **Project**.
6. Name the project `TestConnectivity` and click **Create**.
7. In the **Jobs** tab of the object library, expand the **Real-time Jobs** category; then click, drag, and drop `Job_TestConnectivity` over the TestConnectivity folder in the **Project** area.
8. In the workspace, click the name of the data flow.



9. In the project area, right-click Job_TestConnectivity and select **Execute**.
10. In the Execution Properties window, click **OK**.

    Data Integrator reads the sample file `C:\Temp\TestIn.XML`, inverts the order of the two words in the string, and writes the output to the `C:\Temp\TestOut.XML` file.
11. Read the information in the Job Log window to verify if the `TestOut.XML` file is correct. Use the job log information to troubleshoot problems.
12. If the job was successful, navigate to the C:\Temp\ directory on the Designer's computer and open `TestOut.xml`.

You can display the file in a browser or text editor.



Testing the path from client to service

When your real-time job runs in a normal production environment, you can use it to process a service request from a Web client. In production, Web clients send messages and real-time services receive and process those messages, triggering real-time jobs. Use the Administrator to set up real-time services.

➢ To process a service request from a Web client

1. Add and start the service in the Administrator.

   The Access Server starts service providers which are instances of jobs associated with the listed services.

2. Send a request from your Web client to the Access Server.

   The Access Server sends a request to the appropriate service provider, then the service provider sends a response to the Access Server. The Access Server returns the response to the Web client.

The following procedures briefly describe how to connect a repository to the Administrator, start an Access Server, configure interfaces among components, and use the provided test Web client to verify that your configuration works.

For more detailed information about working with the Administrator, see the *Data Integrator Administrator Guide*.

Configuring the Access Server

The Administrator allows you to view the status of services controlled by the Access Server and to change their configuration.

After you install the Access Server, the Data Integrator service automatically launches the Access Server when your computer restarts or when you stop and start the service.

➢ To start the Administrator

1. Go to **Start > Programs > Business Objects Data Integrator 6.1 > Data Integrator Web Administrator**.

   The Administrator Web browser opens.

   If it does not appear:

   - Check that the port number is not being used by another application.
   - Use a static IP address instead of the host name to locate the Administrator computer.

2. Log in to the Administrator using the default name (admin) and password (admin).

➢ To add your repository for Administrator access

1. In the Administrator, select **Management > Repositories > Add**.
2. Enter your repository connection information and click **Apply**.

➢ To configure the Access Server to listen for responses from services

1. In the Administrator, add a connection to an installed Access Server by selecting **Management** > **Access Servers** > **Add**.
2. Enter your Access Server's machine name and communication port, select **Ping** to test the connection, then click **Apply**.

➢ To add a service

1. In the Administrator's navigation tree, select **Real-time** > **"Access Server Machine Name: Port"**> **Real-time Services** > **Configuration**.
2. Click **Add**.

   The Access Server adds a service to the list (`NewService1`) and displays the configuration boxes for the new service.
3. Enter the following sample information in the **Configuration** section for the service.

   Accept the default values for the service parameters.

Field	Test Values	Description
Service Name	`TestConnectivity`	The message type included in the call from the client.
Job Name	`Job_TestConnectivity`	Browse jobs.

4. Click **Apply**.

   The Administrator prompts you to assign a Job Server to the Service.
5. Under Job Servers for Service, click **Add**.
6. If you have one Job Server, the Administrator automatically selects it for you. Click **Apply**. Otherwise, select a Job Server from the list and click **Apply**.
7. Click the **Status** tab.

The Administrator adds the new service (named after the job) and a service provider to the list for this service. In the status row for each service provider, find the computer name and Job Server port where the service provider is running. The process ID for the service provider is related to this invocation of the Job Server.

8. Verify host name and port number for the new service provider.

   In the list of service providers, verify that the host name and port for the new service provider correctly indicate the computer where the Job Server is installed and the port that the Job Server is configured to use.

   You specified the Job Server port number when you configured the Job Server at installation.

9. In the navigation tree, click **Real-Time Services** to return to the first Real-Time Service Status page.

   The name of the Service should be next to a green icon which indicates that the Service started successfully.

   When a service starts, the Administrator triggers the Access Server which triggers the Job Server to get job information from the repository. The job registers as a service with the Access Server. The Administrator displays the service status (`started`).

   If the service did not start, you can start it manually from the current page. If the service still will not start, See "Resolving connectivity problems" on page 159 of the *Data Integrator Administrator Guide* for help determining the possible cause of the failure.

Web client to Access Server

A Web client opens a connection to the Access Server using a call in the Message Client library. The call includes the host and port information required to make the connection.

The `ClientTest` executable file provided in the Data Integrator installation incorporates the library calls so you can easily test the connection between the Web client computer and the Access Server.

➢ To process a test request

1. On the computer where your Web client application is installed, send a request for Data Integrator to process.

   Copy the command, `C:\Temp\ClientTest.txt` and enter it in a command prompt:

   ```
   ClientTest -AHost -pPort -UUser -PPassword
      -STestConnectivity -XC:\Temp\TestIn.XML
   ```

   where

   Host — Identifies the Access Server host computer by name or IP address.

   Port — Identifies the port on which Access Server listens for Web client requests.

   user — Indicates the user name you specified in the Security section of the Access Server configuration file. The test configuration does not specify a user, but the Client Test utility expects some value. Enter any character.

   password — Indicates the password you specified in the Security section of the Access Server configuration file. The test configuration does not specify a password, but the Client Test utility expects some value. Enter any character.

   TestConnectivity — Specifies the name of the service that you want to invoke.

`C:\Temp\TestIn.XML`
: Indicates the location of the sample XML request that will be sent to the Access Server for processing.

2. Look for the response in the command prompt.

   If the correct response displays, your system is configured properly. If an error occurs, see "Monitoring clients" on page 82 of the *Data Integrator Administrator Guide* for assistance debugging the problem.

3. When you are ready to configure the Access Server to run your own services, first stop the Access Server, then restart it.

Further connectivity tests

There are more tests you can run to define that all the connections you will need. For example, you can use the existing Access Server configuration and make changes to the data flow and input files in the sample real-time job. In particular, consider making simple additions to the data flow in the real-time job to check the following connections:

- From job to Rapid Mart

  To test this connection, add a Rapid Mart table as a source in a data flow in the real-time job and extract a single value to return as a response to the client.

- From job to ERP system

  To test this connection, add a source from the ERP system in a data flow of the real-time job and extract a single value to return as a response to the client.

Updating licenses

You can manage Data Integrator licenses on Windows platforms using local licenses.

➢ To update a license under local license management

You might need to update a license or upgrade a license from evaluation to permanent.

1. Replace the old license file (in the Data Integrator install directory's `\license` subdirectory) with the new license file.
2. If you are adding additional Data Integrator components, reinstall Data Integrator.
3. If you are not adding additional Data Integrator components, simply stop and restart the Data Integrator service.

Installing Message Client libraries

The Message Client libraries support C++, Java, and COM connections between the Data Integrator Web Server and the Access Server.

If you are updating your Message Client library, you must stop your Web application, install the Message Client library files, then restart your Web application computer.

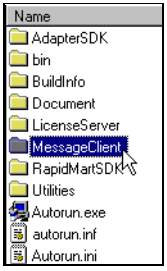
NOTE: The Java client library is a single library (rather than `.class` files); find it in *LINK_DIR*\lib\acta_broker_client.jar.

➢ To install a Message Client library

1. On the computer running the Web application, insert the Data Integrator installation CD.

   The main Data Integrator installation program starts automatically. When it does, click **Exit**.

2. From the Message Client sub-directory on the CD, double-click Setup.exe.



3. Select your interface type.

   The Message Client library supports Java, C++, and COM interfaces.

4. In the Message Client Setup window, use **Browse** to select a location for the Business Objects Client library.

   If you are installing the Message Client library on a computer with other Data Integrator components, choose the same directory. For example, C:\Data Integrator.

   If you are installing the Message Client library on a computer without Data Integrator, choose any installation location.

5. Click **Next** to complete the installation.

Using the Message Client library

The Access Server connector supports C++, Java, and COM interfaces to describe the connection between your Web application and Access Server.

This section describes the Message Client components in general terms, gives specific details according to the interface language, and provides examples of how to include these calls in a typical Web application.

Interface components

The interface between the Access Server and your Web application includes these components:

- Connection definition (Connection) — A class that defines the connection that your application uses to send and receive messages from the Access Server. Initialize the class (using the connect method) each time you initialize your application.
- Connection initialization (Connect) — A method that creates the connection using host and port information supplied by the client.
- Request (Invoke) — A method that indicates request type and details. This method can produce a synchronous or asynchronous return.
- Exception handlers (DIError) — A class that returns exceptions thrown by the connection object and system exceptions, if available.

Creating the connection

The Connection object creates an active connection to the Access Server. This connection supports concurrent use including synchronous and asynchronous calls.

Creating a Connection (calling the Connect method) does the following:

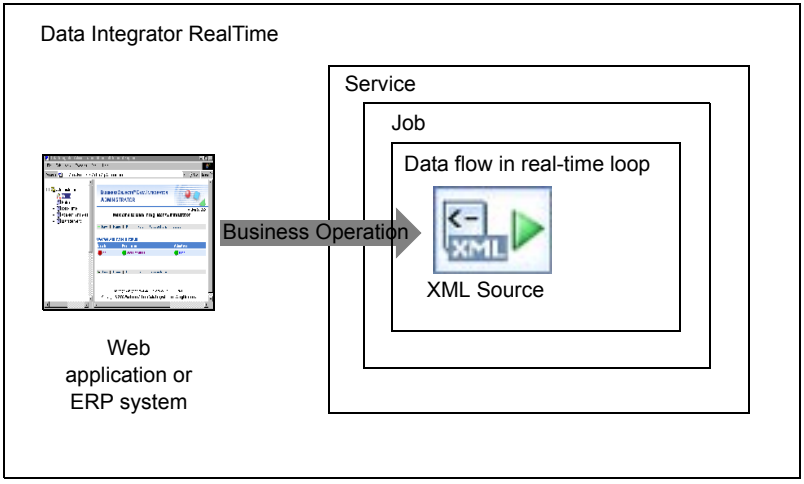
- Authenticates the client as secure
- Produces an open TCP/IP socket between the client and the Access Server
- Encapsulates the connection information into a client identifier (Connection ID)

As soon as you create the Connection, you can use it to send messages to the Access Server. Typically, you would create a single Connection per client. If you attempt to call the Connect method for a Connection that already exists, the Access Server ignores the call.

Sending messages

Send requests from the client application using the Invoke method and the Connection ID.

Each business operation implemented by your Web application can result in a call to the Access Server with a message. The Access Server uses the name of the business operation to determine the path for the message. When you use Data Integrator to process real-time jobs, you pair this business operation name, called a service, with the job and data flow names you defined in Data Integrator to process the message. There is a one-to-one correlation between business operation, service, job, and XML source.



You can choose to send the messages and wait for the response (synchronously). Call the Invoke method with a string return value to process a synchronous response.

Based on your application design and the expected response time, you can also send the message to be processed asynchronously. Call the Invoke method with an integer return value and the name of a ResponseListener object to process an asynchronous response.

Closing the connection

The library provides a method (Disconnect) with the Connection object that allows you to systematically close the TCP/IP socket between the client and the Access Server.

Pseudo code example

```
// Login and authenticate the client
connection = connect(
   accessServerAddress,// TCP/HTTP address
   clientName,// matches Access Server
   clientPassword);// IP & Client
           // security settings

// Invoke Service
String xmlOut = connection.invoke(
   serviceName,// has mapping to RT job
   xmlIn); // according to the RT job DTD

// In case of an error returns the error code
// and error message
```

C++ language interface

The Business Objects Client interface implemented in C++ includes the following classes and elements:

Class	Element	Description
Connection class		Opens a communications link to the Access Server identified by host name and port number.
	Connect method	Creates a new Connection object that can be used to access the connection.
	Invoke method	Sends a message to Access Server for synchronous or asynchronous processing.
	Disconnect method	Closes the connection to the server and removes the Connection object.
ConnectionException class		Constructs a new connection exception when the Connection needs to throw an exception and include a message about another exception that interfered with the Connection's normal operation.
	ConnectionException constructor	Constructs a new connection exception when the Connection needs to throw an exception and include a message about another exception that interfered with the Connection's normal operation.
	getRootCause method	Returns the throwable system exception that makes this Connection exception necessary.

Java language interface

The Business Objects Client interface implemented in Java includes the following classes and elements:

Class	Element	Description
Connection class		Opens a communications link to the Access Server identified by host name and port number.
	Connect method	Creates a new Connection object that can be used to access the connection.
	Invoke method	Sends a message to Access Server for synchronous or asynchronous processing.
	Disconnect method	Closes the connection to the server and removes the Connection object.
ConnectionException class		Constructs a new connection exception when the Connection needs to throw an exception and include a message about another exception that interfered with the Connection's normal operation.
	ConnectionException constructor	Constructs a new connection exception when the Connection needs to throw an exception and include a message about another exception that interfered with the Connection's normal operation.
	getRootCause method	Returns the throwable system exception that makes this Connection exception necessary.

COM interface

The connection between a Web application and the Access Server can be accomplished through a COM object.

The object supports the following Message Client interface elements:

- Connect — Opens a communications link to the Access Server identified by host name and port number.
- Invoke — Sends a message to Access Server for synchronous or asynchronous processing.
- Disconnect — Closes the connection to the Access Server and removes the Connection object.
- ReturnValue — Retrieves the returned (XML) response from the Access Server.
- ErrorMessage — Retrieves the last error message generated by a failure during execution of a Connect or Invoke method.
- ErrorStatus — Retrieves the error status generated by a failure during execution of a Connect or Invoke method.

The object also supports the ErrorInfo interface making it possible to retrieve errors through the Visual Basic Err object.

The remainder of this section describes the syntax for each element.

Connect

Opens a communications link to the Access Server identified by host name and port number.

SYNTAX

```
Connect (host_name, port, user_name, password)
```

WHERE

host_name	An input parameter of type string that indicates the host name of the server on which the Access Server is running.
port	An input parameter of type integer that indicates the number of the port on which Access Server accepts client connections.
user_name	An input parameter of type string that indicates a user login. This is an added level of security that is optional to implement.
password	An input parameter of type string that indicates the encrypted password of the user specified in *user_name*.

After instantiating the Message Client object, use this method to connect to the Access Server.

If an error is encountered while connecting to the Access Server, use the ErrorStatus and ErrorMessage methods to retrieve the error status from the object properties.

EXAMPLE

See "Example" on page 130.

Invoke

Sends a message to Access Server for synchronous or asynchronous processing.

SYNTAX

```
Invoke (ServiceName, InData)
```

WHERE

ServiceName	An input parameter of type string that indicates the name of the service to be executed. This is the same service name specified in the Administrator.
InData	Input parameters of type string that contain the XML string representing the input parameters to the service (the message being processed by the Access Server and corresponding real-time service).

This method submits a request to the Access Server. After a successful call to the `Invoke` method, use the `ReturnValue` call to retrieve the response returned by the Access Server.

If you encounter an error while submitting a request to the Access Server, you can use the `ErrorStatus` and `ErrorMessage` methods to retrieve the error information from the object properties.

EXAMPLE

See "Example" on page 130.

Disconnect

Closes the connection to the Access Server and removes the Connection object.

SYNTAX

```
Disconnect ()
```

ReturnValue

Retrieves the returned (XML) response from the Access Server.

SYNTAX

```
ReturnValue ()
```

RETURNS

`String` XML response from the Access Server.

Use this method to retrieve the response to the request sent by a successful call of the invoke method.

EXAMPLE

See "Example" on page 130.

ErrorMessage

Retrieves the last error message generated by a failure during execution of a Connect or Invoke method.

SYNTAX

```
ErrorMessage ()
```

RETURNS

`String`	The error text returned by the Access Server if an error occurs while executing a COM method.

ErrorStatus

Retrieves the error status generated by a failure during execution of a Connect or Invoke method.

SYNTAX

```
ErrorStatus ()
```

RETURNS

`String`	The error status returned by the Access Server if an error occurs while executing a COM method.

Example

The following example shows how you can use the COM object from within an ASP program using VBScript. The example uses the VB Err object to detect and handle errors.

```
' Instantiate the Message Client COM object
dim ActaConn
set ActaConn = Server.CreateObject("ActaClient.ActaConnection")
' Clear any previous errors and connect to Access Server.
' The parameters are machine, port, user, password
Err.clear
ActaConn.connect "electron", 4000, "user", "password"
if Err.Number <> 0 then
   Response.write("<h4> The connect method returned the following
error: " &Err.Description&"</h4>")
else
   Response.write("<h4> The connect method succeeded.</h4>")
   ' Clear errors and invoke the service needed, pass
   ' service name, XML input data as parameters
Err.clear
ActaConn.invoke "test1", file_content
if Err.Number <> 0 then
   Response.write("<h4> the invoke method returned the following
error:
   " &Err.Description&"</h4>")
else
   Response.write("<h4> The invoke method succeeded</h4>")
       ' Get the return value
       dim outData
       outData = ActaConn.returnValue
       Response.write("<pre>"&outData&"</pre>")
   end if
end if
```

7 Installing Data Integrator on UNIX Systems

All Data Integrator components except the Designer and Repository Manager can run on supported UNIX systems. This chapter discusses the installation of Data Integrator components on UNIX systems and contains the following sections:

- Additional system requirements for UNIX
- Kernel parameters and user resource limits
- Installing Job Servers and Access Servers
- Updating licenses
- Uninstalling Data Integrator
- Troubleshooting

Additional system requirements for UNIX

These requirements supplement the generic ones in "System requirements" on page 46.

Hardware requirements

UNIX computers running Data Integrator Job Server and/or Access Server require:

- A minimum of one processor available to Data Integrator with 512 MB RAM. For best performance, use dual processors (minimum 500 Mhz) with at least 512 MB RAM each. Allow more than 512 MB RAM if you will cache tables or perform memory-intensive operations such as hierarchy flattening, sorts, or lookups.
- 300 MB of disk space.
- Minimum of 1 GB virtual memory (2 GB recommended for best performance)

Software requirements

- Operating systems: see "Data Integrator operating system platforms" on page 31.
- Install database connectivity software on Job Server computers for the database serving as your repository, source, or target. See "Database requirements" on page 47. Microsoft SQL Server is not supported on UNIX.

- Install JDK (1.3 or higher for AIX; 1.3.1 or higher for Solaris and HP-UX), as described in the vendor 's documentation. To support this, AIX must be at the level of AIX 4.3.3.75 (or higher) with the AIX 4330-10 (or higher) Recommended Maintenance Package. In addition, the following filesets must be installed:

Fileset	Level	State	Description
xlC.adt.include	5.0.2.0	COMMITTED	C Set ++ Application Development Toolkit
xlC.aix43.rte	5.0.2.0	COMMITTED	C Set ++ Run-time for AIX 4.3
xlC.cpp	4.3.0.1	COMMITTED	C for AIX Preprocessor
xlC.msg.en_US.rte	5.0.2.0	COMMITTED	C Set ++ Run-time Messages — U.S. English
xlC.rte	5.0.2.0	COMMITTED	C Set ++ Run-time

To find these `xlC` filesets and their levels on your AIX system, use the command `lslpp -l xlC*`

- The following filesets must be installed on AIX. They support the Data Integrator Web Server.

Filesets	AIX 4.3.3	AIX 5.1	AIX 5.2
OpenGL.OpenGL_X.dev.vfb	4.3.3.75	5.1.0.10	5.2.0.0
X11.vfb	4.3.3.50	5.1.0.15	5.2.0.0

- If you are installing the Administrator on HPUX, add the following patches:
  - PHSS 25277 (11.0) PHSS_26637 (for HPUX 11.0)
  - PHSS_26577(11.11) PHSS_26638 (for HPUX 11.11)

  Refer to the $LINK_DIR/log/DITC.log file for browser related errors.

Cron service

Data Integrator schedules UNIX-based jobs by using the UNIX `cron` utility. If `cron` security uses `cron.deny`, the account that starts the Job Server must have an entry in the `cron.allow` file. These files are located as follows:

Operating system	cron.deny location	cron.allow location
HP-UX	/var/adm/cron/cron.deny	/var/adm/cron/cron.allow
AIX	/var/adm/cron/cron.deny	/var/adm/cron/cron.allow
Solaris	/usr/lib/cron/cron.deny	/usr/lib/cron/cron.allow

See your `cron` man pages for more information.

Installing the Data Integrator Job Server without access to `cron` causes the following Job Server behavior:

- Job Server can operate normally
- Data Integrator cannot schedule a job to run on the Job Server
- You can manually execute and monitor jobs from the Designer

User IDs and permissions

If your design, test, or production environment uses tightly controlled root-level administration, Business Objects recommends that you install the Job Server while logged on as a user *without* root-level access. Installing the Job Server without root-level access ensures that root-level access is not required to administer files and processes created by the Job Server.

There are two administration functions that require root-level access as described in the following table:

Functions	Permissions required
Scheduling jobs through Data Integrator	Permission to access `cron`
Automatically restarting the Job Service on restart	Autostart configuration edits need to be made by a user with system-level authority

High-availability support

If you will run Data Integrator on AIX and you have High Availability Clustered Multi-Processing (HACMP) software, version 4.4.0., use HACMP software to minimize downtime of your Data Integrator system.Data Integrator includes start and stop scripts that support HACMP software.

About HACMP software

HACMP software makes applications and software systems "highly available," restoring essential services when a system component or application fails. To support HACMP software, you replicate software and hardware systems to guarantee backup of essential services. Within HACMP software, you define each complete system as a resource group. When a component fails in a resource group, HACMP software quickly restores essential services by starting the failed system on an alternate computer, called a *node*, or group of networked computers, called a *cluster.*

To use HACMP software to make a system "highly available":

- Define the system as a resource group (a logical group that the HACMP software and AIX operating system manages)
- Install the necessary components to support the system on duplicate hardware systems, creating clusters or nodes that can support identical services that the system processes

- Define takeover relationships that determine which cluster or node supports the resource group at any given time

  There are three types of takeover relationships:

  - **Cascading** — Clusters are listed along with a priority rank. Control of the resource group goes to the active cluster with the highest priority ranking. Control can change due to failure or to the reactivation of a cluster with a higher priority rank.
  - **Rotating** — Clusters are listed along with a priority rank. Control of the resource group goes to the active cluster with the highest priority ranking. Control only changes due to the failure of a cluster.
  - **Concurrent** — Multiple clusters support the resource group at the same time.

Using Data Integrator with HACMP software

Data Integrator contains start and stop scripts that support HACMP software. These scripts run when a new cluster takes control of the Data Integrator resource group. The start script:

- Identifies jobs scheduled before the previous cluster went down and launches those jobs in recovery mode
- Identifies jobs scheduled to start during cluster down time and launches those jobs
- Synchronizes the `cron` file on the new cluster with the `cron` file on the previous cluster

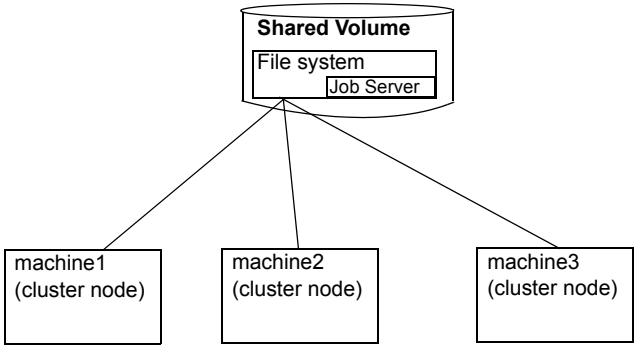
There are several requirements and restrictions for using Data Integrator with HACMP software:

- There can be no single point of failure
- You must use the same Data Integrator user ID on all clusters where you install Data Integrator

- The database systems that support Data Integrator jobs—the repository, sources, and targets—must also be "highly available"
- You cannot use a concurrent takeover relationship for the Data Integrator resource group
- Cannot be used with LiveLoad
- Use only with batch jobs
- You must enable automatic recovery for all scheduled jobs
- You must also enable the recurrent option for all scheduled jobs

➢ To use Data Integrator with HACMP software

1. Install the Data Integrator Job Server on a file system in the shared volume group from a single cluster node.
2. On each cluster node, define the same mount point for the file system. Data Integrator and its configuration is thus made available for the rest of the cluster nodes.



3. Copy the Data Integrator start and stop scripts (`acta_start.sh` and `acta_stop.sh`) from the LINK_DIR/hacmp directory to the HACMP scripts directory on each of the cluster nodes.
4. In the HACMP software:

a. Define Data Integrator as a resource group (a logical group that the HACMP software and AIX operating system manages).

b. Define takeover relationships that determine which cluster or node supports the Data Integrator resource group at any given time.

c. Designate the Data Integrator start and stop scripts as the start and stop scripts for the Data Integrator resource group.

5. Configure Data Integrator to use HACMP as described in the next procedure.

➢ To configure Data Integrator to use HACMP

1. Run the Server Manager by entering:

```
$ cd $LINK_DIR/bin/
$ . ./al_env.sh
$ ./svrcfg
```

2. Select option 1 and stop the Job Service.

3. Select option 6 to configure HACMP.

```
** Data Integrator Server Manager Utility **
     1 : Control Job Service
     2 : Configure Job Server
     3 : Configure Access Server
     4 : Configure Web Server
     5 : Configure SNMP Agent
     6 : Configure HACMP
     x : Exit

     Enter Option: 6
```

4. Configure HACMP.

```
_____________________________________________________________________

*Current HACMP Configuration for Data Integrator Resource Group*
_____________________________________________________________________

      Service IP Label                    Domain Name

     ------------------                   -----------

     aixserver1


_____________________________________________________________________

  e: Edit Configuration    y: Re-sync all repos       q: Quit


Enter Option:
```

- To specify the Service IP Label used while configuring Data Integrator's resource group, enter `e`.

  When you are done, Data Integrator will resynchronize all repositories and the configurations for the Web Server and SNMP agent with the current Service IP Label and domain name. This means that:

  - The Job Service must be stopped prior to editing this configuration.
  - You must re-add repository connections to the Administrator, reconfigure real-time services and adapters, and reschedule batch jobs.

- To resynchronize all repositories manually, enter `y`.

NOTE: Resynchronizing all repositories is not required if you do not see an error message after editing by selecting option `e`. For example, a repository might not get updated to use the configured Service IP Label if the associated database is unavailable. After correcting the problem, you can use this option to resynchronize the repository.

5. To exit the Server Manager, enter `q`, then enter `x`.

Kernel parameters and user resource limits

Business Objects recommends the following kernel parameters and user resource limits when installing Data Integrator on UNIX platforms.

Hewlet Packard kernel configuration

For installations on HP systems, Business Objects recommends the following kernel configuration:

Kernel Parameter	Value	Comments
max_thread_proc	256	
maxdsiz	0X80000000	At least 2GB.
maxdsiz_64bit	0X800000000	Greater than or equal to maxdsiz.
maxfiles	1024	Anything between 1024 - 2048.
maxssiz	0X05000000	You can increase this value but that automatically decrements the maxdsize. Therefore, increase maxdsiz.
maxssiz_64bit	0X05000000	Same as maxssiz.
maxtsiz	0X05000000	You can increase this value but that automatically decrements the maxdsize. Therefore, increase maxdsiz.
maxtsiz_64bit	0X05000000	Same as maxtsiz.
maxuprc	512	Anything between 512 - 1024.
maxusers	128	
nfile	(16*(NPROC+16+MAXUSERS)/10+ 32+2* (NPTY+NSTRPTY+NSTRTEL))	Approximately 1024 (or leave the formula and modify nproc).
nproc		Approximately 1024. Increases the nfile limit and other dependent kernel parameters.

AIX user resource limits

For installations on AIX systems, Business Objects recommends the following user resource limits. You can display these settings by running the ulimit -a command from your Data Integrator installation login.

User resource limit	Value	Comments
file (blocks)	4194302	At least 2GB.
data (kbytes)	2097151	At least 2GB.
stack (kbytes)	512000	At least 500 MB.
memory (kbytes)	2097151	At least 2GB.
nofiles (descriptors)	2000	At least 2000.

Solaris user resource limits

For installations on Solaris systems, Business Objects recommends the following user resource limits. You can display these settings by running the ulimit -a command from your Data Integrator login.

User resource limit	Value	Comments
file (blocks)	unlimited	
data (kbytes)	unlimited	
stack (kbytes)	8192	At least 8K.
time (seconds)	unlimited	
nofiles (descriptors)	1024	At least 1K.
coredump (blocks)	unlimited	
vmemory(kbytes)	unlimited	Unlimited, at least 2GB.

Installing Job Servers and Access Servers

Before installing any Job Servers or Access Servers on UNIX:

- Use your RDBMS system to create databases for your repository and data warehouse.
- Install the Data Integrator Designer on Windows. This installation includes a Repository Manager.
- Use the Repository Manager on Windows to create one or more Data Integrator repositories.

➢ To install a Job Server or Access Server on UNIX

1. Log in as root.
2. Create a user ID, `bodi` for example, for managing Data Integrator and related files running on the UNIX server. (Optional) Create a group for this user ID and for other users accessing the Data Integrator Job Server and related files.

   For multiple Data Integrator installations, each installation requires a unique user ID and unique home directory.
3. Configure the user created in step 2 to log in using the Korn shell.
4. Mount the installation CD on the UNIX server.

   For example, these mount commands mount the CD into `/cdrom`:

Operating system	Command line
HP-UX	`$ mount -F cdfs -o cdcase /dev/dsk/c6t2d0 /cdrom`
AIX	`$ mount -v cdrfs -o ro /dev/cd0 /cdrom`
Solaris[a]	`$ mount -F hsfs -o ro /dev/dsk/c0t6d0s0`

a. Use this command when the Volume Manager daemon is not running. Otherwise, Solaris will automount the CD.

5. Exit from root login.
6. Log in as "bodi" to install the Data Integrator Job Server.
7. Set your database environment variables.

Database	Environment Variables
Oracle	$ORACLE_HOME
	$ORACLE_SID
	$ORACLE_LIB
DB2	$DB2INSTANCE
Sybase	$SYBASE
ODBC	$ODBCINI
DETAIL	$DETAIL_LIBPATH

8. Confirm that you can connect to the local Data Integrator repository for Job Server access.

   To verify the connection, use an RDBMS SQL tool such as SQL*Plus.
9. Configure the network on the Designer computer and the Job Server computer so that they can communicate with each other by host names rather than by IP addresses.
10. Navigate to your UNIX directory in the mount point.

    For example: `cd /cdrom/unix`.
11. Run the installation script: `./install.sh`.

12. Enter the information requested by the script.

The script provides values set for this account as default values. To accept the default at a prompt, press **Enter** or **Return**.

Installation prompt	Example response
CD-ROM mount location	/cdrom
1) Install Data Integrator 2) Exit Enter choice number.	1
Do you have JDK 1.3 or higher installed on this machine? [YES|NO]?	YES
Installation location?	/home/BusinessObjects/Data Integrator
Overwrite existing Data Integrator? [Y|N]?	N (If Y, the installer will not prompt about configuration settings. See step 13.)
License file location	/home/BusinessObjects/al_license.lic
Language code	eng (See "Languages" on page 568 of the *Data Integrator Reference Guide*.)
Territory code	US (See "Territories" on page 569 of the *Data Integrator Reference Guide*.)
Code page	iso88591 (See "Code pages" on page 570 of the *Data Integrator Reference Guide*.)
JDK installation directory	/usr/java
Repository database type	2
For an Oracle repository:	
Location of the database executable and libraries as defined by $ORACLE_HOME	/u01/app/oracle/product/8.1.6
For HP-UX, Oracle version	Enter Oracle Version separated by a period (.). For example: 9.0.1.0
Repository connection string	oraDBS
Repository user name	bodi

Installation prompt	Example response
For a DB2 repository:	
DB2 instance	db2inst1
Repository connection string	db2DBS
Repository user name	bodiRepo

For a Sybase repository:	
Sybase home path	/usr/sybase
Repository server name	sybDBS
Repository database name	bodiRepo
Repository user name	sa

Are you planning to use DETAIL to access DB2/Non-relational databases? [Y|N]?	Y

For DETAIL:	
DETAIL shared library location	/usr/detail/libdtlbase.*extension*

Are you planning to access databases using DataDirect's Connect ODBC drivers for UNIX? [Y|N]?	Y

For ODBC:	
DataDirect's Connect ODBC drivers installation directory	/usr/odbc
Full path of odbc.ini file	/usr/odbc/odbc.ini

Do you want to copy the configuration settings from previous version of Data Integrator? [Y|N]?	Y (**NOTE**: This prompt and the next two only appear if you elected not to overwrite the previous installation.)
Enter the installation directory of old version of Data Integrator.	/usr2/joe/AW5212
Enter the version of old Data Integrator.	5.2
Installation Successful!	

13. If you selected "Y" when prompted to overwrite an earlier version of Data Integrator, a message appears indicating that all existing Data Integrator services will be stopped.

    After completing the installation, the "Installation Successful!" message appears.

    The installer creates a system variable (`LINK_DIR`) that defines the path used by the Data Integrator components.

14. Configure a Data Integrator Web Server (see "Setting Data Integrator environment variables" on page 147).

15. Set Data Integrator environment variables (see "Setting Data Integrator environment variables" on page 147).

16. Configure Job Servers and Access Servers (see "Configuring the Data Integrator Web Server" on page 149).

Locales

The installation script prompts you for a locale for each installed Job Server. This consists of a language code, a territory code, and a code page. For detailed information about using Locales and Multi-byte data types, see Chapter 9, "Locales and Multi-Byte Functionality," in the *Data Integrator Reference Guide*.

For a list of possible values, see "List of supported locales and encodings" on page 567 of the *Data Integrator Reference Guide*.

Setting Data Integrator environment variables

Users who run or administer Data Integrator Job Servers must run a script (`al_env.sh`) to set up environment variables required by the Job Server.

Run this script with the syntax required by your environment. For example, run the following:

```
$ cd $LINK_DIR/bin/
$ . ./al_env.sh
```

You can also add this command to your login script ("bodi") so that it is always configured. For example, add the following line to the `.profile`:

```
$LINK_DIR/bin/al_env.sh
```

No error messages appear if the script fails to run. Check one or more of the following variables to make sure that their values are properly set.

Use the `echo` command to verify environment variable settings.

Variable	Setting
$LINK_DIR	Data Integrator installation directory (set by the DATA INTEGRATOR installation program)
$WEBSERVER_HOME	Home directory for the Web Server. Set to `$LINK_DIR/ext/Web Server`
$SHLIB_PATH	For HP-UX. Must include `$LINK_DIR/bin` and the location of the database libraries. For 64-bit Oracle, `$LINK_DIR/bin` must be listed before any 64-bit Oracle shared library path.
$LD_LIBRARY_PATH	For Solaris. Must include `$LINK_DIR/bin` and the location of the database libraries. For 64-bit Oracle, `$LINK_DIR/bin` must be listed before any 64-bit Oracle shared library path.
$LIBPATH	For AIX. Must include `$LINK_DIR/bin` and the location of the database libraries. For 64-bit Oracle, `$LINK_DIR/bin` must be listed before any 64-bit Oracle shared library path.
$ORACLE_SID	Required for an Oracle source, target, or repository.
$ORACLE_HOME	Required for an Oracle source, target, or repository. For a 64-bit Oracle installation, this must point to the 64-bit Oracle installation.

Variable	Setting
$DB2INSTANCE	Required for a DB2 source, target, or repository.
$SYBASE	Required for a Sybase source, target, or repository.
$ODBCINI	Required for an ODBC source or target.
$DETAIL_LIBPATH	Required for a DETAIL source or target.
$PATH	Must include `$LINK_DIR/bin` and *`databasehome`*`/bin`.

If the variable settings are not properly configured and you start any Data Integrator utility, error messages will appear to indicate that database server files are missing.

If you see such an error, verify that `al_env.sh` contains commands to set the appropriate database home locations. Run `al_env.sh` for the account used by the Job Server, or start the Job Server using an account that has all necessary environment variables defined in its `.profile`.

Configuring the Data Integrator Web Server

The Business Objects Data Integrator (BODI) Web Server uses a Tomcat server as a servlet engine. The Data Integrator Web Server runs as a separate UNIX daemon. To configure the Data Integrator Web Server:

1. Run the Server Manager. Enter:

   ```
   $ cd $LINK_DIR/bin/
   $ . ./al_env.sh
   $ ./svrcfg
   ```

2. Select option `4` to configure Web Server.

3. Configure the Web Server.

   The Server Path field displays the Web Server's home directory: `$LINK_DIR/ext/WebServer`
   It cannot be changed.

The HTTP Port field displays the default value 28080. The Shutdown port field displays the default value 22828. Keep or modify these port numbers and press Enter.

4. Once the Web Server is configured, the confirmation screen appears as follows:

```
_____________________________________________________________

** Current Data Integrator Web Server Information **

_____________________________________________________________

 Server Path                                    HTTP     Shutdown
                                                Port     Port
 ---------                                      -----    -------
 /BusinessObj/DI/ext/Web Server                 28080    22828

_____________________________________________________________

  e: Edit Server                    q: Quit

Enter Option:
```

5. Enter `e` to edit to change the HTTP and shutdown ports, or enter `q` to quit.

An additional message displays if the Web Server is running while you attempt to change port numbers:

```
Data Integrator Web Server is running on
port [9992].
Are you sure you want to stop this and
change port (Y|N) [N]?
```

Configuring Job Servers and Access Servers

Data Integrator installation on UNIX includes a Server Manager (`$LINK_DIR/bin/svrcfg`).

Use the Server Manager on UNIX:

- To configure Job Servers
- To configure an Access Server
- To start or stop the Data Integrator Service

The Server Manager displays the following information:

Job Server information

Server name	This name uniquely identifies the Job Server. The Job Server name cannot be changed.
TCP/IP port number	The port number is a TCP/IP port that the Job Server uses to receive commands from the Designer and an Access Server. If a computer hosts multiple Job Servers, each Job Server must have a unique port number. Choose a port number that is not used by another process on the computer. Business Objects recommends 3500. If you are unsure of which port number to use, use the default port number and increment it for each additional Job Server you configure.
Supports adapter and SNMP communication on port	If this computer hosts adapters or the SNMP agent, you must designate one (and only one) Job Server to support them. Once a Job Server is set to support adapters and SNMP communication (a port is entered and saved), it is marked on the Job Server Configuration screen with this label.
Enable SNMP	After one Job Server per computer is set as the one supporting adapters and SNMP, it and additional Job Servers can be enabled for SNMP.

Access Server information

Server number	This sequence number uniquely identifies the Access Server on this machine. The Access Server number cannot be changed.
Directory	This is the directory containing Access Server information.
Communication port	This port number is used to communicate between the Access Server and the Administrator. The default is 4000.
Parameters	Access Server parameters can be viewed by typing `AL_AccessServer` at the command line and are also described under "Configuration parameters" on page 122 of the *Data Integrator Administrator Guide*.
Enable	Enter "`Y`" to activate the Access Server.

Web Server information

Tomcat home directory	The path in which the DATA INTEGRATOR Web Server and its associated files are located.
HTTP port	The TCP/IP port that is used for communication between the Administrator and the Access Servers. Choose an unused port number. The default is 28080.
Shutdown port	The TCP/IP port that the Data Integrator Web Server service uses to shutdown the Web Server, which in turn controls the Administrator. Choose an unused port number. The default is 22828.

Data Integrator Service information

Data Integrator Service executable path	The directory containing AL_JobService information.
Status	Status of the Data Integrator Service: "Running" or "Not running."

➢ To configure Job Servers

1. Run the Server Manager. Enter:

```
d $LINK_DIR/bin/
$ . ./al_env.sh
$ ./svrcfg
```

NOTE: The second command is used to set the environment variables before running the Server Manager.

2. Select option 2 to configure a Job Server.

```
** Data Integrator Server Manager Utility **
     1 : Control Job Service
     2 : Configure Job Server
     3 : Configure Access Server
     4 : Configure Web Server
     5 : Configure SNMP Agent
     6 : Configure HACMP
     x : Exit

     Enter Option: 2
```

3. Configure or delete a Job Server.

```
_____________________________________________________________

** Current Job Server Information **
_____________________________________________________________

   S#      Job Server Name  TCP      Enable     Repository
                            Port     SNMP       Connection
   --      ---------        -----    -------    ---------
   1*      Server1          19111    Y          repo1@orasvr1
   2       Server2          19112    N          repo2@orasvr1

*:JobServer <S1> supports adapter and SNMP communication
on port:19110
_____________________________________________________________

  c: Create a new JOB SERVER entry   a: Add a REPO to job server
  e: Edit a JOB SERVER entry         y: Resync a REPO
  d: Delete a JOB SERVER entry       r: Remove a REPO from job server
  u: Update a REPO                   s: Set default REPO
  q: Quit

Enter Option:
```

- To create a Job Server, enter `c`. Then enter the Job Server name, Job Server port number, database type (number), repository connection string, repository user name, and repository password.

  If you enter a port number already in use, an error message appears.
- To edit a Job Server, enter `e`. Then enter the serial number of the Job Server you want to edit.
- To delete a Job Server, enter `d`. Then enter the serial number of the Job Server you want to delete.

- To add a repository connection to a Job Server, enter `a`. Then enter the repository serial number (as displayed), database type (number), repository connection string, repository user name, and repository password.
- To remove a repository connection from a Job Server, enter `r`. This is used when you have multiple repository connections to a Job Server. Then enter the Job Server serial number and repository serial number (as displayed).
- To set a default repository connection for a Job Server, enter `s`. Then enter the Job Server serial number and the serial number of a new default repository (from those displayed).
- To resynchronize your Job Server configuration with a repository, enter `y`.

  Cases when you must resynchronize your Job Server and repository include:

  - You have uninstalled Data Integrator and are reinstalling the same Data Integrator version without creating a new repository.
  - You created a new repository using the Repository Manager after installing Data Integrator.

  If you resynchronize your Job Server configuration with a repository, you must re-add a connection for this repository to the Administrator. See the *Data Integrator Administrator Guide*.
- Unlike the Windows Server Manager, the UNIX Server Manager does not prompt for the repository password except when creating a Job Server or adding a repository. To update the repository password in the DSConfig.txt file, enter `u`. All options use the updated password from `DSConfig.txt` file.

Data Integrator's SNMP agent is a license-controlled feature. You must purchase this optional interface to view the following SNMP configuration settings. For more information, see "SNMP support" on page 541 of the *Data Integrator Designer Guide*.

- To enable SNMP for a job server, choose the edit option. Then choose `Y` when prompted with the following question:

  ```
  Do you want to Enable SNMP for this
  JobServer 'Y|N' [N]:
  ```

- To disable SNMP for a job server, choose the edit option. Then choose `Y` when prompted with the following question:

  ```
  Do you want to Disable SNMP for this
  JobServer 'Y|N' [N]:
  ```

4. To exit the Server Manager, enter `q`, then enter `x`.

➢ To configure an Access Server

1. Run the Server Manager. Enter:

   ```
   $ cd $LINK_DIR/bin/
   $ . ./al_env.sh
   $ ./svrcfg
   ```

2. Select option `3` to configure an Access Server.

```
** Data Integrator Server Manager Utility **
     1 : Control Job Service
     2 : Configure Job Server
     3 : Configure Access Server
     4 : Configure Web Server
     5 : Configure SNMP Agent
     x : Exit

     Enter Option: 3
```

3. Configure, edit, or delete an Access Server.

```
_____________________________________________________________________

** Current Access Server Information **
_____________________________________________________________________

 S#      Directory         Communication Port     Enable
 --      ---------         -------                ---------
 1       export/AStest1    7897                   Y
 2       export/AStest2    7898                   Y


_____________________________________________________________________

 c: Create a new access server entry   e: Edit an access server entry
 d: Delete an access server entry      q: Quit

Enter Option:
```

- To configure an Access Server, enter `c`. Then enter the directory, port numbers, and parameters and indicate whether you want to enable the Access Server.

  Access Server parameters can be viewed by typing `AL_AccessServer` at the command line and are also described under "Configuration parameters" on page 122 of the *Data Integrator Administrator Guide*.

  You can configure more than one Access Server on the same computer, but they must each have separate ports. You later add them using the Data Integrator Administrator.

  If you enter a port number already in use, an error message appears.

- To edit an Access Server, enter `e`. Then enter the number of the Access Server you want to edit.

- To delete an Access Server, enter `d`. Then enter the number of the Access Server you want to delete.

  NOTE: When you delete an Access Server, all Access Servers are stopped. When you exit the Server Manager, the remaining Access Servers restart.

4. To exit the Server Manager, enter `q`, then enter `x`.

➢ To start or stop the Data Integrator Service

The Data Integrator Service (`AL_JobService`) is a daemon associated with `$LINK_DIR` that starts locally-configured Job Servers, Access Servers, and Data Integrator Web server, then monitors them and attempts to restart them if they are not running.

After exiting the Server Manager, `AL_JobService` automatically retrieves any changes made to Job Servers or Access Servers. You need not restart `AL_JobService`.

1. Run the Server Manager. Enter:

```
$ cd $LINK_DIR/bin/
$ . ./al_env.sh
$ ./svrcfg
```

2. Select option `1` to control the Data Integrator Service (Job Service).

```
** Data Integrator Server Manager Utility **
     1 : Control Job Service
     2 : Configure Job Server
     3 : Configure Access Server
     4 : Configure Web Server
     5 : Configure SNMP Agent
     x : Exit

     Enter Option: 1
```

3. Start or stop the Job Service.
   - To start the Job Service, enter `s`.
   - To stop the Job Service, enter `o`.
4. To exit the Server Manager, enter `q`, then enter `x`.

Starting AL_JobService at restart

To start `AL_JobService` automatically when the server restarts, you must install the `actaservices` script with **root** privileges.

Run the *`$LINK_DIR`*`/bin/autostrt.sh` script by entering:

```
# cd $LINK_DIR/bin/
# autostrt.sh $LINK_DIR
```

Verifying that Job Servers and Access Servers are running

Determine which Job Servers and Access Servers are running on the computer using the command:

```
$ ps -fu UnixLoginId | grep AL_JobService

$ ps -fu UnixLoginId | grep al_jobserver

$ ps -fu UnixLoginId | grep AL_AccessServer
```

`UnixLoginId` is the user name that you log into UNIX with when installing Data Integrator.

If some are not running, check for error messages in these log files:

Component	Log File Locations
AL_JobService	*$LINK_DIR*/log/service_eventlog.txt
al_jobserver	*$LINK_DIR*/log/*JobServerName*/server_eventlog.txt
AL_AccessServer	*AccessServerPath*/*error_mm_dd_yyyy*.log

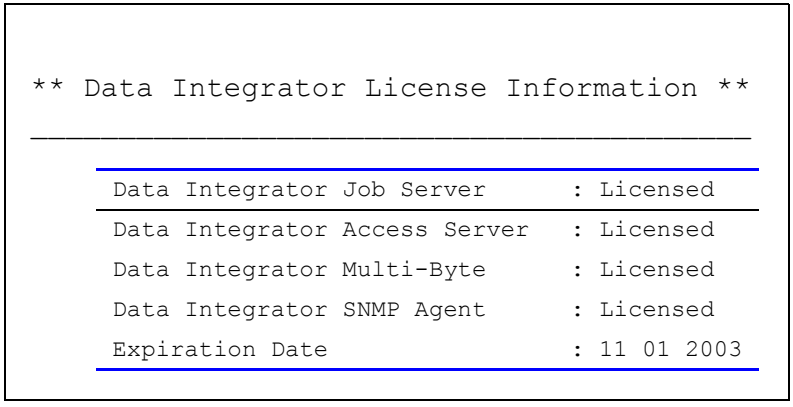
Updating licenses

To update a license (for example, from evaluation to permanent), copy the license file to *$LINK_DIR*/License/al_license.lic.

To verify the licensing information, run the following sequence of commands:

```
$ cd $LINK_DIR/bin
$ . ./al_env.sh
$ LicenseManager
```

This will display an output similar to the following.



Uninstalling Data Integrator

To uninstall Data Integrator:

1. Stop the Data Integrator service. See "To start or stop the Data Integrator Service" on page 158.
2. Remove automatic startup of the Data Integrator Service at restart (if it is enabled) by executing the script as root user:

   ```
   $LINK_DIR/bin/uninstall_root
   ```

3. Remove Data Integrator directories and files recursively in $LINK_DIR by the command:

   ```
   rm -R $LINK_DIR
   ```

Troubleshooting

There are several types of problems that you can troubleshoot:

- Designer-Job Server connection
- Server Manager problems
- Administrator connection
- Memory issues
- Threading issues

Designer-Job Server connection

If the computer on which the Designer is running is unable to connect to the computer on which the Job Server is running, then verify that you can ping, by name, from the Designer's computer to the Job Server's computer. If you cannot ping by name, contact your local network administrator.

Check how long `netstat -a` takes to run and expect at least that amount of time to configure Data Integrator services.

Server Manager problems

If you encounter any of the following situations, reset all Data Integrator processes:

- 5-10 minutes after starting the Server Manager, you still receive messages that the `AL_JobService` is in active mode.
- After starting the Server Manager, the menu does not appear.
- The `AL_JobService` log states that the server configuration is active when the Server Manager is not running.
- Job Servers, Access Servers, or the `AL_JobService` continue to run after you stop the Data Integrator Service.

➢ To reset all Data Integrator processes

1. Obtain the process ID of `AL_JobService`. Enter the command

   ```
   $ ps -fu UnixLoginId
   ```

2. Issue the command:

   ```
   $ kill -15 AL_JobServicePID
   ```

3. Wait at least 30 seconds for the `AL_JobService` and other Data Integrator processes to shut down. This time period might be longer if there are many instances of Job Servers and Access Servers in the installation.

4. To verify that the AL_JobService, al_jobserver, AL_AccessServer, and httpd processes were stopped, enter the command:

   ```
   $ ps -fu UnixLoginId
   ```

5. For any processes that still exist, enter a specific kill command:

   - For each instance of the process al_jobserver, enter the command:

     ```
     $ kill -9 al_jobserverPID
     ```

   - For each instance of the process AL_AccessServer, enter the command:

     ```
     $ kill -9 AL_AccessServerPID
     ```

   - For the process AL_JobService, enter the command:

     ```
     $ kill -9 AL_JobServicePID
     ```

   - For the Web Server, enter the single command:

     ```
     $WEBSERVER_HOME/bin/shutdown.sh -config \
     $WEBSERVER_HOME/conf/server-acta-1.xml
     ```

6. Remove the file `$LINK_DIR/log/AL_JobService.pid`.

7. If the process svrcfg exists, enter the command:

   ```
   $ kill -9 svrcfgPID
   ```

8. Remove semaphores created by Data Integrator if any exist.
   a. To list the Data Integrator semaphores, enter the command:

      ```
      $ ipcs -s|grep UnixLoginId
      ```

   b. For each semaphore listed, note the ID (second column of the ipcs output), and enter the command:

      ```
      $ ipcrm -s id
      ```

Administrator connection

If you cannot connect to the Data Integrator Administrator, stop and restart the AL_JobService process.

Memory issues

As with any computer on which a Job Server is running, reserve enough space for the Data Integrator engine to cache tables and extract all data necessary to perform hierarchy flattening, sorts, or lookups.

Consider changing the following system settings if you see memory-related errors:

- Maximum virtual memory per process

  If this setting is low enough to interfere with Data Integrator operations, you might see messages describing low-memory errors.

- Maximum number of files per process

  If this setting is low enough to interfere with Data Integrator or database server operations, you might see messages describing file open errors.

- Maximum stack size per process

If this setting is low enough to interfere with Data Integrator operations, you might see messages describing segmentation- violation errors.

- Maximum data segment size per process

  If this setting is low enough to interfere with Data Integrator operations, you might see messages describing segmentation- violation errors.

For more information, see the your UNIX system administration documentation.

Threading issues

If you are running the Data Integrator Job Server on the HP-UX platform and you get an error that reads "Cannot Start Thread", check the Kernel Parameter (max_thread_proc) on HP-UX. Set the value to 256. If you encounter additional errors, please contact Business Objects Customer Support.

A Glossary

This glossary defines terms used in this documentation set including common terms that have a special meaning with respect to Data Integrator. Within a definition, other terms that are defined in the glossary appear in **bold type**.

ABAP

Advanced Business Application Programming.
A fourth-generation programming language developed by SAP in which R/3 applications are written.

ABAP program

A program that executes database operations on an **SAP R/3** server. Data Integrator **R/3 data flows** generate ABAP programs.

Access Server

A process that starts, stops, and monitors real-time services (**real-time jobs**). The Access Server dispatches requests to real-time services, ensuring optimal load balancing and complete life cycle management. The Access Server also starts, stops, and monitors **adapters**.

adapter

An external Data Integrator **interface**. There are two types of adapters:

- Custom adapters — Adapters developed using the Data Integrator Adapter Development Kit
- Prepackaged adapters — Adapters prebuilt and purchased from Business Objects, such as the Data Integrator MQSeries Adapter

Administrator

A browser-based system administration application that supports Data Integrator. Use the Administrator to configure and manage:

- Services (**real-time jobs**)
- Service providers (instances of real-time services)
- Clients (Web application clients)
- Batch jobs
- Adapters
- Job logs

after-image

The values in an UPDATE row after the row changes. You might want to pair before- and after-images of UPDATE rows when designing log-based changed-data capture jobs which Data Integrator supports.

aggregate data

Data that results when a process combines elements. This data can be presented collectively or in summary form.

ALE

Application Link Enabling. An SAP R/3 programming-related interface designed to allow reliable communication across a distributed environment.

Analytic Rapid Mart

An **Data Integrator application** that produces a predefined **data warehouse** for a specific area of business analysis. Analytic Rapid Marts are typically based on dimensional modelling. With Analytic Rapid Marts, users can manage large volumes of data.

attribute

A **property** created for a type of **object**.

BAPI

Business Application Programming Interface. A standardized SAP R/3 programming interface that allows non-SAP R/3 applications to access specific business processes and data.

back-office application

A software system that an enterprise uses to run business operations, such as human resources, general ledger, and finance. Back-office applications store data in a relational **database** optimized for operational use.

There are several types of back-office applications:

- Enterprise resource planning (ERP) — For example, SAP R/3, BAAN, J.D. Edwards, PeopleSoft
- Supply chain management (SCM) — For example, i2
- Customer relationship management (CRM) — For example, Siebel, Clarify
- Legacy

Basis

The SAP infrastructure. Basis is the foundation for all SAP products.

batch job

A group of objects (data flows, work flows, conditionals, scripts, and so forth) that executes on a schedule. This type of job is recommended for operations on large amounts of data.

before-image

The values in an UPDATE row before the row changes. You might want to pair before- and after-images of UPDATE rows when designing log-based changed-data capture jobs which Data Integrator supports.

bulk loading

A software-based mechanism that moves large data files. This mechanism supports compression, blocking, and buffering to optimize transfer times.

business component

A set of tables Siebel applications use to create a logical object called a business object.

Business Objects Data Platform

A set of applications from Business Objects, Americas, that enterprises can use to manage all types of data—from data for supply chain partners to data for customers. The Business Objects Data Platform consists of **Data Integrator** and one or more **Rapid Marts**.

CDC datastore

A CDC datastore allows you to limit extracted data to changed data only. A CDC datastore connects a **changed-data capture** table on a source database to Data Integrator.

CDC checkpoint

A CDC checkpoint enables Data Integrator to restrict CDC subscription reads. After you enable a checkpoint, the next time the CDC job runs, it reads only the rows inserted into the CDC table since the last checkpoint.

CDC Subscription

A CDC subscription name allows different data flows to extract data from the same table without corrupting data extracted by other data flows. The datastore and table name can be the same; however, the table owner name must be different in each source CDC table.

changed-data capture

The process of retrieving changes made to a production data source. This process consolidates units of work, ensures data is synchronized with the original source, and reduces data volume in a warehouse environment.

Citrix MetaFrame XP

Citrix MetaFrame XP software provides an access infrastructure for enterprise applications. You can use this software to run Data Integrator on a server which publishes instances of the Designer and other Data Integrator interfaces to users on client computers.

client/server

A distributed technology approach where the processing is divided by function. The server performs shared functions (such as managing communications and providing database services), while the client performs individual user functions.

COM

Component Object Model. Provides a set of interfaces that allow clients and servers to communicate within the same computer (running on Windows 95 or later versions).

Commerce Rapid Mart

An **Data Integrator application** that produces a predefined **data warehouse** and that includes logic to support real-time requests for data from a **back-office application**. A Commerce Rapid Mart is usually based on different data model than an **Analytic Rapid Mart**. With a Commerce Rapid Mart, users can integrate a company's back-office system with an **e-commerce** application.

Communication Structure

In **SAP BW**, a data structure that defines a set of **InfoObject**s available from an **InfoSource** to put into **InfoCube**s.

conditional

A **single-use object**, available in **work flow**s**,** that allows you to branch the execution logic based on the results of an expression. The conditional takes the form of an if/then/else statement.

custom ABAP program

A custom ABAP program runs an **ABAP program** and generates a **data set**. With a custom ABAP program, you can run an existing ABAP program as part of a job. Use a custom ABAP program as a **source** in a **data flow** or an **R/3 data flow**.

data extraction

The process of moving data from a **database** or application source to a database target (either from a legacy database to a **data mart**, or from one data mart to another).

data flow

A **reusable object** containing **steps** to define the transformation of data from **source** to **target**. Data flows are called from inside a **work flow** or **job**. You can pass information into or out of data flows using **parameters**.

Data Integrator

A software system that allows users to build and execute applications with which they can create and maintain **data warehouses**.

Data Integrator consists of several components:

- **Access Server**
- **Data Integrator engine**
- **Data Integrator repository**
- **Data Integrator Service**
- **Administrator**
- **Designer**
- **Job Server**

Data Integrator application

A set of **jobs** that process batch data extractions and real-time requests using data from a variety of data sources.

Data Integrator engine

The core process that reads job information from the **Data Integrator repository** and sets up run-time processes that execute the job. The run-time processes extract, transform, and load relational and hierarchical data. The **Job Server** starts the Data Integrator engine to execute batch or real-time jobs.

Data Integrator interface

A program that Data Integrator uses to access data sources. Specific interfaces vary by installation. There are internal interfaces—those native to the installation—and external interfaces—those that you install separately. Internal interfaces allow Data Integrator to access applications like **SAP R/3** and **SAP BW**, **messages**, relational database systems, and **legacy systems**. An external interface is also known as an **adapter**. It allows Data Integrator to access applications using information exchange technologies such as Web Services and MQ Series.

Data Integrator repository

The database that contains information about an Data Integrator application. The repository contains information about defined **reusable objects**, the **metadata** for tables, and built-in **transforms** and **functions**. When you invoke Data Integrator, you log in to the repository containing the **objects** you want to use.

Data Integrator Service

The process that ensures that the **Access Server** and the **Job Server** are running. You can configure the Data Integrator Service to restart the Access Server and Job Server whenever the computer where they are located restarts.

data loading

The process of populating a **data warehouse**. Data loading is provided by **DBMS**-specific load processes, DBMS insert processes, and independent fast-load processes.

data mapping

The process of assigning a source data element to a target data element.

data mart

A highly-focused version of a **data warehouse**. Typically, created by a department or division of a company, data marts contain data for a specific subject area, such as finance or sales. Data Integrator can populate a data mart.

data transformation

The process of filtering, merging, decoding, and translating source data to create validated data for a **data warehouse**.

data set

Rows of data with a defined schema. A step in a **data flow**—such as reading data from a **source**, joining data in a **Query transform**, or transforming data though another **transform**—yields a data set. You can view individual data sets by placing a **target** table or file at that point in the **data flow**.

data transport

A step in an **R/3 data flow** that defines a **target** to store the **data set** extracted during the flow. You can locate the target file on the **SAP R/3** server or in a location accessible to both the SAP R/3 server and to Data Integrator across a network.

data type

The format used to store a value. Data types can imply a default format for displaying and entering the value. Data read from a **source** is converted to the appropriate Data Integrator data types; data loaded to a **target** is converted from its Data Integrator data type to the type appropriate for the target.

data warehouse

A relational **database** containing subject-oriented information that supports decision analysis. Typically, information in a data warehouse is integrated from multiple sources, including operational sources. Data warehouses provide a single, non-volatile point of reference for a particular time period. Data warehouses are composed of **data marts.** Data Integrator can populate a data warehouse.

database

A collection of **table**s managed by a **DBMS** such as Microsoft SQL Server or Oracle.

DataConnector

DataConnector operator instances are used to read data files generated by Data Integrator when performing bulk loading using the Teradata Warehouse Builder.

datastore

A logical channel connecting Data Integrator to a **database**. The datastore definition includes the name and location of the database as well as user authentication information. Data Integrator uses a datastore definition to qualify a table name wherever a table is indicated in a diagram or expression. You can access the datastore definition through the **object library**.

DBMS

Database management system. A software system that builds and maintains database tables.

debug mode

Allows you to debug a job using the **interactive debugger** features in the Designer.

declarative specification

A way of indicating the desired results of a **data transformation** without placing constraints on the method of reaching the results. When you create a **data flow** in Data Integrator, you specify what data you want to read from which **source**, what you want done to that data, and where you want it loaded. What you specify constitutes the declarative specification. Data Integrator then determines the most efficient way to reach the requirements you set, including pushing operations to the source **DBMS** for execution.

degree of parallelism

You can set a value at the data flow level that allows you to parallel-process a **transform** within a data flow.

For example, if from the Properties window of a data flow you set the **Degree of parallelism** to 5, then when the job executes, Data Integrator replicates each transform in the data flow five times. Each of these replicated transforms executes in parallel using a separate process thread. Using a value for degree of parallelism can improve job performance if a **Job Server**'s computer memory and number of CPUs supports a job's parallel-processing configuration settings.

dependency analysis

A category of metadata reports that produces reports based on search criteria you specify to find specific objects in the repository. For each object returned, dependency analysis reports provide object lineage as well as object impact information.

derived data

Data that results from a computational step applied to reference or event data. Derived data is the result of either relating two or more elements of a single transaction (such as aggregation), or relating one or more elements of a transaction to an external algorithm or rule.

Designer

A graphical user interface that allows you to design and test an **Data Integrator application**.

diagram

The icons and connections between the icons that make up the definition of a **job**, **work flow**, or **data flow**.

DMZ

Demilitarized zone. A perimeter network that is more accessible than your most protected network, but located inside a security layer protected from the internet.

drill down

A method of exploring detailed data that was used in creating a summary level of data. Drill down levels depend on the granularity of the data in the **data warehouse**.

DTD

Document type definition. A text file that describes the elements (tags) in an **XML** document and the relationship among them. When an XML document is used to describe a transaction, the DTD describes the data schema used in the transaction.

e-commerce

Buy and sell transactions conducted over the internet.

EDI

Electronic Data Interchange. Electronic exchange of structured data between businesses. This exchange is not dependent on hardware, software, or communication protocols.

element

A component found within XML Schemas and DTDs.

embedded data flow

A data flow created within one or more data flows.

enterprise data

Data that is defined for use across a corporate environment.

ERP system

Enterprise resource planning system. A **back-office application** from which Data Integrator can extract data.

exception

An error that occurs while executing a **job**. You can catch individual or groups of exceptions using a **try/catch block** inside a **work flow**. Catching an exception allows you to automatically execute a solution for the error.

Executive Information Systems (EIS)

Tools programmed to provide canned reports or briefing books to executives. They offer strong reporting and drill-down capabilities. These tools allow ad hoc querying against a multi-dimensional **database**, and most offer analytical applications along functional lines such as sales or financial analysis.

extract date

The date data was extracted.

extract frequency

The interval at which data is extracted, such as daily, weekly, monthly, or quarterly. The frequency that data extracts are needed in the **data warehouse** is determined by the shortest frequency requested through an order, or by the frequency required to maintain consistency of the other associated data types in the source data warehouse.

file

A **source** or **target** for data. To read or load data to a file within Data Integrator, define a **file format** that instructs Data Integrator about the structure of the data in the file.

file format

A description of how data is or should be organized in a **file** Data Integrator reads from or loads to. A file format can be specific to a single file or generic for many files.

front-office application

Software that companies use to support reporting, **e-commerce**, and other tasks. Types of front-office applications include:

- Supply chain collaboration
- Analytical
- Sell-side e-commerce
- Customer relationship management

function

A program that operates on values that are passed to it. Data Integrator functions are available through a function wizard in a **script**, **conditional**, or **Query transform**. Data Integrator also gives you access to functions provided by the **DBMS** you are using. In addition, you can define your own functions using the Data Integrator scripting language or by writing C-language programs.

IDoc

Intermediate Document. An SAP-specific format. Used for **EDI** and **ALE**.

IDoc type

Indicates the SAP format that is used to interpret the data of a business transaction. Consists of the following components:

- A control record

  Identical for each IDoc type.

- Several data records

  A single data record consists of a fixed key part and a variable data part. The data part is interpreted using segments, which differ depending on the IDoc type selected.

- Several status records

  Identical for each IDoc type. Describe the status states an IDoc has already passed through or the status an IDoc has attained.

import

The process of acquiring information for the **Data Integrator repository**. Import the following kinds of information into Data Integrator:

- The **metadata** for **source** and **target** databases
- Descriptions and code for user-defined and DBMS **functions** and **transforms**

InfoArea

In **SAP BW**, an arbitrary grouping of **InfoCube**s.

InfoCube

In **SAP BW**, a multidimensional star schema that is limited to one fact table and 16 dimensions.

InfoCubes store only numeric data at the fact level.

SAP BW reserves three of the dimensions for currencies, time, and units of measure. Dimensions have to be linked back to master data and hierarchy tables to gain access to the information stored there.

When a user query is initiated, BW creates a temporary **OLAP** cube that stores a constrained subset of the permanent OLAP cube. The temporary InfoCube persists until the user disconnects from BW or changes the query.

InfoObject

In **SAP BW**, an InfoObject is equivalent to a data element and its associated metadata (for example, description or business rules). Can be a **Key Figure** or a Characteristic.

InfoPackage

In **SAP BW**, a schedule for running BW updates.

Information resource (IR)

A producer or consumer of enterprise information. An information resource can be a flat file, a **back-office application**, an EAI system, or a Web application.

InfoSource

In **SAP BW**, a logical construct that defines a set of **InfoObject**s and their mappings to source data elements. InfoSources do not store data, but rather serve as conduits via **Transfer Structure**s.

Integration Rapid Mart

An **Data Integrator application** that provides point-to-point content mapping and data movement mechanisms between different **back-office applications**.

interactive debugger

A Designer feature that allows you to step through the data of a job one row at a time using filters and breakpoints on a line. Like executing a job, you can start the interactive debugger from the Debug menu when a job is active in the workspace. While in debug mode, all other Designer features are set to read-only.

interface

Data Integrator offers two types of interfaces:

An internal **Data Integrator interface**, allows you to create datastore connections to natively supported applications.

An external **Data Integrator interface** (or **adapter**), allows Data Integrator to communicate with information exchange technologies such as Web Services and MQ Series.

IR

See **Information resource (IR)**.

job

The unit of work that can be scheduled independently for execution by the **Administrator**. Jobs are special **work flows** that can be called by other work flows or jobs.

Job Server

A process that receives requests from the **Designer** and the **Administrator** to start and stop **jobs**. To start batch or real-time jobs, the Job Server triggers the **Data Integrator engine**. Engine processes run on the same computer as the Job Server process that triggers them.

join rank

A value given to or calculated for all data sets in a **data flow**. Data Integrator uses the join rank to determine which **source** to read first when assembling the **data set** in a join. Data Integrator uses the source with the lower join rank as the inner source of the join and uses the source with the higher join rank as the outer source of the join. If more than one of the sources is located in the same **DBMS**, the highest rank is used for all of the sources from that DBMS. When a source is processed through a **Query transform** or another **transform**, and then used in a join, the maximum join rank defined for the source is used as the join rank in the query.

Key Figure

In **SAP BW**, an **InfoObject** that represents a numeric fact.

legacy system

An information or transaction processing system used to store data such as bank balances, inventories, payroll, and manufacturing parts.

license-controlled feature

A Data Integrator feature that is enabled or disabled based on the product license. The product license controls which icons and settings are available in Data Integrator as an internal **Data Integrator interface**.

line

The line you use to connect objects in a workspace diagram. Lines represent a left-to-right flow path for data moving between two objects during job execution.

memory datastore

A datastore connection/container for memory tables.

memory table

Internal Data Integrator table used to store a data set in memory while a job runs. Use instead of staging tables to improve performance of a real-time job built with multiple data flows. Use a memory table to move a data set between data flows.

message

Represents hierarchical data, such as a header with line items, for document-oriented transactions, such as a purchase order.

metadata

In Data Integrator, information acquired and maintained to describe tables in **source** and **target** databases. This information includes the names of tables and their columns, and the data types of the columns. All the metadata for all user tables is imported at one time for a given **datastore**.

In general, metadata typically includes a description of data models, a description of the layouts used in database design, the definition of the system of record, the mapping of data from the system of record to other places in the environment, and specific database design definitions.

multiple data flow model

One of three data flow models you can use in real-time jobs. Use this model to develop real-time jobs in separate logical object segments (data flows) rather than in a single data flow. Allows for any number of data flows in a real-time processing loop under these conditions: The first data flow must have a single message source, the last data flow must have a single message target. Other data flows in the same real-time job cannot have a message source or message target. See also **put/acknowledge data flow model** and **single data flow model**.

nested data

Data in one **table** that is related to a single row of another table. A nested table appears in Data Integrator as a column in a parent table. Columns in the nested table can themselves contain tables.

object

Any item that you create in the **Designer**. Data Integrator distinguishes two classes of objects: **reusable objects** that are complete and can be reused in your projects (such as **data flows**) and **single-use objects** that only appear as components of other objects (such as a **try/catch block**). This distinction affects how you create and retrieve each type of object.

object definition

The options that describe the operation of an **object**. To view and modify an object definition, open the object so that its definition appears in the **workspace**.

object library

A tool in the **Designer** that gives you access to reusable objects.

ODBC

Open database connectivity. A standard for **database** access initiated by the SQL Access Group consortium and now owned by Microsoft.

ODS

Operational data store. An OLAP-designed relational **database** that an enterprise has designated as the operational database of record (for example, a finance department might use an ODS to close its books).

OLAP

Online analytical processing. A relational **database** design that is optimized for random access. OLAP systems are used in a query environment, such as for a **front-office application**.

OLTP

Online transaction processing. A relational database design optimized for operational use. OLTP systems are used in an operational environment, such as for a **back-office application**.

operation code

A flag associated with a row in a **data set** that indicates the status of the data in the row. The operation codes are INSERT, UPDATE, DELETE, and NORMAL.

operational statistics

A category of reports within the metadata reporting tool that provides you with job and data flow performance information over a given period of time or number of executions.

parameter

A value passed to a **work flow** or **data flow** when that flow is called.

partition

To divide table data into sets based on a criteria such as a range or list of values in each row. You can configure Data Integrator to read and write partitioned table data in parallel threads. Designing jobs with partitioned table data can improve job performance if a Job Server's computer memory and number of CPUs supports the job's parallel-processing configuration settings.

project

The collection of **jobs** available in the **Designer** at a given time. A project provides a way to organize the **objects** you create.

property

Detailed descriptive information about **objects**.

put/acknowledge data flow model

One of three data flow models you can use in real-time jobs. Similar to the multiple data flow model, this model does not require a data flow with a target message. Data Integrator creates a target message to automatically acknowledge a request message. Designed for use with IDoc sources in SAP R/3 systems. See also **multiple data flow model** and **single data flow model**.

Query transform

A **data transformation object** that you can use to map columns from a **source** to a **target** schema, add new columns to the target schema, determine the data to extract, and perform operations on the data. Like a SQL SELECT statement, a query creates a **data set** that satisfies the conditions you specify.

Rapid Mart

A prepackaged **Data Integrator application** that delivers data for a specific business purpose. Easily deployed and customized, BusinessObjects™ Rapid Marts include the extraction, transformation, and data loading logic to support analysis or **e-commerce**. There are three types of Rapid Mart:

- **Analytic Rapid Mart**
- **Commerce Rapid Mart**
- **Integration Rapid Mart**

R/3 data flow

A **data flow** that extracts data from an **SAP R/3** source table. Data Integrator translates steps you define in an R/3 data flow into **ABAP** and then passes the ABAP program back to your SAP R/3 system for execution. The resulting **table** or **file** resides on the SAP R/3 system to be used as a **source** in the parent data flow.

real-time job

A group of objects (data flows, work flows, conditionals, scripts, and so forth) that execute on-demand as a "request-response" system. You design real-time jobs in the **Designer**, then configure them as real-time services and associate them with an **Access Server** in the **Administrator**, where they are started, managed and monitored. When a real-time service receives a request from a caller, it processes the request and returns a reply. This type of job is recommended for operations on small amounts of data.

relational data

A **data set** in which data in each column contains a scalar value. Data Integrator can process relational data; it can also process **nested data**.

repository

See **Data Integrator repository**.

reusable object

An **object** that can be defined, stored, and reused independent of other objects. Create reusable objects from the **object library**.

RFC server

Remote Function Call server. The Data Integrator RFC server allows third-party programs, including SAP R/3, to schedule and initiate Data Integrator jobs and return the results to Data Integrator.

SAP BW

Business Information Warehouse, an SAP product.

SAP R/3

An **ERP system**.

script

A step in a **work flow** that allows you to calculate values to pass to other parts of the work flow. The script can call **function**s, execute if-then-else statements, and assign values to **variable**s. Write a script in the Data Integrator scripting language.

segment

Format with which the data records of **IDocs** are interpreted.

server group

A defined collection of Job Servers on different computers. A server group automatically measures resource availability on each Job Server in the group and distributes scheduled batch jobs to the Job Server with the lightest load at run-time. Use the Server Groups node in the Administrator's navigation tree to group Job Servers that are associated with the same repository into a server group.

service request

Any message sent from a Web client that requires processing by a **real-time job**.

session

Associated with the **Adapter Properties** tab in an adapter datastore, a session is created by a stream-oriented adapter to group adapter components and maintain state for interactivity between Data Integrator and the adapter.

single data flow model

One of three data flow models you can use in real-time jobs. Allows for one data flow in a real-time processing loop. This data flow must have a single message source and a single message target. See also **multiple data flow model** and **put/acknowledge data flow model**.

single-use object

A step in a **work flow** or **data flow** that cannot be saved independently of the flow. Create single-use objects from the tool palette.

smart editor

A flexible editing tool in Data Integrator used for creating scripts, expressions, and custom functions without having to type the names of existing elements like column, function, and variable names.

SNMP

System Network Management Protocol helps network administrators manage network routing hardware. The protocol can manage a variety of hardware and software devices.

source

An **object** from which Data Integrator reads data.

SQL

Structured Query Language. A query language for accessing relational, **ODBC**, DRDA, or non-relational **database** systems.

SQL query tool

An end-user tool that accepts **SQL** to be processed against one or more relational databases.

star schema

A **database** design you can use to format data in a **data mart**. This design is based on a single fact table to which any number of dimensional tables may be joined. This type of database design supports multi-dimensional database analysis.

step

An **object** that is part of the definition of a **work flow** or **data flow**. Each step is represented by an icon in the diagram of the flow and is connected to other steps to indicate the flow of data through the data flow or the order of execution in the work flow.

table

A **database** table that Data Integrator reads data from or loads data into. The path and mechanisms for reading and loading data and apportioning the data among rows and columns are defined in the **datastore** that the table is associated with. Writing a **data set** to a database table means sending a combination of rows with appropriate **operation code** to the database table.

target

An object in which Data Integrator loads extracted and transformed data in a **data flow**. Data Integrator only loads rows flagged as INSERT or UPDATE.

TCP/IP

Transmission Control Protocol/Internet Protocol. The basic communication protocol of the internet, and often intranets and extranets. A computer having direct access to the internet contains a copy of the TCP/IP program. TCP/IP makes it possible for computers to communicate with each other.

Tdpid

(Teradata Director Program ID) The server name Data Integrator uses when loading with the bulk loader option. Data Integrator uses tdpid as a Teradata Warehouse Builder operator attribute.

Transfer Structure

In **SAP BW**, a data structure that defines a set of **InfoObjects** to be moved into BW. To an ETL tool like Data Integrator, a Transfer Structure looks like a table.

transform

A step in a **data flow** that acts on a **data set**. Data Integrator transforms are available through the **object library**. You can also import user-defined transforms and transforms from your **DBMS**.

Transformation Rule (Transfer Rule)

In **SAP BW**, a data manipulation rule between a **Transfer Structure** and a **Communication Structure**. The results of these transformation are globally available to all **InfoCubes**.

try/catch block

A combination of a try object and one or more catch objects that define alternate execution paths in case an error occurs during the execution of a **job**. You can tune try/catch blocks to trap specific errors and to provide general alerts or messages if an error occurs.

Update Rule

In **SAP BW**, a data manipulation rule between a **Communication Structure** and an **InfoCube**. The result of the data manipulation is specific to one InfoCube.

variable

A symbolic placeholder for a value. Data Integrator lets you define local variables and global variables.

Local variables pertain to the **work flow** in which they are defined. You can pass the value into another **work flow** or **data flow** using a **parameter**.

Global variables pertain to the **job** in which they are defined. With global variables, there is no need to define parameters between **objects** in the job. Global variables can also be selected at execution time. This eliminates the need to open the Designer to set global variable values.

Web Services

A standard platform for integrating applications. Web Services allow different programs, constructed in different languages, on different platforms to communicate with each other. DATA INTEGRATOR supports three XML web service technologies: SOAP, WSDL, and **XML Schema**.

work flow

A **reusable object** containing **steps** to define the order of **job** execution. Work flows call **data flow**s, but cannot manipulate data themselves.

You call work flows from inside other work flows or **job**s. You pass information into or out of work flows using **parameter**s. You reuse work flows by dragging existing work flows from the **object library**.

workspace

The window inside the **Designer** in which you define, display, and modify **objects**. The workspace for a **data flow** contains an area to build a **diagram** representing the data flow definition. The workspace for a **transform** contains an editor for modifying the transform options.

WSDL

Web Services Definition Language. Web services are self-contained, modular business process applications based on open Internet standards.

XML

Extensible Markup Language. This markup language is like HTML (Hypertext Markup Language) in that it specifies a standard with which you can define your own markup languages with their own sets of tags. XML allows you to define various tags with various rules, such as tags that represent business rules, tags that represent data description, or tags that represent data relationships.

XML Schema

The XML format used by Data Integrator to support message processing that includes **Web Services**. XML Schemas describe the data structure of an **XML** file or message. Data flows can read and write data to messages or files based on a specified XML Schema format. You can use the same XML Schema to describe multiple XML sources or targets. XML Schema properties include: Name, Description, Imported from, Root element name, and Namespace.

Index

Business Objects, Americas
3030 Orchard Parkway
San Jose, CA 95134
USA

info@businessobjects.com
www.businessobjects.com

Data Integrator™

Designer Guide

Version 6.5

Document Number DI-65-0102-001

January 30, 2004

Contents

1 Welcome

Welcome to the *Data Integrator Designer Guide*. The Data Integrator Designer provides a graphical user interface (GUI) development environment in which you define data application logic to extract, transform, and load data from databases and applications into a data warehouse used for analytic and on-demand queries. You can also use the Designer to define logical paths for processing message-based queries and transactions from Web-based, front-office, and back-office applications.

This chapter discusses these topics:

- Overview of this document
- Audience and assumptions
- More Data Integrator product documentation

Overview of this document

The book contains two kinds of information:

- *Conceptual information* that helps you understand the Data Integrator Designer and how it works
- *Procedural information* that explains in a step-by-step manner how to accomplish a task

You will find this book most useful:

- While you are learning about the product
- While you are performing tasks in the design and early testing phase of your data-movement projects
- As a general source of information during any phase of your projects

Audience and assumptions

This and other Data Integrator product documentation assumes the following:

- You are an application developer, consultant, or database administrator working on data extraction, data warehousing, or data integration.
- You understand your source data systems, RDBMS, business intelligence, and messaging concepts.
- You understand your organization's data needs.
- You are familiar with SQL (Structured Query Language).
- If you are interested in using this product to design real-time processing, you should be familiar with:
  - DTD and XML Schema formats for XML files
  - Publishing Web Services (WSDL, HTTP, and SOAP protocols, etc.)
- You are familiar Data Integrator installation environments—Microsoft Windows or UNIX.

More Data Integrator product documentation

Consult the *Data Integrator Getting Started Guide* for:

- An overview of Data Integrator products and architecture
- Data Integrator installation and configuration information
- A list of product documentation and a suggested reading path

After you install Data Integrator, you can view technical documentation from many locations. To view documentation in PDF format, you can:

- Select **Start** > **Programs** > **Data Integrator** ***version*** > **Data Integrator Documentation** and choose:
  - Release Notes
  - Release Summary
  - Technical Manuals
- Select one of the following from the Designer's **Help** menu:
  - Release Notes
  - Release Summary
  - Technical Manuals
- Select **Help** from the Data Integrator Administrator

You can also view and download PDF documentation by visiting Business Objects Customer Support online. To access this Web site, you must have a valid user name and password. To obtain your user name and password, go to http://www.techsupport.businessobjects.com and click **Register**.

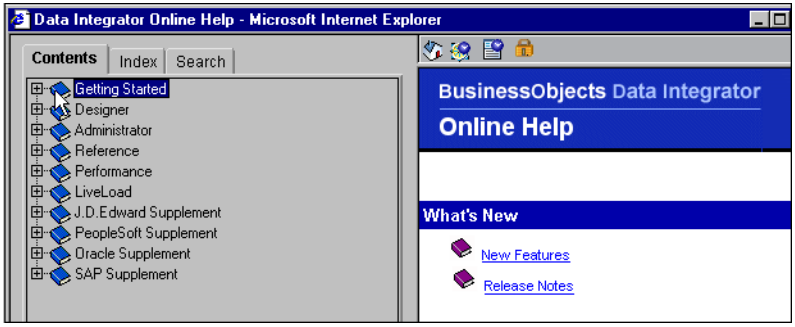
After installing Data Integrator Designer you can also view documentation in HTML format. **Online Help** opens automatically the first time you open the Designer.

You can also open **Help,** using one of the following methods:

- Choose **Contents** from the Designer's Help menu.
- Click objects in the object library or workspace and press **F1**.

  Online Help opens to the subject you selected.

Use **Online Help's** links and tool bar to navigate:



Online help tool bar

Icon	Icon Name	Description
	Home	Returns to the Online Help home page.
	Show/Hide Navigation	Shows and hides the Table of Contents for the core suite of Data Integrator Technical Manuals. Includes a master **Index** and **Search** tabs.
	Show/Hide Descriptions	The Online Help window automatically updates its information when you click an object in the Designer window. In addition, you can click any object in the Designer window then click this button to show or hide an object's local and global variables.
	Start/Stop Help Updates	When the lock appears open, each time you click an object in the Designer window, the Online Help information automatically updates.
		When the lock icon appears closed, the Online Help window does not automatically update when you click objects in the Designer window; you must press F1.

2 Logging in to the Designer

This chapter describes how to log in to the Data Integrator Designer. When you log in to the Data Integrator Designer, you are actually logging in to the database you defined for the Data Integrator repository.

Data Integrator repositories can reside on Oracle, Microsoft SQL Server, Informix, IBM DB2, Sybase, or any ODBC-compliant database.

This chapter discusses:

- Creating a Data Integrator repository
- Associating the repository with a Job Server
- Entering repository information
- Resetting users

Creating a Data Integrator repository

You must configure a local repository to log in to Data Integrator. Typically, you create a repository during installation. However, you can create a repository at any time using the Data Integrator Repository Manager.

➢ To create a local repository

1. Define a database for the local repository using your database management system.
2. From the **Start** menu, choose **Programs** > **BusinessObjects Data Integrator 6.1** > **Repository Manager** (assuming you installed Data Integrator in the Data Integrator program group).
3. In the Repository Manager window, enter the database connection information for the repository and select **Local** for repository type.
4. Click **Create**. This adds the Data Integrator repository schema to the specified database.

Associating the repository with a Job Server

Each repository must be associated with at least one Job Server, which is the process that starts jobs. When running a job from a repository, you select one of the associated repositories. In production environments, you can balance loads appropriately. You can link any number of repositories to a single Job Server. The same Job Server can run jobs stored on multiple repositories.

Typically, you define a Job Server and link it to a repository during installation. However, you can define or edit Job Servers or links between repositories and Job Servers at any time using the Data Integrator Server Manager.

➢ To create a Job Server for your local repository

1. Open the Data Integrator Server Manager.

   From the **Start** menu, choose **Programs** > **BusinessObjects Data Integrator 6.1** > **Server Manager** (assuming you installed Data Integrator in the Data Integrator program group).

2. See "To configure Job Servers" on page 90 of the *Data Integrator Getting Started Guide* for detailed instructions.

Entering repository information

To log in, enter the connection information for your Data Integrator repository. The required information varies with the type of database containing the repository. This section discusses:

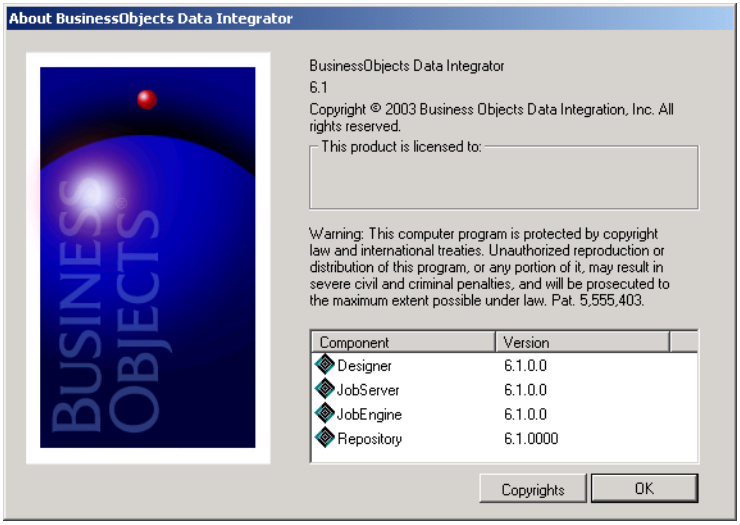
- Version restrictions
- Oracle login
- Microsoft SQL Server login
- Informix login
- IBM DB2 login
- Sybase login

Version restrictions

Your repository version must be associated with the same major release as the Designer and must be less than or equal to the version of the Designer. For example, Designer 5.2 can access repositories 5.0, 5.1, and 5.2 (equal to or less than), but not repository 4.4 (different major release version). So, in this example, repository 5.0 is the earliest repository version that could be used with Designer version 5.2.

During login, Data Integrator alerts you if there is a mismatch between your Designer version and your repository version.

After you log in, you can view Data Integrator and repository versions by selecting **Help** > **About Data Integrator**.



Some features in the current release of the Designer might not be supported if you are not logged in to the latest version of the repository.

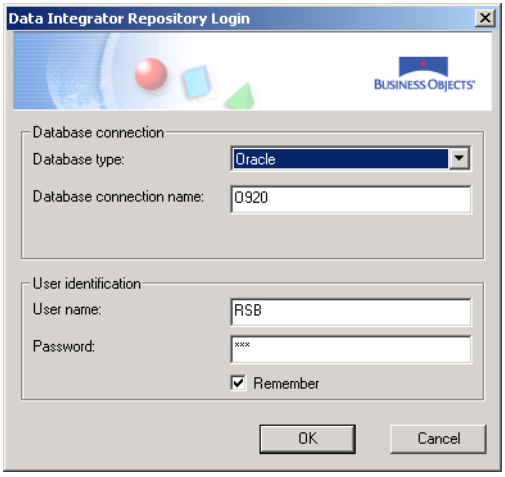
Oracle login

From the Windows **Start** menu, select **Programs** > **BusinessObjects Data Integrator 6.1** > **Data Integrator Designer**.

In the Repository Login window, complete the following fields:

- **Database type** — Select **Oracle**.
- **Database connection name** — The TNSnames.ora entry or Net Service Name of the database.
- **User name** and **Password** —The user name and password for a Data Integrator repository defined in an Oracle database.

- **Remember** — Check this box if you want the Designer to store this information for the next time you log in.



Microsoft SQL Server login

From the Windows **Start** menu, select **Programs** > **BusinessObjects Data Integrator 6.1** > **Data Integrator Designer**.

For a Microsoft SQL Server repository, you must complete the following fields:

- **Database type** — Select **Microsoft_SQL_Server**.
- **Database server name** —The database server name.
- **Database name** — The name of the specific database to which you are connecting.
- **User name** and **Password** — The user name and password for a Data Integrator repository defined in a Microsoft SQL Server database.
- **Remember** — Check this box if you want the Designer to store this information for the next time you log in.

Informix login

From the Windows **Start** menu, select **Programs** > **BusinessObjects Data Integrator 6.1** > **Data Integrator Designer**.

For an Informix repository, you must complete the following fields:

- **Database type** — Select **Informix**.
- **Informix datasource** — The data source name.
- **User name** and **Password** — The user name and password for a Data Integrator repository defined in an Informix database.
- **Remember** — Check this box if you want the Designer to store this information for the next time you log in.

IBM DB2 login

From the Windows **Start** menu, select **Programs** > **BusinessObjects Data Integrator 6.1** > **Data Integrator Designer**.

For a DB2 repository, you must complete the following fields:

- **Database type** — Select **DB2**.
- **DB2 datasource** — The data source name.
- **User name** and **Password** — The user name and password for a Data Integrator repository defined in a DB2 database.
- **Remember** — Check this box if you want the Designer to store this information for the next time you log in.

Sybase login

From the Windows **Start** menu, select **Programs** > **BusinessObjects Data Integrator 6.1** > **Data Integrator Designer**.

For a Sybase repository, you must complete the following fields:

- **Database type** — Select **Sybase**.

- **Database server name** — Enter the database's server name.
- **Database name** — Enter the name of the specific database to which you are connecting.
- **User name** and **Password** — Enter the user name and password for this database.
- **Remember** — Check this box if you want the Designer to store this information for the next time you log in.

Resetting users

Occasionally, more than one person may attempt to log in to a single repository. If this happens, the Reset Users window appears, listing the users and the time they logged in to the repository.

From this window, you have several options. You can:

- **Reset Users** to clear the users in the repository and set yourself as the currently logged in user.
- **Continue** to log in to the system regardless of who else might be connected.
- **Exit** to terminate the login attempt and close the session.

NOTE: Only use **Reset Users** or **Continue** if you know that you are the only user connected to the repository. Subsequent changes could corrupt the repository.

3 Designer user interface

This chapter provides basic information about the Designer's graphical user interface. It contains the following sections:

Data Integrator objects

All "entities" you define, edit, or work with in Data Integrator Designer are called *objects*. The local object library shows objects such as source and target metadata, system functions, projects, and jobs.

Objects are hierarchical and consist of:

- *Options*, which control the operation of objects. For example, in a datastore, the *name* of the database to which you connect is an option for the datastore object.
- *Properties*, which document the object. For example, the *name of the object* and the *date it was created* are properties. Properties describe an object, but do not affect its operation.

Data Integrator has two types of objects:

- Reusable objects
- Single-use objects

The object type affects how you define and retrieve the object.

Reusable objects

Most objects defined in Data Integrator are available for reuse and can be replicated.

After you define and save a reusable object, Data Integrator stores the definition in the local repository. You can then reuse the definition as often as necessary by creating calls to the definition. Access reusable objects through the local object library.

A reusable object has a single definition; all calls to the object refer to that definition. If you change the definition of the object in one place, you are changing the object in all other places in which it appears.

A data flow, for example, is a reusable object. Multiple jobs, like a weekly load job and a daily load job, can call the same data flow. If the data flow changes, both jobs use the new version of the data flow.

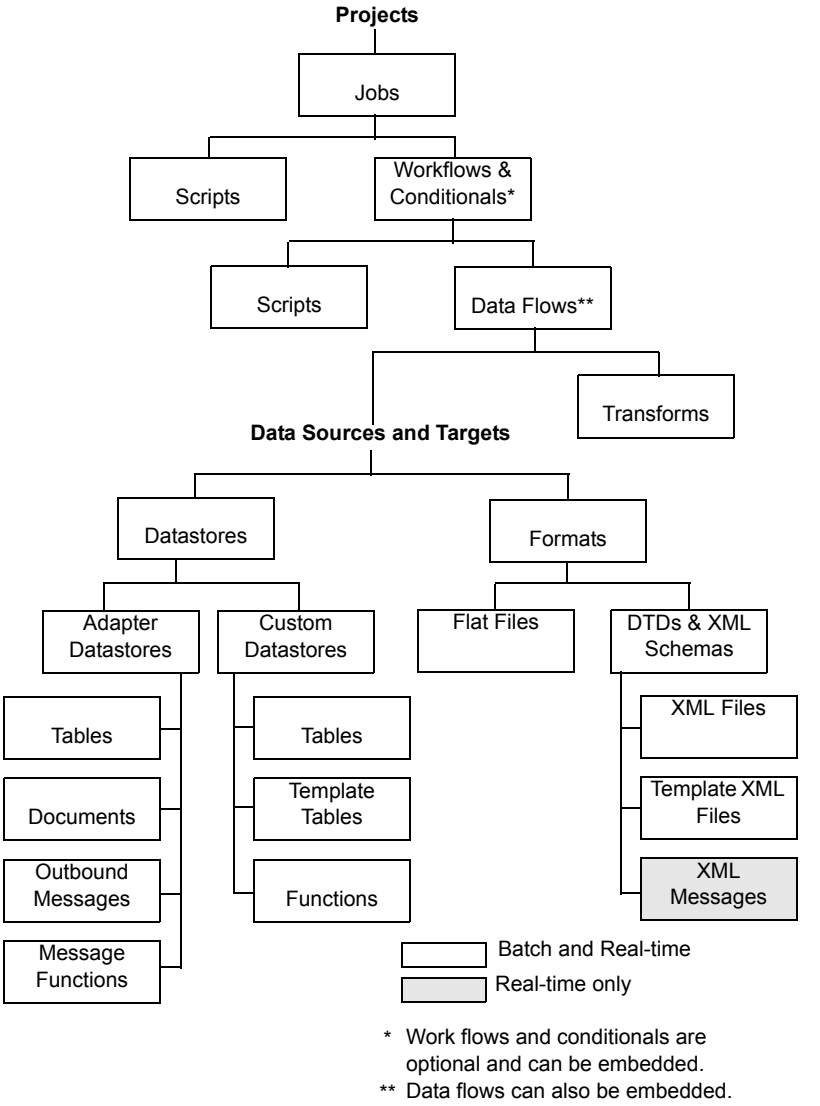
The object library contains object definitions. When you drag and drop an object from the object library, you are really creating a new reference (or *call*) to the existing object definition.

Single-use objects

Some objects are defined only within the context of a single job or data flow, for example scripts and specific transform definitions.

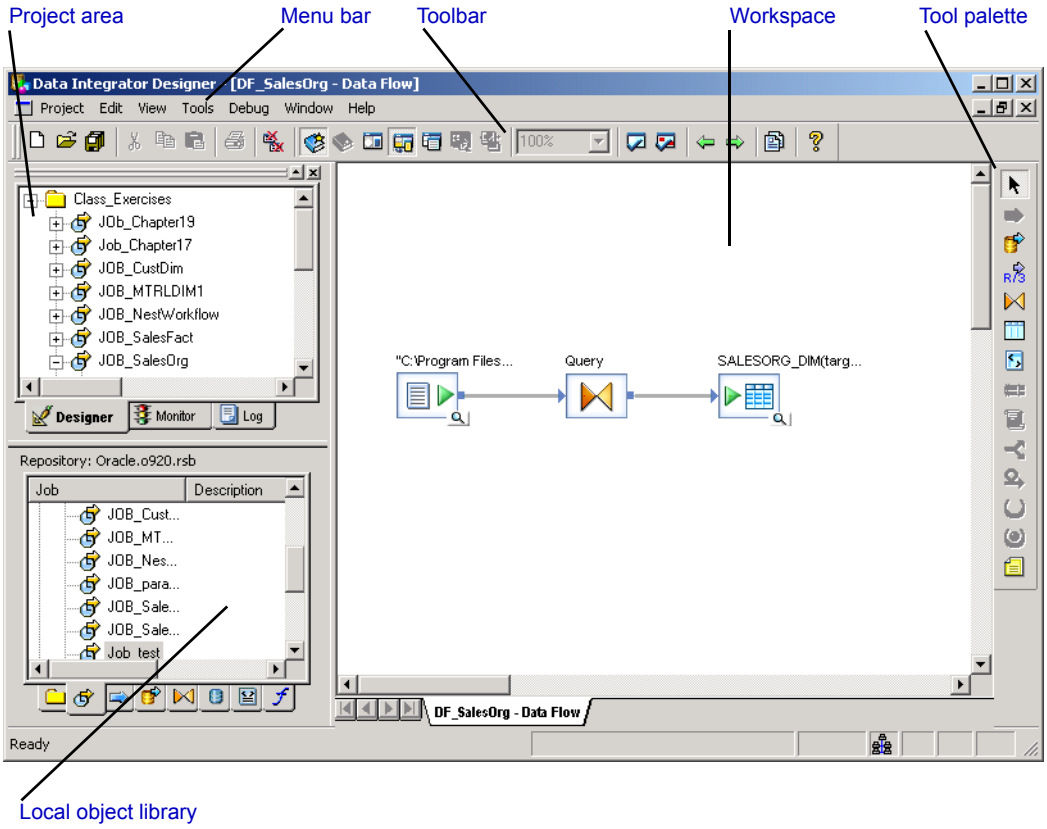
Object hierarchy

Data Integrator object relationships are hierarchical. The following figure shows the relationships between major Data Integrator object types:



Designer window

The Data Integrator Designer user interface consists of a single application window and several embedded supporting windows.



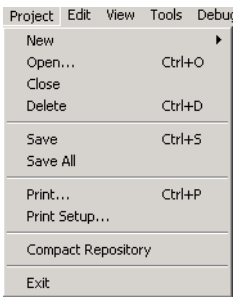
The application window contains the Menu bar, Toolbar, Project area, Tool palette, tabbed Workspace, and tabbed Local object library.

Menu bar

This section contains a brief description of the Designer's menus:

- Project menu
- Edit menu
- View menu
- Tools menu
- Debug menu
- Validation menu
- Window menu
- Help menu

Project menu

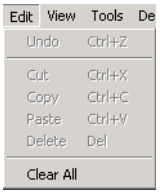


The project menu contains standard Windows as well as Data Integrator-specific options.

- **New** — Define a new project, batch job, real-time job, work flow, data flow, transform, datastore, file format, DTD, XML Schema, or custom function.
- **Open** — Open an existing project.
- **Close** — Close the currently open project.
- **Delete** — Delete the selected object.
- **Save** — Save the object open in the workspace.
- **Save All** — Save all changes to objects in the current Designer session.
- **Print** — Print the active workspace.
- **Print Setup** — Set up default printer information.
- **Compact Repository** — Remove redundant and obsolete objects from the repository tables.

- **Exit** — Exit Data Integrator Designer.

Edit menu



The Edit menu provides standard Windows commands with a few restrictions.

- **Undo** — Undo the last operation (text edits only).
- **Cut** — Cut the selected object or text and place it on the clipboard.
- **Copy** — Copy the selected object or text to the clipboard.

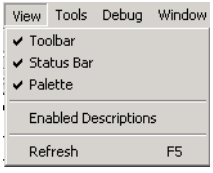
  NOTE: You cannot copy reusable objects using the **Copy** command; instead, use **Replicate** in the object library to make an independent copy of an object.

- **Paste** — Paste the contents of the clipboard into the active workspace or text box.

  NOTE: You can only paste clipboard contents once. To paste again, you must cut or copy the objects again.

- **Delete** — Delete the selected object.
- **Clear All** — Clear all objects in the active workspace (no undo).

View menu



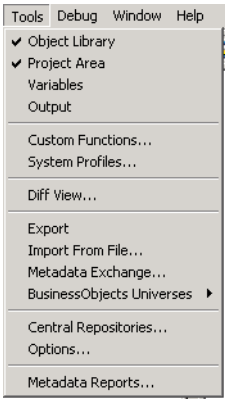
A check mark indicates that the tool is active.

- **Toolbar** — Display or remove the toolbar in the Designer window.
- **Status Bar** — Display or remove the status bar in the Designer window.
- **Palette** — Display or remove the floating tool palette.
- **Enabled Descriptions** — View descriptions for objects with enabled descriptions.

- **Refresh** — Redraw the display.

  Use this command to ensure the content of the workspace represents the most up-to-date information from the repository.

Tools menu



A check mark indicates that the tool is active.

- **Object Library** — Open or close the object library window. For more information, see "Local object library" on page 39.
- **Project Area** — Display or remove the project area from the Data Integrator window. For more information, see "Project area" on page 30.
- **Variables** — Open or close the Variables and Parameters window. For more information, see "Variables and Parameters" on page 289.
- **Output** — Open or close the Output window. The Output window shows errors that occur such as during job validation or object export.
- **Custom Functions** — Display the Custom Functions window. For more information, see "Custom functions" on page 333 of the *Data Integrator Reference Guide*.
- **System Profiles** — Display the System Profiles editor. For more information, see "Defining profiles" on page 111.
- **Diff View** — Display the Differences Among Objects window. For more information, see "Comparing objects" on page 576.
- **Export** — Export individual repository objects to another repository or file. This command opens the Export editor in the workspace. You can drag objects from the object library into the editor for export. To export your whole repository, in the object library right-click and select **Repository > Export to file**. For more information, see "Exporting/importing objects in Data Integrator" on page 340.

- **Import From File** — Import objects into the current repository from a file. For more information, see "Importing from a file" on page 347.
- **Metadata Exchange** — Import and export metadata to third-party systems via a file. See "Metadata exchange" on page 406.
- **BusinessObjects Universes** — Export (create or update) metadata in Business Objects Universes. See "Creating Business Objects universes" on page 410.
- **Central Repositories** — Create or edit connections to a central repository for managing object versions among multiple users. See "Multi-user environment setup" on page 546.
- **Options** — Display the Options window. See "General and environment options" on page 63.
- **Metadata Reports** — Display the Metadata Reports window. Select the object type, report type, and the objects in the repository that you want to list in the report. See "Metadata reporting tool" on page 419.

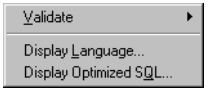
Debug menu

The only options available on this menu at all times are **Show Filters/Breakpoints** and **Filters/Breakpoints**. The **Execute** and **Start Debug** options are only active when a job is selected. All other options are available as appropriate when a job is running in the Debug mode. For more information, see "Using the interactive debugger" on page 382.

- **Execute** - Opens the Execution Properties window which allows you to execute the selected job.
- **Start Debug** - Opens the Debug Properties window which allows you to run a job in the debug mode.
- **Show Filters/Breakpoints** - Shows and hides filters and breakpoints in workspace diagrams.

- **Filters/Breakpoints** - Opens a window you can use to manage filters and breakpoints. For more information, see "Filters and Breakpoints window" on page 399.

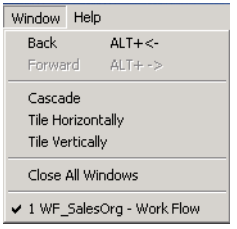
Validation menu



The Designer displays options on this menu as appropriate when an object is open in the workspace.

- **Validate** — Validate the objects in the current workspace view or all objects in the job before executing the application.
- **Display Language** — View a read-only version of the language associated with the job.
- **Display Optimized SQL** — Display the SQL that Data Integrator generated for a selected data flow. See "Viewing SQL" on page 34 of the *Data Integrator Performance Optimization Guide*.

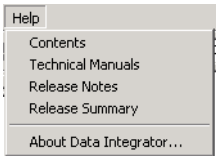
Window menu



The Window menu provides standard Windows options.

- **Back** — Move back in the list of active workspace windows.
- **Forward** — Move forward in the list of active workspace windows.
- **Cascade** — Display window panels overlapping with titles showing.
- **Tile Horizontally** — Display window panels side by side.
- **Tile Vertically** — Display window panels one above the other.
- **Close All Windows** — Close all open windows.
- A list of objects open in the workspace also appears on the Windows menu. The name of the currently-selected object is indicated by a check mark. Navigate to another open object by selecting its name in the list.

Help menu



- **Contents** — Display on-line help. Data Integrator's on-line help works with Microsoft Internet Explorer version 5.5 and higher.
- **Technical Manuals**— Display a PDF version of Data Integrator documentation. This file contains the same content as on-line help. It is provided for users who prefer to print out their documentation. This format prints graphics clearly and includes a master index and page numbers/references. You can also access the same file from the **Help** menu in the Administrator or from the <linkdir>\Doc\Books directory.
- **Release Notes** — Display current release notes.
- **Release Summary** — Display summary of new features in the current release.
- **About Data Integrator** — Display information about Data Integrator including versions of the Designer, Job Server and engine, copyright information, and a link to the Business Objects Web site.

Toolbar

In addition to many of the standard Windows tools, Data Integrator provides application-specific tools, including:

Icon	Tool	Description
	Close all windows	Closes all open windows in the workspace.
	Local Object Library	Opens and closes the local object library window.
	Central Object Library	Opens and closes the central object library window.
	Variables	Opens and closes the variables and parameters creation window.
	Project Area	Opens and closes the project area.
	Output	Opens and closes the output window.
	Diff View	Displays the differences between objects.
	View Enabled Descriptions	Enables the system level setting for viewing object descriptions in the workspace.
	Validate Current View	Validates the object definition open in the workspace. Other objects included in the definition are also validated.
	Validate All Objects in View	Validates the object definition open in the workspace. Objects included in the definition are also validated.
	View Where Used	Opens the Output window, which lists parent objects (such as jobs) of the object currently open in the workspace (such as a data flow). Use this command to find other jobs that use the same data flow, before you decide to make design changes. To see if an object in a data flow is reused elsewhere, right-click one and select **View Where Used**.
	Go Back	Move back in the list of active workspace windows.
	Go Forward	Move forward in the list of active workspace windows.
	Metadata Reports	Opens and closes the Metadata Reports window.
	About	Opens the Data Integrator About box, with product component version numbers and a link to the Business Objects Web site.

Use the tools to the right of the About tool with the interactive debugger. See "Menu options and tool bar" on page 400.

Project area

The *project area* provides a hierarchical view of the objects used in each project. Tabs on the bottom of the project area support different tasks. Tabs include:

Create, view and manage projects. Provides a hierarchical view of all objects used in each project.

View the status of currently executing jobs. Selecting a specific job execution displays its status, including which steps are complete and which steps are executing. These tasks can also be done using the Data Integrator Administrator.

View the history of complete jobs. Logs can also be viewed with the Data Integrator Administrator.

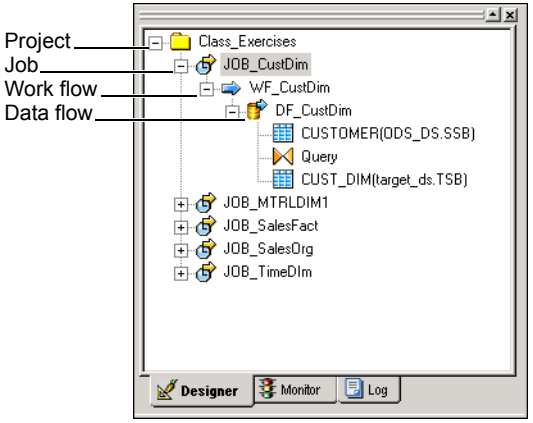
To control project area location, right-click its gray border and select/deselect **Allow Docking**, or select **Hide** from the menu.

- When you select **Allow Docking**, you can click and drag the project area to dock at and undock from any edge within the Designer window. When you drag the project area away from a Designer window edge, it stays undocked. To quickly switch between your last docked and undocked locations, just double-click the gray border.

  When you deselect **Allow Docking**, you can click and drag the project area to any location on your screen and it will not dock inside the Designer window.

- When you select **Hide**, the project area disappears from the Designer window. To unhide the project area, click its toolbar icon.

Here's an example of the Project window's **Designer** tab, which shows the project hierarchy:



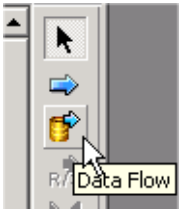
As you drill down into objects in the Designer workspace, the window highlights your location within the project hierarchy.

Tool palette

The tool palette is a separate window that appears by default on the right edge of the Designer workspace. You can move the tool palette anywhere on your screen or dock it on any edge of the Designer window.

The icons in the tool palette allow you to create new objects in the workspace. The icons are disabled when they are not allowed to be added to the diagram open in the workspace.

To show the name of each icon, hold the cursor over the icon until the tool tip for the icon appears, as shown.



When you create an object from the tool palette, you are creating a new definition of an object. If a new object is reusable, it will be automatically available in the object library after you create it.

For example, if you select the data flow icon from the tool palette and define a new data flow, later you can drag that existing data flow from the object library, adding a call to the existing definition.

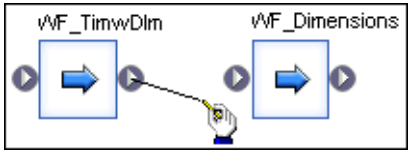
The tool palette contains the following icons:

Icon	Tool	Description (class)	Available
	Pointer	Returns the tool pointer to a selection pointer for selecting and moving objects in a diagram.	Everywhere
	Work flow	Creates a new work flow. (reusable)	Jobs and work flows
	Data flow	Creates a new data flow. (reusable)	Jobs and work flows
	R/3 data flow	Used only with the SAP licensed extension. See the *Data Integrator Supplement for SAP*.	
	Query transform	Creates a template for a query. Use it to define column mappings and row selections. (single-use)	Data flows
	Template table	Creates a table for a target. (single-use)	Data flows
	Template XML	Creates an XML template. (single-use)	Data flows
	Data transport	Used only with the SAP Licensed extension. See the *Data Integrator Supplement for SAP*.	
	Script	Creates a new script object. (single-use)	Jobs and work flows
	Conditional	Creates a new conditional object. (single-use)	Jobs and work flows
	Try	Creates a new try object. (single-use)	Jobs and work flows
	Catch	Creates a new catch object. (single-use)	Jobs and work flows
	Annotation	Creates an annotation. (single-use)	Jobs, work flows, and data flows

Workspace

When you open or select a job or any flow within a job hierarchy, the *workspace* becomes "active" with your selection. The workspace provides a place to manipulate system objects and graphically assemble data movement processes.

These processes are represented by icons that you drag and drop into a workspace to create a *workspace diagram***.** This diagram is a visual representation of an entire data movement application or some part of a data movement application.



This section describes major workspace area tasks, such as:

- Moving objects in the workspace area
- Connecting and disconnecting objects
- Describing objects
- Scaling the workspace
- Arranging workspace windows
- Closing workspace windows

Moving objects in the workspace area

Use standard mouse commands to move objects in the workspace.

➢ To move an object to a different place in the workspace area

1. Click to select the object.

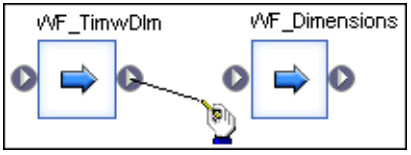
2. Drag the object to where you want to place it in the workspace.

Connecting and disconnecting objects

You specify the flow of data through jobs and work flows by connecting objects in the workspace from left to right in the order you want the data to be moved:

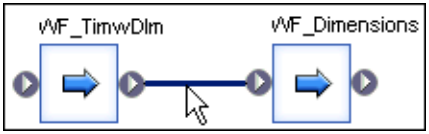
➢ To connect objects

1. Place the objects you want to connect in the workspace.
2. Click and drag from the triangle on the right edge of an object to the triangle on the left edge of the next object in the flow.



➢ To disconnect objects

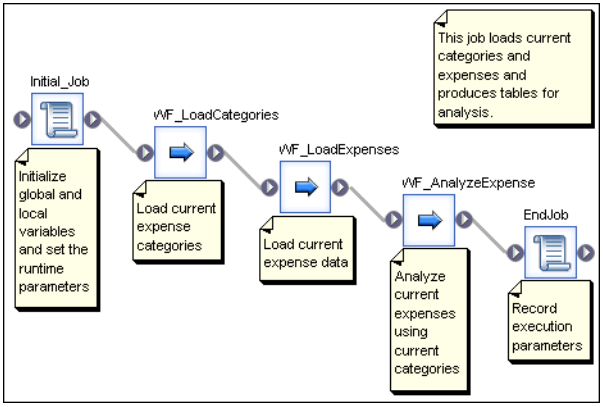
1. Click the connecting line.



2. Press the **Delete** key.

Describing objects

You can use descriptions to add comments about objects. You can use annotations to explain a job, work flow, or data flow. You can view object descriptions and annotations in the workspace. Together, descriptions and annotations allow you to document a Data Integrator application. For example, you can describe the incremental behavior of individual jobs with numerous annotations and label each object with a basic description.



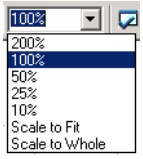
For more information, see "Creating descriptions" on page 52 and "Creating annotations" on page 56.

Scaling the workspace

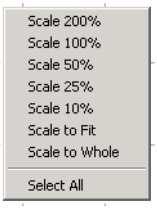
You can control the scale of the workspace. By scaling the workspace, you can change the focus of a job, work flow, or data flow. For example, you might want to increase the scale to examine a particular part of a work flow, or you might want to reduce the scale so that you can examine the entire work flow without scrolling.

➢ To change the scale of the workspace

1. In the drop-down list on the tool bar, select a predefined scale or enter a custom value.



2. Alternatively, right-click in the workspace and select a desired scale.



NOTE: You can also select **Scale to Fit** and **Scale to Whole**:

- Select **Scale to Fit** and the Designer calculates the scale that fits the entire project in the current view area.
- Select **Scale to Whole** to show the entire workspace area in the current view area.

Arranging workspace windows

The Window menu allows you to arrange multiple open workspace windows in the following ways: cascade, tile horizontally, or tile vertically.

Closing workspace windows

When you drill into an object in the project area or workspace, a view of the object's definition opens in the workspace area. The view is marked by a tab at the bottom of the workspace area, and as you open more objects in the workspace, more tabs appear. (You can show/hide these tabs from the **Tools > Options** menu. Go to **Designer > General** options and select/deselect **Show tabs in workspace**. For more information, see the "General and environment options" section.)

NOTE: These views use system resources. If you have a large number of open views, you might notice a decline in performance.

Close the views individually by clicking the close box in the top right corner of the workspace. Close all open views by selecting **Window > Close All Windows** or clicking the **Close All Windows** icon on the toolbar.

Local object library

The *local object library* provides access to reusable objects. These objects include built-in system objects, such as transforms, and the objects you build and save, such as datastores, jobs, data flows, and work flows.

The local object library is a window into your local *Data Integrator repository* and eliminates the need to access the repository directly. Updates to the repository occur through normal Data Integrator operation. Objects you create are added to the repository when you save them, and you access saved objects through the local object library. For more information about repositories, see Chapter 14, "Migration and Repositories".

To control object library location, right-click its gray border and select/deselect **Allow Docking**, or select **Hide** from the menu.

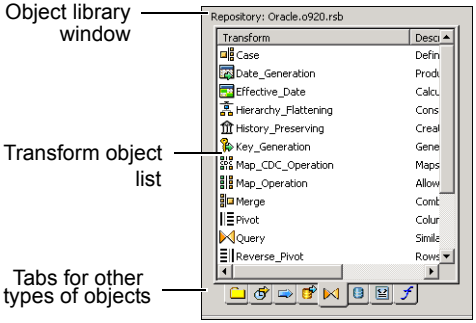
- When you select **Allow Docking**, you can click and drag the object library to dock at and undock from any edge within the Designer window. When you drag the object library away from a Designer window edge, it stays undocked. To quickly switch between your last docked and undocked locations, just double-click the gray border.

  When you deselect **Allow Docking**, you can click and drag the object library to any location on your screen and it will not dock inside the Designer window.

- When you select **Hide**, the object library disappears from the Designer window. To unhide the object library, click its toolbar icon.

➢ To open the object library

Choose **Tools** > **Object Library**, or click the object library icon in the icon bar.

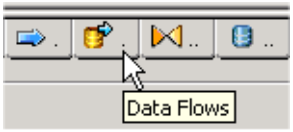


The object library gives you access to the object types listed in the following table. The table shows the tab on which the object type appears in the object library and describes the Data Integrator context in which you can use each type of object.

Tab	Description
	Projects are sets of jobs available at a given time.
	Jobs are executable work flows. There are two job types: batch jobs and real-time jobs.
	Work flows order data flows and the operations that support data flows, defining the interdependencies between them.
	Data flows describe how to process a task.
	Transforms operate on data, producing output data sets from the sources you specify. The object library lists both built-in and custom transforms.
	Datastores represent connections to databases and applications used in your project. Under each datastore is a list of the tables, documents, and functions imported into Data Integrator.
	Formats describe the structure of a flat file, XML file, or XML message.
	Custom Functions are functions written in the Data Integrator Scripting Language. They can be used in Data Integrator jobs.

➢ To display the name of each tab as well as its icon, do one of the following:

- Make the object library window wider until the names appear.
- Hold the cursor over the tab until the tool tip for the tab appears, as shown.



➢ To sort columns in the object library

- Click the column heading.

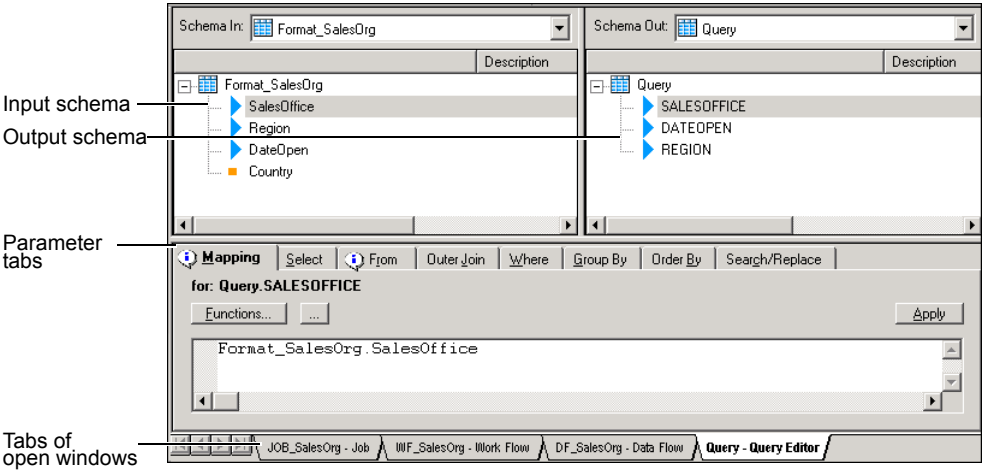
  For example, you can sort data flows by clicking the **Data Flow** column heading once. Names are listed in ascending order. To list names in descending order, click the **Data Flow** column heading again.

Object editors

To work with the options for an object, in the workspace click the name of the object to open its editor. The editor displays the input and output schemas for the object and a panel below them listing options set for the object. If there are many options, they are grouped in tabs in the editor.

A *schema* is a data structure that can contain columns, other nested schemas, and functions (the contents are called *schema elements*). A table is a schema containing only columns.

A common example of an editor is the editor for the query transform, as shown in the following illustration:



For specific information about the query editor, see "Query editor" on page 174.

In an editor, you can:

- Undo or redo previous actions performed in the window (right-click and choose **Undo** or **Redo**)
- Find a string in the editor (right-click and choose **Find**)

- Drag-and-drop column names from the input schema into relevant option boxes
- Use colors to identify strings and comments in text boxes where you can edit expressions (keywords appear blue; strings are enclosed in quotes and appear pink; comments begin with a pound sign and appear green)

Working with objects

This section discusses common tasks you complete when working with objects in the Designer. With these tasks, you use various parts of the Designer—the toolbar, tool palette, workspace, and local object library.

Tasks in this section include:

- Creating new reusable objects
- Changing object names
- Viewing and changing object properties
- Creating descriptions
- Creating annotations
- Saving and deleting objects
- Searching for objects

Creating new reusable objects

You can create reusable objects from the object library or by using the tool palette. After you create an object, you can work with the object, editing its definition and adding calls to other objects.

➢ To create a reusable object (in the object library)

1. Open the object library by choosing **Tools** > **Object Library**.
2. Click the tab corresponding to the object type.
3. Right-click anywhere except on existing objects and choose **New**.
4. Right-click the new object and select **Properties**. Enter options such as name and description to define the object.

➢ To create a reusable object (using the tool palette)

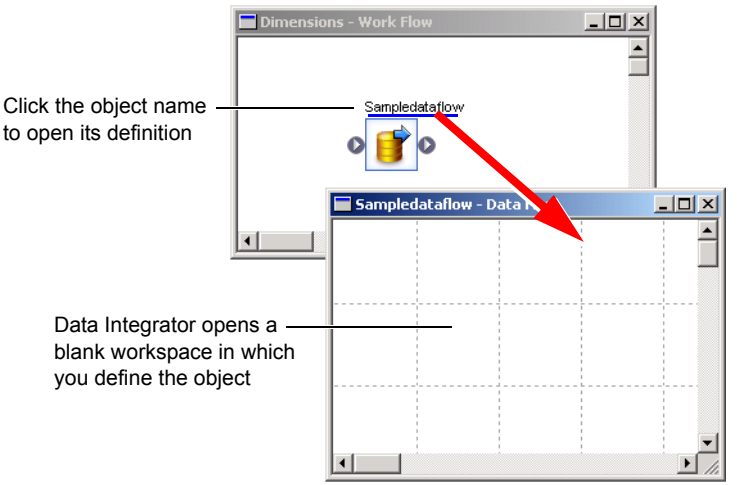
1. In the tool palette, left-click the icon for the object you want to create.
2. Move the cursor to the workspace and left-click again.

   The object icon appears in the workspace where you have clicked.

➢ To open an object's definition

You can open an object's definition in one of two ways:

- From the workspace, click the object name.



- From the project area, click the object.

You define an object using other objects. For example, if you click the name of a batch data flow, a new workspace opens for you to assemble sources, targets, and transforms that make up the actual flow.

➢ To add an existing object (create a new call to an existing object)

1. Open the object library by choosing **Tools** > **Object Library**.
2. Click the tab corresponding to any object type.
3. Select an object.
4. Drag the object to the workspace.

NOTE: Objects dragged into the workspace must obey the hierarchy logic explained in "Object hierarchy" on page 20. For example, you can drag a data flow into a job, but you cannot drag a work flow into a data flow.

Changing object names

You can change the name of an object from the workspace or the object library. You can also create a copy of an existing object.

NOTE: You cannot change the names of built-in objects.

➢ To change the name of an object in the workspace

1. Click to select the object in the workspace.
2. Right-click and choose **Edit Name**.
3. Edit the text in the name text box.
4. Click outside the text box or press Enter to save the new name.

➢ To change the name of an object in the object library

1. Select the object in the object library.
2. Right-click and choose **Properties**.
3. Edit the text in the first text box.
4. Click **OK**.

➢ To copy an object

1. Select the object in the object library.
2. Right-click and choose **Replicate**.

   Data Integrator makes a copy of the top-level object (but not of objects that it calls) and gives it a new name, which you can edit.

Viewing and changing object properties

You can view (and, in some cases, change) an object's properties through its property page.

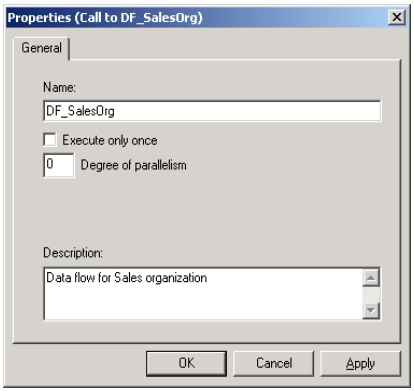
➢ To view, change, and add object properties

1. Select the object in the object library.
2. Right-click and choose **Properties**. The **General** tab of the Properties window opens.
3. Complete the property sheets. The property sheets vary by object type, but General, Attributes and Class Attributes are the most common and are described in the following sections.
4. When finished, click **OK** to save changes you made to the object properties and to close the window.

   Alternatively, click **Apply** to save changes without closing the window.

General tab

The **General** tab contains two main object properties: name and description.



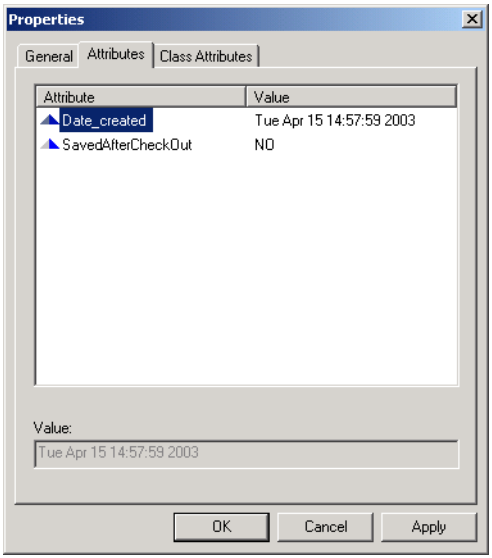
From the General tab, you can change the object name as well as enter or edit the object description. You can add object descriptions to single-use objects as well as to reusable objects. Note that you can toggle object descriptions on and off by right-clicking any object in the workspace and selecting/deslecting **View Enabled Descriptions**.

Depending on the object, there can be other properties, such as:

- **Execute only once** — See "Creating and defining data flows" in Chapter 7, "Data Flows" for more information.
- **Recover as a unit** — See "Marking recovery units" in Chapter 18, "Recovery Mechanisms" for more information.
- **Degree of parallelism** — See the *Data Integrator Performance Optimization Guide* for more information on this advanced feature.

Attributes tab

The **Attributes** tab allows you to assign values to the attributes of the current object.

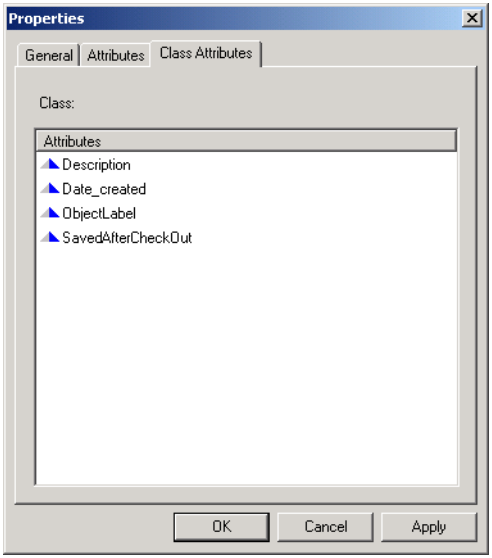


To assign a value to an attribute, select the attribute and enter the value in the **Value** box at the bottom of the window.

Some attribute values are set by Data Integrator and cannot be edited. When you select an attribute with a system-defined value, the **Value** field is unavailable.

Class Attributes tab

The **Class Attributes** tab shows the attributes available for the type of object selected. For example, all data flow objects have the same class attributes.

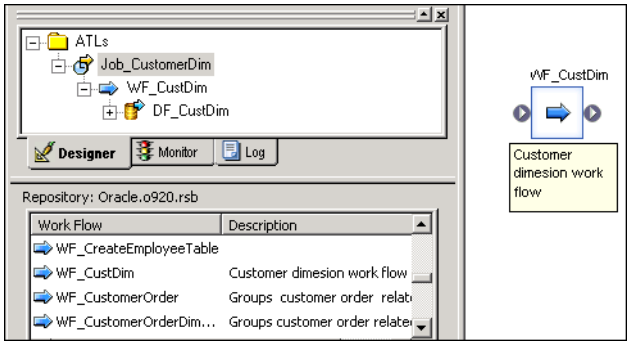


To create a new attribute for a class of objects, right-click in the attribute list and select **Add**. The new attribute is now available for all of the objects of this class.

To delete an attribute, select it then right-click and choose **Delete**. You cannot delete the class attributes predefined by Data Integrator.

Creating descriptions

Use descriptions to document objects. You can see descriptions on workspace diagrams. Therefore, descriptions are a convenient way to add comments to workspace objects.

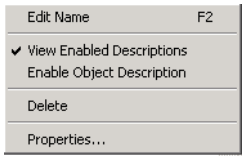


A description is associated with a particular object. When you import or export that repository object (for example, when migrating between development, test, and production environments), you also import or export its description.

The Designer determines when to show object descriptions based on a system-level setting and an object-level setting. Both settings must be activated to view the description for a particular object.

The system-level setting is unique to your setup. The system-level setting is disabled by default. To activate that system-level setting, select **View** > **Enabled Descriptions**, or click the **View Enabled Descriptions** button on the toolbar.

The object-level setting is saved with the object in the repository. The object-level setting is also disabled by default unless you add or edit a description from the workspace. To activate the object-level setting, right-click the object and select **Enable object description**.



An ellipses after the text in a description indicates that there is more text. To see all the text, resize the description by clicking and dragging it. When you move an object, its description moves as well. To see which object is associated with which selected description, view the object's name in the status bar.

➢ To add a description to an object

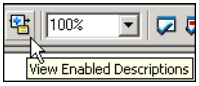
1. In the project area or object library, right-click an object and select **Properties**.
2. Enter your comments in the **Description** text box.
3. Click **OK**.

   The description for the object displays in the object library.

➢ To display a description in the workspace

1. In the project area, select an existing object (such as a job) that contains an object to which you have added a description (such as a work flow).
2. From the **View** menu, select **Enabled Descriptions**.

Alternately, you can select the **View Enabled Descriptions** button on the toolbar.



3. Right-click the work flow and select **Enable Object Description**.

   The description displays in the workspace under the object.

➢ To add a description to an object from the workspace

1. From the **View** menu, select **Enabled Descriptions**.
2. In the workspace, right-click an object and select **Properties**.
3. In the Properties window, enter text in the **Description** box.
4. Click **OK**.

   The description displays automatically in the workspace (and the object's **Enable Object Description** option is selected).

➢ To hide a particular object's description

1. In the workspace diagram, right-click an object.

   Alternately, you can select multiple objects by:

   - Pressing and holding the Control key while selecting objects in the workspace diagram, then right-clicking one of the selected objects.
   - Dragging a selection box around all the objects you want to select, then right-clicking one of the selected objects.

2. In the pop-up menu, deselect **Enable Object Description.**

   The description for the object selected is hidden, even if the **View Enabled Descriptions** option is checked, because the object-level switch overrides the system-level switch.

➢ To edit object descriptions

1. In the workspace, double-click an object description.
2. Enter, cut, copy, or paste text into the description.
3. In the **Project** menu, select **Save**.

   Alternately, you can right-click any object and select **Properties** to open the object's Properties window and add or edit its description.

NOTE: If you attempt to edit the description of a reusable object, Data Integrator alerts you that the description will be updated for every occurrence of the object, across all jobs. You can select the **Do not show me this again** check box to avoid this alert. However, after deactivating the alert, you can only reactivate the alert by calling Technical Support.

Creating annotations

Annotations describe a flow, part of a flow, or a diagram in a workspace. An annotation is associated with the job, work flow, or data flow where it appears. When you import or export that job, work flow, or data flow, you import or export associated annotations.

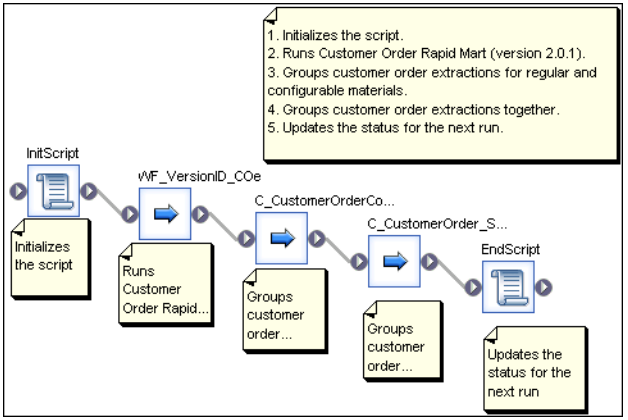
➢ To annotate a workspace diagram

1. Open the workspace diagram you want to annotate.

   You can use annotations to describe any workspace such as a job, work flow, data flow, catch, conditional, or while loop.
2. In the tool palette, click the annotation icon.
3. Click a location in the workspace to place the annotation.

   An annotation appears on the diagram.

You can add, edit, and delete text directly on the annotation. In addition, you can resize and move the annotation by clicking and dragging. You can add any number of annotations to a diagram.



You cannot hide annotations that you have added to the workspace. However, you can move them out of the way or delete them.

➢ To delete an annotation

1. Right-click an annotation.
2. Select **Delete**.

   Alternately, you can select an annotation and press the **Delete** key.

Saving and deleting objects

"Saving" an object in Data Integrator means storing the language that describes the object to the repository. You can save reusable objects; single-use objects are saved only as part of the definition of the reusable object that calls them.

You can choose to save changes to the reusable object currently open in the workspace. When you save the object, the object properties, the definitions of any single-use objects it calls, and any calls to other reusable objects are recorded in the repository. The content of the included reusable objects is not saved; only the call is saved.

Data Integrator stores the description even if the object is not complete or contains an error (does not validate).

➢ To save changes to a single reusable object

1. Open the project in which your object is included.
2. Choose **Project > Save**.

   This command saves all objects open in the workspace.

Repeat these steps for other individual objects you want to save.

➢ To save all changed objects in the repository

1. Choose **Project > Save All**.

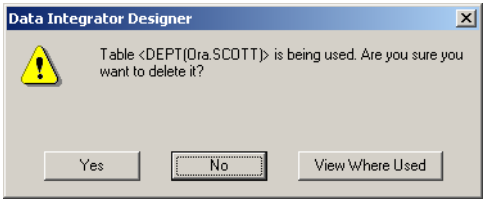
   Data Integrator lists the reusable objects that have been changed since the last save operation.

2. Click **OK**.

NOTE: Saving a reusable object saves any single-use object included in it.

➢ To delete an object definition from the repository

1. In the object library, select the object.
2. Right-click and choose **Delete**.
   - If you attempt to delete an object that is being used, Data Integrator provides a warning message and the option of using the **View Where Used** feature.



For more information, see "Using View Where Used" on page 354.

- If you select **Yes**, Data Integrator marks all calls to the object with a red "deleted" icon to indicate that the calls are invalid. You must remove or replace these calls to produce an executable job.

NOTE: Built-in objects such as transforms cannot be deleted from the object library.

➢ To delete an object call

1. Open the object that contains the call you want to delete.
2. Right-click the object call and choose **Delete**.

If you delete a reusable object from the workspace or from the project area, only the object call is deleted. The object definition remains in the object library.

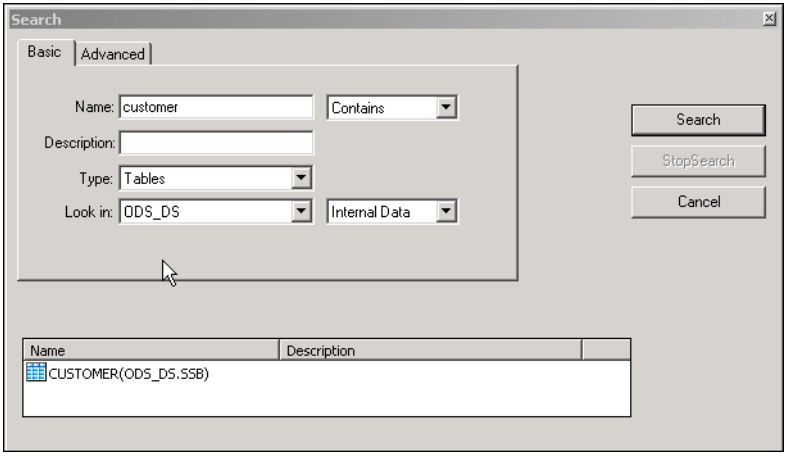
Searching for objects

From within the object library, you can search for objects defined in the repository or objects available through a datastore.

➢ To search for an object

1. Right-click in the object library and choose **Search**.

   Data Integrator displays the Search window.



2. Enter the appropriate values for the search.

   Options available in the Search window are described in detail following this procedure.

3. Click **Search**.

   The objects matching your entries are listed in the window. From the search results window you can use the context menu to:

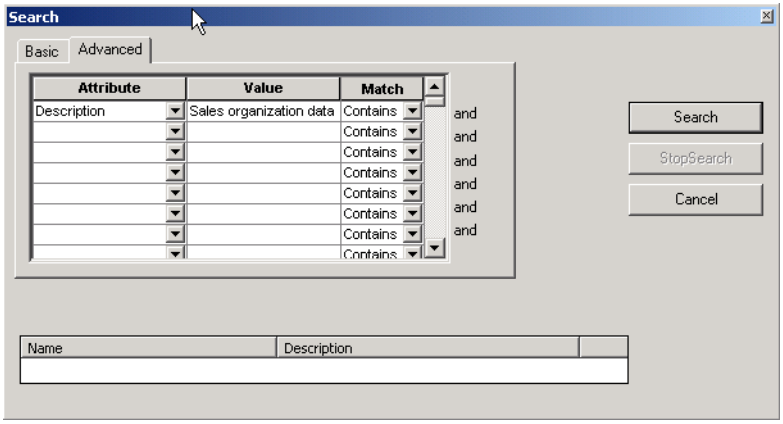
   - **Open** an item
   - **View** the attributes (**Properties**)
   - **Import** external tables as repository metadata

   You can also drag objects from the search results window and drop them in the desired location.

The **Basic** tab in the Search window provides you with the following options:

Option	Description
Name	The object name to find. If you are searching in the repository, the name is not case sensitive. If you are searching in a datastore and the name is case sensitive in that datastore, enter the name as it appears in the database or application and use double quotation marks (") around the name to preserve the case. You can designate whether the information to be located **Contains** the specified name or **Equals** the specified name using the drop-down box next to the **Name** field.
Description	The object description to find. Objects imported into the Data Integrator repository have a description from their source. By default, objects you create in the Designer have no description unless you add a one. The search returns objects whose description attribute contains the value entered.
Type	The type of object to find. When searching the repository, choose from Tables, Files, Data flows, Work flows, Jobs, Hierarchies, IDOCs, and Domains. When searching a datastore or application, choose from object types available through that datastore.
Look in	Where to search. Choose from the repository or a specific datastore. When you designate a datastore, you can also choose to search the imported data (**Internal Data**) or the entire datastore (**External Data**).

The Search window also includes an **Advanced** tab. From the **Advanced** tab, you can choose to search for objects based on their Data Integrator attribute values. You can search by attribute values only when searching in the repository.



The **Advanced** tab provides the following options:

Option	Description
Attribute	The object attribute in which to search. The attributes are listed for the object type specified on the **Basic** tab.
Value	The attribute value to find.
Match	The type of search performed. Select **Contains** to search for any attribute that contains the value specified. Select **Equals** to search for any attribute that contains only the value specified.

General and environment options

To open the Options window, select **Tools** > **Options**. The window displays option groups for Designer, Data, and Job Server options.

Expand the options by clicking the plus icon. As you select each option group or option, a description appears on the right.

SAP and Liveload options also appear if you have installed these licensed extensions. See the *Data Integrator Supplement for SAP* and *Data Integrator LiveLoad User's Guide* for more information about these options.

The standard options include:

- Designer — Environment
- Designer — General
- Designer — Graphics
- Designer — Central Repository Connections
- Data — General
- Job Server — Environment
- Job Server — General

Designer — Environment

Default Administrator for Metadata Reporting:

Administrator — Select the Administrator the metadata reporting tool uses. An Administrator is defined by host name and port.

Default Job Server:

If a repository is associated with several Job Servers, one Job Server must be defined as the default Job Server to use at login.

Current — Displays the current value of the default Job Server.

New — Allows you to specify a new value for the default Job Server from a drop-down list of Job Servers associated with this repository. Changes are effective immediately.

NOTE: Job-specific options and path names specified in Designer refer to the current default Job Server. If you change the default Job Server, modify these options and path names.

Designer Communication Ports:

Allow Designer to set the port for Job Server communication — If checked, Designer automatically sets an available port to receive messages from the current Job Server. The default is checked. Uncheck to specify a listening port or port range.

Specify port range — Only activated when you deselect the previous control. Allows you to specify a range of ports from which the Designer can choose a listening port.

You may choose to constrain the port used for communication between Designer and Job Server when the two components are separated by a firewall.

Enter port numbers in the **From port** and **To port** text boxes. To specify a specific listening port, enter the same port number in both the **From port** and **To port** text boxes. Changes will not take effect until you restart Data Integrator.

Interactive Debugger — Allows you to set a communication port for the Designer to communicate with a Job Server while running in Debug mode. For more information, see "Changing the interactive debugger port" on page 388.

Server group for local repository — If the local repository that you logged in to when you opened the Designer is associated with a server group, the name of the server group appears.

Designer — General

View data sampling size (rows) — Controls the sample size used to display the data in sources and targets in open data flows in the workspace. View data by clicking the magnifying glass icon on source and target objects. For more information, see "Design and Debug using View Data" on page 329.

Number of characters in workspace icon name — Controls the length of the object names displayed in the workspace. Object names are allowed to exceed this number, but the Designer only displays the number entered here. The default is 17 characters.

Maximum schema tree elements to auto expand — The number of elements displayed in the schema tree. Element names are not allowed to exceed this number. Enter a number for the **Input schema** and the **Output schema**. The default is 100.

Default parameters to variables of the same name — When you declare a variable at the work-flow level, Data Integrator automatically passes the value as a parameter with the same name to a data flow called by a work flow.

Automatically import domains — Select this check box to automatically import domains when importing a table that references a domain.

Perform complete validation before job execution — If checked, Data Integrator performs a complete job validation before running a job. The default is unchecked. If you keep this default setting, you should validate your design manually before job execution.

Open monitor on job execution — Affects the behavior of the Designer when you execute a job. With this option enabled, the Designer switches the workspace to the monitor view during job execution; otherwise, the workspace remains as is. The default is on.

Calculate column mapping while saving data flow — Calculates information about target tables and columns and the sources used to populate them. This information is stored in the AL_COLMAP table (ALVW_MAPPING view) automatically upon saving a data flow. This information can be seen when you generate metadata reports. If you select this option, be sure to validate your entire job before saving it. This functionality is highly sensitive to errors and will skip data flows with validation problems. For more information, see, "Tools" on page 454.

Show dialog when job is completed — Allows you to choose if you want to see an alert or just read the trace messages.

Show tabs in workspace — Allows you to decide if you want to use the tabs at the bottom of the workspace to navigate.

Designer — Graphics

Choose and preview stylistic elements to customize your workspaces. Using these options, you can easily distinguish your job/work flow design workspace from your data flow design workspace.

- **Workspace flow type** — Switch between the two workspace flow types (Job/Work Flow and Data Flow) to view default settings. Modify settings for each type using the remaining options.
- **Line Type** — Choose a style for object connector lines.
- **Line Thickness** — Set the connector line thickness.
- **Background style** — Choose a plain or tiled background pattern for the selected flow type.
- **Color scheme** — Set the background color to blue, gray, or white.
- **Use navigation watermark** — Add a watermark graphic to the background of the flow type selected. Note that this option is only available with a plain background style.

Designer — Central Repository Connections

Displays the central repository connections and the active central repository. To activate a central repository, right-click one of the central repository connections listed and select **Activate**.

Reactivate automatically — Select if you want the active central repository to be reactivated whenever you log in to Data Integrator using the current local repository.

Data — General

Century Change Year — Indicates how Data Integrator interprets the century for two-digit years. Two-digit years greater than or equal to this value are interpreted as 19##. Two-digit years less than this value are interpreted as 20##. The default value is 15.

For example, if the **Century Change Year** is set to 15:

Two-digit year	Interpreted as
99	1999
16	1916
15	1915
14	2014

DB2 client version — Indicates the version of DB2 client being used.

Convert blanks to nulls for Oracle bulk loader — Converts blanks to NULL values when loading data using the Oracle bulk loader utility and:

- the column is not part of the primary key
- the column is nullable

Job Server — Environment

Maximum number of engine processes — Sets a limit on the number of engine processes that this Job Server can have running concurrently.

Job Server — General

Use this window to reset Job Server options (see "Changing Job Server options" on page 330) or with guidance from Business Objects Customer Support. For contact information, visit Business Objects Customer Support Online.

4 Projects and Jobs

Project and job objects represent the top two levels of organization for the application flows you create using the Designer. This chapter contains the following sections:

- Projects
- Jobs

Projects

A *project* is a reusable object that allows you to group jobs. A project is the highest level of organization offered by Data Integrator. Opening a project makes one group of objects easily accessible in the user interface.

You can use a project to group jobs that have schedules that depend on one another or that you want to monitor together.

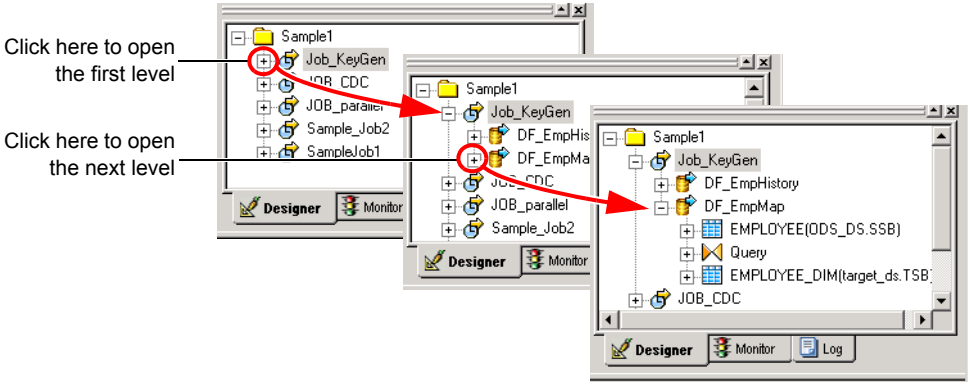
Projects have common characteristics:

- Projects are listed in the object library.
- Only one project can be open at a time.
- Projects cannot be shared among multiple users.

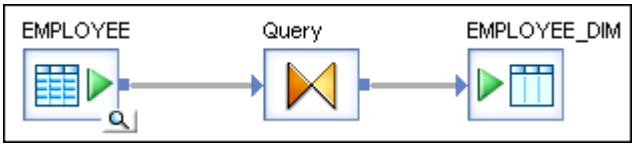
Objects that make up a project

The objects in a project appear hierarchically in the project area. If a plus sign (+) appears next to an object, expand it to view the lower-level objects contained in the object. Data Integrator shows you the contents as both names in the project area hierarchy and icons in the workspace.

In the following example, the Job_KeyGen job contains two data flows, and the DF_EmpMap data flow contains multiple objects.



Each item selected in the project area also displays in the workspace:



Creating new projects

➢ To create a new project:

1. Choose **Project** > **New** > **Project**.
2. Enter the name of your new project.

   The name can include alphanumeric characters and underscores (_). It cannot contain blank spaces.
3. Click **Create**.

The new project appears in the project area. As you add jobs and other lower-level objects to the project, they also appear in the project area.

Opening existing projects

➢ To open an existing project

1. Choose **Project** > **Open**.
2. Select the name of an existing project from the list.
3. Click **Open**.

NOTE: If another project was already open, Data Integrator closes that project and opens the new one.

Saving projects

➢ To save all changes to a project

1. Choose **Project > Save All**.

   Data Integrator lists the jobs, work flows, and data flows that have changed since the last save.

2. Click **OK**.

Jobs

A *job* is the only object you can execute. You can manually execute and test jobs in development. In production, you can schedule batch jobs and set up real-time jobs as services that execute a process when Data Integrator receives a message request.

A job is made up of steps you want executed together. Each step is represented by an object icon that you place in the workspace to create a *job diagram*. A job diagram is made up of two or more objects connected together. You can include any of the following objects in a job definition:

- Data flows
  - Sources
  - Targets
  - Transforms
- Work flows
  - Scripts
  - Conditionals
  - While Loops
  - Try/catch blocks

If a job becomes complex, organize its content into individual work flows, then create a single job that calls those work flows. For more information on work flows, see Chapter 8, "Work Flows".

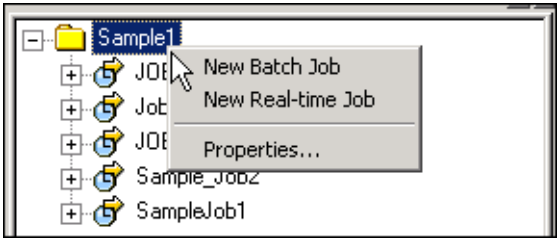
Real-time jobs use the same components as batch jobs. You can add work flows and data flows to both batch and real-time jobs. When you drag a work flow or data flow icon into a job, you are telling Data Integrator to validate these objects according the requirements of the job type (either batch or real-time).

There are some restrictions regarding the use of some Data Integrator features with real-time jobs. For more information, see Chapter 10, "Real-time jobs".

Creating jobs

➢ To create a job in the project area

1. In the project area, select the project name.
2. Right-click and choose **New Batch Job** or **Real Time Job**.



3. Edit the name.

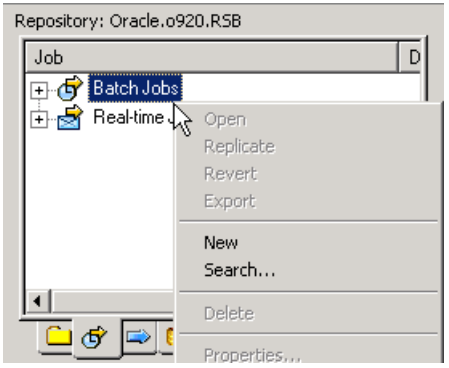
   The name can include alphanumeric characters and underscores (_). It cannot contain blank spaces.

   Data Integrator opens a new workspace for you to define the job.

➢ To create a job in the object library

1. Go to the **Jobs** tab.

2. Right-click **Batch Jobs** or **Real Time Jobs** and choose **New.**



3. A new job with a default name appears.
4. Right-click and select Properties to change the object's name and add a description.

   The name can include alphanumeric characters and underscores (_). It cannot contain blank spaces.
5. To add the job to the open project, drag it into the project area.

Naming conventions for objects in jobs

We recommend that you follow consistent naming conventions to facilitate object identification across all systems in your enterprise. This allows you to more easily work with metadata across all applications such as:

- Data-modeling applications
- ETL applications
- Reporting applications
- Adapter software development kits

Examples of conventions recommended for use with jobs and other objects are shown in the following table.

Prefix	Suffix	Object	Example
DF_		Data flow	DF_Currency
EDF_	_NoOutput_Input	Embedded data flow	EDF_ERP_NoOutput_Input
EDF_	_Input	Embedded data flow	EDF_Example_Input
EDF_	_Output	Embedded data flow	EDF_Example_Output
EDF_	_NoInput_Output	Embedded data flow	EDF_ERP_NoInput_Output
RTJob_		Real-time job	RTJob_OrderStatus
WF_		Work flow	WF_SalesOrg
JOB_		Job	JOB_SalesOrg
	_DS	Datastore	ORA_DS
DP_		Datastore profile	DP_DB2_production
SP_		System profile	SP_ORA_test
	_Memory_DS	Memory datastore	Catalog_Memory_DS
PROC_		Stored procedure	PROC_SalesStatus

Although Data Integrator Designer is a graphical user interface with icons representing objects in its windows, other interfaces might require you to identify object types by the text alone. By using a prefix or suffix, you can more easily identify your object's type.

In addition to prefixes and suffixes, you might want to provide standardized names for objects that identify a specific action across all object types. For example: DF_OrderStatus, RTJob_OrderStatus.

In addition to prefixes and suffixes, naming conventions can also include path name identifiers. For example, the stored procedure naming convention can look like either of the following:

<datastore>.<owner>.<PROC_Name>
<datastore>.<owner>.<package>.<PROC_Name>

5 Datastores

This chapter contains the following sections:

What are datastores?

Datastores represent connections between Data Integrator and databases or applications, directly or through adapters. Datastore connections allow Data Integrator to access metadata from a database or application and read from or write to that database or application.

Data Integrator datastores can connect to:

- Databases and mainframe file systems. See "Custom datastores" on page 80.
- Applications that have pre-packaged or user-written Data Integrator adapters. See "Adapter datastores" on page 106.
- J.D. Edwards One World and J.D. Edwards World, Oracle Applications, PeopleSoft, SAP R/3, SAP BW, and Siebel Applications. See the appropriate *Data Integrator Supplement*.

NOTE: Data Integrator reads and writes data stored in flat files through flat file formats as described in "File Formats" on page 117. Data Integrator reads and writes data stored in XML documents through DTDs and XML Schemas. See "Formatting XML documents" on page 224.

The specific information that a datastore contains depends on the connection. When your database or application changes, you must make corresponding changes in the datastore information in Data Integrator—Data Integrator does not automatically detect the new information.

NOTE: Objects deleted from a datastore connection are identified in the project area and workspace by a red "deleted" icon  . This visual flag allows you to find and update data flows affected by datastore changes.

You can configure your datastore to have multiple profiles. A datastore profile specifies connection information. By using datastore profiles, you can specify connection information at run-time. You group datastore profiles into system profiles. When running a job, you select a system profile, and thus, the datastore profile you want to use for the source and target objects associated with a datastore.

For example, you might have identical databases for different departments in your firm and want to run the same job, such as a job that tracks compensation for both departments. Because the source information is stored on different databases, you need to use a different database connection name. Rather than creating a datastore for each source, you can create a single datastore with multiple datastore profiles. You would create a system profile that contains each datastore profile and select the appropriate system profile at run-time.

Note that LiveLoad-enabled datastores cannot have multiple profiles. In addition, multiple profiles are not available for the following types of datastores:

- Adapter
- DETAIL
- J.D. Edwards

Custom datastores

Custom datastores can represent Data Integrator connections with:

- IBM DB2, Informix, Microsoft SQL Server, Oracle, Sybase, and Teradata databases (using native connections)
- Other databases (through ODBC)
- A Data Integrator repository using a memory datastore
- DB2, IMS, VSAM, IDMS, ADABAS, and flat file systems on IBM MVS, NT, and OS/400 platforms through BusinessObjects Data Integrator Mainframe Interfaces. These interfaces are available through Striva DETAIL licenses.

This section discusses:

- Mainframe Interfaces (DETAIL)
- Defining a custom datastore
- Browsing metadata through a custom datastore
- Importing metadata through a custom datastore
- Memory datastores

Mainframe Interfaces (DETAIL)

DETAIL datastores provide connections to relational databases and non-relational data on mainframe systems. DETAIL is certified to work with IBM MVS and AS/400 mainframe systems.

Choose from two types of DETAIL datastores to read data from data sources. You cannot write data using a DETAIL datastore.

Datastore Editor's options for DETAIL

Client Driver (connection)	Data Source	Client Operating System	Source Operating System
DETAIL_DB2[a]	DB2	NT, AIX, HPUX, Solaris	Not applicable.[b]
DETAIL_NRDB	IMS VSAM ADABAS IDMS Flat files[c]	NT, AIX, HPUX, Solaris	Not applicable.[b]

[a] Only DETAIL_DB2 datastores can contain functions.

[b] Connection is made using the DETAIL driver and the data source.

[c] DETAIL_NRDB reads database sources as flat files.

The DETAIL_DB2 datastore can be used as an alternative to access through IBM iSeries Access Driver (native) or IBM DB2 Connect (via ODBC). DETAIL connections are read-only, however.

Comparison of DB2 Client Drivers

Client Driver (connection)	Datastore Editor option	Data Source	Client Operating System	Source Operating System
IBM iSeries Access (formerly known as Client Access)	DB2	DB2	Windows only	OS/400
DB2 Connect	ODBC	DB2	Any	OS/400, MVS
DETAIL_DB2	DETAIL_DB2	DB2	NT, AIX, HPUX, Solaris	OS/400, MVS (tested)

Before using a DETAIL datastore, you must:

- Define a node in DETAIL. A node identifies the computer where the database and the DETAIL Listener are located.

- Make sure that the DETAIL Listener is running. The DETAIL Listener is a background process that facilitates communication between Data Integrator and the mainframe computer.
- Define a data map in DETAIL if you are using an NRDB datastore.
- Define a browser profile in DETAIL to enable metadata browsing.

Refer to the DETAIL documentation on the BusinessObjects Data Integrator Mainframe Interfaces CD for installation and setup instructions.

Defining a custom datastore

You need to define at least one datastore for each database or mainframe file system with which you are exchanging data.

To define a datastore, you must have appropriate access privileges to the database or file system that the datastore describes. If you do not have access, ask your database administrator to create an account for you.

NOTE: To allow Data Integrator to use parameterized SQL when reading or writing to DB2 databases, authorize the user (of the datastore/database) to create, execute and drop stored procedures. If a user is not authorized to create, execute and drop stored procedures jobs will still run. However they will produce a warning message and will run less efficiently.

➢ To define a custom datastore

1. In the **Datastores** tab of the object library, right-click and select **New**.
2. Enter the name of the new datastore in the **Name** field.

   The name can contain any alphabetical or numeric characters or underscores (_). It cannot contain spaces.

3. Select the **Application type**.

   Choose **Custom**.

4. Select the **Database type**.

   Choose from DB2, DETAIL _DB2, DETAIL_NRDB, DETAIL_NRDB2, Informix, Memory, Microsoft SQL Server, ODBC, Oracle, Sybase, or Teradata.

5. Enter the appropriate information for the selected database type. See "Datastore" on page 34 of the *Data Integrator Reference Guide* for a description of the required fields.

6. Click **OK.**

7. See "Ways of importing metadata" on page 92 for the procedures you will use to import metadata from the connected database or application.

Changing a datastore definition

Like all Data Integrator objects, datastores are defined by both options and properties:

- *Options* control the operation of objects. For example, the name of the database to connect to is a datastore option.
- *Properties* document the object. For example, the name of the datastore and the date on which it was created are datastore properties. Properties are merely descriptive of the object and do not affect its operation.

➢ To change datastore options

1. Go to the **Datastores** tab in the object library.
2. Right-click the datastore name and choose **Edit**.

   The datastore editor appears in the workspace.

3. Change the datastore options.

   The options take effect immediately.

The individual options available for each datastore type are described in "Custom application datastores" on page 35 of the *Data Integrator Reference Guide*.

➢ To change datastore properties

1. Go to the datastore tab in the object library.
2. Right-click the datastore name and select **Properties**.

   The Properties window opens.
3. Change the datastore properties.

   The individual properties available for a datastore are described in "Datastore" on page 34 of the *Data Integrator Reference Guide*.
4. Click **OK**.

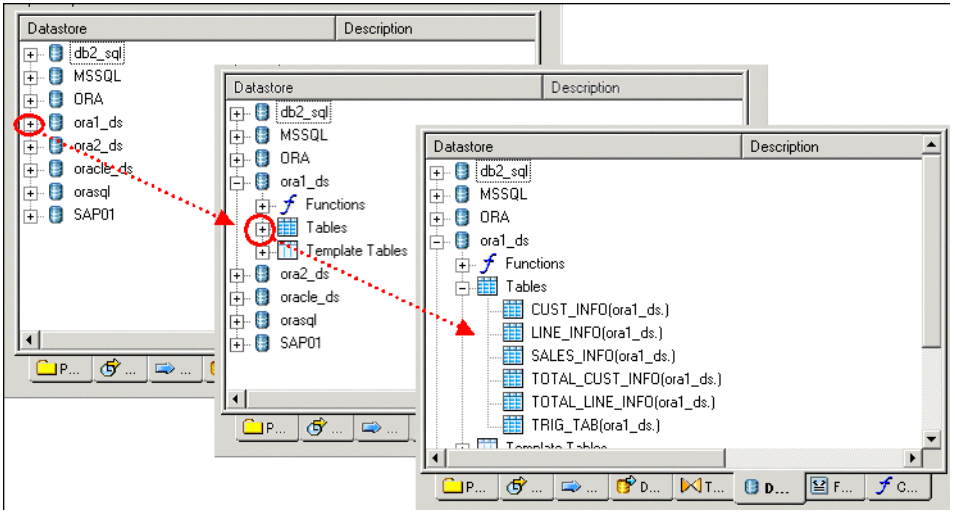
Browsing metadata through a custom datastore

Data Integrator stores metadata information for all imported objects in a datastore. You can use Data Integrator to view metadata for imported or non-imported objects and to check whether the metadata has changed for objects already imported.

➢ To view imported objects

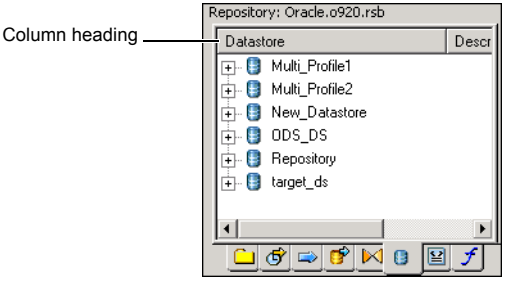
1. Go to the **Datastores** tab in the object library.
2. Click the plus sign (+) next to the datastore name to view the object types in the datastore. For example, custom datastores have functions, tables, and template tables.
3. Click the plus sign (+) next to an object type to view the objects of that type imported from the datastore.

For example, click the plus sign (+) next to tables to view the imported tables.



➢ To sort the list of objects

Click the column heading to sort the objects in each grouping and the groupings in each datastore alphabetically. Click again to sort in reverse-alphabetical order.



➢ To view datastore metadata

1. Select the **Datastores** tab in the object library.

2. Choose a datastore, right-click, and select **Open**. (Alternatively, you can double-click the datastore icon.)

   Data Integrator opens the datastore explorer in the workspace. The datastore explorer lists the tables in the datastore. You can view tables in the external database or tables in the internal repository. You can also search through them. For more information about the search feature, see "To import by searching" on page 97.

3. Select **External metadata** to view tables in the external database.

   If you select one or more tables, you can right-click for further options.

Command	Description
Open[a]	Opens the editor for the table metadata.
Import	Imports (or re-imports) metadata from the database into the repository.
Reconcile	Checks for differences between metadata in the database and metadata in the repository.

a. Only available if you select one table.

4. Select **Repository metadata** to view imported tables.

   If you select one or more tables, you can right-click for further options.

Command	Description
Open[a]	Opens the editor for the table metadata.
Reconcile	Checks for differences between metadata in the repository and metadata in the database.
Reimport	Reimports metadata from the database into the repository.
Delete	Deletes the table or tables from the repository.
Properties[a]	Shows the properties of the selected table.
View Data	Opens the View Data window which allows you to see the data currently in the table.

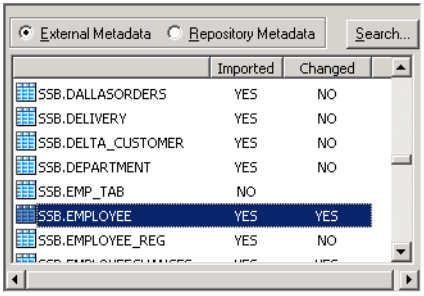
a. Only available if you select one table.

➢ To determine if a table schema is imported or has changed since it was imported

1. In the browser window showing the list of repository tables, select **External Metadata**.
2. Choose the table or tables you want to check for changes.
3. Right-click and choose **Reconcile**.

   The **Changed** column displays YES to indicate that the database tables differ from the metadata imported into Data Integrator. To use the most recent metadata from Data Integrator, reimport the table.

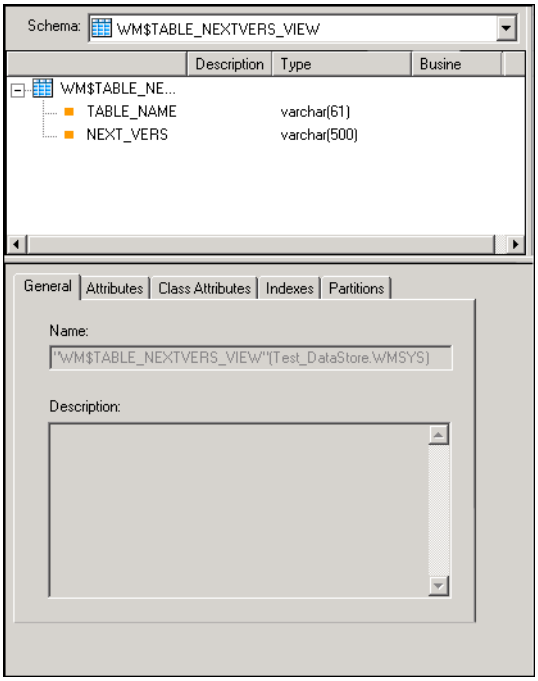
   The **Imported** column displays YES to indicate that the table has been imported into the repository.



➢ To browse the metadata for an external table

1. In the browser window showing the list of external tables, select the table you want to view.
2. Right-click and choose **Open**.

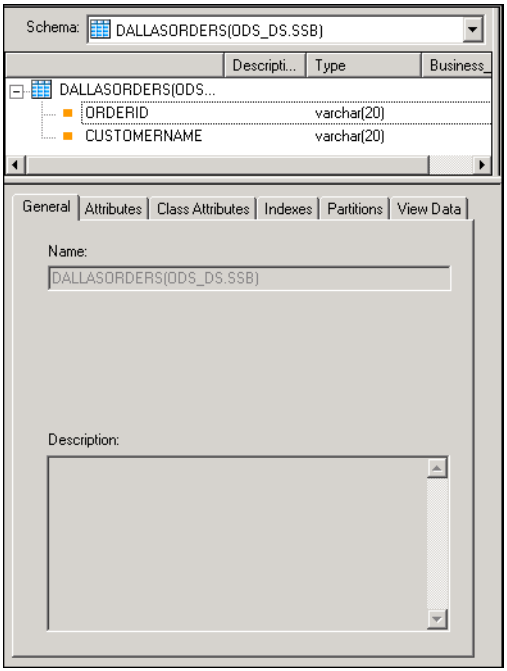
A table editor appears in the workspace and displays the schema and attributes of the table.



➢ To view the metadata for an imported table

1. Select the table name in the list of imported tables.
2. Right-click and select **Open**.

A table editor appears in the workspace and displays the schema and attributes of the table.

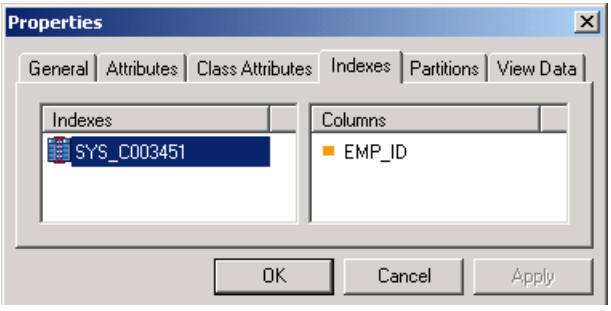


➢ To view secondary index information for tables

Secondary index information can help you understand the schema of an imported table.

1. From the datastores tab in the Designer, right-click a table to open the shortcut menu.
2. From the shortcut menu, click **Properties** to open the Properties window.
3. In the Properties window, click the **Indexes** tab. The left portion of the window displays the Index list.

4. Click an index to see the contents.



Importing metadata through a custom datastore

For custom datastores, you can import metadata about tables and functions.

This section discusses:

- Imported table information
- Imported stored function and procedure information
- Ways of importing metadata

Imported table information

Data Integrator determines and stores a specific set of metadata information for tables. After importing metadata, you can edit column names, descriptions, and data types. The edits are propagated to all objects that call these objects.

Metadata	Description
Table name	The name of the table as it appears in the database.
Table description	The description of the table.
Column name	The name of the table column.
Column description	The description of the column.
Column data type	The data type for each column. If a column is defined as an unsupported data type, Data Integrator converts the data type to one that is supported. In some cases, if Data Integrator cannot convert the data type, it ignores the column entirely.
Primary key column	The column(s) that comprise the primary key for the table. After a table has been added to a data flow diagram, these columns are indicated in the column list by a key icon next to the column name.
Table attribute	Information Data Integrator records about the table such as the date created and date modified if these values are available.
Owner name	Name of the table owner.

Imported stored function and procedure information

Data Integrator can import stored procedures from DB2, MS SQL Server, Oracle, and Sybase databases. You can also import stored functions and packages from Oracle. You can use these functions and procedures in the extraction specifications you give Data Integrator.

Information that is imported for functions includes:

- Function parameters
- Return type
- Name, owner

Imported functions and procedures appear on the **Datastores** tab of the object library. Functions and procedures appear in the **Function** branch of each datastore tree.

You can configure imported functions and procedures through the function wizard and the smart editor in a category identified by the datastore name. For more information, see "About procedures" on page 491 of the *Data Integrator Reference Guide*.

Ways of importing metadata

This section discusses methods you can use to import metadata:

- To import by browsing
- To import by name
- To import by searching

➢ To import by browsing

NOTE: Functions cannot be imported by browsing.

1. Open the object library.
2. Go to the **Datastores** tab.
3. Select the datastore you want to use.
4. Right-click and choose **Open**.

   The items available to import through the datastore appear in the workspace.

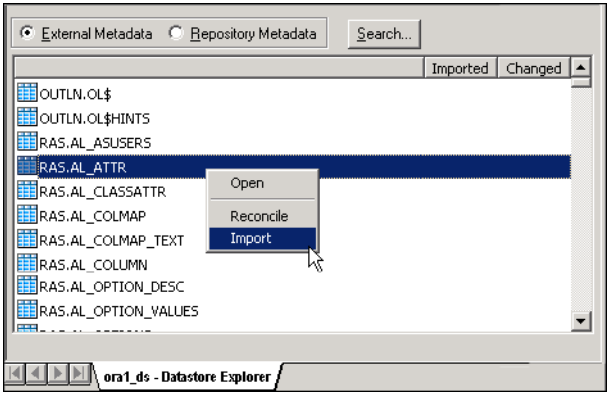
In some environments, the tables are organized and displayed as a tree structure. If this is true, there is a plus sign (+) to the left of the name. Click the plus sign to navigate the structure.

The workspace contains columns that indicate whether the table has already been imported into Data Integrator (**Imported**) and if the table schema has changed since it was imported (**Changed**). To verify whether the repository contains the most recent metadata for an object, right-click the object and choose **Reconcile**.

5. Select the items for which you want to import metadata.

   For example, to import a table, you must select a table rather than a folder that contains tables.

6. Right-click and choose **Import**.



7. In the object library, go to the **Datastores** tab to display the list of imported objects.

➢ To import by name

1. Open the object library.
2. Click the **Datastores** tab.
3. Select the datastore you want to use.

4. Right-click and choose **Import By Name**.
5. In the Import By Name window, choose the type of item you want to import from the **Type** list.

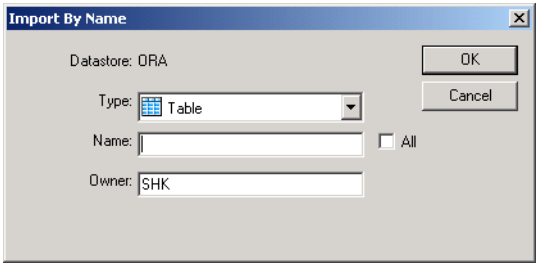
   If you are importing a stored procedure, select **Function**.
6. Specify the items you want imported.

   NOTE: Options vary by database type.

   - For tables:
     - Enter a table name in the **Name** box to specify a particular table, or select the **All** check box, if available, to specify all tables.

       If the name is case-sensitive in the database (and not all uppercase), enter the name as it appears in the database and use double quotation marks (") around the name to preserve the case.
     - Enter an owner name in the **Owner** box to limit the specified tables to a particular owner. If you leave the owner name blank, you specify matching tables regardless of owner (that is, either any table with the specified table name, or all tables).

When importing database objects into a datastore configured for multiple profiles, Data Integrator sets the **Owner** to the **Owner name** of the default datastore profile when searching the database. You cannot change the **Owner** in this window. In the repository, Data Integrator sets the owner to DSOWNER.



You can view the connection information for the default datastore profile including the owner name in both the datastore editor and the datastore profile editor.

- For functions and procedures:
  - In the **Name** box, enter the name of the function or stored procedure.

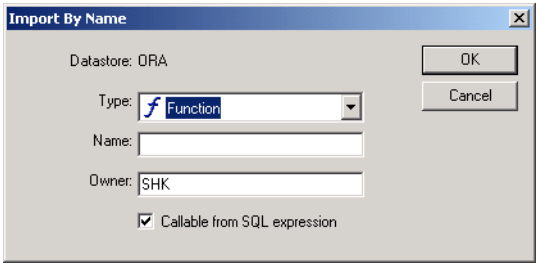
    If the name is case-sensitive in the database (and not all uppercase), enter the name as it appears in the database and use double quotation marks (") around the name to preserve the case. Otherwise, Data Integrator will convert names into all upper-case characters.

    You can also enter the name of a package. An Oracle package is an encapsulated collection of related program objects (e.g., procedures, functions, variables, constants, cursors, and exceptions) stored together in the database. Data Integrator allows you to import procedures or functions created within packages and use them as top-level procedures or functions.

If you enter a package name, Data Integrator imports all stored procedures and stored functions defined within the Oracle package. You cannot import an individual function or procedure defined within a package.

- Enter an owner name in the **Owner** box to limit the specified functions to a particular owner. If you leave the owner name blank, you specify matching functions regardless of owner (that is, any function with the specified name).

  When importing database objects into a datastore configured for multiple profiles, Data Integrator sets the **Owner** to the **Owner name** of the default datastore profile when searching the database. You cannot change the **Owner** in this window. In the repository, Data Integrator sets the owner to DSOWNER.



- If you are importing an Oracle function or stored procedure and any of the following conditions apply, clear the **Callable from SQL expression** check box. A stored procedure cannot be pushed down to a database inside another SQL statement when the stored procedure contains a DDL statement, ends the current transaction with COMMIT or ROLLBACK, or issues any ALTER SESSION or ALTER SYSTEM commands.

7. Click **OK**.

➢ To import by searching

NOTE: Functions cannot be imported by searching.

1. Open the object library.
2. Click the **Datastores** tab.
3. Select the name of the datastore you want to use.
4. Right-click and select **Search**.

   The Search window appears.
5. Enter the entire item name or some part of it in the **Name** text box.

   If the name is case-sensitive in the database (and not all uppercase), enter the name as it appears in the database and use double quotation marks (") around the name to preserve the case.
6. Select **Contains** or **Equals** from the drop-down list to the right depending on whether you provide a complete or partial search value.

   Equals qualifies only the full search string. That is, you need to search for *owner*.*table_name* rather than simply *table_name*.
7. (Optional) Enter a description in the **Description** text box.
8. Select the object type in the **Type** box.
9. Select the datastore in which you want to search from the **Look In** box.
10. Select **External** from the drop-down box to the right of the **Look In** box.

    External indicates that Data Integrator searches for the item in the entire database defined by the datastore.

    Internal indicates that Data Integrator searches only the items that have been imported.

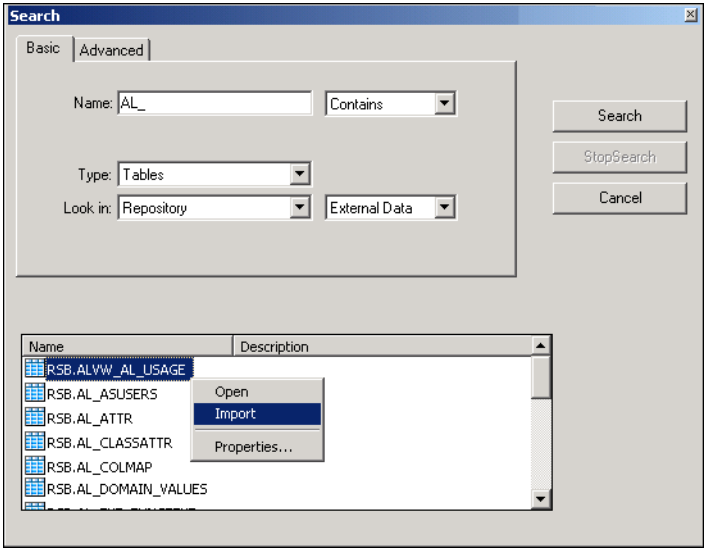
11. Go to the **Advanced** tab to search using Data Integrator attribute values.

    The advanced options only apply to searches of imported items.

12. Click **Search**.

    Data Integrator lists the tables matching your search criteria.

13. To import a table from the returned list, select the table, right-click, and choose **Import**.



Memory datastores

Data Integrator also allows you to create a custom datastore using **Memory** as the **Database type**. Memory datastores are designed to enhance processing performance of data flows executing in real-time jobs. Data (typically small amounts in a real-time job) is stored in memory to provide immediate access instead of going to the original source data.

A memory datastore is a container for memory tables. In Data Integrator, a datastore normally provides a connection to a database, application, or adapter. By contrast, a memory datastore contains memory table schemas saved in the repository.

Memory tables are schemas that allow you to cache intermediate data. Memory tables can cache data from relational database tables and hierarchical data files such as XML messages and SAP IDocs (both of which contain nested schemas).

Memory tables can be used to:

- Move data between data flows in real-time jobs. By caching intermediate data, the performance of real-time jobs with multiple data flows is far better than it would be if files or regular tables were used to store intermediate data. For best performance, only use memory tables when processing small quantities of data.

Store table data in memory for the duration of a job. By storing table data in memory, the LOOKUP_EXT function and other transforms and functions that do not require database operations can access data without having to read it from a remote database.

The lifetime of memory table data is the duration of the job. The data in memory tables cannot be shared between different real-time jobs. Support for the use of memory tables in batch jobs is not available.

Creating memory datastores

You can create memory datastores using the Datastore Editor window.

➢ To define a memory datastore

1. From the **Project** menu, select **New** > **Datastore**.
2. In the **Name** box, enter the name of the new datastore.

Be sure to use the naming convention "Memory_DS". Datastore names are appended to table names when table icons appear in the workspace. Memory tables are represented in the workspace with regular table icons. Therefore, label a memory datastore to distinguish its memory tables from regular database tables in the workspace.

3. In the **Application Type** box select **Custom**.
4. In the **Database Type** box select **Memory**.

   No additional attributes are required for the memory datastore.
5. Click **OK**.

Creating memory tables

When you create a memory table, you do not have to specify the table's schema or import the table's metadata. Instead, Data Integrator creates the schema for each memory table automatically based on the preceding schema, which can be either a schema from a relational database table or hierarchical data files such as XML messages. The first time you save the job, Data Integrator defines the memory table's schema and saves the table. Subsequently, the table appears with a table icon in the workspace and in the object library under the memory datastore.

➢ To create a memory table:

1. From the tool pallet, click the template table icon. 
2. Click inside a data flow to place the template table.

   The Create Table window opens.
3. From the Create Table window, select the memory datastore.
4. Enter a table name.

5. If you want a system-generated row ID column in the table, click the **Create Row ID** check box.

   See "Create Row ID option" on page 103 for more information.

6. Click **OK**.

   The memory table appears in the workspace as a template table icon.

7. Connect the memory table to the data flow as a target.

8. From the **Project** menu select **Save**.

   In the workspace, the memory table's icon changes to a target table icon and the table appears in the object library under the memory datastore's list of tables.

Using memory tables as sources and targets

After you create a memory table as a target in one data flow, you can use a memory table as a source or target in any data flow. See Chapter 10, "Real-time jobs" for an example of how to use memory tables as sources and targets in a job.

➢ To use a memory table as a source or target

1. In the object library, click the **Datastores** tab.
2. Expand the memory datastore that contains the memory table you want to use.
3. Expand **Tables.**

   A list of tables appears.

4. Select the memory table you want to use as a source or target, and drag it into an open data flow.
5. Connect the memory table as a source or target in the data flow.

- If you are using a memory table as a target, open the memory table's target table editor to set table options. See "Memory table target options" on page 102 for more information.

6. Save the job.

Update Schema option

You might want to quickly update a memory target table's schema if the preceding schema changes. To do this, use the Update Schema option. Otherwise, you would have to add a new a memory table to update a schema.

➢ To update the schema of a memory target table

1. Right-click the memory target table's icon in the work space.
2. Select **Update Schema**.

   The schema of the preceding object is used to update the memory target table's schema. The current memory table is updated in your repository. All occurrences of the current memory table are updated with the new schema.

Memory table target options

The **Delete data from table before loading** option is available for memory table targets. The default is yes. To set this option, open the memory target table editor. If you deselect this option, new data will append to the existing table data.

Create Row ID option

If the **Create Row ID** is checked in the Create Memory Table window, Data Integrator generates an integer column called **DI_Row_ID** in which the first row inserted gets a value of 1, the second row inserted gets a value of 2, etc. This new column allows you to use a LOOKUP_EXT expression as an iterator in a script.

NOTE: The same functionality is available for other datastore types using the SQL function.

Use the **DI_Row_ID** column to iterate through a table using a `lookup_ext` function in a script. For example:

```
$NumOfRows = total_rows(memory_DS..table1)

$I = 1;

$count=0;

while ($count < $NumOfRows)

begin

   $data =
   lookup_ext([memory_DS..table1,'NO_CACHE',
   'MAX'],[A],[O],[DI_Row_ID,'=',$I]);

   $I = $I + 1;

   if ($data != NULL)

   begin

   $count = $count + 1;

   end
```

In the preceding script, `table1` is a memory table. The table's name is preceded by its datastore name (memory_DS), a dot, a blank space (where a table owner would be for a regular table), then a second dot. There are no owners for memory datastores, so tables are identified by just the datastore name and the table name as shown.

Select the LOOKUP_EXT function arguments (line 7) from the function editor when you define a LOOKUP_EXT function.

The TOTAL_ROWS(*DatastoreName.Owner.TableName*) function returns the number of rows in a particular table in a datastore. This function can be used with any type of datastore. If used with a memory datastore, use the following syntax: TOTAL_ROWS(*DatastoreName..TableName*)

Data Integrator also provides a built-in function that you can use to explicitly expunge data from a memory table. This provides finer control than the active job has over your data and memory usage. The TRUNCATE_TABLE(*DatastoreName..TableName*) function can only be used with memory tables.

For more information about these and other Data Integrator functions, see "Descriptions of built-in functions" on page 342 of the *Data Integrator Reference Guide*.

Troubleshooting memory tables

- One possible error, particularly when using memory tables, is that Data Integrator runs out of virtual memory space. If Data Integrator runs out of memory while executing any operation, Data Integrator exits.
- A validation and run time error occurs if the schema of a memory table does not match the schema of the preceding object in the data flow.

  To correct this error, use the **Update Schema** option or create a new memory table to match the schema of the preceding object in the data flow.

- Two log files contain information specific to memory tables: `trace_memory_reader` log and `trace_memory_loader` log.

Adapter datastores

Depending on the adapter implementation, Data Integrator adapters can provide:

- Application metadata browsing
- Application metadata importing into the Data Integrator repository
- Batch and real-time data movement between Data Integrator and applications

Business Objects offers an Adapter Software Development Kit (SDK) to develop your own custom adapters. Also, you can buy Data Integrator pre-packaged adapters to access application metadata and data in any application. For more information on these products, contact your Business Objects Sales Representative.

For information on installing, configuring, and starting adapters, see Chapter 8, "Adapters," in the *Data Integrator Administrator Guide*. For information on configuring adapter connections for a Job Server, see the *Data Integrator Getting Started Guide*.

Adapters are represented in Designer by adapter datastores. Data Integrator jobs provide batch and real-time data movement between Data Integrator and applications through an adapter datastore's subordinate objects:

Subordinate Objects	Use as	For
Tables	Source or target	Batch data movement
Documents	Source or target	
Functions	Function call in query	
Message functions	Function call in query	Real-time data movement
Outbound messages	Target only	

These objects are described in "Source and target objects" on page 161 and "Real-time source and target objects" on page 252.

Adapters can provide access to an application's data and metadata or just metadata. For example, if the data source is SQL-compatible, the adapter might be designed to access metadata, while Data Integrator extracts data from or loads data directly to the application.

Defining an adapter datastore

You need to define at least one datastore for each adapter through which you are extracting or loading data.

To define a datastore, you must have appropriate access privileges to the application that the adapter serves. If you do not have access, ask your system administrator to create an account for you.

➢ To define an adapter datastore

1. In the Object Library, click to select the **Datastores** tab.
2. Right-click and select **New** from the menu to view the Datastore Editor window.
3. In the **Name** text box of the **Connection** tab, enter a unique identifying name for the datastore.

   The datastore name appears in the Designer only. It can be the same as the adapter name.
4. In the **Application type** list, select **Adapter**.
5. Select a **Job server** from the list.

   To create an adapter datastore, you must first select a Job Server to run the adapter. Adapters residing on the Job Server computer and registered with the selected Job Server appear as potential selections.
6. Select an adapter instance from the **Adapter instance name** list.

7. Select the **Adapter Properties** tab.

   NOTE: If you do not select a Job Server/Adapter instance name location, you cannot access the Adapter Properties tab.

8. Enter all adapter property information required to complete the datastore connection.

   NOTE: Adapter documentation should list all information required for datastore connection.

9. If the adapter supports direct SQL access to the data source, you will see an **SQL Connection** tab. Select this tab and enter the requested database properties to include such information as database type, connection name, database version, user name, and password.

   If all data and metadata pass through the adapter, you will not see this tab.

10. Click **OK**.

    The datastore configuration is saved in your metadata repository and the new datastore appears in the Object Library Datastores tab.

After you complete your datastore connection, you can browse and/or import metadata from the data source through the adapter.

➢ To change an adapter datastore's configuration

1. Right-click the datastore you want to browse and select **Edit** to open the Datastore Editor window.
2. Edit configuration information in all available tabs.
3. Click **OK**.

   The edited datastore configuration is saved in your metadata repository.

➢ To delete an adapter datastore and associated metadata objects

1. Right-click the datastore you want to delete and select **Delete**.
2. Click **OK** in the confirmation window.

   Data Integrator removes the datastore and all metadata objects contained within that datastore from the metadata repository. Any of these objects used in established flows will appear marked with a red "deleted" icon  .

Browsing metadata through an adapter datastore

The metadata you can browse depends on the specific adapter.

➢ To browse application metadata

1. Right-click the datastore you want to browse and select **Open**.

   A window opens showing source metadata.
2. Scroll to view metadata name and description attributes.
3. Click plus signs [+] to expand objects and view subordinate objects.
4. Right-click any object to check importability.

Importing metadata through an adapter datastore

The metadata you can import depends on the specific adapter. After importing metadata, you can edit it. Your edits propagate to all objects that call these objects.

➢ To import application metadata while browsing

1. Right-click the datastore you want to browse, then select **Open**.

2. Find the metadata object you want to import from the browsable list.
3. Right-click the object and select **Import**.
4. The object is imported into one of the adapter datastore containers (documents, functions, tables, outbound messages, or message functions).

➢ To import application metadata by name

1. Right-click the datastore from which you want metadata, then select **Import by name**.

   The Import by name window appears containing import parameters with corresponding text boxes.
2. Click each import parameter text box and enter specific information related to the object you want to import.
3. Click **OK**. Any object(s) matching your parameter constraints are imported to one of the corresponding Data Integrator categories specified under the datastore.

Defining profiles

When you configure a datastore to have multiple profiles, you must create each profile you want associated with that datastore. There are two types of profiles:

- **Datastore profiles** — These profiles define the connections for a particular datastore.
- **System profiles** — These profiles define a set of datastore profiles that you want to use together when running a job.

When designing jobs, you must determine and create the required datastore profiles and system profiles depending on your business environment and rules. Create datastore profiles for the datastores in your repository before you create the system profiles.

You select a system profile to use at run-time. In many enterprises, a job designer defines the required datastore and system profiles, and a system administrator determines which system profile to use when scheduling or starting a job.

Datastore profiles are part of the datastore object. When you import or export a job, datastore profiles are included. Similarly, when you check in or check out a datastore, you also check in or check out the corresponding datastore profiles.

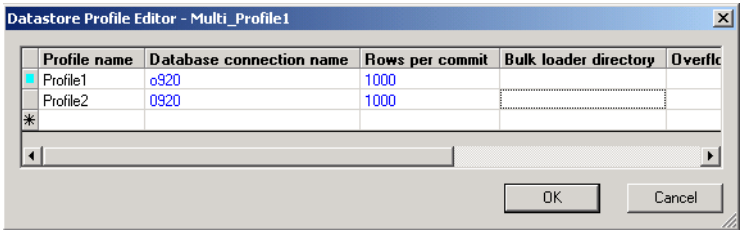
Data Integrator maintains system profiles separately. You cannot check in or check out system profiles. However, you can export system profiles to a separate flat file which you can later import. By maintaining system profiles in a separate file, you avoid modifying your datastore each time you import or export a job, or each time you check in and check out the datastore.

➢ To create a datastore profile

1. You must first create or edit a datastore and select the **Multiple profiles** checkbox.
2. On the datastores tab of the object library, right-click a datastore to see whether it is configured for multiple profiles.

   If the **Datastore Profiles** command is available, the datastore is configured for multiple profiles.
3. Select the **Datastore Profiles** command.

   The Datastore Profile Editor window opens, showing datastore profiles in a table format.



   The information you entered in the datastore editor when creating the datastore appears as a profile labeled "New_Profile" and initially appears as the default profile. You can change this (and any other) profile name. When you define two or more profiles, you can also change the default profile.

   The current default profile for the datastore is indicated by a blue square. Data Integrator uses the default profile information when importing metadata, when executing a job with no system profile specified, or when executing a job and the system profile does not specify a particular datastore profile. The datastore editor shows connection information for the current default profile.

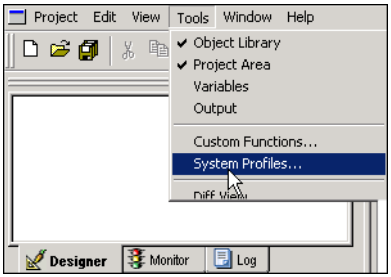
4. Use the Datastore Profile Editor window to add a new profile or to edit an existing profile.
   - To add a new profile, type in the row marked with an asterisk.
   - Business Objects recommends that you use the `DP_` prefix in the profile name so that you can easily identify it as a datastore profile.
   - Profile options depend on datastore type. See "Datastore" on page 34 of the *Data Integrator Reference Guide* for a description of options.

     You cannot modify the following profile options from the original values used to create the datastore:
     - Datastore application type
     - Database type
     - Database version
   - Right-click the gray box at the beginning of each row to use the **Cut**, **Copy**, **Paste**, and **Delete** commands or to set one profile as the default.
5. (Optional) Customize your view of profile data.
   - Select columns that you would like to hide, then right-click and select **Hide**. Select a range of columns, then right-click the first row and select **Show** to display hidden columns within the range.
   - Right-click the **Profile name** column and select **Sort** > **Ascending** or **Sort** > **Descending** to sort the datastore profiles by name.
6. Click **OK**.

➢ To create a system profile

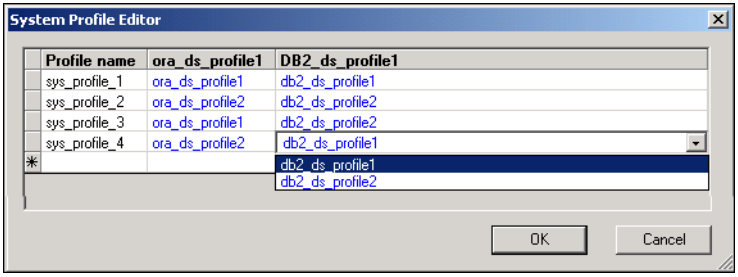
NOTE: You cannot define a system profile if your repository does not contain at least one datastore with multiple profiles.

1. Select **Tools** > **System Profiles**.



The System Profile Editor window opens.

In this window, each column is a datastore that has multiple profiles in your repository. Each row is a system profile and lists the datastore profiles used when this system profile is selected.



2. Create a new system profile or edit the datastore profiles in an existing system profile.
   - To add a new system profile, type in the row marked with an asterisk.
   - Business Objects recommends that you use the `SP_` prefix in the profile name so that you can easily identify it as a system profile, particularly when exporting.

- For each datastore, select a datastore profile you want to use when you run a job using the selected system profile. Data Integrator displays datastore profile names in lists under the datastore names. If you do not specify a datastore profile, the default datastore profile is used for this system profile.
- Right-click the gray box at the beginning of a row to use the **Cut**, **Copy**, **Paste**, and **Delete** commands.

3. Click **OK**.

➢ To export a system profile

1. In the object library, right-click a datastore. If the datastore is configured for multiple profiles, select **Repository** > **Export System Profile**.
2. Business Objects recommends that you add the `SP_` prefix to your exported system profile's .atl file to easily identify the file as a system profile.
3. Click **OK**.

6 File Formats

This chapter contains the following sections:

- What are file formats?
- File format editor
- Creating file formats
- Editing file formats
- Replicating and renaming file formats
- File format features
- File transfers

For full details of file format properties, see "File format" on page 62 of the *Data Integrator Reference Guide*.

What are file formats?

A *file format* is a set of properties describing the structure of a flat file (ASCII). File formats describe the metadata structure. A file format describes a specific file. A file format template is a generic description that can be used for many data files.

Data Integrator can use data stored in files for data sources and targets. A file format defines a connection to a file. Therefore, you use a file format to connect Data Integrator to source or target data when the data is stored in a file rather than a database table. The object library stores file format templates that you use to define specific file formats as sources and targets in data flows.

When working with file formats, you must:

- Create a file format template that defines the structure for a file.
- Create a specific source or target file format in a data flow. The source or target file format is based on a template and specifies connection information such as the file name.

File format objects can describe files in:

- **Delimited format** — Characters such as commas or tabs separate each field
- **Fixed width format** — The column width is specified by the user
- **SAP R/3 format** — For details, see "Defining SAP R/3 file formats" on page 70 of the *Data Integrator Supplement for SAP*

File format editor

Use the file format editor to set properties for file format templates and source and target file formats. Available properties vary by the mode of the file format editor:

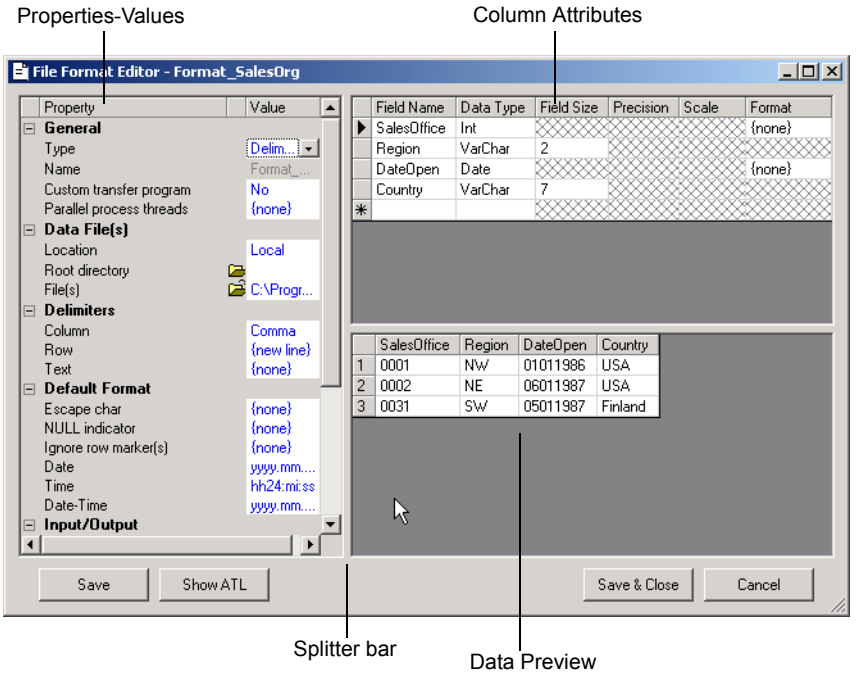
- **New mode** — Create a new file format template
- **Edit mode** — Edit an existing file format template
- **Source mode** — Edit the file format of a particular source file
- **Target mode** — Edit the file format of a particular target file

The file format editor has three work areas:

- **Properties-Values** — Edit the values for file format properties. Expand and collapse the property groups by clicking the leading plus or minus.
- **Column Attributes** — Edit and define the columns or fields in the file. Field-specific formats override the default format set in the Properties-Values area.
- **Data Preview** — View how the settings affect sample data.

The file format editor contains "splitter" bars to allow resizing of the window and all the work areas. You can expand the file format editor to the full screen size.

The properties and appearance of the work areas vary with the format of the file.

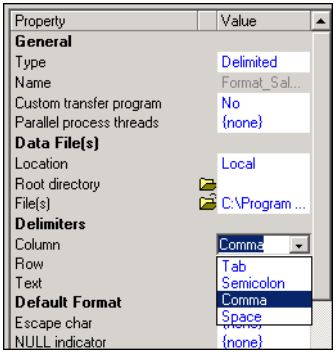


For more information about the properties in the file format editor, see "File format" on page 62 of the *Data Integrator Reference Guide*.

You can navigate within the file format editor as follows:

- Switch between work areas using the **Tab** key.
- Navigate through fields in the Data Preview area with the **Page Up**, **Page Down**, and arrow keys.

- Open a drop-down menu in the Properties-Values area by pressing the **ALT**-down arrow key combination.



- When the file format type is fixed-width, you can also edit the column metadata structure in the Data Preview area.

NOTE: The **Show ATL** button displays a view-only copy of the Transformation Language file generated for your file format. You might be directed to use this by Business Objects Technical Support.

Creating file formats

To specify a source or target file:

- Create a file format template that defines the structure for a file.
- When you drag and drop a file format into a data flow; the format represents a file that is based on the template and specifies connection information such as the file name.

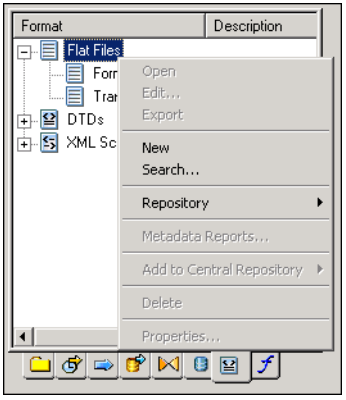
Create a file format template using any of the following methods:

- To create a new file format
- To model a file format on a sample file
- To create a file format from an existing file format
- To create a file format from an existing flat table schema

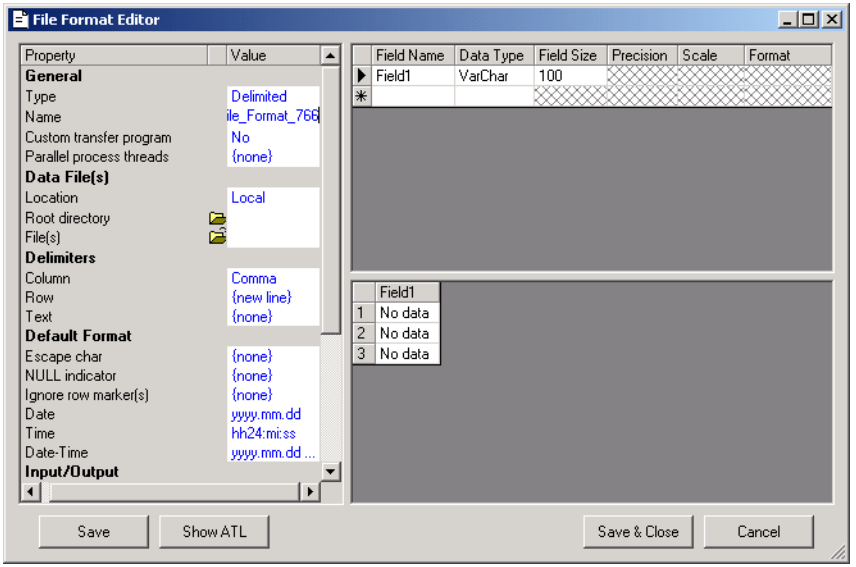
To use a file format to create a metadata file, see "To create a specific source or target file" on page 130.

➢ To create a new file format

1. In the local object library, go to the **Formats** tab, right-click **Flat Files**, and select **New**.



2. The file format editor opens.



3. In **Type**, specify the file type:
   - **Delimited** — Select **Delimited** if the file uses a character sequence to separate columns
   - **Fixed width** — Select **Fixed width** if the file uses specified widths for each column.

     NOTE: Data Integrator represents column sizes (field-size) in number of characters for all sources except fixed-width file formats, which it always represents in bytes. Consequently, if a fixed-width, file format uses a multi-byte code page, then no data is displayed in the data preview section of the file format editor for its files. For more information about multi-byte support, see Chapter 9, "Locales and Multi-Byte Functionality," in the *Data Integrator Reference Guide*.

4. In **Name**, enter a name that describes this file format template.

After you save this file format template, you cannot change the name.

5. If you want to read and load files using a third-party file-transfer program, select YES for **Custom transfer program** and see "File transfers" on page 139.

6. Complete the other properties to describe files that this template represents.

   Properties vary by file type. All properties are described in "File format" on page 62 of the *Data Integrator Reference Guide*. Look for properties available when the file format editor is in new mode.

7. For source files, specify the structure of the columns in the Column Attributes work area:

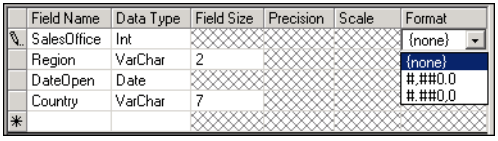
   a. Enter field name.

   b. Set data types.

   c. Enter field lengths for **VarChar** data types.

   d. Enter scale and precision information for **Numeric** and **Decimal** data types.

   e. Enter **Format** field information for appropriate data types, if desired. This information overrides the default format set in the Properties-Values area for that data type.



You can model a file format on a sample file. See "To model a file format on a sample file" on page 125.

NOTE: You do not need to specify columns for files used as targets. If you do specify columns and they do not match the output schema from the preceding transform, Data Integrator writes to the target file using the transform's output schema.

NOTE: For a decimal or real data type, if you only specify a source column format, and the column names and data types in the target schema do not match those in the source schema, Data Integrator cannot use the source column format specified. Instead, it defaults to the format used by the code page on the computer where the Job Server is installed.

8. Click **Save & Close** to save the file format template and close the file format editor.

➢ To model a file format on a sample file

1. From the **Formats** tab in the local object library, create a new flat file format template or edit an existing flat file format template.
2. Under **Data File(s)**:
   - If the sample file is on your Designer computer, set **Location** to **Local**. Browse to set the **Root directory** and **File(s)** to specify the sample file.

     NOTE: During design, you can specify a file located on the computer where the Designer runs or on the computer where the Job Server runs. Indicate the file location in the **Location** property. During execution, you must specify a file located on the Job Server computer that will execute the job.

- If the sample file is on the current Job Server computer, set **Location** to **Job Server**. Enter the **Root directory** and **File(s)** to specify the sample file. When you select **Job Server**, the **Browse** button is disabled, so you must type the absolute path to the file. (For example, a path on UNIX might be `/usr/data/abc.txt`. A path on Windows might be `C:\DATA\abc.txt`.)

  NOTE: In the Windows operating system, files are not case sensitive; however, file names are case sensitive in the UNIX environment. (For example, `abc.txt` and `aBc.txt` would be two different files in the same UNIX directory.)

  To reduce the risk of typing errors, you can telnet to the Job Server (UNIX or Windows) computer and find the full path name of the file you want to use. Then, copy and paste the path name from the telnet application directly into the **Root directory** text box in the file format editor. You cannot use the Windows Explorer to determine the exact file location on Windows.

3. If the file type is delimited, set the appropriate column delimiter for the sample file.
4. Under **Input/Output**, set **Skip row header** to **Yes** if you want to use the first row in the file to designate field names.

   The file format editor will show the column names in the Data Preview area and create the metadata structure automatically.
5. Edit the metadata structure as needed.

   For both delimited and fixed-width files, you can edit the metadata structure in the Column Attributes work area:

   a. Right-click to insert or delete fields.
   b. Rename fields.
   c. Set data types.
   d. Enter field lengths for the **VarChar** data type.

e. Enter scale and precision information for **Numeric** and **Decimal** data types.

f. Enter **Format** field information for appropriate data types, if desired. This format information overrides the default format set in the Properties-Values area for that data type.

For fixed-width files, you can also edit the metadata structure in the Data Preview area:

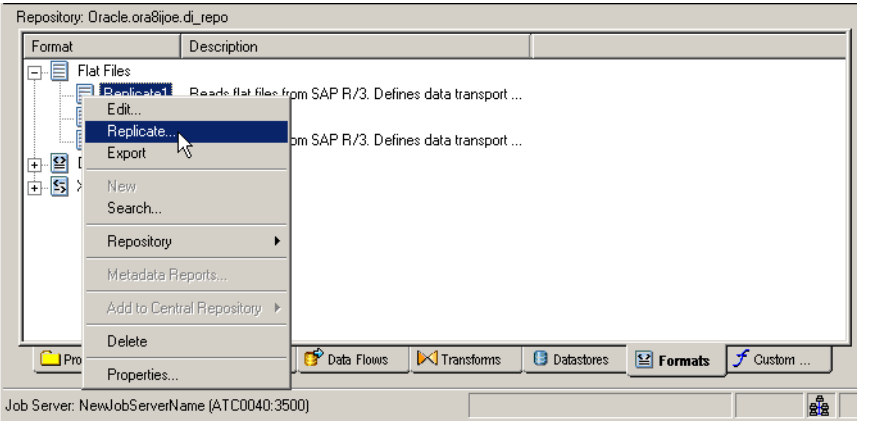
a. Click to select and highlight columns.

b. Right-click to insert or delete fields.

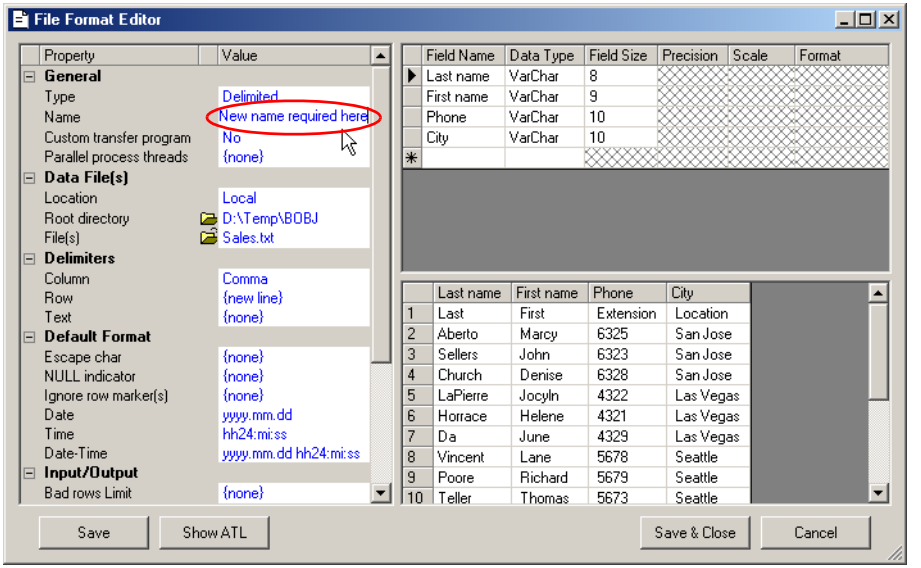
6. Click **Save & Close** to save the file format template and close the file format editor.

➢ To create a file format from an existing file format

1. In the Formats tab of the object library, right-click an existing file format and choose **Replicate** from the menu.



The File Format Editor opens, displaying the schema of the copied file format.



2. Double-click to select the **Name** property value (which contains the same name as the original file format object).
3. Type a new, unique name for the replicated file format.

   NOTE: You must enter a new name for the replicated file. Data Integrator does not allow you to save the replicated file with the same name as the original (or any other existing File Format object). Also, this is your only opportunity to modify the **Name** property value. Once saved, you cannot modify the name again.

4. Edit other properties as desired.

   Properties are described in "File format" on page 62 of the *Data Integrator Reference Guide*. Look for properties available when the file format editor is in new mode.

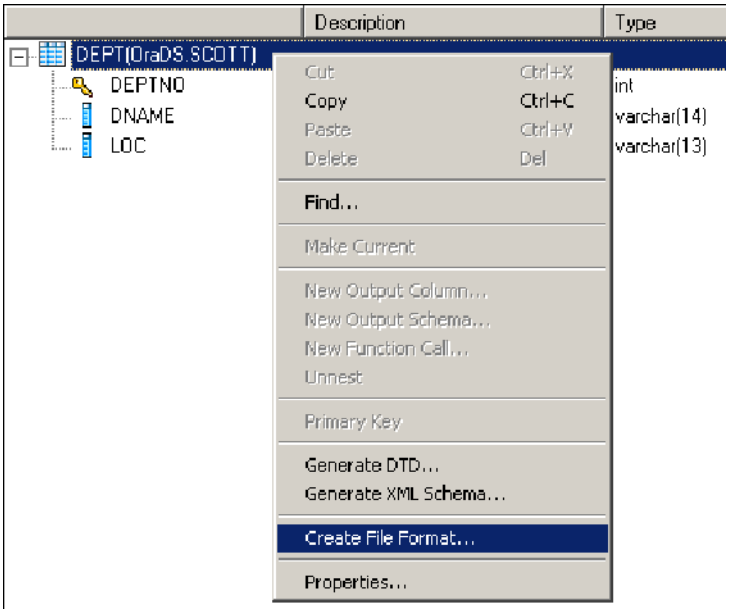
5. To save and view your new file format schema, click **Save**.

To terminate the replication process (even after you have changed the name and clicked **Save**), click **Cancel** or press the **Esc** button on your keyboard.

6. Click **Save & Close.**

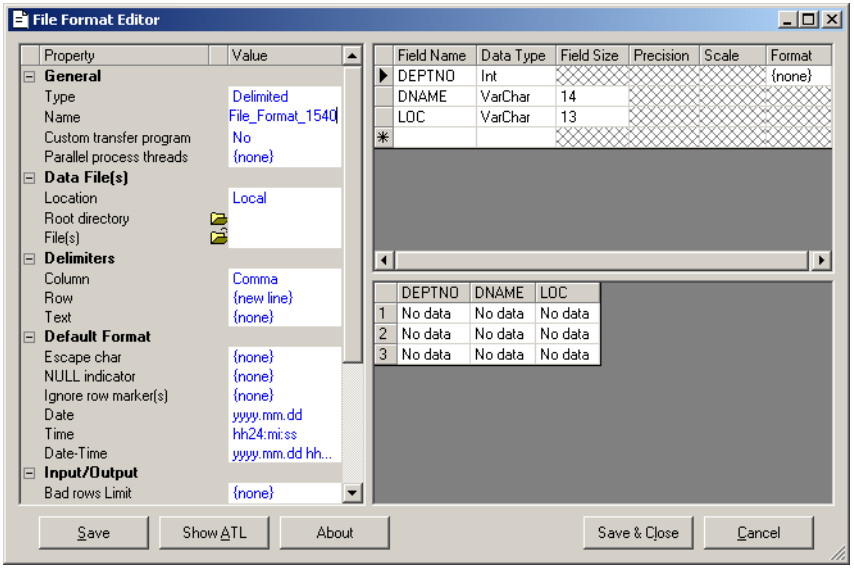
➢ To create a file format from an existing flat table schema

1. From the Query editor, right-click a schema and select **Create File format**.



The File Format editor opens populated with the schema you selected.

2. Edit the new schema as appropriate and click **Save &Close.**



Data Integrator saves the file format in the repository. You can access it from the Formats tab of the object library.

➢ To create a specific source or target file

1. Select a flat file format template on the **Formats** tab of the local object library.
2. Drag the file format template to the data flow workspace.
3. Select **Make Source** to define a source file format, or select **Make Target** to define a target file format.
4. Click the name of the file format object in the workspace to open the file format editor.
5. Enter the properties specific to the source or target file.

   For a description of available properties, refer to "File format" on page 62 of the *Data Integrator Reference Guide*. Look for properties available when the file format editor is in source mode or target mode.

Under **File(s)**, be sure to specify the file name and location in the **File** and **Location** properties.

NOTE: You can use variables as file names. Refer to "Setting file names at run-time using variables" on page 313.

6. Connect the file format object to other objects in the data flow as appropriate.

Editing file formats

You can modify existing file format templates to match changes in the format or structure of a file. You cannot change the name of a file format template.

For example, if you have a date field in a source or target file that is formatted as mm/dd/yy and the data for this field changes to the format dd-mm-yy due to changes in the program that generates the source file, you can edit the corresponding file format template and change the date format information.

For specific source or target file formats, you can edit properties that uniquely define that source or target such as the file name and location.

➢ To edit a file format template

1. In the object library **Formats** tab, double-click an existing flat file format (or right-click and choose **Edit**).

   The file format editor opens with the existing format values.

2. Edit the values as needed.

   Properties are described in "File format" on page 62 of the *Data Integrator Reference Guide*. Look for properties available when the file format editor is in edit mode.

   Any changes effect every source or target file that is based on this file format template.

3. Click **Save**.

➢ To edit a source or target file

1. From the workspace, click the name of a source or target file.

   The file format editor opens, displaying the properties for the selected source or target file.

2. Edit the desired properties.

Properties are described in "File format" on page 62 of the *Data Integrator Reference Guide*. Look for properties available when the file format editor is in source or target mode as appropriate.

To change properties that are not available in source or target mode, you must edit the file's file format template.

Any changes you make to values in a source or target file editor override those on the original file format.

3. Click **Save**.

Replicating and renaming file formats

After you create one file format schema, you can quickly create another file format object with the same schema by replicating the existing file format and renaming it. To save time in creating file format objects, replicate and rename instead of configuring from scratch. For more information see, "To create a file format from an existing file format" on page 127.

File format features

Data Integrator offers several capabilities for processing files:

- Reading multiple files at one time
- Number formats
- Ignoring rows with specified markers
- Date formats at the field level
- Replicating and renaming file formats

Reading multiple files at one time

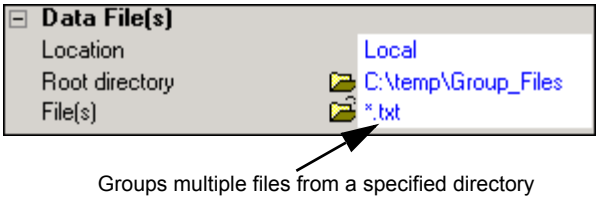
Data Integrator can read multiple files with the same format from a single directory using a single source object.

➢ To specify multiple files to read

1. Under **Data File(s)** in the file format editor:
   a. Set the **Location** of the source files to **Local** or **Job Server**
   b. Set the root directory in **Root directory**.
   c. Under **File(s)**, enter one of the following:
      - A list of file names separated by commas, or
      - A file name containing a wild card character (* or ?). For example:

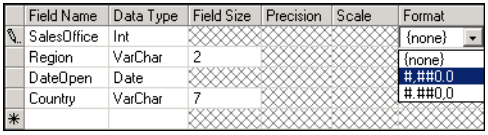
        `*.txt` reads all files with the `txt` extension

        `1999????.txt` might read files from the year 1999.



Number formats

The period (.) and the comma (,) are the two most common formats used to determine decimal and thousand separators for numeric data types. When formatting files in Data Integrator, data types in which these symbols can be used include Decimal, Numeric, Int (integer), and Double. You can use either symbol for the thousands indicator and either symbol for the decimal separator. For example: `2,098.65` or `2.089,65`.



Leading and trailing decimal signs are also supported. For example: `+12,000.00` or `32.32-`.

Ignoring rows with specified markers

The file format editor provides a way to ignore rows containing a specified marker (or markers) when reading files.

For example, you might want to ignore comment line markers such as `#` and `//`.

Associated with this feature, two special characters — the semicolon (;) and the backslash (\) — make it possible to define multiple markers in your ignore row marker string. Use the semicolon to delimit each marker, and use the backslash to indicate special characters as markers (such as the backslash and the semicolon).

The default marker value is an empty string. When you specify the default value, no rows are ignored.

➢ To specify markers for rows to ignore

1. Open the file format editor from the Object Library or by opening a source object in the workspace.

2. Find **Ignore row marker(s)** under the **Format** Property.
3. Click in the associated text box and enter a string to indicate one or more markers representing rows that Data Integrator should skip during file read and/or metadata creation.

   The following table provides some ignore row marker(s) examples. (Each value is delimited by a semicolon unless the semicolon is preceded by a backslash.)

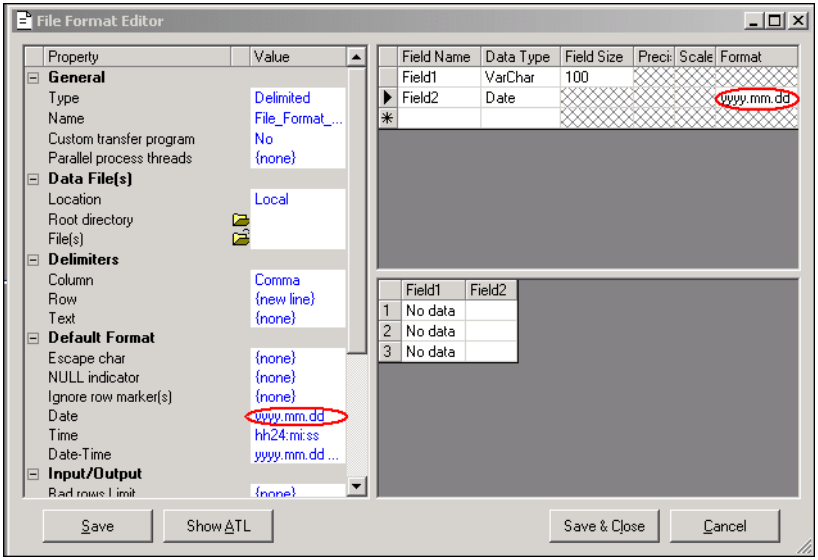
Marker Value(s)	Row(s) Ignored
	None (this is the default value)
`abc`	Any that begin with the string `abc`
`abc;def;hi`	Any that begin with `abc` or `def` or `hi`
`abc;\;`	Any that begin with `abc` or `;`
`abc;\\;\;`	Any that begin with `abc` or `\` or `;`

Date formats at the field level

You can specify a date format at the field level to overwrite the default date, time, or date-time formats set in the Properties-Values area.

For example, when the **Data Type** is set to Date, you can edit the value in the corresponding **Format** field to a different date format such as:

- yyyy.mm.dd
- mm/dd/yy
- dd.mm.yy


File Format Editor
Property
Value
General
Type
Delimited
Name
File_Format_...
Custom transfer program
No
Parallel process threads
{none}
Data File(s)
Location
Local
Root directory
File(s)
Delimiters
Column
Comma
Row
{new line}
Text
{none}
Default Format
Escape char
{none}
NULL indicator
{none}
Ignore row marker(s)
{none}
Date
yyyy.mm.dd
Time
hh24:mi:ss
Date-Time
yyyy.mm.dd ...
Input/Output
Bad rows Limit
{none}
Field Name
Data Type
Field Size
Preci
Scale
Format
Field1
VarChar
100
Field2
Date
yyyy.mm.dd
Field1
Field2
1
No data
2
No data
3
No data
Save
Show ATL
Save & Close
Cancel

File transfers

Data Integrator can read and load files using a third-party file transfer program for flat files. You can use third-party (custom) transfer programs to:

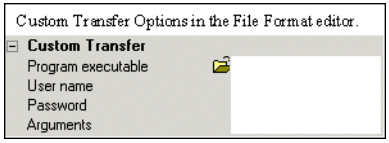
- Incorporate company-standard file-transfer applications as part of Data Integrator job execution
- Provide high flexibility and security for files transferred across a firewall

The custom transfer program option allows you to specify:

- A custom transfer program (invoked during job execution)
- Additional arguments, based on what is available in your program, such as:
  - Connection data
  - Encryption/decryption mechanisms
  - Compression mechanisms

Custom transfer system variables for flat files

When you set **Custom Transfer program** to YES in the **Property** column of the file format editor, the following options are added to the column. To view them, scroll the window down.



When you set custom transfer options for external file sources and targets, some transfer information, like the name of the remote server that the file is being transferred to or from, may need to be entered literally as a transfer program argument. You can enter other information using the following Data Integrator system variables:

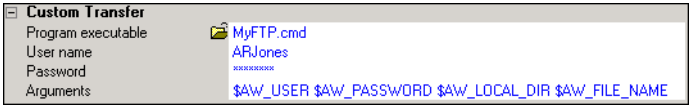
Data entered for:	Is substituted for this variable if it is defined in the Arguments field
User name	$AW_USER
Password	$AW_PASSWORD
Local directory	$AW_LOCAL_DIR
File(s)	$AW_FILE_NAME

By using these variables as custom transfer program arguments, you can collect connection information entered in Data Integrator and use that data at run-time with your custom transfer program.

For example, the following custom transfer options use a Windows command file (Myftp.cmd) with five arguments. Arguments 1 through 4 are Data Integrator system variables:

- User and Password variables are for the external server
- The Local Directory variable is for the location where the transferred files will be stored in Data Integrator
- The File Name variable is for the names of the files to be transferred

Argument 5 provides the literal external server name.



The content of the Myftp.cmd script is as follows:

NOTE: If you do not specify a standard output file (such as ftp.out in the example below), Data Integrator writes the standard output into the job's trace log.

```
@echo off

set USER=%1
set PASSWORD=%2
set LOCAL_DIR=%3
set FILE_NAME=%4
set LITERAL_HOST_NAME=%5

set INP_FILE=ftp.inp

echo %USER%>%INP_FILE%
echo %PASSWORD%>>%INP_FILE%
echo lcd %LOCAL_DIR%>>%INP_FILE%
echo get %FILE_NAME%>>%INP_FILE%
echo bye>>%INP_FILE%

ftp -s:%INP_FILE% >ftp.out
```

Custom transfer options for flat files

Of the custom transfer program options, only the **Program executable** option is mandatory.

Entering **User Name**, **Password,** and **Arguments** values is optional. These options are provided for you to specify arguments that your custom transfer program can process (such as connection data).

You can also use **Arguments** to enable or disable your program's built-in features such as encryption/decryption and compression mechanisms. For example, you might design your transfer program so that when you enter `-sSecureTransportOn` or `-CCompressionYES` security or compression is enabled.

NOTE: Available arguments depend on what is included in your custom transfer program. See your custom transfer program documentation for a valid argument list.

You can use the **Arguments** box to enter a user name and password. However, Data Integrator also provides separate **User name** and **Password** boxes. By entering the *$AW_USER* and *$AW_PASSWORD* variables as **Arguments** and then using the **User** and **Password** boxes to enter literal strings, these extra boxes are useful in two ways:

- You can more easily update users and passwords in Data Integrator both when you configure Data Integrator to use a transfer program and when you later export the job. For example, when you migrate the job to another environment, you might want to change login information without scrolling through other arguments.
- You can use the mask and encryption properties of the **Password** box. Data entered in the **Password** box is masked in log files and on the screen, stored in the repository, and encrypted by Data Integrator.

NOTE: Data Integrator sends password data to the custom transfer program in clear text. If you do not allow clear passwords to be exposed as arguments in command-line executables, then set up your custom program to either:

- Pick up its password from a trusted location
- Inherit security privileges from the calling program (in this case, Data Integrator)

Setting custom transfer options

The custom transfer option allows you to use a third-party program to transfer flat file sources and targets. You can configure your custom transfer program in the File Format Editor window. Like other file format settings, you can override custom transfer program settings if they are changed for a source or target in a particular data flow. You can also edit the custom transfer option when exporting a file format.

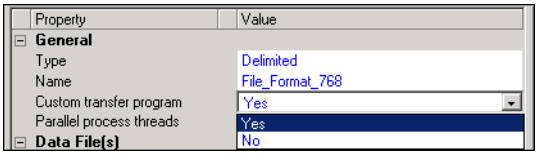
➢ To configure a custom transfer program in the file format editor

1. Select the **Formats** tab in the object library.
2. Right-click **Flat Files** in the tab and select **New**.

   The File Format Editor opens.
3. Select either the **Delimited** or the **Fixed width** file type.

   NOTE: While the custom transfer program option is not supported with R/3 file types, you can use it as a data transport method for an R/3 data flow. See "Custom Transfer method" on page 171 of the *Data Integrator Supplement for SAP* for more information.
4. Enter a format name.

5. Select **Yes** for the **Custom transfer program** option.



6. Enter the custom transfer program name and arguments.

7. Complete the other boxes in the file format editor window. See "File format" on page 62 of the *Data Integrator Reference Guide* for more information.

   In the **Data Files(s)** section, specify the location of the file in Data Integrator.

   To specify system variables for **Root directory** and **File(s)** in the **Arguments** box:

   - Associate the Data Integrator system variable `$AW_LOCAL_DIR` with the local directory argument of your custom transfer program.
   - Associate the Data Integrator system variable `$AW_FILE_NAME` with the file name argument of your custom transfer program.

   For example, enter: `-l$AW_LOCAL_DIR\$AW_FILE_NAME`.

   When the program runs, the **Root directory** and **File(s)** settings are substituted for these variables and read by the custom transfer program.

   NOTE: The flag `-l` used in the example above is a custom program flag. Arguments you can use as custom program arguments in Data Integrator depend upon what your custom transfer program expects.

8. Click **Save**.

Design tips

Keep the following concepts in mind when using the custom transfer options:

- Variables are not supported in file names when invoking a custom transfer program for the file.
- You can only edit custom transfer options in the File Format Editor (or Datastore Editor in the case of SAP R/3) window before they are exported. You cannot edit updates to file sources and targets at the data flow level when exported. After they are imported, you can adjust custom transfer option settings at the data flow level. They override file format level settings.

When designing a custom transfer program to work with Data Integrator, keep in mind that:

- Data Integrator expects the called transfer program to return 0 on success and non-zero on failure.
- Data Integrator provides trace information before and after the custom transfer program executes. The full transfer program and its arguments with masked password (if any) is written in the trace log. When "Completed Custom transfer" appears in the trace log, the custom transfer program has ended.
- If the custom transfer program finishes successfully (the return code = 0), Data Integrator checks the following:
  - For an R/3 dataflow, if the transport file does not exist in the local directory, it throws an error and Data Integrator stops. See "Custom Transfer method" on page 171 of the *Data Integrator Supplement for SAP* for information about file transfers from SAP R/3.

  - For a file source, if the file or files to be read by Data Integrator do not exist in the local directory, Data Integrator writes a warning message into the trace log.

- If the custom transfer program throws an error or its execution fails (return code is not 0), then Data Integrator produces an error with return code and `stdout`/`stderr` output.
- If the custom transfer program succeeds but produces standard output, Data Integrator issues a warning, logs the first 1,000 bytes of the output produced, and continues processing.
- The custom transfer program designer must provide valid option arguments to ensure that files are transferred to and from the Data Integrator local directory (specified in Data Integrator). This might require that the remote file and directory name be specified as arguments and then sent to the Data Integrator Designer interface using Data Integrator system variables.

Web log support

Web logs are flat files generated by Web servers and are used for business intelligence. Web logs typically track details of Web site hits such as:

- Client domain names or IP addresses
- User names
- Timestamps
- Requested action (might include search string)
- Bytes transferred
- Referred address
- Cookie ID

Web logs use a common file format and an extended common file format.

Common Web Log Format

```
151.99.190.27 - - [01/Jan/1997:13:06:51 -0600]
"GET /~bacuslab HTTP/1.0" 301 -4
```

Extended Common Web Log Format

```
saturn5.sun.com - - [25/Jun/1998:11:19:58 -0500]
"GET /wew/js/mouseover.html HTTP/1.0" 200 1936
"http://av.yahoo.com/bin/query?p=mouse+over+javascript+source+code&hc=0&hs=0"
"Mozilla/4.02 [en] (X11; U; SunOS 5.6 sun4m)"
```

Data Integrator supports both common and extended common Web log formats as sources. The file format editor also supports the following:

- Dash as NULL indicator
- Time zone in date-time, e.g. 01/Jan/1997:13:06:51 –0600

Data Integrator includes several functions for processing Web log data:

- Word_ext function
- Concat_date_time function
- WL_GetKeyValue function

Word_ext function

The `word_ext` is a Data Integrator string function that extends the word function by returning the word identified by its position in a delimited string. This function is useful for parsing URLs or file names.

FORMAT

```
word_ext(string, separator(s), word_number)
```

A negative word number means count from right to left

EXAMPLES

`word_ext('www.bodi.com', '.', 2)` returns `'bodi'`.

`word_ext('www.cs.wisc.edu', '.', -2)` returns `'wisc'`.

`word_ext('www.cs.wisc.edu', '.', 5)` returns `NULL`.

`word_ext('aaa+=bbb+=ccc+zz=dd', '+=', 4)` returns `'zz'`. If 2 separators are specified (+=), the function looks for either one.

`word_ext(',,,,,aaa,,,,bb,,,c ', '.', 2)` returns `'bb'`. This function skips consecutive delimiters.

Concat_date_time function

The concat_date_time is a Data Integrator date function that returns a datetime from separate date and time inputs.

FORMAT

```
concat_date_time(date, time)
```

EXAMPLE

```
concat_date_time(MS40."date",MS40."time")
```

WL_GetKeyValue function

The WL_GetKeyValue is a custom function (written in the Data Integrator Scripting Language) that returns the value of a given keyword. It is useful for parsing search strings.

FORMAT

```
WL_GetKeyValue(string, keyword)
```

EXAMPLE

A search in Google for `bodi B2B` is recorded in a Web log as:

```
GET "http://www.google.com/search?hl=en&lr=&
safe=off&q=bodi+B2B&btnG=Google+Search"
```

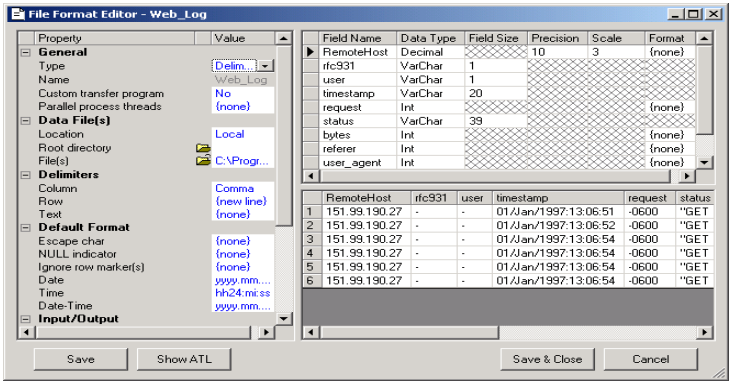
`WL_GetKeyValue('http://www.google.com/searc h?hl=en&lr=&safe=off&q=bodi+B2B&btnG=Google +Search','q')` returns `'bodi+B2B'`.

Sample Web log formats in Data Integrator

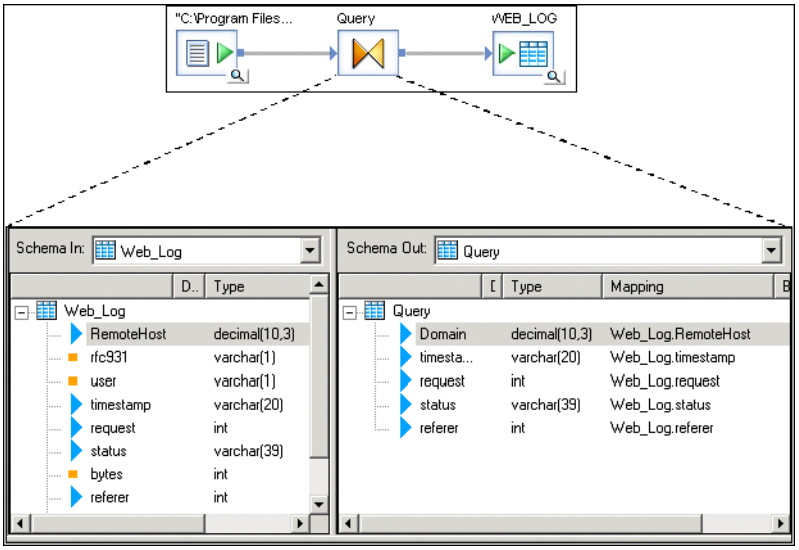
Below is a file with a common Web log file format:

```
151.99.190.27 - - [01/Jan/1997:13:06:51 -0600]
"GET /~bacuslab HTTP/1.0" 301 -4

151.99.190.27 - - [01/Jan/1997:13:06:52 -0600]
"GET /~bacuslab/ HTTP/1.0" 200 1779

151.99.190.27 - - [01/Jan/1997:13:06:54 -0600]
"GET /~bacuslab/BLI_Logo.jpg HTTP/1.0" 200 8210

151.99.190.27 - - [01/Jan/1997:13:06:54 -0600]
"GET /~bacuslab/BulletA.gif HTTP/1.0" 200 1151

151.99.190.27 - - [01/Jan/1997:13:06:54 -0600]
"GET /~bacuslab/Email4.gif HTTP/1.0" 200 3218

151.99.190.27 - - [01/Jan/1997:13:06:54 -0600]
"GET /~bacuslab/HomeCount.xbm HTTP/1.0" 200 890
```

Below is the file format editor view of this Web log:



Below is a representation of a sample data flow for this Web log. Data flows are described in Chapter 7, "Data Flows".



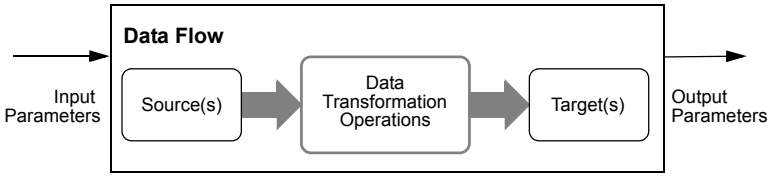
7 Data Flows

This chapter contains the following sections:

What is a data flow?

Data flows extract, transform, and load data. Everything having to do with data, including reading sources, transforming data, and loading targets, occurs inside a data flow. The lines connecting objects in a data flow represent the flow of data through data transformation steps.

After you define a data flow, you can add it to a job or work flow. From inside a work flow, a data flow can send and receive information to and from other objects through input and output parameters.



Naming data flows

Data flow names can include alphanumeric characters and underscores (_). They cannot contain blank spaces.

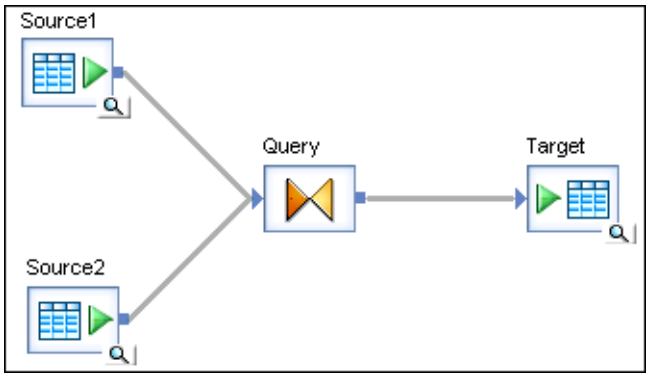
Data flow example

Suppose you want to populate the fact table in your data warehouse with new data from two tables in your source transaction database.

Your data flow consists of the following:

- Two source tables
- A join between these tables, defined in a query transform
- A target table where the new rows are placed

You indicate the flow of data through these components by connecting them in the order that data moves through them. The resulting data flow looks like the following:



Steps in a data flow

Each icon you place in the data flow diagram becomes a step in the data flow. This chapter discusses objects that you can use as steps in a data flow:

- Source and target objects
- Transforms

The connections you make between the icons determine the order in which Data Integrator completes the steps.

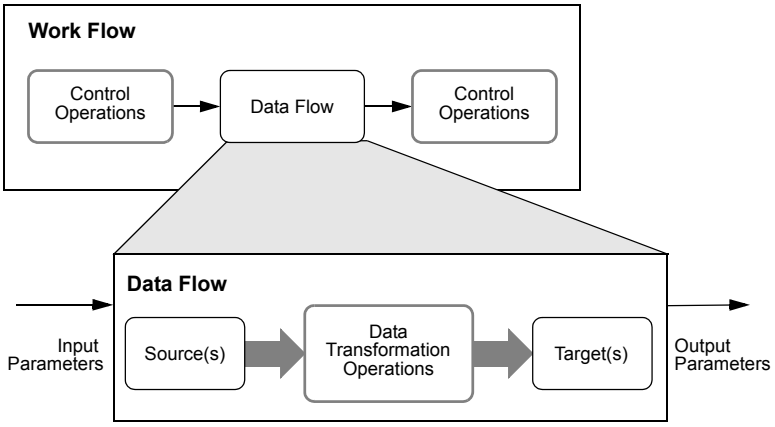
Data flows as steps in work flows

Data flows are closed operations, even when they are steps in a work flow. Any data set created within a data flow is not available to other steps in the work flow.

A work flow does not operate on data sets and cannot provide more data to a data flow; however, a work flow can do the following:

- Call data flows to perform data movement operations

- Define the conditions appropriate to run data flows
- Pass parameters to and from data flows



Intermediate data sets in a data flow

Each step in a data flow—up to the target definition—produces an intermediate result (for example, the results of a SQL statement containing a WHERE clause), which flows to the next step in the data flow. The intermediate result consists of a set of rows from the previous operation and the schema in which the rows are arranged. This result is called a *data set*. This data set may, in turn, be further "filtered" and directed into yet another data set.

Operation codes

Each row in a data set is flagged with an *operation code* that identifies the status of the row. The operation codes are as follows:

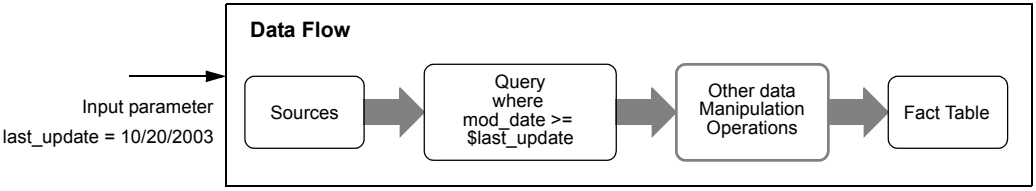
Operation code	Description
NORMAL	Creates a new row in the target. All rows in a data set are flagged as NORMAL when they are extracted from a source. If a row is flagged as NORMAL when loaded into a target, it is inserted as a new row in the target.
INSERT	Creates a new row in the target. Rows can be flagged as INSERT by transforms in the data flow to indicate that a change occurred in a data set as compared with an earlier image of the same data set. The change is recorded in the target separately from the existing data.
DELETE	Is ignored by the target. Rows flagged as DELETE are not loaded. Rows can be flagged as DELETE only by the Map_Operation transform.
UPDATE	Overwrites an existing row in the target. Rows can be flagged as UPDATE by transforms in the data flow to indicate that a change occurred in a data set as compared with an earlier image of the same data set. The change is recorded in the target in the same row as the existing data.

Passing parameters to data flows

Data does not flow outside a data flow, not even when you add a data flow to a work flow. You can, however, pass parameters into and out of a data flow. Parameters evaluate single values rather than sets of values.

When a data flow receives parameters, the steps inside the data flow can reference those parameters as variables.

Parameters make data flow definitions more flexible. For example, a parameter can indicate the last time a fact table was updated. You can use this value in a data flow to extract only rows modified since the last update. The following figure shows the parameter `last_update` used in a query to determine the data set used to load the fact table.



For more information about parameters, see "Variables and Parameters" on page 289.

Creating and defining data flows

You can create data flows using objects from:

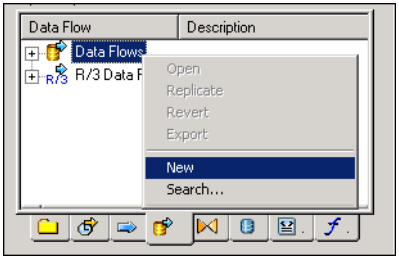
- The object library
- The tool palette

After creating a data flow, you can specify that a batch job only execute the data flow one time, even if the data flow appears in the job multiple times.

➢ To define a new data flow using the object library

1. In the object library, go to the **Data Flows** tab.

2. Select the data flow category, right-click and select **New**.



3. Select the new data flow.
4. Drag the data flow into the workspace for a job or a work flow.
5. Add the sources, transforms, and targets you need.

➢ To define a new data flow using the tool palette

1. Select the data flow icon in the tool palette. 

2. Click the workspace for a job or work flow to place the data flow.

   You can add data flows to batch and real-time jobs. When you drag a data flow icon into a job, you are telling Data Integrator to validate these objects according the requirements of the job type (either batch or real-time).

3. Add the sources, transforms, and targets you need.

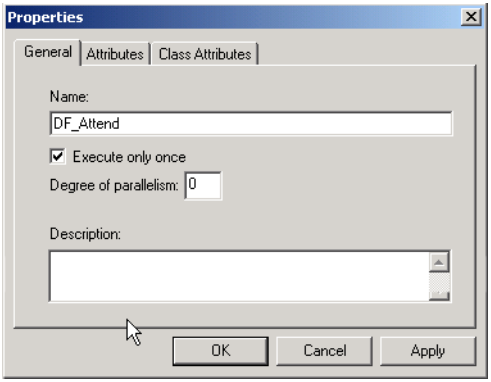
➢ To specify that a batch job executes the data flow one time

When you specify that a data flow should only execute once, a batch job will never re-execute that data flow after the data flow completes successfully, even if the data flow is contained in a work flow that is a recovery unit that re-executes. Business Objects recommends that you do not mark a data flow as **Execute only once** if a parent work flow is a recovery unit.

1. Right-click the data flow and select **Properties**.

   The Properties window opens for the data flow.

2. Select the **Execute only once** check box.



3. Click **OK**.

Source and target objects

A data flow directly reads and loads data using two types of objects:

Source objects — Define sources to read data from

Target objects — Define targets to write (or load) data to

Source objects

Source objects represent data sources read from data flows.

Source object	Description	Data Integrator access
Table	A file formatted with columns and rows as used in relational databases	Direct or through adapter
Template table	A template table that has been created and saved in another data flow (used in development). See "Template tables" on page 165.	Direct
File	A delimited or fixed-width flat file	Direct
Document	A file with an application-specific format (not readable by SQL or XML parser)	Through adapter
XML file	A file formatted with XML tags	Direct
XML message	Used as a source in real-time jobs. See "Real-time source and target objects" on page 252).	Direct

If you have the SAP licensed extension, you can also use IDocs as sources. For more information, see the "IDoc sources in real-time jobs" on page 135 of the *Data Integrator Supplement for SAP*.

Target objects

Target objects represent data targets that can be written to in data flows.

Target object	Description	Data Integrator access
Table	A file formatted with columns and rows as used in relational databases	Direct or through adapter
Template table	A table whose format is based on the output of the preceding transform (used in development)	Direct
File	A delimited or fixed-width flat file	Direct
Document	A file with an application-specific format (not readable by SQL or XML parser)	Through adapter
XML file	A file formatted with XML tags	Direct
XML template file	An XML file whose format is based on the preceding transform output (used in development, primarily for debugging data flows)	Direct
XML message	See "Real-time source and target objects" on page 252).	
Outbound message	See "Real-time source and target objects" on page 252).	

If you have the SAP licensed extension, you can also use IDocs as targets. For more information, see the "IDoc targets in real-time jobs" on page 147 of the *Data Integrator Supplement for SAP*.

Adding source or target objects to data flows

Fulfill the following prerequisites before using a source or target object in a data flow:

For	Prerequisite	Reference
Tables accessed directly from a database	Define a custom datastore and import table metadata.	"Custom datastores" on page 80
Template tables	Define a custom datastore.	"Template tables" on page 165
Files	Define a file format and import the file	"File Formats" on page 117
XML files and messages	Import an XML file format	"To import a DTD or XML Schema format" on page 231
Objects accessed through an adapter	Define an adapter datastore and import object metadata.	"Adapter datastores" on page 106

➢ To add a source or target object to a data flow

1. Open the data flow in which you want to place the object.
2. If the object library is not already open, select **Tools** > **Object Library** to open it.
3. Select the appropriate object library tab:
   - **Formats** tab for flat files, DTDs, or XML Schemas
   - **Datastores** tab for database and adapter objects
4. Select the object you want to add as a source or target. (Expand collapsed lists by clicking the plus sign next to a container icon.)

   For a new template table, select the Template Table icon from the tool palette.

   For a new XML template file, select the Template XML icon from the tool palette.
5. Drop the object in the workspace.
6. For objects that can be either sources or targets, when you release the cursor, a popup menu appears. Select the kind of object to make.

   For new template tables and XML template files, when you release the cursor, a secondary window appears. Enter the requested information for the new template object. Names can include alphanumeric characters and underscores (_). Template tables cannot have the same name as an existing table within a datastore.
7. The source or target object appears in the workspace.

8. Click the object name in the workspace

   Data Integrator opens the editor for the object. Set the options you require for the object.

NOTE: Ensure that any files that reference flat file, DTD, or XML Schema formats are accessible from the Job Server where the job will be run and specify the file location relative to this computer.

Template tables

During the initial design of an application, you might find it convenient to use template tables to represent database tables. With template tables, you do not have to initially create a new table in your DBMS and import the metadata into Data Integrator. Instead, Data Integrator automatically creates the table in the database with the schema defined by the data flow when you execute a job.

After creating a template table as a target in one data flow, you can use it as a source in other data flows. Though a template table can be used as a source table in multiple data flows, it can only be used as a target in one data flow.

Template tables are particularly useful in early application development when you are designing and testing a project. If you modify and save the data transformation operation in the data flow where the template table is a target, the schema of the template table automatically changes. Any updates to the schema are automatically made to any other instances of the template table. During the validation process, Data Integrator warns you of any errors such as those resulting from changing the schema.

After you are satisfied with the design of your data flow, save it. When the job is executed, Data Integrator uses the template table to create a new table in the database you specified when you created the template table. Once a template table is created in the database, you can convert the template table in the repository to a regular table.

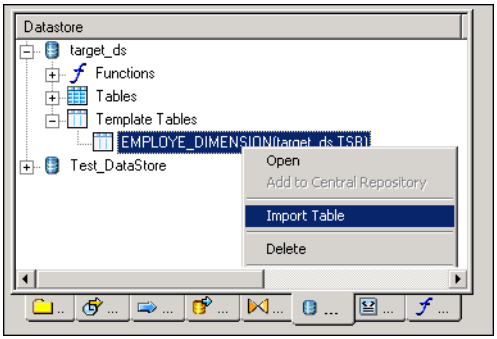
Once a template table is converted, you can no longer alter the schema. Note that you must convert template tables to take advantage of some features such as bulk loading. Other features, such as exporting an object, are available for template tables.

➢ To convert a template table into a regular table from the object library

1. Open the object library and go to the **Datastores** tab.
2. Click the plus sign (+) next to the datastore that contains the template table you want to convert.

   A list of objects appears.
3. Click the plus sign (+) next to **Template Tables**.

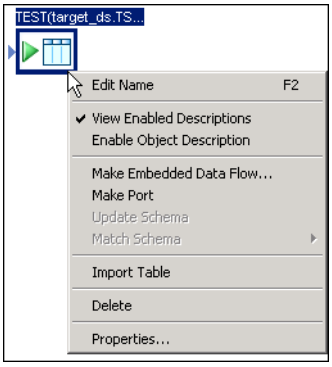
   The list of template tables appears.
4. Right-click a template table you want to convert and select **Import Table**.



Data Integrator converts the template table in the repository into a regular table by importing it from the database. To update the icon in all data flows, choose **View** > **Refresh**. In the datastore object library, the table is now listed under Tables rather than Template Tables.

➢ To convert a template table into a regular table from a data flow

1. Open the data flow containing the template table.
2. Right-click on the template table you want to convert and select **Import Table**.



After a template table is converted into a regular table, you can no longer change the table's schema.

Transforms

Data Integrator includes objects called *transforms*. Transforms operate on data sets. Transforms manipulate input sets and produce one or more output sets. By contrast, *functions* operate on single values in specific columns in a data set.

Data Integrator includes several built-in transforms. These transforms are available from the object library on the **Transforms** tab. See Chapter 5, "Transforms," in the *Data Integrator Reference Guide* for detailed information about these transforms.

Transform	Description
Case	The Case transform simplifies branch logic in data flows by consolidating case or decision-making logic in one transform. Paths are defined in an expression table.
Date_Generation	Generates a column filled with date values based on the start and end dates and an increment that you provide.
Effective_Date	Generates an additional "effective to" column based on the primary key's "effective date."
Hierarchy_Flattening	Flattens hierarchical data into relational tables so that it can participate in a star schema. Hierarchy_flattening can be both vertical and horizontal.
History_Preserving	Converts rows flagged as UPDATEs to INSERT rows so that the original values are preserved in the target. You specify in which column to look for updated data.
Key_Generation	Generates new keys for source data, starting from existing key values in the table you specify.
Map_Operation	Allows conversions between operation codes (map UPDATE to INSERT).
Merge	Inserts the rows from two or more sources into a single target.
Pivot (Columns to Rows)	Rotates the values in specified columns to rows.
Query	Retrieves a data set that satisfies conditions that you specify (similar to a SQL SELECT statement) and can perform joins, transformations, and functions on that data.
Pivot (Columns to Rows)	Rotates the values in specified rows to columns.
Row_Generation	Generates a data set from a single column. The new column values start from zero and increment by one to the end value you specify.
SQL	Performs the indicated SQL query operation. Use this transform when the SQL operation you want cannot be performed by native transforms.
Table_Comparison	Compares two data sets and produces the difference between them as a data set with rows flagged as INSERT and UPDATE.

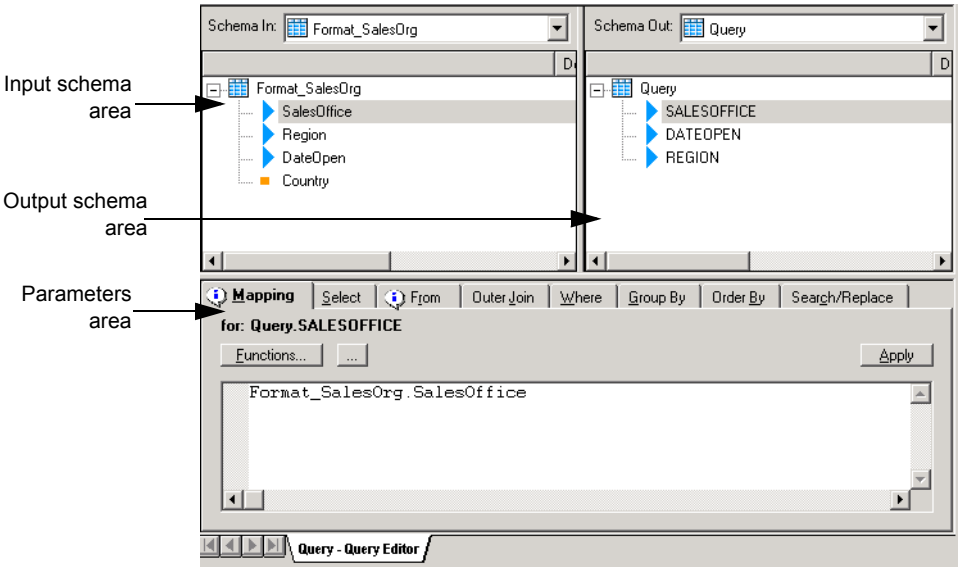
Transform editors

Transform editors can have two areas:

- An input schema area and/or output schema area

- A parameters area (or options area) that allows you to set all the values the transform requires

Here is an example of the Query transform editor.



For general information about editors, see "Object editors" on page 43. For specific information about the editor for a particular transform, see Chapter 5, "Transforms," in the *Data Integrator Reference Guide*.

Adding transforms to data flows

Use the Designer to add transforms to data flows.

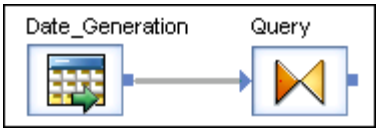
➢ To add a transform to a data flow

1. Open a data flow object.

   From the work flow, click the data flow name. Alternatively, from the object library, click the **Data Flow** tab and double-click the data flow name.

2. Open the object library if it is not already open.
3. Go to the **Transforms** tab.
4. Select the transform you want to add to the data flow.
5. Drag the transform icon into the data flow workspace.
6. Draw the data flow connections.

   To connect a source to a transform, click the arrow on the right edge of the source and drag the cursor to the arrow on the left edge of the transform.



   Continue connecting inputs and outputs as required for the transform.

   - The input for the transform might be the output from another transform or the output from a source; or, the transform may not require source data.
   - You can connect the output of the transform to the input of another transform or target.

7. Click the name of the transform.

   This opens the transform editor, which lets you complete the definition of the transform.

8. Enter option values.

   To specify a data column as a transform option, enter the column name as it appears in the input schema or drag the column name from the input schema into the option box.

To specify dates or times as option values, use the following formats:

Format	Description
yyyy.mm.dd	Where yyyy represents a four-digit year, mm represents a two-digit month, and dd represents a two-digit day
hh.mi.ss	Where hh represents the two hour digits, mi the two minute digits, and ss the two second digits of a time

Query transform overview

Query transform

The Query transform is by far the most commonly used transform, so this section provides an overview. For a full description, see Chapter 5, "Transforms," in the *Data Integrator Reference Guide*.

The Query transform can perform the following operations:

- Choose (filter) the data to extract from sources
- Join data from multiple sources
- Map columns from input to output schemas
- Perform transformations and functions on the data
- Perform data nesting and unnesting (see "Nested Data" on page 205)
- Add new columns, nested schemas, and function results to the output schema
- Assign primary keys to output columns

Adding a Query transform to a data flow

Because it is so commonly used, the Query transform icon is included in the tool palette, providing an easier way to add a Query transform.

➢ To add a Query transform to a data flow

1. Click the Query icon in the tool palette.
2. Click anywhere in a data flow workspace.
3. Connect the Query to inputs and outputs.
   - The inputs for a Query can include the output from another transform or the output from a source.
   - The outputs from a Query can include input to another transform or input to a target.

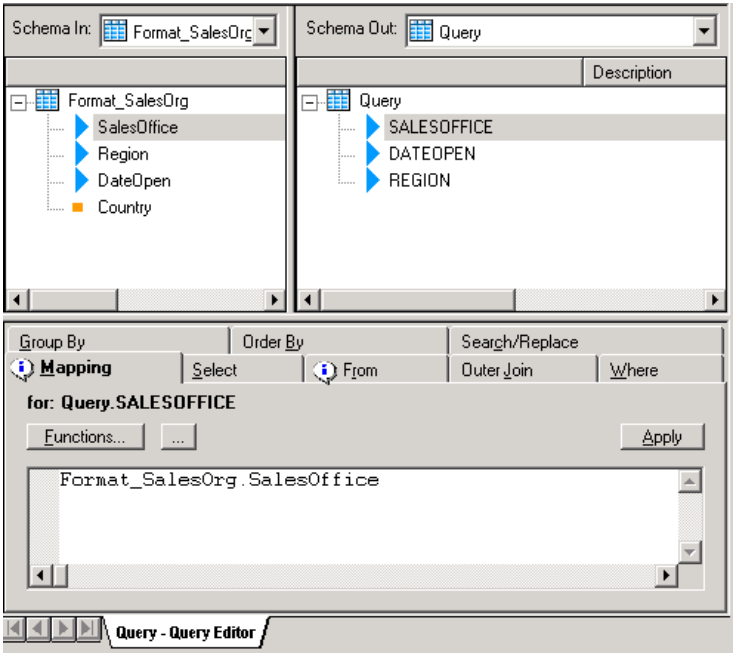
If you connect a target table to a Query with an empty output schema, Data Integrator automatically fills the Query's output schema with the columns from the target table, without mappings.

Query editor

The query editor, a graphical interface for performing query operations, contains the following areas:

- Input schema area (upper left)
- Output schema area (upper right)
- Parameters area (lower tabbed area)

  The "i" icon indicates tabs containing user-defined entries.



The input and output schema areas can contain:

- Columns
- Nested schemas
- Functions (output only)

The Schema In and Schema Out lists display the currently selected schema in each area. The currently selected output schema is called the *current schema* and determines:

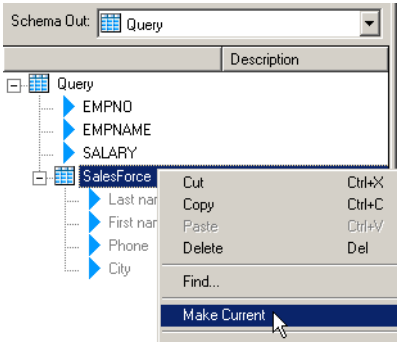
- The output elements that can be modified (added, mapped, or deleted)
- The scope of the **Select** through **Order by** tabs in the parameters area

The current schema is highlighted while all other (non-current) output schemas are gray.

➢ To change the current schema

You can change the current schema in the following ways:

- Select a schema from the Output list.
- Right-click a schema, column, or function in the output schema area and select **Make Current**.
- Double-click one of the non-current (grayed-out) elements in the output schema area.



➢ To modify output schema contents

You can modify the output schema in several ways:

- Drag and drop (or copy and paste) columns or nested schemas from the input schema area to the output schema area to create simple mappings.
- Use right-click menu options on output elements to:
  - Add new output columns and schemas
  - Use adapter functions or (with the SAP license extension) SAP R/3 functions to generate new output columns
  - Assign or reverse primary key settings on output columns. Primary key columns are flagged by a key icon.
  - Unnest or re-nest schemas.
- Use the **Mapping** tab to provide complex column mappings. Drag and drop input schemas and columns into the output schema to enable the editor. Use the function wizard and the expression editor to build expressions. When the text editor is enabled, you can access these features using the buttons above the editor.

- Use the **Select** through **Order By** tabs to provide additional parameters for the current schema (similar to SQL SELECT statement clauses). You can drag and drop schemas and columns into these areas.

Select	Specifies whether to output only distinct rows (discarding any identical duplicate rows).
From	Specifies all input schemas that are used in the current schema.
Outer Join	Specifies an inner table and outer table for any joins (in the **Where** sheet) that are to be treated as outer joins.

Where	Specifies conditions that determine which rows are output.
	The syntax is like an SQL SELECT WHERE clause; for example:
	`TABLE1.EMPNO = TABLE2.EMPNO`
	`AND TABLE1.EMPNO > 1000`
	`OR TABLE2.EMPNO < 9000`
	Use the buttons above the editor to build expressions.

Group By	Specifies how the output rows are grouped (if required).
Order By	Specifies how the output rows are sequenced (if required).

- Use the **Search** tab to locate input and output elements containing a specific word or term.

Data flow execution

A data flow is a *declarative specification* from which Data Integrator determines the correct data to process. For example in data flows placed in batch jobs, the transaction order is to extract, transform, then load data into a target. Data flows are similar to SQL statements. The specification declares the desired output.

From the information in the specification, Data Integrator produces output while optimizing performance. For example, for SQL sources and targets, Data Integrator creates database-specific SQL statements based on a job's data flow diagrams. To optimize performance, Data Integrator pushes down as many transform operations as possible to the source or target database and combines as many operations as possible into one request to the database. For example, Data Integrator tries to push down joins and function evaluations. By pushing down operations to the database, Data Integrator reduces the number of rows and operations that the Data Integrator engine must process.

Data flow design influences the number of operations that Data Integrator can push to the source or target database. Before running a job, you can examine the SQL that Data Integrator generates and alter your design to produce the most efficient results. See "Maximizing the number of pushed-down operations" on page 31 of the *Data Integrator Performance Optimization Guide* for more information.

Data Integrator executes a data flow each time the data flow occurs in a job. However, you can specify that a batch job execute a particular data flow only one time. In that case, Data Integrator only executes the first occurrence of the data flow; Data Integrator skips subsequent occurrences in the job.

You might use this feature when developing complex batch jobs with multiple paths, such as jobs with try/catch blocks or conditionals, and you want to ensure that Data Integrator only executes a particular data flow one time.

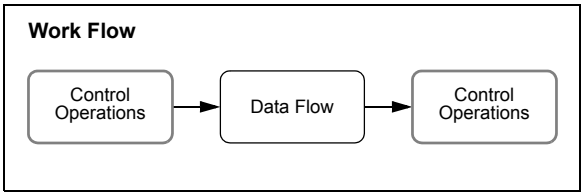
See "Creating and defining data flows" on page 159 for information on how to specify that a job execute a data flow only one time.

8 Work Flows

This chapter contains the following sections:

What is a work flow?

A *work flow* defines the decision-making process for executing data flows. For example, elements in a work flow can determine the path of execution based on a value set by a previous job or can indicate an alternative path if something goes wrong in the primary path. Ultimately, the purpose of a work flow is to prepare for executing data flows and to set the state of the system after the data flows are complete.



Jobs (introduced in Chapter 4, "Projects and Jobs") are special work flows. Jobs are special because you can execute them. Almost all of the features documented for work flows also apply to jobs, with one exception: jobs do not have parameters.

Steps in a work flow

Work flow steps take the form of icons that you place in the work space to create a *work flow diagram*. The following objects can be elements in work flows:

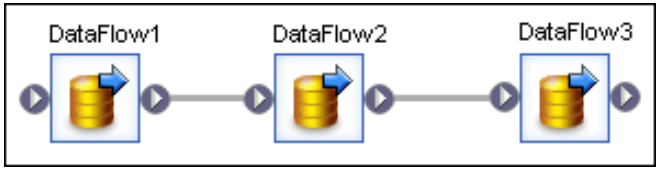
- Work flows
- Data flows
- Scripts
- Conditionals
- While loops
- Try/catch blocks

Work flows can call other work flows, and you can nest calls to any depth. A work flow can also call itself.

The connections you make between the icons in the workspace determine the order in which work flows execute, unless the jobs containing those work flows execute in parallel.

Order of execution in work flows

Steps in a work flow execute in a left-to-right sequence indicated by the lines connecting the steps. Here is the diagram for a work flow that calls three data flows:

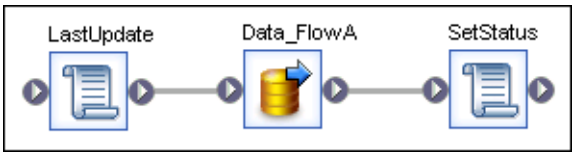


Note that Data_Flow1 has no connection from the left but is connected on the right to the left edge of Data_Flow2 and that Data_Flow2 is connected to Data_Flow3. There is a single thread of control connecting all three steps. Execution begins with Data_Flow1 and continues through the three data flows.

Connect steps in a work flow when there is a dependency between the steps. If there is no dependency, the steps need not be connected. In that case, Data Integrator can execute the independent steps in the work flow as separate processes. In the following work flow, Data Integrator executes data flows 1 through 3 in parallel:

To execute more complex work flows in parallel, define each sequence as a separate work flow, then call each of the work flows from another work flow as in the following example:

Define work flow A



Define work flow B

Call work flows A and B from work flow C

You can specify that a job execute a particular work flow or data flow only one time. In that case, Data Integrator only executes the first occurrence of the work flow or data flow; Data Integrator skips subsequent occurrences in the job. You might use this feature when developing complex jobs with multiple paths, such as jobs with try/catch blocks or conditionals, and you want to ensure that Data Integrator only executes a particular work flow or data flow one time.

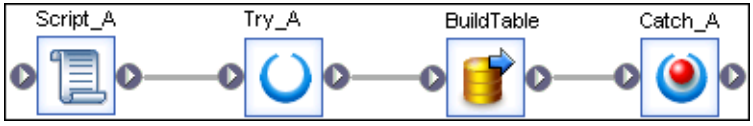
Example of a work flow

Suppose you want to update a fact table. You define a data flow in which the actual data transformation takes place. However, before you move data from the source, you want to determine when the fact table was last updated so that you only extract rows that have been added or changed since that date.

You need to write a script to determine when the last update was made. You can then pass this date to the data flow as a parameter.

In addition, you want to check that the data connections required to build the fact table are active when data is read from them. To do this in Data Integrator, you define a try/catch block. If the connections are not active, the catch runs a script you wrote, which automatically sends mail notifying an administrator of the problem.

Scripts and error detection cannot execute in the data flow. Rather, they are steps of a decision-making process that influences the data flow. This decision-making process is defined as a work flow, which looks like the following:



Data Integrator executes these steps in the order that you connect them.

Creating work flows

You can create work flows using one of two methods:

- Object library
- Tool palette

After creating a work flow, you can specify that a job only execute the work flow one time, even if the work flow appears in the job multiple times.

➢ To create a new work flow using the object library

1. Open the object library.
2. Go to the **Work Flows** tab.
3. Right-click and choose **New**.
4. Drag the work flow into the diagram.
5. Add the data flows, work flows, conditionals, try/catch blocks, and scripts that you need.

➢ To create a new work flow using the tool palette

1. Select the work flow icon in the tool palette.
2. Click where you want to place the work flow in the diagram.

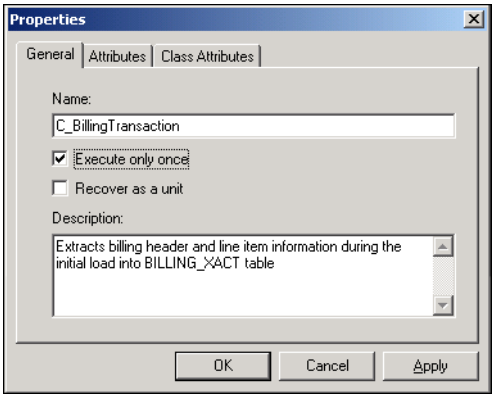
If more than one instance of a work flow appears in a job, you can improve execution performance by running the work flow only one time.

➢ To specify that a job executes the work flow one time

When you specify that a work flow should only execute once, a job will never re-execute that work flow after the work flow completes successfully, even if the work flow is contained in a work flow that is a recovery unit that re-executes. Business Objects recommends that you not mark a work flow as **Execute only once** if the work flow or a parent work flow is a recovery unit.

1. Right click on the work flow and select **Properties**.

   The Properties window opens for the work flow.



2. Select the **Execute only once** check box.
3. Click **OK**.

Conditionals

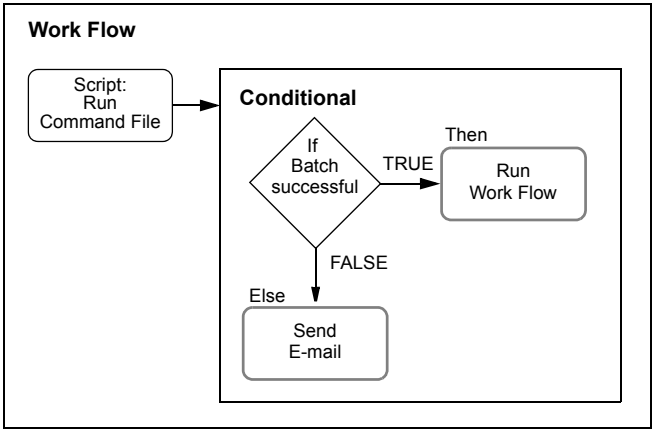
Conditionals are single-use objects used to implement if/then/else logic in a work flow. Conditionals and their components (*if* expressions, *then* and *else* diagrams) are included in the scope of the parent control flow's variables and parameters.

To define a conditional, you specify a condition and two logical branches:

If	A Boolean expression that evaluates to TRUE or FALSE. You can use functions, variables, and standard operators to construct the expression.
Then	Work flow elements to execute if the **If** expression evaluates to TRUE.
Else	(Optional) Work flow elements to execute if the **If** expression evaluates to FALSE.

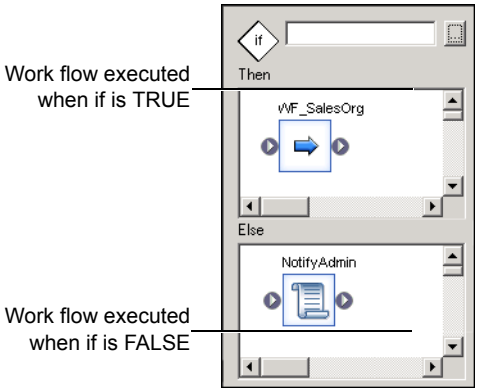
Define the **Then** and **Else** branches inside the definition of the conditional.

A conditional can fit in a work flow. Suppose you use a Windows command file to transfer data from a legacy system into Data Integrator. You write a script in a work flow to run the command file and return a success flag. You then define a conditional that reads the success flag to determine if the data is available for the rest of the work flow.



To implement this conditional in Data Integrator, you define two work flows—one for each branch of the conditional. If the elements in each branch are simple, you can define them in the conditional editor itself.

The definition of the conditional shows the two branches as follows:



Both the **Then** and **Else** branches of the conditional can contain any object that you can have in a work flow including other work flows, nested conditionals, try/catch blocks, and so on.

➢ To define a conditional

1. Define the work flows that are called by the **Then** and **Else** branches of the conditional.

   Business Objects recommends that you define, test, and save each work flow as a separate object rather than constructing these work flows inside the conditional editor.

2. Open the work flow in which you want to place the conditional.
3. Click the icon for a conditional in the tool palette.
4. Click the location where you want to place the conditional in the diagram.

   The conditional appears in the diagram.

5. Click the name of the conditional to open the conditional editor.
6. Click **if**.

7. Enter the Boolean expression that controls the conditional.

   Continue building your expression. You might want to use the function wizard or smart editor.

8. After you complete the expression, click **OK**.

9. Add your predefined work flow to the **Then** box.

   To add an existing work flow, open the object library to the **Work Flows** tab, select the desired work flow, then drag it into the **Then** box.

10. (Optional) Add your predefined work flow to the **Else** box.

    If the **If** expression evaluates to FALSE and the **Else** box is blank, Data Integrator exits the conditional and continues with the work flow.

11. After you complete the conditional, choose **Debug** > **Validate**.

    Data Integrator tests your conditional for syntax errors and displays any errors encountered.

12. The conditional is now defined. Click the **Back** button to return to the work flow that calls the conditional.

While loops

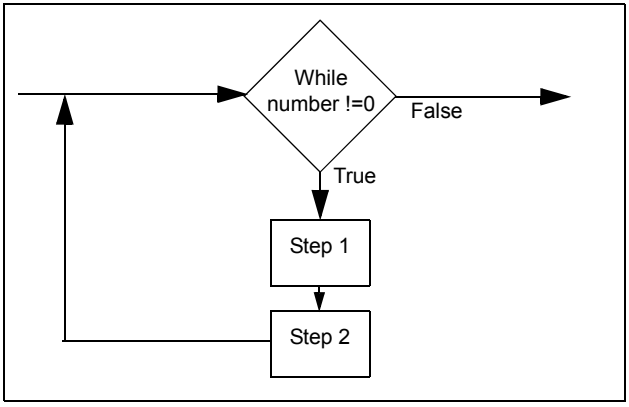
Use a while loop to repeat a sequence of steps in a work flow as long as a condition is true.

This section discusses:

- Design considerations
- Defining a while loop
- Using a while loop with View Data

Design considerations

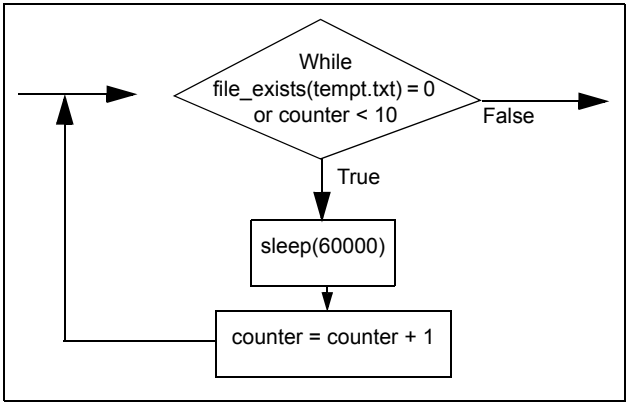
The while loop is a single-use object that you can use in a work flow. The while loop repeats a sequence of steps as long as a condition is true.



Typically, the steps done during the while loop result in a change in the condition so that the condition is eventually no longer satisfied and the work flow exits from the while loop. If the condition does not change, the while loop will not end.

For example, you might want a work flow to wait until the system writes a particular file. You can use a while loop to check for the existence of the file using the `file_exists` function. As long as the file does not exist, you can have the work flow go into sleep mode for a particular length of time, say one minute, before checking again.

Because the system might never write the file, you must add another check to the loop, such as a counter, to ensure that the while loop eventually exits. In other words, change the while loop to check for the existence of the file and the value of the counter. As long as the file does not exist and the counter is less than a particular value, repeat the while loop. In each iteration of the loop, put the work flow in sleep mode and then increment the counter.



Defining a while loop

You can define a while loop in any work flow.

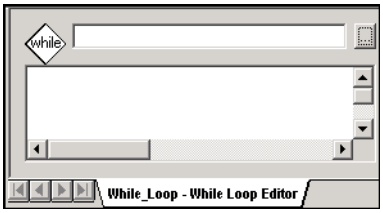
➢ To define a while loop

1. Open the work flow where you want to place the while loop.
2. Click the while loop icon on the tool palette.

3. Click the location where you want to place the while loop in the workspace diagram.

   The while loop appears in the diagram.

4. Click the while loop to open the while loop editor.



5. In the **While** box at the top of the editor, enter the condition that must apply to initiate and repeat the steps in the while loop.

   Alternatively, click  to open the expression editor, which gives you more space to enter an expression and access to the function wizard. Click **OK** after you enter an expression in the editor.

6. Add the steps you want completed during the while loop to the workspace in the while loop editor.

You can add any objects valid in a work flow including scripts, work flows, and data flows. Connect these objects to represent the order that you want the steps completed.

NOTE: Although you can include the parent work flow in the while loop, recursive calls can create an infinite loop.

7. After defining the steps in the while loop, choose **Debug > Validate**.

   Data Integrator tests your definition for syntax errors and displays any errors encountered.

8. Close the while loop editor to return to the calling work flow.

Using a while loop with View Data

When using View Data, a job stops when Data Integrator has retrieved the specified number of rows for all scannable objects.

Depending on the design of your job, Data Integrator might not complete all iterations of a while loop if you run a job in view data mode:

- If the while loop contains scannable objects and there are no scannable objects outside the while loop (for example, if the while loop is the last object in a job), then the job will complete after the scannable objects in the while loop are satisfied, possibly after the first iteration of the while loop.
- If there are scannable objects after the while loop, the while loop will complete normally. Scanned objects in the while loop will show results from the last iteration.
- If there are no scannable objects following the while loop but there are scannable objects completed in parallel to the while loop, the job will complete as soon as all scannable objects are satisfied. The while loop might complete any number of iterations.

Try/catch blocks

A *try/catch block* is a combination of one try object and one or more catch objects that allow you to specify alternative work flows if errors occur while Data Integrator is executing a job. Try/catch blocks:

- "Catch" classes of exceptions "thrown" by Data Integrator, the DBMS, or the operating system
- Apply solutions that you provide
- Continue execution

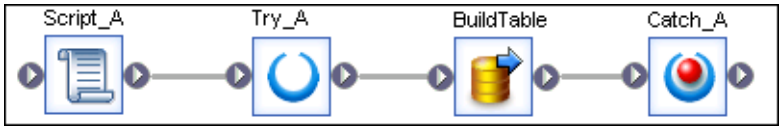
Try and catch objects are single-use objects.

Here's the general method to implement exception handling:

- Insert a try object before the steps for which you are handling errors.
- Insert one or more catches in the work flow after the steps.
- In each catch, do the following:
  - Indicate the error or group of errors that you want to catch.
  - Define the work flows that a thrown exception executes.

If an exception is thrown during the execution of a try/catch block and if no catch is looking for that exception, then the exception is handled by normal error logic.

The following work flow shows a try/catch block surrounding a data flow:



In this case, if the data flow BuildTable causes any system-generated exceptions handled by the catch, then the work flow defined in the catch executes.

The action initiated by the catch can be simple or complex. Here are some examples of possible exception actions:

- Send a prepared e-mail message to a system administrator.
- Rerun a failed work flow or data flow.
- Run a scaled-down version of a failed work flow or data flow.

➢ To define a try/catch block

1. Define the work flow that is called by each catch you expect to define.

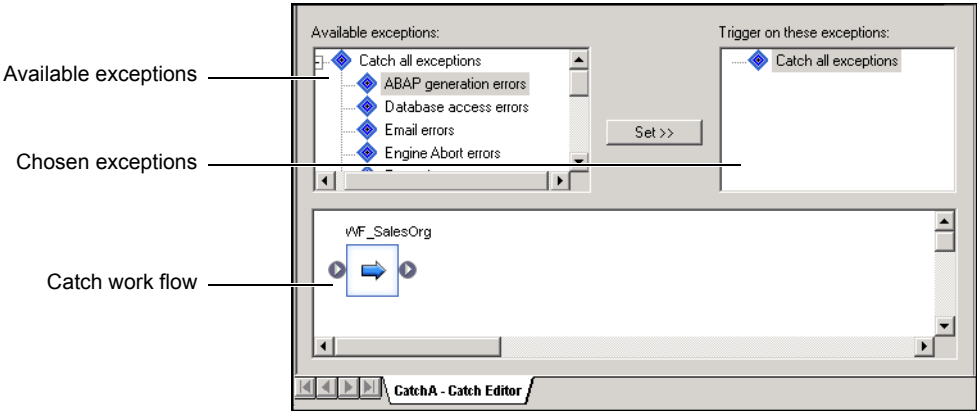
   We recommend that you define, test, and save each work flow as a separate object rather than constructing these work flows inside the catch editor.
2. Open the work flow that includes the try/catch block.
3. Click the try icon in the tool palette.
4. Click the location where you want to place the try in the diagram.

   The try icon appears in the diagram.

   NOTE: There is no editor for a try; the try merely initiates the try/catch block.
5. Click the catch icon in the tool palette.
6. Click the location where you want to place the catch in the diagram.

   The catch icon appears in the diagram.
7. Repeat steps 5 and 6 if you want to define a separate catch for other exceptions in this try/catch block.
8. Connect the try and catch to the objects they enclose.

9. Click the name of the object to open the catch editor.



10. Select an exception or a group of exceptions from the list of **Available Exceptions**. (For a complete list of available exceptions, see "Categories of available exceptions" on page 200.)

    Click the plus sign (+) next to an exception group to see the individual exceptions included in the group. You can select any single exception or any group of exceptions. Each catch supports one exception selection.

11. Click **Set**.

12. Repeat steps 10 and 11 until you have chosen all of the exceptions for this catch.

13. Add your predefined work flow to the catch work flow box.

    To add an existing work flow, open the object library to the **Work Flows** tab, select the desired work flow, and then drag it into the box.

14. After you have completed the catch, choose **Debug** > **Validate**.

    Data Integrator tests your definition for syntax errors and displays any errors encountered.

15. Click the **Back** button to return to the work flow that calls the catch.
16. Repeat steps 9 to 14 for each catch in the work flow.

If any exception listed in the catch occurs during the execution of this try/catch block, Data Integrator executes the catch work flow.

Categories of available exceptions

Categories of available exceptions include:

- Database access errors
- Email errors
- Engine abort errors
- Execution errors
- File access errors
- Microsoft connection errors
- Parser errors
- Predefined transform errors
- Repository access errors
- Resolver errors
- System exception errors
- User transform errors

Scripts

Scripts are single-use objects used to call functions and assign values to variables in a work flow.

For example, you can use the `SQL` function in a script to determine the most recent update time for a table and then assign that value to a variable. You can then assign the variable to a parameter that passes into a data flow and identifies the rows to extract from a source.

A script can contain the following statements:

- Function calls
- If statements
- While statements
- Assignment statements
- Operators

The basic rules for the syntax of the script are as follows:

- Each line ends with a semicolon (;).
- Variable names start with a dollar sign ($).
- String values are enclosed in single quotation marks (').
- Comments start with a pound sign (#).
- Function calls always specify parameters even if the function uses no parameters.

For example, the following script statement determines today's date and assigns the value to the variable `$TODAY`:

```
$TODAY = sysdate();
```

You cannot use variables unless you declare them in the work flow that calls the script.

For more information about scripts and the Business Objects scripting language, see Chapter 7, "Data Integrator Scripting Language," in the *Data Integrator Reference Guide*.

➢ To create a script

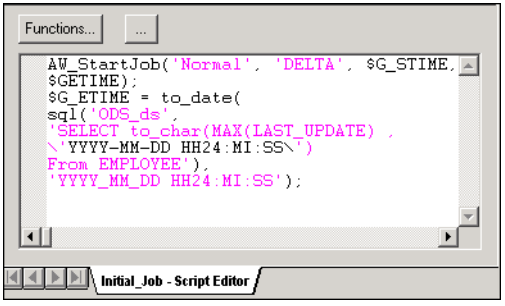
1. Open the work flow.
2. Click the script icon in the tool palette.
3. Click the location where you want to place the script in the diagram.

   The script icon appears in the diagram.
4. Click the name of the script to open the script editor.
5. Enter the script statements, each followed by a semicolon.

   Click the function button to include functions in your script.
6. After you complete the script, select **Debug** > **Validate**.

   Data Integrator tests your script for syntax errors and displays any errors encountered.

The following figure shows the script editor displaying a script that determines the start time from the output of a custom function.



➢ To save a script

If you want to save a script that you use often, create a custom function containing the script steps.

Debugging scripts using the print function

Data Integrator has a debugging feature that allows you to print:

- The values of variables and parameters during execution
- The execution path followed within a script

You can use the print function to write the values of parameters and variables in a work flow to the trace log. For example, this line in a script:

```
print ('The value of parameter $x: [$x]');
```

produces the following output in the trace log:

```
The following output is being printed via the
Print function in <Session job_name>.

The value of parameter $x: value
```

For details about the print function, see "print" on page 445 of the *Data Integrator Reference Guide*.

9 Nested Data

This chapter contains the following sections:

- What is nested data?
- Operations on nested data
- Formatting XML documents

If you do not plan to use nested data sources or targets, skip this chapter.

What is nested data?

Real-world data often has hierarchical relationships that are represented in a relational database with master-detail schemas using foreign keys to create the mapping. However, some data sets, such as XML documents and SAP R/3 IDocs, handle hierarchical relationships through *nested data*.

Data Integrator maps nested data to a separate schema *implicitly* related to a single row and column of the parent schema. This mechanism is called *Nested Relational Data Modelling (NRDM)*. NRDM provides a way to view and manipulate hierarchical relationships within data flow sources, targets, and transforms.

Sales orders are often presented using nesting: the line items in a sales order are related to a single header and are represented using a nested schema. Each row of the sales order data set contains a nested line item schema.

Order data set

OrderNo	CustID	ShipTo1	ShipTo2	LineItems
9999	1001	123 State St.	Town, CA	

Item	Qty	ItemPrice
001	2	10
002	4	5

Representing hierarchical data

You can represent the same hierarchical data in several ways. Examples include:

- Multiple rows in a single data set

Order data set

OrderNo	CustID	ShipTo1	ShipTo2	Item	Qty	ItemPrice
9999	1001	123 State St.	Town, CA	001	2	10
9999	1001	123 State St.	Town, CA	002	4	5

- Multiple data sets related by a join

Order header data set

OrderNo	CustID	ShipTo1	ShipTo2
9999	1001	123 State St.	Town, CA

Line-item data set

OrderNo	Item	Qty	ItemPrice
9999	001	2	10
9999	002	4	5

WHERE
Header.OrderNo = LineItem.OrderNo

- Nested data

Order data set

OrderNo	CustID	ShipTo1	ShipTo2	LineItems
9999	1001	123 State St.	Town, CA	

Item	Qty	ItemPrice
001	2	10
002	4	5

Using the nested data method can be more concise (no repeated information), and can scale to present a deeper level of hierarchical complexity. For example, columns inside a nested schema can also contain columns. There is a unique instance of each nested schema for each row at each level of the relationship.

Order data set

OrderNo	CustID	ShipTo1	ShipTo2	LineItems
9999	1001	123 State St.	Town, CA	

Item	ItemQty	ItemPrice
001	2	
002	4	

Qty	Sell Price
1	25
20	23

Qty	SellPrice
1	200
10	190

Generalizing further with nested data, each row at each level can have any number of columns containing nested schemas.

Order data set

OrderNo	LineItems	CustInfo
9999		

Item	ItemQty	ItemPrice
001	2	
002	4	

Qty	Sell Price
1	25
20	23

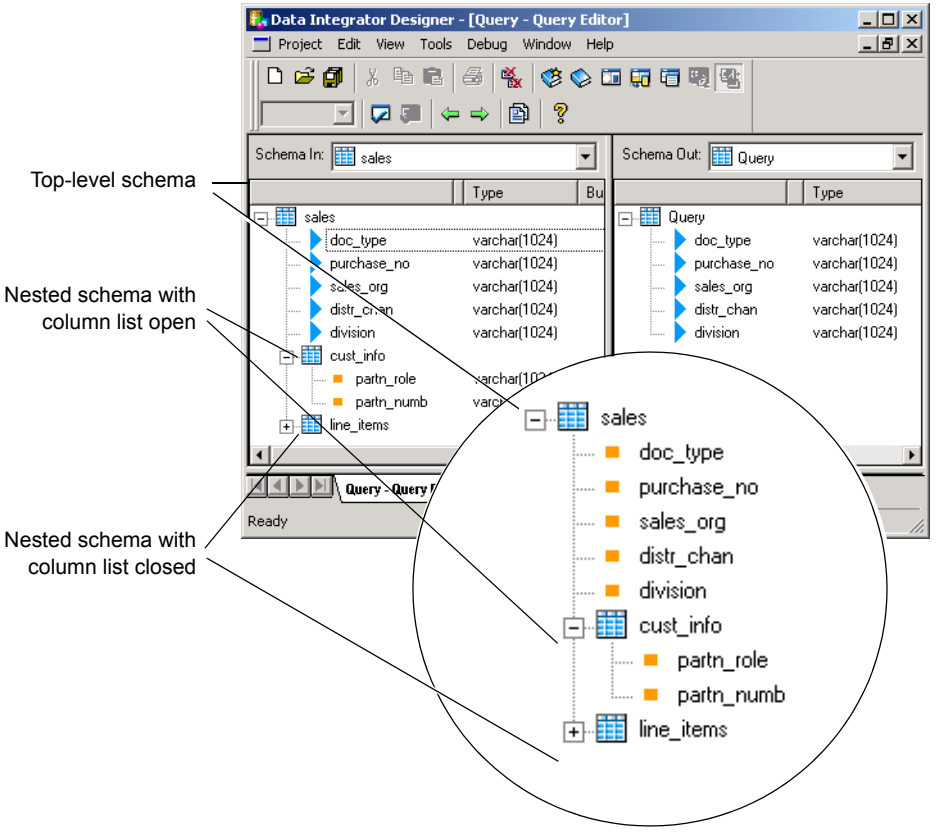
Qty	SellPrice
1	200
10	190

Type	CustID	Contacts
Ship	1001	
Bill	7777	

Name	Phone
Alvarez	555-1234
Tanaka	555-6666

Name	Phone
Samp	333-1234

In Data Integrator, you can see the structure of nested data in the input and output schemas of sources, targets, and transforms in data flows. Nested schemas appear with a schema icon paired with a plus sign, which indicates that the object contains columns. The structure of the schema shows how the data is ordered.



Operations on nested data

In Data Integrator, you use the Query transform to manipulate nested data. This section discusses:

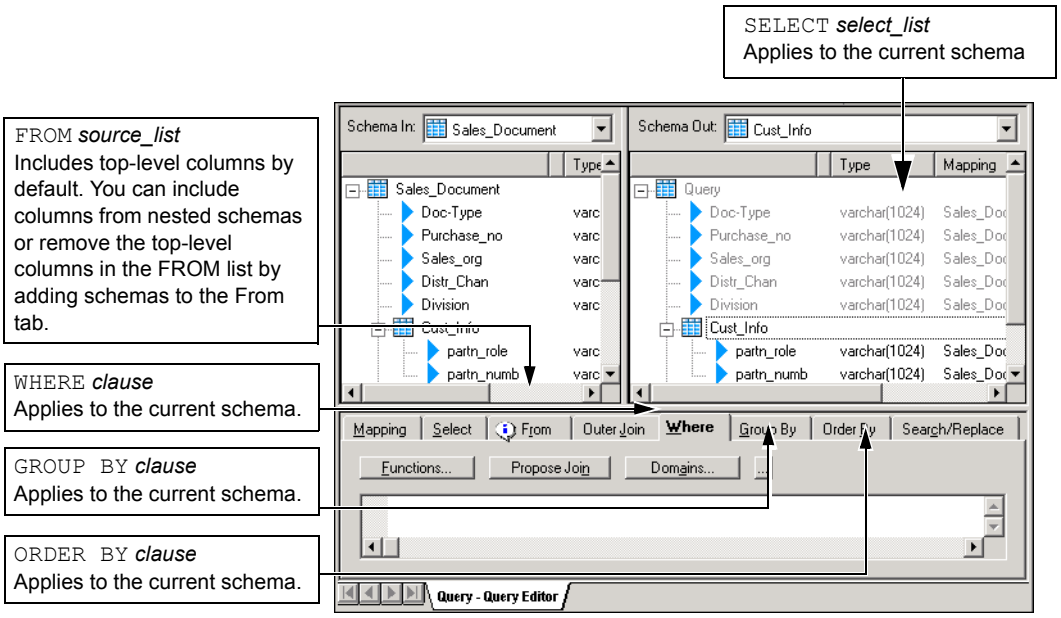
- Overview of nested data and the Query transform
- FROM clause construction
- Nesting columns
- Using correlated columns in nested data
- Distinct rows and nested data
- Grouping values across nested schemas
- Unnesting nested data
- How transforms handle nested data

Overview of nested data and the Query transform

With relational data, a Query transform allows you to execute a SELECT statement. The mapping between input and output schemas defines the project list for the statement. When working with nested data, the query provides an interface to perform SELECTs at each level of the relationship that you define in the output schema.

Without nested schemas, the Query transform assumes that the FROM clause in the SELECT statement contains the data sets that are connected as inputs to the query object. When working with nested data, you must explicitly define the FROM clause in a query. Data Integrator assists by setting the top-level inputs as the default FROM clause values for the top-level output schema.

The other SELECT statement elements defined by the query work the same with nested data as they do with flat data. However, because a SELECT statement can only include references to relational data sets, a query that includes nested data includes a SELECT statement to define operations for each parent and child schema in the output.



The parameters you enter apply only to the current schema (as displayed in the Output list at the top right). For information on setting the current schema and completing the parameters, see "Query editor" on page 174.

FROM clause construction

When you include a schema in the FROM clause, you indicate that all of the columns in the schema—including columns containing nested schemas—are available to be included in the output.

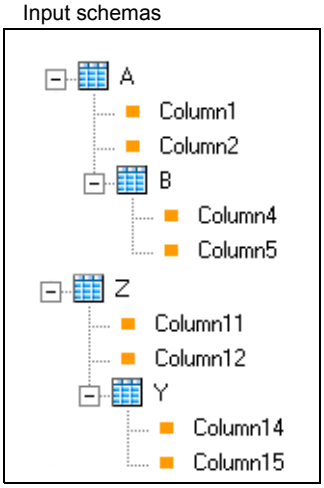
If you include more than one schema in the FROM clause, you indicate that the output will be formed from the cross product of the two schemas, constrained by the WHERE clause for the current schema.

These FROM clause descriptions and the behavior of the query are exactly the same with nested data as with relational data. The current schema allows you to distinguish multiple SELECT statements from each other within a single query. However, because the SELECT statements are dependent upon each other—and because the user interface makes it easy to construct arbitrary data sets—determining the appropriate FROM clauses for multiple levels of nesting can be complex.

A FROM clause can contain:

- Any top-level schema from the input
- Any schema that is a column of a schema in the FROM clause of the parent schema

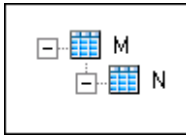
The FROM clauses form a path that can start at any level of the output. The first schema in the path must always be a top-level schema from the input.



Correct combination of FROM clauses for the two output schemas:

```
SELECT columns FROM A
   SELECT columns FROM A.B, Z
```

Output schema



Incorrect combination of FROM clauses for the two output: schemas

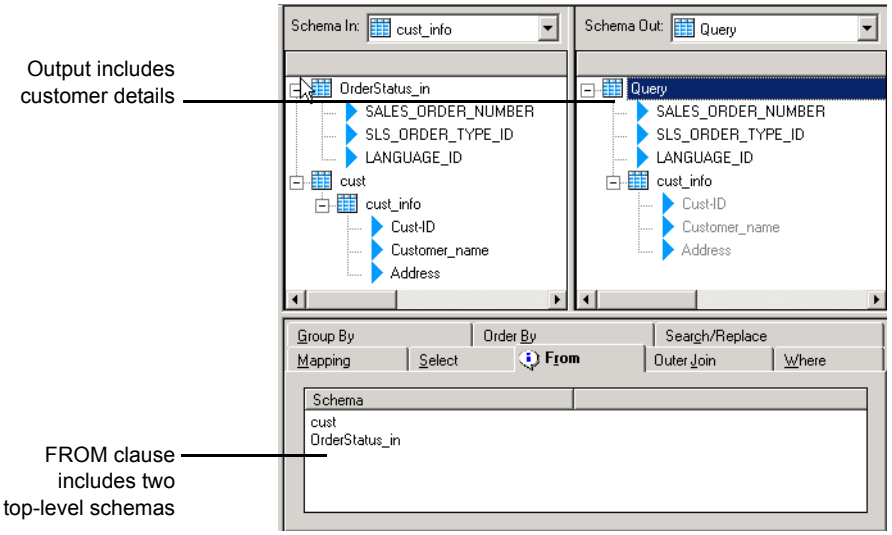
```
SELECT columns FROM A
   SELECT columns FROM A.B, Z.Y
```

The data that a SELECT statement from a lower schema produces differs depending on whether or not a schema is included in the FROM clause at the top-level.

The next two examples use the sales order data set to illustrate scenarios where FROM clause values change the data resulting from the query.

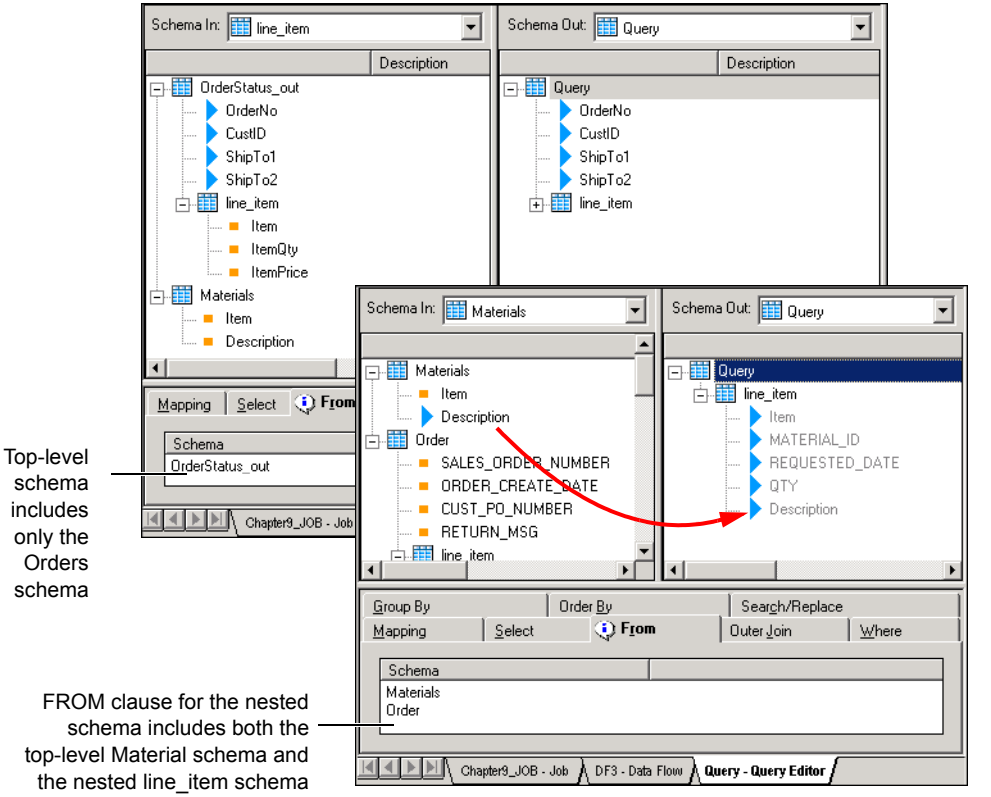
Example: FROM clause includes all top-level inputs

Joining the order schema at the top-level with a customer schema, the output can include detailed customer information for all of the orders in the data set. Including both input schemas in the FROM clause at the top-level contents produces the appropriate data.



Example: Lower level FROM clause contains top-level input

If instead the input includes a material schema, you would define the join constraint for the line-item schema so that the detailed material information appears for each row in the nested data set. The FROM clause for the line-item schema would include the material schema (top level) and the line-item schema. For the line-item schema to be available in the FROM clause, the order schema would have to be included in the FROM clause of the top-level schema.



Nesting columns

When you nest rows of one schema inside another, the data set produced in the nested schema is the result of a query against the first one using the related values from the second one.

For example, if you have sales-order information in a header schema and a line-item schema, you can nest the line items under the header schema. The line items for a single row of the header schema are equal to the results of a query including the order number:

```
SELECT * FROM LineItems
   WHERE Header.OrderNo = LineItems.OrderNo
```

In Data Integrator, you can use a query transform to construct a nested data set from relational data. When you indicate the columns included in the nested schema, specify the query used to define the nested data set for each row of the parent schema.

➢ To construct a nested data set

1. Create a data flow with the sources that you want to include in the nested data set.
2. Place a query in the data flow and connect the sources to the input of the query.
3. Indicate the FROM clause, source list, and WHERE clause to describe the SELECT statement that the query executes to determine the top-level data set.
   - **FROM clause**: Include the input sources in the list on the **From** tab.
   - **Source list**: Drag the columns from the input to the output. You can also include new columns or include mapping expressions for the columns.
   - **WHERE clause**: Include any filtering or joins required to define the data set for the top-level output.

4. Create a new schema in the output.

   In the output of the query, right-click and choose **New Output Schema**. A new schema icon appears in the output, nested under the top-level schema.

   You can also drag an entire schema from the input to the output, if that schema is included in the FROM clause for this schema.

5. Change the current schema to the nested schema. (For information on setting the current schema and completing the parameters, see "Query editor" on page 174.)

   The query editor changes to display the new current schema.

6. Indicate the FROM clause, source list, and WHERE clause to describe the SELECT statement that the query executes to determine the top-level data set.

   - **FROM clause**: If you created a new output schema, you need to drag schemas from the input to populate the FROM clause. If you dragged an existing schema from the input to the top-level output, that schema is automatically listed.
   - **Select list**: Only columns are available that meet the requirements for the FROM clause as described in "FROM clause construction" on page 211.
   - **WHERE clause**: Only columns are available that meet the requirements for the FROM clause.

7. If the output requires it, nest another schema at this level.

   Repeat steps 4 through 6 in this current schema.

8. If the output requires it, nest another schema under the top level.

   Make the top-level schema the current schema.

Using correlated columns in nested data

Correlation allows you to use columns from a higher-level schema to construct a nested schema. In a nested-relational model, the columns in a nested schema are implicitly related to the columns in the parent row. To take advantage of this relationship, you can use columns from the parent schema in the construction of the nested schema. The higher-level column is a *correlated column*.

Including a correlated column in a nested schema can serve two purposes:

- The correlated column is a key in the parent schema. Including the key in the nested schema allows you to maintain a relationship between the two schemas after converting them from the nested data model to a relational model.
- The correlated column is an attribute in the parent schema. Including the attribute in the nested schema allows you to use the attribute to simplify correlated queries against the nested data.

To include a correlated column in a nested schema, you do not need to include the schema that includes the column in the FROM clause of the nested schema.

➢ To used a correlated column in a nested schema

1. Create a data flow with a source that includes a parent schema with a nested schema.

   For example, the source could be an order header schema that has a LineItems column that contains a nested schema.

2. Connect a query to the output of the source.

3. In the query editor, copy all columns of the parent schema to the output.

   In addition to the top-level columns, Data Integrator creates a column called LineItems that contains a nested schema that corresponds to the LineItems nested schema in the input.

4. Change the current schema to the LineItems schema. (For information on setting the current schema and completing the parameters, see "Query editor" on page 174.)

5. Include a correlated column in the nested schema.

   Correlated columns can include columns from the parent schema and any other schemas in the FROM clause of the parent schema.

   For example, drag the OrderNo column from the Header schema into the LineItems schema. Including the correlated column creates a new output column in the LineItems schema called OrderNo and maps it to the Order.OrderNo column. The data set created for LineItems includes all of the LineItems columns and the OrderNo.

   If the correlated column comes from a schema other than the immediate parent, the data in the nested schema includes only the rows that match both the related values in the current row of the parent schema and the value of the correlated column.

You can always remove the correlated column from the lower-level schema in a subsequent query transform.

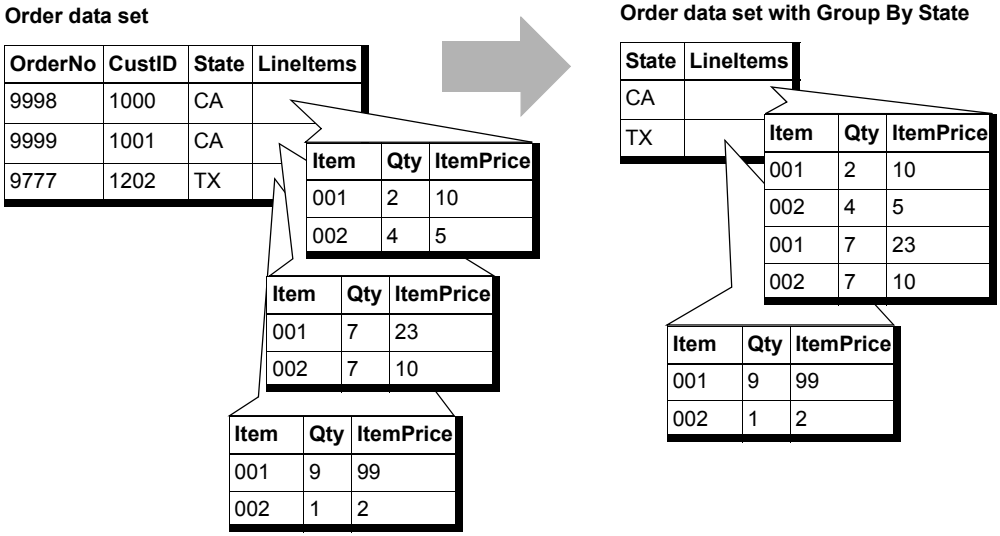
Distinct rows and nested data

The **Distinct rows** option in Query transforms removes any duplicate rows at the top level of a join. This is particularly useful to avoid cross products in joins that produce nested output.

Grouping values across nested schemas

When you specify a Group By clause for a schema with a nested schema, the grouping operation combines the nested schemas for each group.

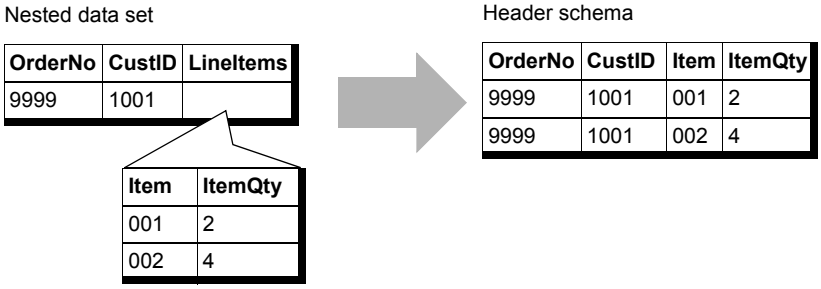
For example, to assemble all the line items included in all the orders for each state from a set of orders, you can set the Group By clause in the top level of the data set to the state column (Order.State) and create an output schema that includes State column (set to Order.State) and LineItems nested schema.



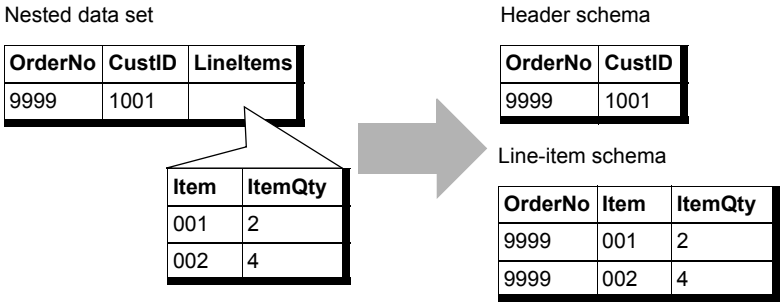
The result is a set of rows (one for each state) that has the State column and the LineItems nested schema that contains all the LineItems for all the orders for that state.

Unnesting nested data

Loading a data set that contains nested schemas into a relational (non-nested) target requires that the nested rows be unnested. For example, a sales order may use a nested schema to define the relationship between the order header and the order line items. To load the data into relational schemas, the multi-level must be *unnested*. Unnesting a schema produces a cross-product of the top-level schema (parent) and the nested schema (child).



It is also possible that you would load different columns from different nesting levels into different schemas. A sales order, for example, may be flattened so that the order number is maintained separately with each line item and the header and line item information loaded into separate schemas.



Data Integrator allows you to unnest any number of nested schemas at any depth. No matter how many levels are involved, the result of unnesting schemas is a cross product of the parent and child schemas. When more than one level of unnesting occurs, the inner-most child is unnested first, then the result—the cross product of the parent and the inner-most child—is then unnested from its parent, and so on to the top-level schema.

Order data set

OrderNo	CustID	LineItems
9999	1001	

Item	ItemQty	ItemPrice
001	2	
002	4	

Qty	Sell Price
1	25
20	23

Qty	SellPrice
1	200
10	190

Unnested data set

OrderNo	CustID	Item	ItemQty	Qty	SellPrice
9999	1001	001	2	1	200
9999	1001	001	2	10	190
9999	1001	002	4	1	25
9999	1001	002	4	20	23

Unnesting all schemas (cross product of all data) might not produce the results you intend. For example, if an order includes multiple customer values such as ship-to and bill-to addresses, flattening a sales order by unnesting customer and line-item schemas produces rows of data that might not be useful for processing the order.

Nested data set

OrderNo	LineItems	CustInfo
9999		

Item	ItemQty
001	2
002	4

Type	CustID
Ship	1001
Bill	7777

Unnested data set

OrderNo	Item	ItemQty	Type	CustID
9999	001	2	Ship	1001
9999	002	4	Ship	1001
9999	001	2	Bill	7777
9999	002	4	Bill	7777

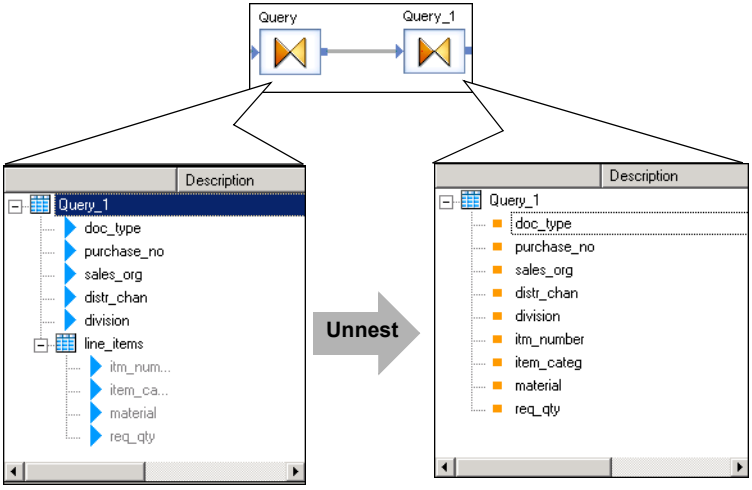
➢ To unnest nested data

1. Create the output that you want to unnest in the output schema of a query.

   Data for unneeded columns or schemas might be more difficult to filter out after the unnesting operation. You can use the **Cut** command to remove columns or schemas from the top level; to remove nested schemas or columns inside nested schemas, make the nested schema the current schema, and then cut the unneeded columns or nested columns.

2. For each of the nested schemas that you want to unnest, right-click the schema name and choose **Unnest**.

   The output of the query (the input to the next step in the data flow) includes the data in the new relationship.



How transforms handle nested data

Nested data included in the input to transforms (with the exception of a query transform) passes through the transform without being included in the transform's operation. Only the columns at the first level of the input data set are available for transformation. Therefore, to transform values in nested schemas, you must use a query transform to unnest the data, perform the transformation, then nest the data again to reconstruct the nested relationships.

Formatting XML documents

Data Integrator allows you to import and export metadata for XML documents (files or messages), which you can use as sources or targets in jobs. XML documents are hierarchical. Their valid structure is stored in separate format documents.

The format of an XML file or message (.xml) can be specified using either an XML Schema (.xsd for example) or a document type definition (.dtd).

When you import a format document's metadata, it is structured into Data Integrator's internal schema for hierarchical documents which uses the nested relational data model (NRDM).

Using XML Schemas

Data Integrator supports WC3 XML Schema Specification 1.0. See "XML Schema" on page 152 of the *Data Integrator Reference Guide* for this XML Schema's URL. For an XML document that contains information to place a sales order—order header, customer, and line items—the corresponding XML Schema includes the order structure and the relationship between data.

Message with data

OrderNo	CustID	ShipTo1	ShipTo2	LineItems
9999	1001	123 State St.	Town, CA	

Item	ItemQty	ItemPrice
001	2	10
002	4	5

Each column in the XML document corresponds to an ELEMENT or ATTRIBUTE definition in the XML Schema.

Corresponding XML Schema

```
<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Order">
    <xs:complexType>
      <xs:sequence>
          <xs:element name="OrderNo"  type="xs:string" />
          <xs:element name="CustID"   type="xs:string" />
          <xs:element name="ShipTo1"  type="xs:string" />
          <xs:element name="ShipTo2"  type="xs:string" />
          <xs:element maxOccurs="unbounded" name="LineItems">
             <xs:complexType>
                <xs:sequence>
                   <xs:element name="Item"      type="xs:string" />
                   <xs:element name="ItemQty"   type="xs:string" />
                   <xs:element name="ItemPrice" type="xs:string" />
                </xs:sequence>
             </xs:complexType>
          </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Import the metadata for each XML Schema you use. The object library lists imported XML Schemas in the **Formats** tab.

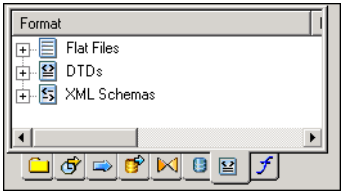
When importing an XML Schema, Data Integrator reads the defined elements and attributes, then imports the following:

- Document structure
- Table and column names
- Data type of each column
- Nested table and column attributes

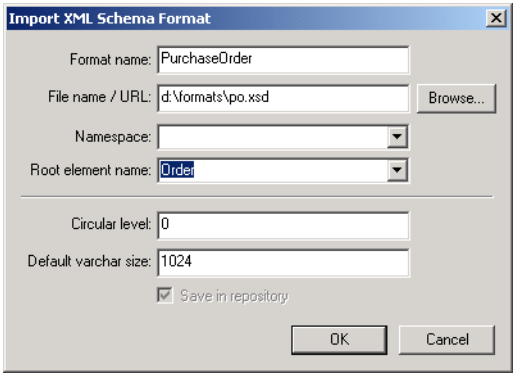
  While XML Schemas make a distinction between elements and attributes, Data Integrator imports and converts them all to Data Integrator nested table and column attributes. See "XML Schema" on page 152 of the *Data Integrator Reference Guide* for the list of Data Integrator attributes.

➢ To import an XML Schema

1. From the object library, click the **Format** tab.
2. Right-click the **XML Schemas** icon.



3. Enter settings into the Import XML Schema Format window:



When importing an XML Schema:

- Enter the name you want to use for the format in Data Integrator.
- Enter the file name of the XML Schema or its URL address.
- If the root element name is not unique within the XML Schema, select a namespace to identify the imported XML Schema.
- In the **Root element name** box, select the name of the primary node you want to import. Data Integrator only imports elements of the XML Schema that belong to this node or any subnodes.
- If the XML Schema contains recursive elements (element A contains B, element B contains A), specify the number of levels it has by entering a value in the **Circular level** box. This value must match the number of recursive levels in the XML Schema's content. Otherwise, the job that uses this XML Schema will fail.
- You can set Data Integrator to import strings as a varchar of any size. Varchar 1024 is the default.

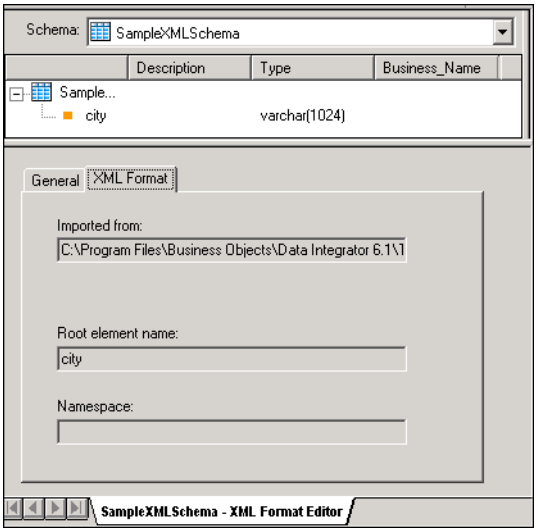
4. Click **OK**.

After you import an XML Schema, you can edit its column properties such as data type using the **General** tab of the Column Properties window. You can also view and edit nested table and column attributes from the Column Properties window.

➢ To view and edit nested table and column attributes for XML Schema

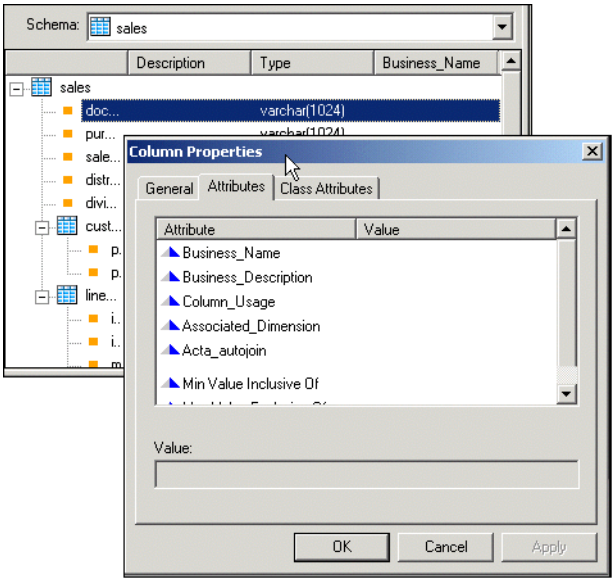
1. From the object library, select the **Formats** tab.
2. Expand the **XML Schema** category.
3. Double-click an XML Schema name.

   The XML Schema Format window appears in the workspace.



The **Type** column displays the data types that Data Integrator uses when it imports the XML document metadata. See "XML Schema" on page 152 of the *Data Integrator Reference Guide* for more information about data types supported by XML Schema.

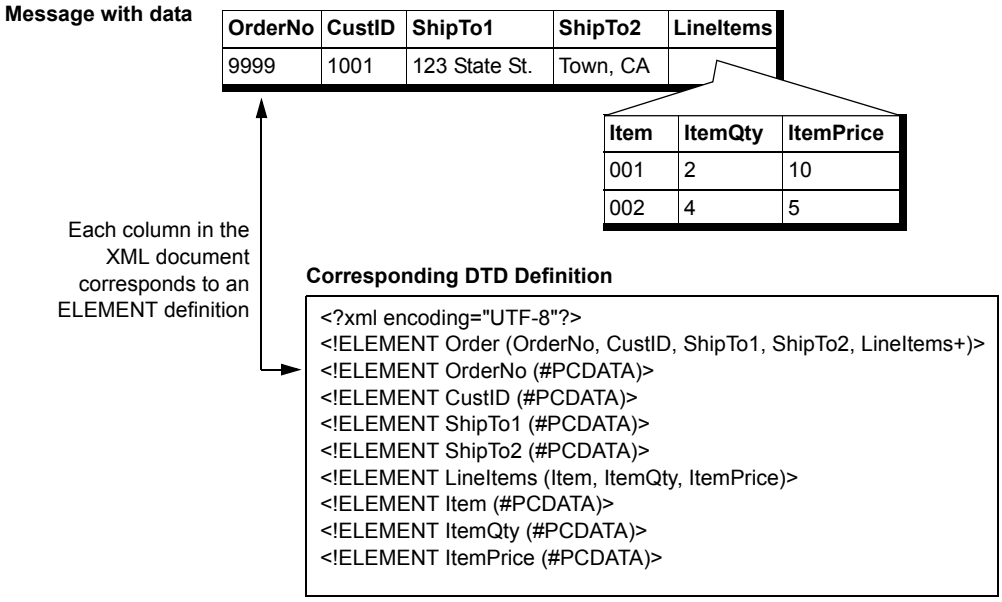
4. Double-click a nested table or column and select **Attributes** to view or edit XML Schema attributes.



Using Document Type Definitions (DTDs)

The format of an XML document (file or message) can be specified by a document type definition (DTD). The DTD describes the data contained in the XML document and the relationships among the elements in the data.

For an XML document that contains information to place a sales order—order header, customer, and line items—the corresponding DTD includes the order structure and the relationship between data.



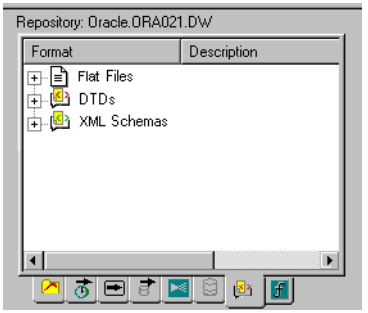
Import the metadata for each DTD you use. The object library lists imported DTDs in the **Formats** tab.

You can import metadata from either an existing XML file (with a reference to a DTD) or DTD file. If you import the metadata from an XML file, Data Integrator automatically retrieves the DTD for that XML file.

When importing a DTD, Data Integrator reads the defined elements and attributes. Data Integrator ignores other parts of the definition, such as text and comments. This allows you to modify imported XML data and edit the data type as needed. See "DTD" on page 50 of the *Data Integrator Reference Guide* for information about Data Integrator attributes that support DTDs.

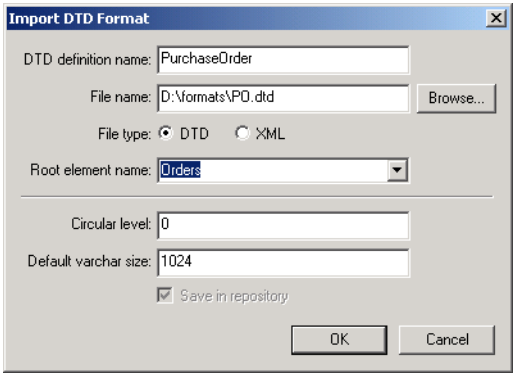
➢ To import a DTD or XML Schema format

1. From the object library, click the **Format** tab.



2. Right-click the **DTDs** icon and select **New**.

   The Import DTD Format window opens.



3. Enter settings into the Import DTD Format window:
   - In the **Format name** box, enter the name you want to give the imported DTD format in Data Integrator.
   - Enter the file that specifies the DTD you want to import.
   - If importing an XML file, select the **XML file** option button. If importing a DTD file, select the **DTD file** option button.

- In the **Root element name** box, select the name of the primary node you want to import. Data Integrator only imports elements of the DTD that belong to this node or any subnodes.
- If the DTD contains recursive elements (element A contains B, element B contains A), specify the number of levels it has by entering a value in the **Circular level** box. This value must match the number of recursive levels in the DTD's content. Otherwise, the job that uses this DTD will fail.
- You can set Data Integrator to import strings as a varchar of any size. Varchar 1024 is the default.

4. Click **OK**.

After you import a DTD, you can edit its column properties such as data type using the **General** tab of the Column Properties window. You can also view and edit DTD nested table and column attributes from the Column Properties window.

➢ To view and edit nested table and column attributes for DTDs

1. From the object library, select the **Formats** tab.
2. Expand the **DTDs** category.
3. Double-click a DTD name.

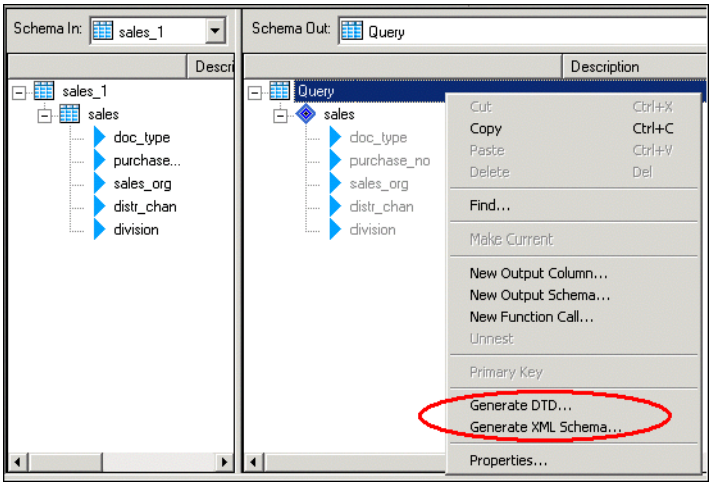
   The DTD Format window appears in the workspace.
4. Double-click a nested table or column.

   The Column Properties window opens.
5. Select the **Attributes** tab to view or edit DTD attributes.

Generating DTDs and XML Schemas from an NRDM schema

You can right-click any schema from within a query editor in the Designer and generate a DTD or an XML Schema that corresponds to the structure of the selected schema (either NRDM or relational).

This feature is useful if you want to stage data to an XML file and subsequently read it into another data flow. First generate a DTD/XML Schema. Then use it to setup an XML format, which in turn is used to set up an XML source for the staged file.



The DTD/XML Schema generated will be based on the following information:

- Columns become either elements or attributes based on whether the *XML Type* attribute is set to ATTRIBUTE or ELEMENT.
- If the *Required* attribute is set to NO, the corresponding element or attribute is marked optional.
- Nested tables become intermediate elements.
- The *Native Type* attribute is used to set the type of the element or attribute.

- While generating XML Schemas, the MinOccurs and MaxOccurs values will be set based on the *Minimum Occurrence* and *Maximum Occurrence* attributes of the corresponding nested table.

No other information is considered while generating the DTD or XML Schema.

See "DTD" on page 50 of the *Data Integrator Reference Guide* and "XML Schema" on page 152 of the *Data Integrator Reference Guide* for details about how Data Integrator creates internal attributes when importing a DTD or XML Schema.

10 Real-time jobs

Data Integrator supports real-time data transformation. *Real-time* means that Data Integrator can receive requests from ERP systems and Web applications and send replies immediately after getting the requested data from a data cache or a second application. You define operations for processing on-demand messages by building real-time jobs in the Designer.

This chapter contains the following sections:

Request-response message processing

The message passed through a real-time system includes the information required to perform a business transaction. The content of the message can vary:

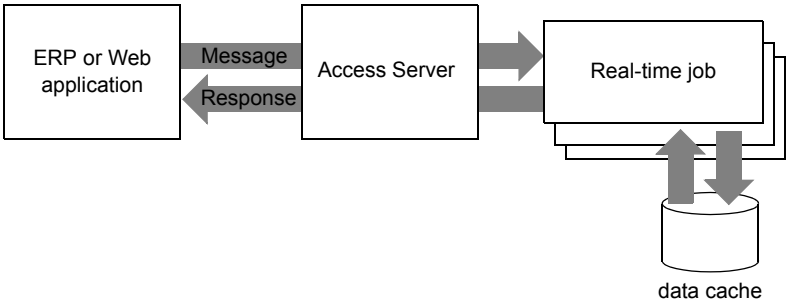
- It could be a sales order or an invoice processed by an ERP system destined for a data cache.
- It could be an order status request produced by a Web application that requires an answer from a data cache or back-office system.

The Data Integrator Access Server constantly listens for incoming messages. When a message is received, the Access Server routes the message to a waiting process that performs a predefined set of operations for the message type. The Access Server then receives a response for the message and replies to the originating application.

Two Data Integrator components support request-response message processing:

- **Access Server** — Listens for messages and routes each message based on message type.
- **Real-time job** — Performs a predefined set of operations for that message type and creates a response.

Processing might require that additional data be added to the message from a data cache or that the message data be loaded to a data cache. The Access Server returns the response to the originating application.



What is a real-time job?

The Data Integrator Designer allows you to define the processing of real-time messages using a real-time job. You create a different real-time job for each type of message your system can produce.

Real-time versus batch

Like a batch job, a real-time job extracts, transforms, and loads data. Real-time jobs "extract" data from the body of the message received and from any secondary sources used in the job. Each real-time job can extract data from a single message type. It can also extract data from other sources such as tables or files.

The same powerful transformations you can define in batch jobs are available in real-time jobs. However, you might use transforms differently in real-time jobs. For example, you might use branches and logic controls more often than you would in batch jobs. If a customer wants to know when they can pick up their order at your distribution center, you might want to create a CheckOrderStatus job using a look-up function to count order items and then a case transform to provide status in the form of strings: "No items are ready for pickup" or "X items in your order are ready for pickup" or "Your order is ready for pickup".

Also in real-time jobs, Data Integrator writes data to message targets and secondary targets in parallel. This ensures that each message receives a reply as soon as possible.

Unlike batch jobs, real-time jobs do not execute in response to a schedule or internal trigger; instead, real-time jobs execute as real-time services started through the Administrator. Real-time services then wait for messages from the Access Server. When the Access Server receives a message, it passes the message to a running real-time service designed to process this message type. The real-time service processes the message and returns a response. The real-time service continues to listen and process messages on demand until it receives an instruction to shut down.

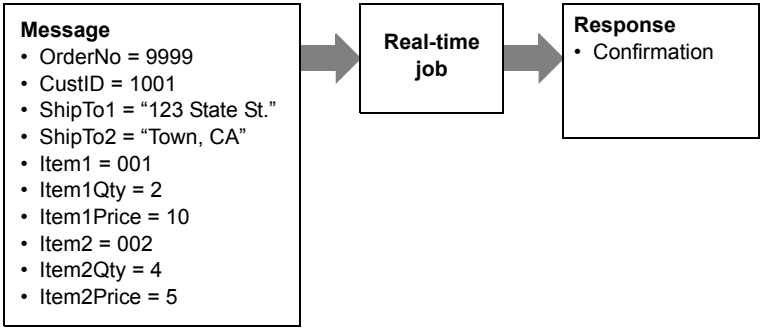
Messages

How you design a real-time job depends on what message you want it to process. Typical messages include information required to implement a particular business operation and to produce an appropriate response.

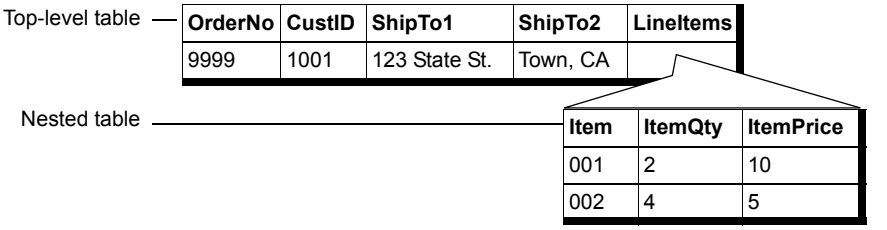
For example, suppose a message includes information required to determine order status for a particular order. The message contents might be as simple as the sales order number. The corresponding real-time job might use the input to query the right sources and return the appropriate product information.

In this case, the message contains data that can be represented as a single column in a single-row table.

In a second case, a message could be a sales order to be entered into an ERP system. The message might include the order number, customer information, and the line-item details for the order. The message processing could return confirmation that the order was submitted successfully.



In this case, the message contains data that cannot be represented in a single table; the order header information can be represented by a table and the line items for the order can be represented by a second table. Data Integrator represents the header and line item data in the message in a nested relationship.



When processing the message, the real-time job processes all of the rows of the nested table for each row of the top-level table. In this sales order, both of the line items are processed for the single row of header information.

Real-time jobs can send only one row of data in a reply message (message target). However, you can structure message targets so that all data is contained in a single row by nesting tables within columns of a single, top-level table.

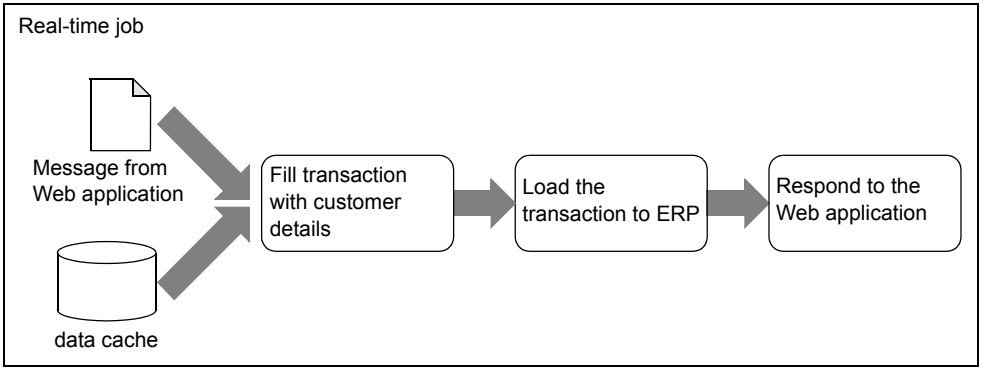
Data Integrator data flows support the nesting of tables within other tables. See Chapter 9, "Nested Data" for details.

Real-time job examples

These examples provide a high-level description of how real-time jobs address typical real-time scenarios. Later sections describe the actual objects that you would use to construct the logic in the Designer.

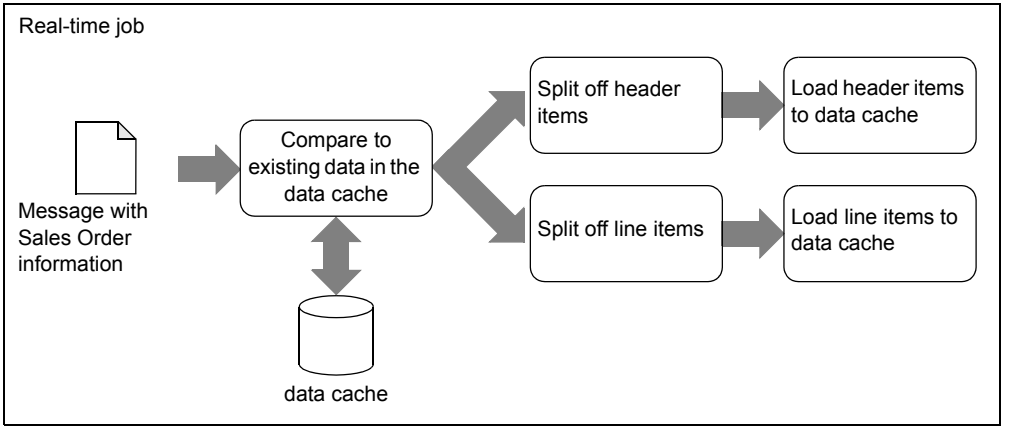
Loading transactions into a back-office application

A real-time job can receive a transaction from a Web application and load it to a back-office application (ERP, SCM, legacy). Using a query transform, you can include values from a data cache to supplement the transaction before applying it against the back-office application (such as an ERP system).



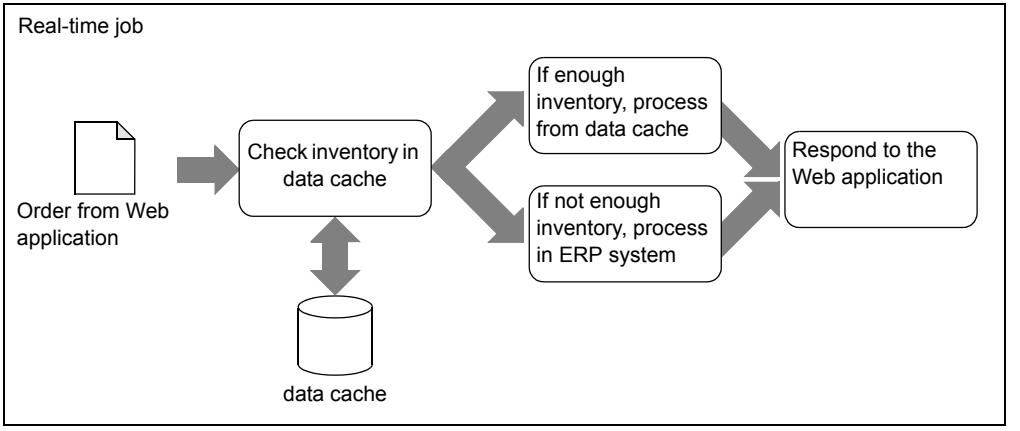
Collecting back-office data into a data cache

You can use messages to keep the data cache current. Real-time jobs can receive messages from a back-office application and load them into a data cache or data warehouse.



Retrieving values from a data cache or back-office application

You can create real-time jobs that use values from a data cache to determine whether or not to query the back-office application (such as an ERP system) directly.



Creating real-time jobs

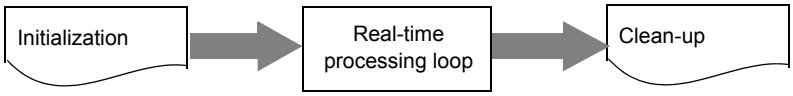
You can create real-time jobs using the same objects as batch jobs (data flows, work flows, conditionals, scripts, while loops, etc.). However, object usage must adhere to a valid real-time job model.

Real-time job models

In contrast to batch jobs, which typically move large amounts of data at scheduled times, a real-time job, once started as a real-time service, listens for a request. When a real-time job receives a request (typically to access small number of records), Data Integrator processes the request, returns a reply, and continues listening. This listen-process-listen logic forms a processing loop.

A real-time job is divided into three processing components: initialization, a real-time processing loop, and clean-up.

Real-time job



- The initialization component (optional) can be a script, work flow, data flow, or a combination of objects. It runs only when a real-time service starts.
- The real-time processing loop is a container for the job's single process logic. You can specify any number of work flows and data flows inside it.
- The clean-up component (optional) can be a script, work flow, data flow, or a combination of objects. It runs only when a real-time service is shut down.

In a real-time processing loop, a single message source must be included in the first step and a single message target must be included in the last step.

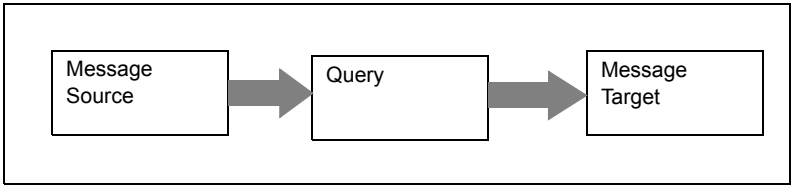
The following models support this rule:

- Single data flow model
- Multiple data flow model
- Request/Acknowledge data flow model (see "IDoc sources in real-time jobs" on page 135 of the *Data Integrator Supplement for SAP*)

Single data flow model

With the single data flow model, you create a real-time job using a single data flow in its real-time processing loop. This single data flow must include a single message source and a single message target.

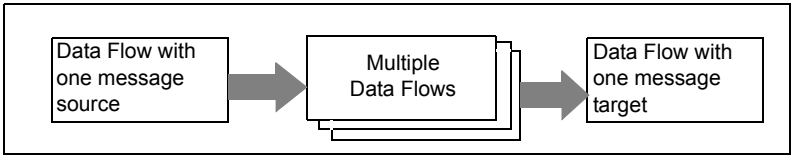
Real-time processing loop



Multiple data flow model

The multiple data flow model allows you to create a real-time job using multiple data flows in its real-time processing loop.

Real-time processing loop



By using multiple data flows, you can ensure that data in each message is completely processed in an initial data flow before processing for the next data flows starts. For example, if the data represents 40 items, all 40 must pass though the first data flow to a staging or memory table before passing to a second data flow. This allows you to control and collect all the data in a message at any point in a real-time job for design and troubleshooting purposes.

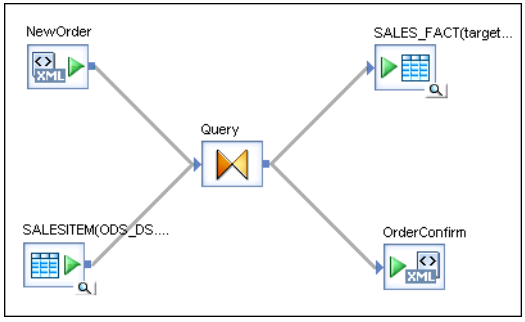
If you use multiple data flows in a real-time processing loop:

- The first object in the loop must be a data flow. This data flow must have one and only one message source.
- The last object in the loop must be a data flow. This data flow must have a message target.
- Additional data flows cannot have *message* sources or targets.
- You can add any number of additional data flows to the loop, and you can add them inside any number of work flows.
- All data flows can use input and/or output *memory tables* to pass data sets on to the next data flow. Memory tables store data in memory while a loop runs. They improve the performance of real-time jobs with multiple data flows.

Using real-time job models

Single data flow model

When you use a single data flow within a real-time processing loop your data flow diagram might look like this:



Notice that the data flow has one message source and one message target.

Multiple data flow model

When you use multiple data flows within a real-time processing loop your data flow diagrams might look like those in the following example scenario in which Data Integrator writes data to several targets according to your multiple data flow design.

Example scenario requirements:

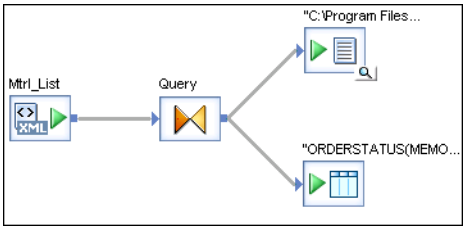
Your job must do the following tasks, completing each one before moving on to the next:

- Receive requests about the status of individual orders from a web portal and record each message to a backup flat file
- Perform a query join to find the status of the order and write to a customer database table.
- Reply to each message with the query join results

Solution:

First, create a real-time job and add a data flow, a work flow, and an another data flow to the real-time processing loop. Second, add a data flow to the work flow. Next, set up the tasks in each data flow:

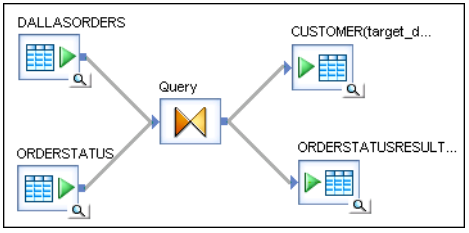
- The first data flow receives the XML message (using an XML message source) and records the message to the flat file (flat file format target). Meanwhile, this same data flow writes the data into a memory table (table target).



NOTE: You might want to create a memory table to move data to sequential data flows. For more information, see "Memory datastores" on page 98.

- The second data flow reads the message data from the memory table (table source), performs a join with stored data (table source), and writes the results to a database table (table target) and a new memory table (table target).

  Notice this data flow has neither a message source nor a message target.



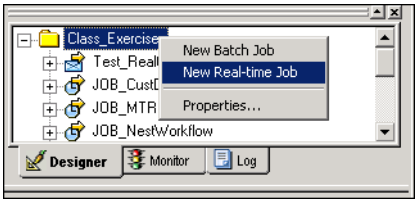
- The last data flow sends the reply. It reads the result of the join in the memory table (table source) and loads the reply (XML message target).

For more information about building real-time jobs, see the "Building blocks for real-time jobs" and "Designing real-time applications" sections.

Creating a real-time job

➢ To create a real-time job

1. In the Designer, create or open an existing project.
2. From the project area, right-click the white space and select **New Real-time job** from the shortcut menu.



New_RTJob1 appears in the project area. The workspace displays the job's structure, which consists of two markers:

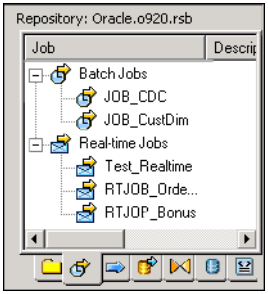
- RT_Process_begins
- Step_ends

These markers represent the beginning and end of a real-time processing loop.

3. In the project area, rename **New_RTJob1**.

   Always add a prefix to job names with their job type. In this case, use the naming convention: `RTJOB_JobName`.

Although saved real-time jobs are grouped together under the **Job** tab of the object library, job names may also appear in text editors used to create adapter or Web Services calls. In these cases, a prefix saved with the job name will help you identify it.



4. If you want to create a job with a single data flow:

   a. Click the data flow icon in the tool pallet. 

      You can add data flows to either a batch or real-time job. When you place a data flow icon into a job, you are telling Data Integrator to validate the data flow according the requirements of the job type (batch or real-time).

   b. Click inside the loop.

      The boundaries of a loop are indicated by begin and end markers. One message source and one message target are allowed in a real-time processing loop.

   c. Connect the begin and end markers to the data flow.

   d. Build the data flow including a message source and message target. For more information about sources and targets, see "Real-time source and target objects" on page 252.

   e. Add, configure, and connect initialization object(s) and clean-up object(s) as needed.

A real-time job with a single data flow might look like this:

5. If you want to create a job with multiple data flows:
   a. Drop and configure a data flow.

      This data flow must include one message source.

   b. After this data flow, drop other objects such as work flows, data flows, scripts, or conditionals from left to right between the first data flow and end of the real-time processing loop.

   c. Just before the end of the loop, drop and configure your last data flow.

      This data flow must include one message target.

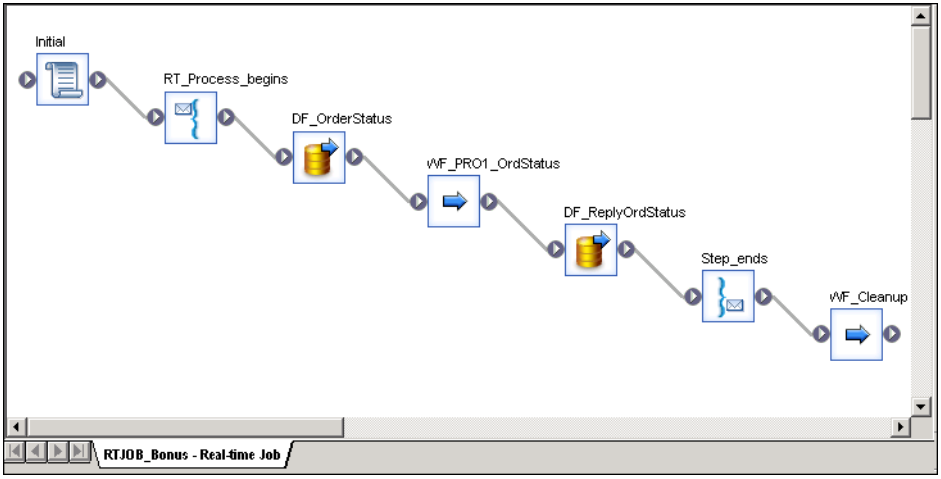
   d. Open each object and configure it.

   e. Return to the real-time job window and connect all the objects.

      NOTE: Objects at the real-time job level in Designer diagrams must be connected. Connected objects run in sequential order. To include parallel processing in a real-time job, drop data flows within job-level work flows. Do not connect these secondary-level data flows. These data flows will run in parallel when job processing begins.

   f. Drop, configure, and connect the initialization, and clean up objects outside the real-time processing loop as needed.

A real-time job with multiple data flows might look like this:



6. After adding and configuring all objects, validate your job.
7. Save the job.
8. Assign test source and target files for the job and execute it in test mode. For more information see, "Testing real-time jobs" on page 258.
9. Using the Administrator, configure a service and service providers for the job and run it in your test environment.

Real-time source and target objects

Real-time jobs must contain a real-time source and/or target object. Those normally available are:

Object	Description	Used as a:	Data Integrator Access
XML message	An XML message structured in a DTD or XML Schema format	Source or target	Directly or through adapters
Outbound message	A real-time message with an application-specific format (not readable by XML parser)	Target	Through an adapter

If you have the SAP licensed extension, you can also use IDoc messages as real-time sources. For more information, see the *Data Integrator Supplement for SAP*.

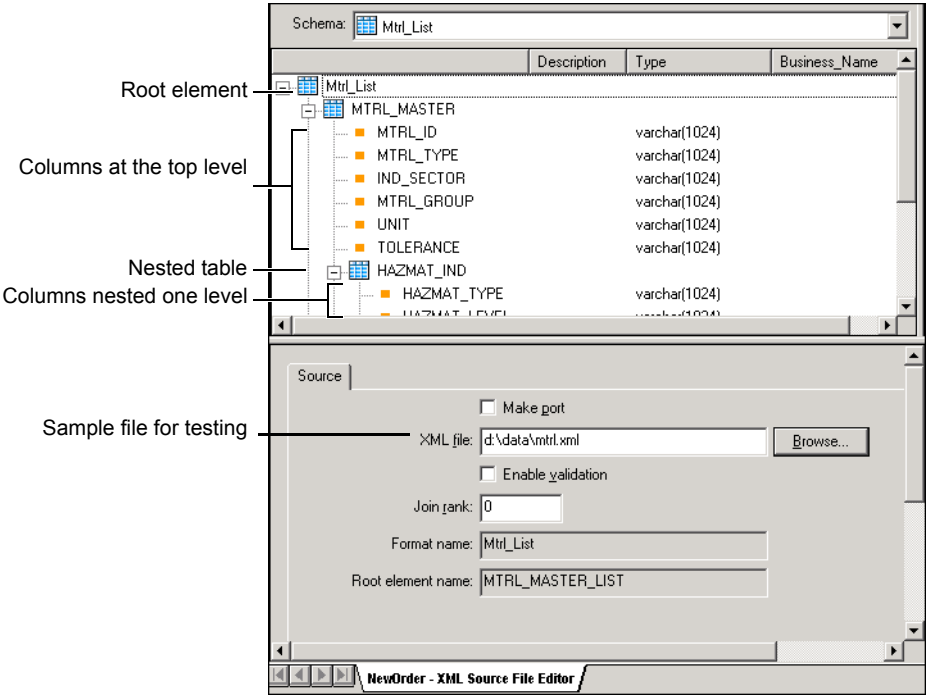
Adding sources and targets to real-time jobs is similar to adding them to batch jobs, with the following additions:

For	Prerequisite	Reference	Object library location
XML messages	Import a DTD or XML Schema to define a format	"To import a DTD or XML Schema format" on page 231	**Formats** tab
Outbound message	Define an adapter datastore and import object metadata.	"Adapter datastores" on page 106	**Datastores** tab, under adapter datastore

➢ To view an XML message source or target schema

In the workspace of a real-time job, click the name of an XML message source or XML message target to open its editor.

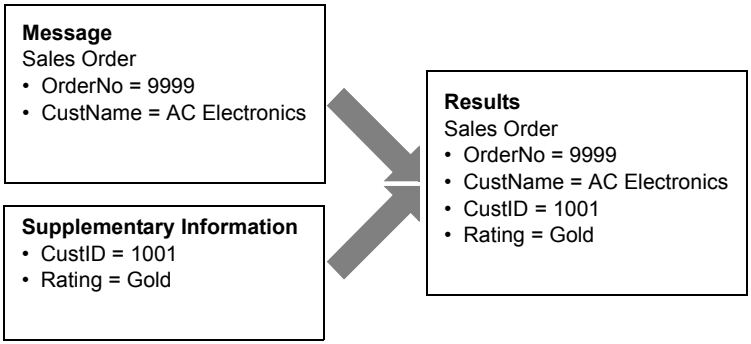
If the XML message source or target contains nested data, the schema displays nested tables to represent the relationships among the data.



Secondary sources and targets

Real-time jobs can also have secondary sources or targets (see "Source and target objects" on page 161). For example, suppose you are processing a message that contains a sales order from a Web application. The order contains the customer name, but when you apply the order against your ERP system, you need to supply more detailed customer information.

Inside a data flow of a real-time job, you can supplement the message with the customer information to produce the complete document to send to the ERP system. The supplementary information might come from the ERP system itself or from a data cache containing the same information.



Tables and files (including XML files) as sources can provide this supplementary information.

Data Integrator reads data from secondary sources according to the way you design the data flow. Data Integrator loads data to secondary targets in parallel with a target message.

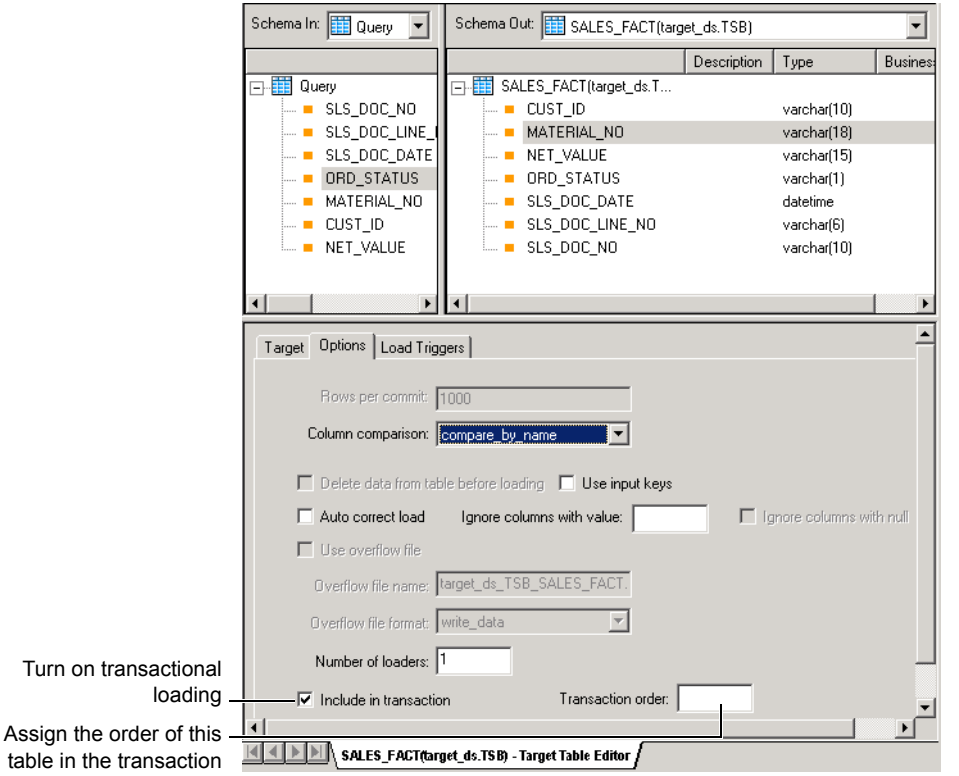
Add secondary sources and targets to data flows in real-time jobs as you would to data flows in batch jobs (See "Adding source or target objects to data flows" on page 163).

Transactional loading of tables

Target tables in real-time jobs support *transactional loading*, in which the data resulting from the processing of a single data flow can be loaded into multiple tables as a single transaction. No part of the transaction applies if any part fails.

NOTE: Target tables in batch jobs also support transactional loading. However, use caution when you consider enabling this option for a batch job because it requires the use of memory, which can reduce performance when moving large amounts of data.

You can specify the order in which tables in the transaction are included using the target table editor. This feature supports a scenario in which you have a set of tables with foreign keys that depend on one with primary keys.



You can use transactional loading only when all the targets in a data flow are in the same datastore. If the data flow loads tables in more than one datastore, targets in each datastore load independently. If you use transactional loading, you cannot use bulk loading, pre-load, or post-load commands.

Design tips for data flows in real-time jobs

Keep in mind the following when you are designing data flows:

- If you include a table in a join with a real-time source, Data Integrator includes the data set from the real-time source as the outer loop of the join. If more than one supplementary source is included in the join, you can control which table is included in the next outer-most loop of the join using the join ranks for the tables.
- In real-time jobs, do *not* cache data from secondary sources unless the data is static. The data will be read when the real-time job starts and will not be updated while the job is running.
- If no rows are passed to the XML target, the real-time job returns an empty response to the Access Server. For example, if a request comes in for a product number that does not exist, your job might be designed in such a way that no data passes to the reply message. You might want to provide appropriate instructions to your user (exception handling in your job) to account for this type of scenario.
- If more than one row passes to the XML target, the target reads the first row and discards the other rows. To avoid this issue, use your knowledge of Data Integrator's Nested Relational Data Model (NRDM) and structure your message source and target formats so that one "row" equals one message. With NRDM, you can structure any amount of data into a single "row" because columns in tables can contain other tables. See Chapter 9, "Nested Data" for more information.
- Recovery mechanisms are not supported in real-time jobs.
- The LiveLoad feature is not supported with real-time jobs.

For more detailed information about real-time job processing see "Real-time job" on page 86 of the *Data Integrator Reference Guide*.

Testing real-time jobs

There are several ways to test real-time jobs during development. These include:

- Executing a real-time job in test mode
- Using View Data
- Using an XML file target

Executing a real-time job in test mode

You can test real-time job designs without configuring the job as a service associated with an Access Server. In *test mode*, you can execute a real-time job using a sample source message from a file to determine if Data Integrator produces the expected target message.

➢ To specify a sample XML message and target test file

1. In the XML message source and target editors, enter a file name in the **XML test file** box.

   Enter the full path name for the source file that contains your sample data. Use paths for both test files relative to the computer that runs the Job Server for the current repository.

2. Execute the job.

   Test mode is always enabled for real-time jobs. Data Integrator reads data from the source test file and loads it into the target test file.

Using View Data

To ensure that your design returns the results you expect, execute your job using *View Data*. With View Data, you can capture a sample of your output data to ensure your design is working. See Chapter 14, "Design and Debug using View Data" for more information.

Using an XML file target

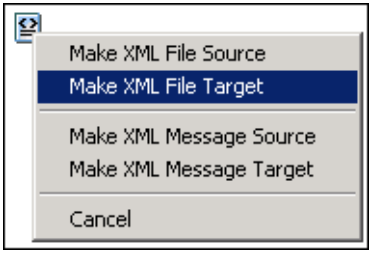
You can use an "XML file target" to capture the message produced by a data flow while allowing the message to be returned to the Access Server.

Just like an XML message, you define an XML file by importing a DTD or XML Schema for the file, then dragging the format into the data flow definition. Unlike XML messages, you can include XML files as sources or targets in batch and real-time jobs.

➢ To use a file to capture output from steps in a real-time job

1. In the **Formats** tab of the object library, drag the DTD or XML Schema into a data flow of a real-time job.

   A menu prompts you for the function of the file.



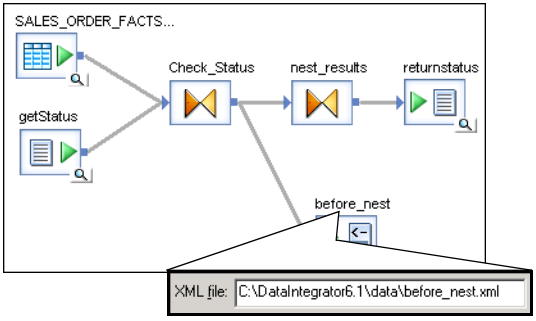
2. Choose **Make XML File Target**.

   The XML file target appears in the workspace.

3. In the file editor, specify the location to which Data Integrator writes data.

   Enter a file name relative to the computer running the Job Server.

4. Connect the output of the step in the data flow that you want to capture to the input of the file.



Building blocks for real-time jobs

This section describes some of the most common operations that real-time jobs can perform and how to define them in the Designer:

- Supplementing message data
- Branching data flow based on a data cache value
- Calling application functions

Also read about "Embedded Data Flows" on page 277 and the Case transform in "Descriptions of transforms" on page 225 of the *Data Integrator Reference Guide*.

Supplementing message data

The data included in messages from real-time sources might not map exactly to your requirements for processing or storing the information. If not, you can define steps in the real-time job to supplement the message information.

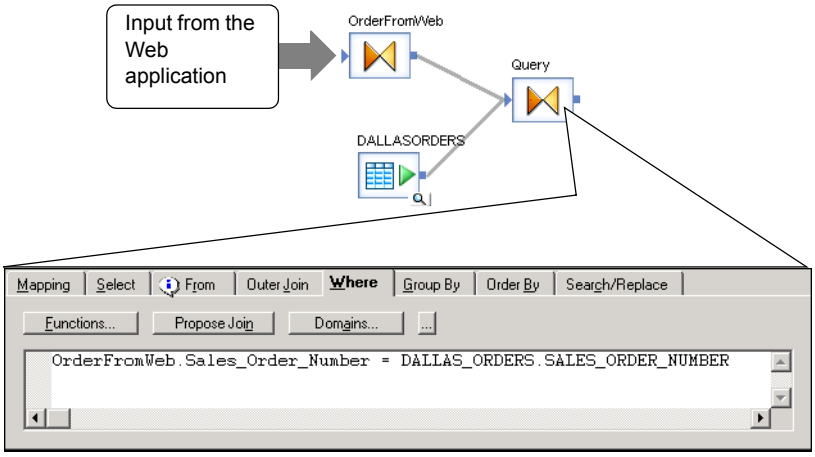
One technique for supplementing the data in a real-time source includes these steps:

1. Include a table or file as a source.

   In addition to the real-time source, include the files or tables from which you require supplementary information.

2. Use a query to extract the necessary data from the table or file.
3. Use the data in the real-time source to find the necessary supplementary data.

You can include a join expression in the query to extract the specific values required from the supplementary source.



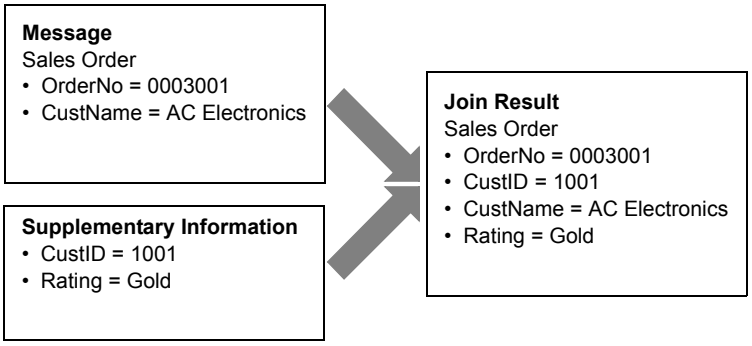
The WHERE clause joins the two inputs, resulting in output for only the sales document and line items included in the input from the application.

Be careful to use data in the join that is guaranteed to return a value. If no value returns from the join, the query produces no rows and the message returns to the Access Server empty. If you cannot guarantee that a value returns, consider these alternatives:

- **Lookup function call** — Returns a default value if no match is found
- **Outer join** — Always returns a value, even if no match is found

➢ To supplement message data

In this example, a request message includes sales order information and its reply message returns order status. The business logic uses the customer number and priority rating to determine the level of status to return. The message includes only the customer name and the order number. A real-time job is then defined to retrieve the customer number and rating from other sources before determining the order status.



1. Include the real-time source in the real-time job.
2. Include the supplementary source in the real-time job.

   This source could be a table or file. In this example, the supplementary information required doesn't change very often, so it is reasonable to extract the data from a data cache rather than going to an ERP system directly.
3. Join the sources.

   In a query transform, construct a join on the customer name:

   ```
   (Message.CustName = Cust_Status.CustName)
   ```

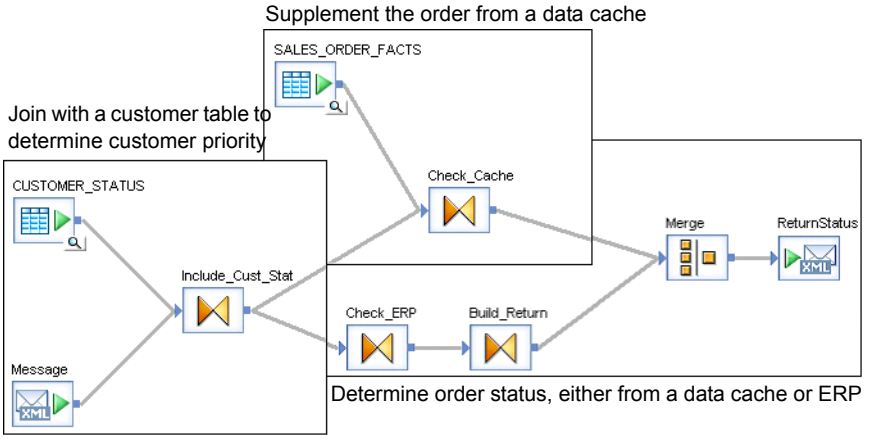
   You can construct the output to include only the columns that the real-time job needs to determine order status.

4. Complete the real-time job to determine order status.

   The example shown here determines order status in one of two methods based on the customer status value. Order status for the highest ranked customers is determined directly from the ERP. Order status for other customers is determined from a data cache of sales order information.

   The logic can be arranged in a single or multiple data flows. The illustration below shows a *single data flow model*.

   Both branches return order status for each line item in the order. The data flow merges the results and constructs the response. The next section describes how to design branch paths in a data flow.



Branching data flow based on a data cache value

One of the most powerful things you can do with a real-time job is to design logic that determines whether responses should be generated from a data cache or if they must be generated from data in a back-office application (ERP, SCM, CRM).

One technique for constructing this logic includes these steps:

1. Determine the rule for when to access the data cache and when to access the back-office application.
2. Compare data from the real-time source with the rule.
3. Define each path that could result from the outcome.

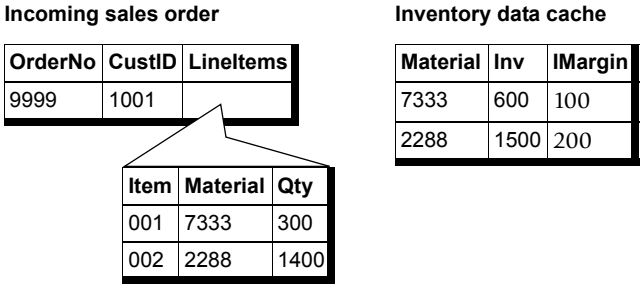
   You might need to consider the case where the rule indicates back-office application access, but the system is not currently available.
4. Merge the results from each path into a single data set.
5. Route the single result to the real-time target.

   You might need to consider error-checking and exception-handling to make sure that a value passes to the target. If the target receives an empty set, the real-time job returns an empty response (begin and end XML tags only) to the Access Server.

This example describes a section of a real-time job that processes a new sales order. The section is responsible for checking the inventory available of the ordered products—it answers the question, "is there enough inventory on hand to fill this order?"

The rule controlling access to the back-office application indicates that the inventory (Inv) must be more than a pre-determined value (IMargin) greater than the ordered quantity (Qty) to consider the data cached inventory value acceptable.

Data Integrator makes a comparison for each line item in the order.



The quantity of items in the sales order is compared to inventory values in the data cache.

➢ To branch a data flow based on a rule

1. Create a real-time job and drop a data flow inside it.
2. Add the XML source in the data flow.

   See "To import a DTD or XML Schema format" on page 231 to define the format of the data in the XML message.
3. Determine the values you want to return from the data flow.

   The XML source contains the entire sales order, yet the data flow compares values for line items inside the sales order. The XML target that ultimately returns a response to the Access Server requires a single row at the top-most level. Because this data flow needs to be able to determine inventory values for multiple line items, the structure of the output requires the inventory information to be nested. The input is already nested under the sales order; the output can use the same convention.

In addition, the output needs to include some way to indicate whether the inventory is or is not available.

Incoming sales order

OrderNo	CustID	LineItems
9999	1001	

Item	Material	Qty
001	7333	300
002	2288	1400

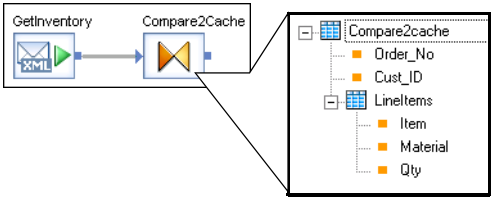
Outgoing sales order

OrderNo	CustID	LineItems
9999	1001	

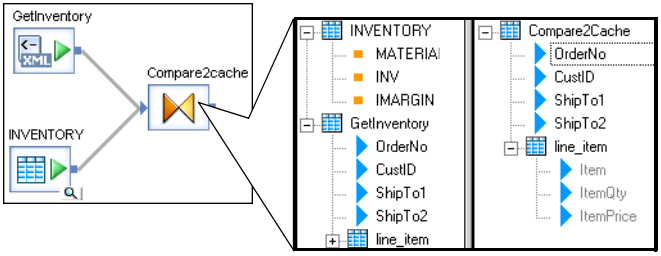
Item	Material	Qty	Inv
001	7333	300	
002	2288	1400	1300

4. Connect the output of the XML source to the input of a query and map the appropriate columns to the output.

   You can drag all of the columns and nested tables from the input to the output, then delete any unneeded columns or nested tables from the output.



5. Add the comparison table from the data cache to the data flow as a source.

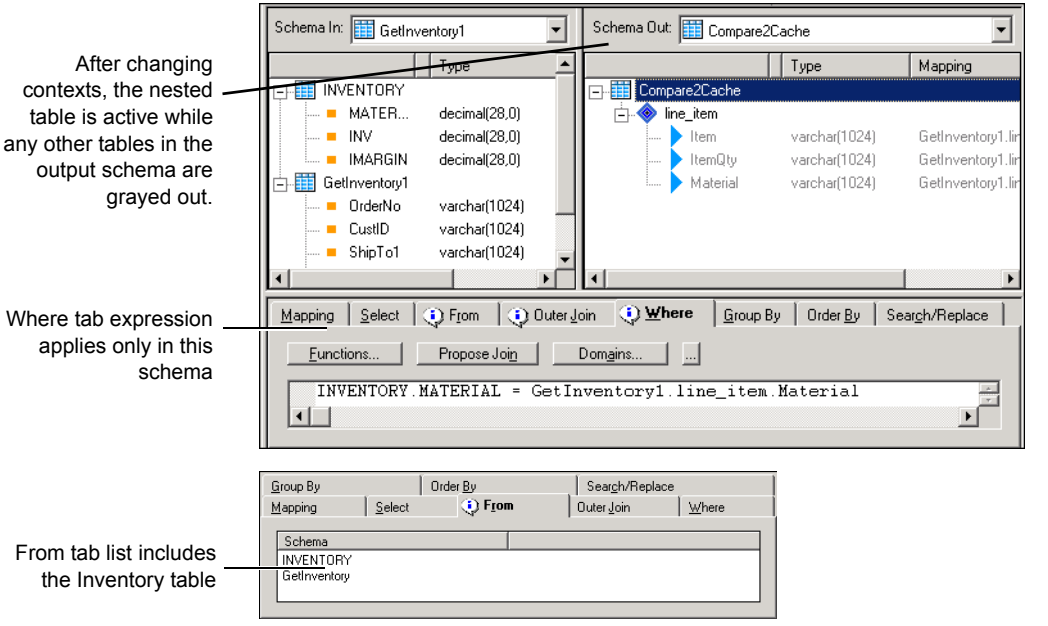


6. Construct the query so you extract the expected data from the inventory data cache table.

Without nested data, you would add a join expression in the WHERE clause of the query. Because the comparison occurs between a nested table and another top-level table, you have to define the join more carefully:

- Change context to the LineItem table
- Include the Inventory table in the FROM clause in this context (the LineItem table is already in the From list)
- Define an outer join with the Inventory table as the inner table
- Add the join expression in the WHERE clause in this context

In this example, you can assume that there will always be exactly one value in the Inventory table for each line item and can therefore leave out the outer join definition.



7. Include the values from the Inventory table that you need to make the comparison.

Drag the Inv and IMargin columns from the input to the LineItem table.

8. Split the output of the query based on the inventory comparison.

   Add two queries to the data flow:

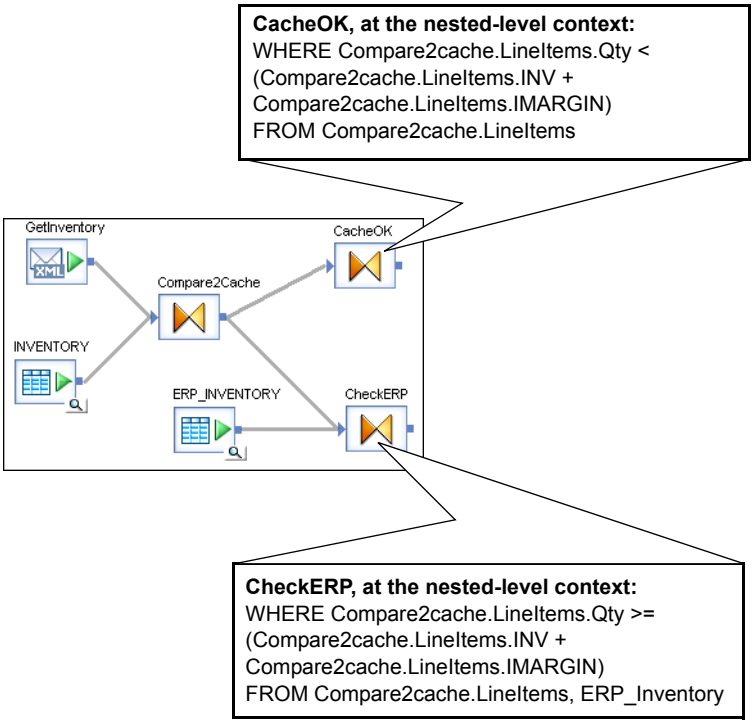
   - Query to process valid inventory values from the data cache

     The WHERE clause at the nested level (LineItem) of the query ensures that the quantities specified in the incoming line item rows are appropriately accounted for by inventory values from the data cache.

   - Query to retrieve inventory values from the ERP

     The WHERE clause at the nested level (LineItem) of the query ensures that the quantities specified in the incoming line item rows are not accounted for by inventory values from the data cache. The inventory values in the ERP inventory table are then substituted for the data cache inventory values in the output.

There are several ways to return values from the ERP. For example, you could use a lookup function or a join on the specific table in the ERP system. This example uses a join so that the processing can be performed by the ERP system rather than Data Integrator. As in the previous join, if you cannot guarantee that a value will be returned by the join, make sure to define an outer join so that the line item row is not lost.



9. Show inventory levels only if less than the order quantity.

   The goal from this section of the data flow was an answer to, "is there enough inventory to fill this order?"

   To complete the order processing, each branch returns an inventory value that can then be compared to the order quantity to answer the question.

The "CacheOK" branch of this example always returns line-item rows that include enough inventory to account for the order quantity; you can remove the inventory value from the output of these rows.

The "CheckERP" branch can return line item rows without enough inventory to account for the order quantity; the available inventory value can be useful if customers want to change their order quantity to match the inventory available.

Change the mapping of the Inv column in each of the branches to show available inventory values only if they are less than the order quantity.

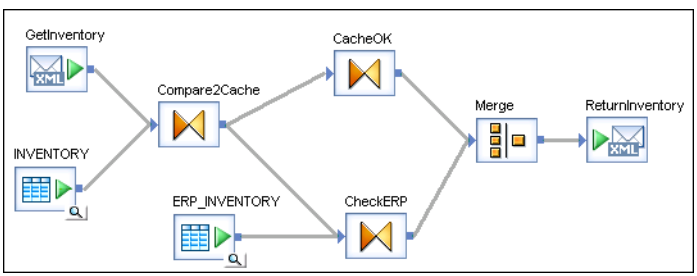
- For data cache OK: Inv maps from `'NULL'`
- For CheckERP: Inv maps from `ERP_Inventory.INV - ERP_Inventory.IMARGIN`

10. Merge the branches into one response.

    Both branches of the data flow include the same column and nested tables. The Merge transform combines the results of the two branches into a single data set.

11. Complete the processing of the message.

    Add the XML target to the output of the Merge transform.



Calling application functions

A real-time job can use application functions to operate on data. You can include tables as input or output parameters to the function.

Application functions require input values for some parameters and some can be left unspecified. You must determine the requirements of the function to prepare the appropriate inputs.

To make up the input, you can specify the top-level table, top-level columns, and any tables nested one-level down relative to the tables listed in the FROM clause of the context calling the function. If the application function includes a structure as an input parameter, you must specify the individual columns that make up the structure.

A data flow may contain several steps that call a function, retrieve results, then shape the results into the columns and tables required for a response.

Designing real-time applications

Data Integrator provides a reliable and low-impact connection between a Web application and an back-office applications such as an enterprise resource planning (ERP) system. Because each implementation of an ERP system is different and because Data Integrator includes versatile decision support logic, you have many opportunities to design a system that meets your internal and external information and resource needs.

This section discusses:

- Reducing queries requiring back-office application access
- Messages from real-time jobs to adapter instances
- Real-time service invoked by an adapter instance

Reducing queries requiring back-office application access

This section provides a collection of recommendations and considerations that can help reduce the time you spend experimenting in your development cycles.

The information you allow your customers to access through your Web application can impact the performance that your customers see on the Web. You can maximize performance through your Web application design decisions. In particular, you can structure your application to reduce the number of queries that require direct back-office (ERP, SCM, Legacy) application access.

For example, if your ERP system supports a complicated pricing structure that includes dependencies such as customer priority, product availability, or order quantity, you might not be able to depend on values from a data cache for pricing information. The alternative might be to request pricing information directly from the ERP system. ERP system access is likely to be much slower than direct database access, reducing the performance your customer experiences with your Web application.

To reduce the impact of queries requiring direct ERP system access, modify your Web application. Using the pricing example, design the application to avoid displaying price information along with standard product information and instead show pricing only after the customer has chosen a specific product and quantity. These techniques are evident in the way airline reservations systems provide pricing information—a quote for a specific flight—contrasted with other retail Web sites that show pricing for every item displayed as part of product catalogs.

Messages from real-time jobs to adapter instances

If a real-time job will send a message to an adapter instance, refer to the adapter documentation to decide if you need to create a message function call or an outbound message.

- Message function calls allow the adapter instance to collect requests and send replies.
- Outbound message objects can only send outbound messages. They cannot be used to receive messages.

For information on importing message function calls and outbound messages, see "Importing metadata through an adapter datastore" on page 109.

Using these objects in real-time jobs is the same as in batch jobs. See "To modify output schema contents" on page 176.

Real-time service invoked by an adapter instance

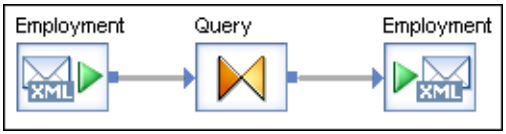
This section uses terms consistent with Java programming. (Please see your adapter SDK documentation for more information about terms such as *operation instance* and *information resource*.)

When an operation instance (in an adapter) gets a message from an information resource, it translates it to XML (if necessary), then sends the XML message to a real-time service.

In the real-time service, the message from the adapter is represented by a DTD or XML Schema object (stored in the **Formats** tab of the object library). The DTD or XML Schema represents the data schema for the information resource.

The real-time service processes the message from the information resource (relayed by the adapter) and returns a response.

In the example data flow below, the Query processes a message (here represented by "Employment") received from a source (an adapter instance), and returns the response to a target (again, an adapter instance).



11 Embedded Data Flows

Data Integrator provides an easy-to-use option to create embedded data flows.

This section covers the following topics:

- Overview
- Example of when to use embedded data flows
- Creating embedded data flows
- Using embedded data flows
- Testing embedded data flows
- Troubleshooting embedded data flows

Overview

An embedded data flow is a data flow that is called from inside another data flow. Data passes into or out of the embedded data flow from the parent flow through a single source or target. The embedded data flow can contain any number of sources or targets, but only one input or one output can pass data to or from the parent data flow.

There are four types of embedded data flows:

Type	Use when you want to...
One input	Add an embedded data flow at the end of a data flow
One input and one output	Add an embedded data flow in the middle of a data flow.
One output	Add an embedded data flow at the beginning of a data flow
No input or output	Replicate an existing data flow.

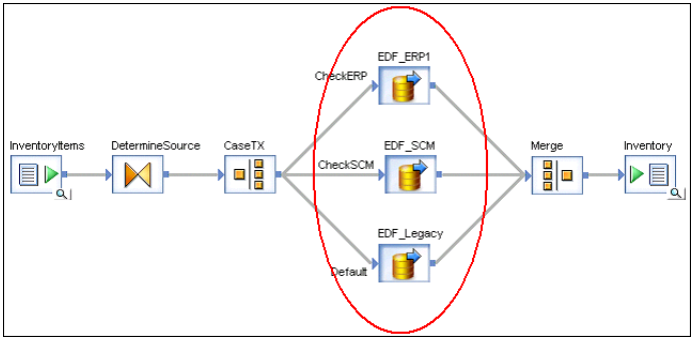
An embedded data flow is a design aid that has no effect on job execution. When Data Integrator executes the parent data flow, it expands any embedded data flows, optimizes the parent data flow, then executes it.

Use embedded data flows to:

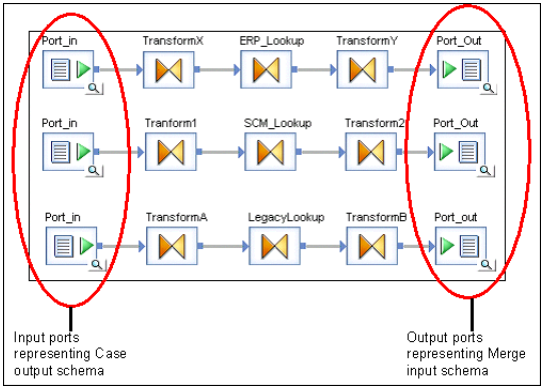
- Simplify data flow display. Group sections of a data flow in embedded data flows to allow clearer layout and documentation.
- Reuse data flow logic. Save logical sections of a data flow so you can use the exact logic in other data flows, or provide an easy way to replicate the logic and modify it for other flows.
- Debug data flow logic. Replicate sections of a data flow as embedded data flows so you can execute them independently.

Example of when to use embedded data flows

In this example, a parent data flow is simplified by using embedded data flows for the three different cases.



Each embedded data flow takes data from the source and processes it to get a unique target output:



Creating embedded data flows

There are two ways to create embedded data flows.

- Select objects within a data flow, right-click, and select **Make Embedded Data Flow**.
- Drag a complete and fully validated data flow from the object library into an open data flow in the workspace. Then:
  - Open the data flow you just added.
  - Right-click the objects you want to use as input and output ports and select **Make Port**.

    Data Integrator marks the objects you select as connection points for this embedded data flow.

Using the Make Embedded Data Flow option

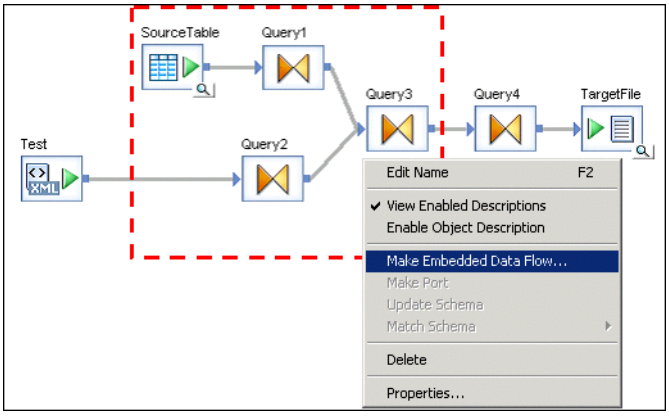
➢ To create an embedded data flow

1. Select objects from an open data flow using one of the following methods:
   - Click and drag
   - CTRL-click

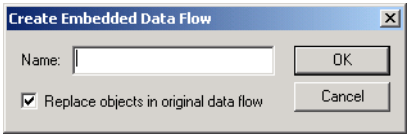
   Ensure that the set of objects you select are:
   - All connected to each other
   - Connected to other objects according to the type of embedded data flow you want to create:
     - One input
     - One output
     - Both one input and one output
     - No input or output

In the example shown in step 2, the embedded data flow is connected to the parent by both one input and one output object.

2. Right-click and select **Make Embedded Data Flow**.



The Create Embedded Data Flow window opens.



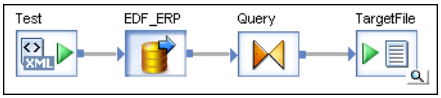
3. Name the embedded data flow using the convention `EDF_EDFName` for example `EDF_ERP`.

   If you deselect the **Replace objects in original data flow** box, Data Integrator will not make a change in the original data flow. Data Integrator the new embedded data flow object to the repository and displays it in the object library under the **Data Flows** tab.

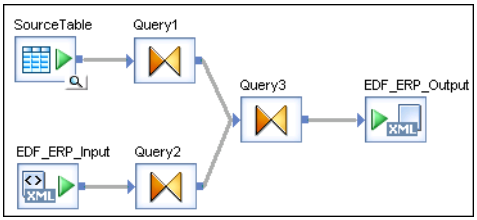
   You can use an embedded data flow created without replacement as a stand-alone data flow for troubleshooting.

If **Replace objects in original data flow** is selected, the original data flow becomes a parent data flow, which has a call to the new embedded data flow. The embedded data flow is represented in the new parent data flow as shown in step 4.

4. Click **OK**.



5. Click the name of the embedded data flow to open it.
6. Notice that Data Integrator created new objects (in this case, `EDF_ERP_Input` and `EDF_ERP_Output`) for a embedded data flow based on the objects that were connected to the embedded data flow when it was created.



When you use the **Make Embedded Data flow** option, Data Integrator automatically creates input and/or output objects based the objects that are connected to the embedded data flow when it is created.

For example, if an embedded data flow has an output connection but no input connection (no preceding object in the original data flow), the embedded data flow will include a target XML file object labeled *`EDFName`*`_NoInput_Output`.

The naming conventions for each embedded data flow type are:

Type	Naming Conventions
One input	*EDFName*_NoOutput_Input
One input and output	*EDFName*_Input
	*EDFName*_Output
One output	*EDFName*_NoInput_Output
No input or output	Data Integrator creates an embedded data flow without input or output objects

Creating embedded data flows from existing flows

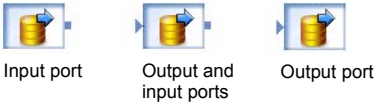
To call an existing data flow from inside another data flow, put the data flow inside the parent data flow, then mark which source and or target should be used to pass data between the parent and the embedded data flows.

➢ To create an embedded data flow out of an existing data flow

1. Drag an existing valid data flow from the object library into a data flow that is open in the workspace.
2. Consider renaming the flow using the EDF_*EDFName* naming convention.

   The embedded data flow appears without any arrowheads (ports) in the workspace.
3. Open the embedded data flow.
4. Right-click a source and/or target object (file or table) and select **Make Port**.

Different types of embedded data flow ports are indicated by directional markings on the embedded data flow icon.



Using embedded data flows

When you create and configure an embedded data flow using the **Make Embedded Data Flow** option, Data Integrator creates new input and/or output XML files and saves their schemas in the repository as XML Schemas. You can reuse an embedded data flow by dragging it from the **Data Flow** tab of the object library into other data flows. To save mapping time, you might want to use the Update Schema option or the Match Schema option.

The following example scenario uses both options:

- Create data flow 1.
- Select objects in data flow 1, and create embedded data flow 1 so that parent data flow 1 calls embedded data flow 1.
- Create data flow 2 and data flow 3 and add embedded data flow 1 to both of them.
- Go back to data flow 1. Change the schema of the object preceding embedded data flow 1 and use the Update Schema option with embedded data flow 1. It updates the schema of embedded data flow 1 in the repository.
- Now the schemas in data flow 2 and data flow 3 that are feeding into embedded data flow 1 will be different from the schema the embedded data flow expects.
- Use the Match Schema option for embedded data flow 1 in both data flow 2 and data flow 3 to resolve the mismatches at runtime. The Match Schema option only affects settings in the current data flow.

The following sections describe the use of the Update Schema and Match Schema options in more detail.

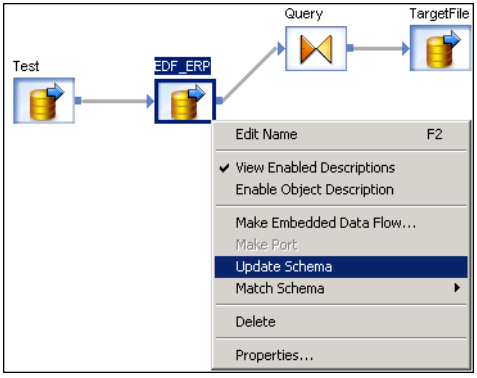
Updating Schemas

Data Integrator provides an option to update an input schema of an embedded data flow. This option updates the schema of an embedded data flow's input object with the schema of the preceding object in the parent data flow. All occurrences of the embedded data flow update when you use this option.

➢ To update a schema

1. Open the embedded data flow's parent data flow.
2. Right-click the embedded data flow object and select **Update Schema**.

For example, in the data flow shown below, Data Integrator copies the schema of `Test` to the input of `EDF_ERP`.



Matching data between parent and embedded data flow

The schema of an embedded data flow's input object can match the schema of the preceding object in the parent data flow by name or position. A match by position is the default.

➢ To specify how schemas should be matched

1. Open the embedded data flow's parent data flow.

2. Right-click the embedded data flow object and select **Match Schema** > **By Name** or **Match Schema** > **By Position**.

   The Match Schema option only affects settings for the current data flow.

Data Integrator also allows the schema of the preceding object in the parent data flow to have more or fewer columns than the embedded data flow. The embedded data flow ignores additional columns and reads missing columns as NULL.

Columns in both schemas must have identical or convertible data types. See the section on "Type conversion" in the *Data Integrator Reference Guide* for more information.

Deleting embedded data flow objects

You can delete embedded data flow ports, or remove entire embedded data flows.

➢ To remove a port

Right-click the input or output objects within the embedded data flow and deselect **Make Port**. Data Integrator removes the connections to parent objects.

NOTE: You cannot remove a port simply by deleting the connection in the parent flow.

➢ To remove an embedded data flow

Select it from the open parent data flow and choose **Delete** from the right-click menu or edit menu.

If you delete embedded data flows from the object library, the embedded data flow icon appears with a red circle-slash flag in the parent data flow.

Delete these defunct embedded data flow objects from the parent data flows.

Testing embedded data flows

You might find it easier to test embedded data flows by running them separately as regular data flows.

➢ To separately test an embedded data flow

1. Specify an XML file for input port and/or output ports.

   When you use the **Make Embedded Data Flow** option, input and /or output XML file objects are created then (optional) connected to the preceding and succeeding objects in the parent data flow. To test these XML files without a parent data flow, click the name of each XML file to open its source or target editor to specify a file name. For more configuration information see "XML file" on page 145 of the *Data Integrator Reference Guide*.

2. Put the embedded data flow into a job.
3. Run the job.

You can also use the view data feature to sample data passed into an embedded data flow. See Chapter 14, "Design and Debug using View Data" for more information.

Troubleshooting embedded data flows

The following situations produce errors:

- Trapped defunct data flows. See "To remove an embedded data flow" on page 286.
- Deleted connections to the parent data flow while the **Make Port** option, in the embedded data flow, remains selected. See "To remove a port" on page 286.

- Transforms with splitters (such as the Case transform) specified as the output port object because a splitter produces multiple outputs, and embedded data flows can only have one.
- Variables and parameters declared in the embedded data flow that are not also declared in the parent data flow.
- Embedding the same data flow at any level within itself.

  You can however have unlimited embedding levels. For example, DF1 data flow calls EDF1 embedded data flow which calls EDF2.

12 Variables and Parameters

This chapter covers creating local and global variables for Data Integrator jobs. It also introduces the use of environment variables.

This chapter contains the following sections:

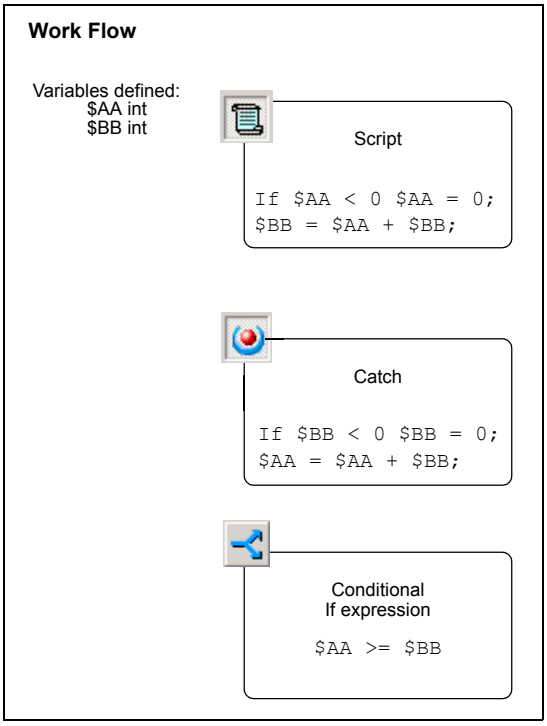
Overview

You can increase the flexibility and reusability of work flows and data flows using local and global variables when you design your jobs. *Variables* are symbolic placeholders for values. The data type of a variable can be any supported by Data Integrator such as an integer, decimal, date, or text string.

You can use variables in expressions to facilitate decision-making or data manipulation (using arithmetic or character substitution). For example, a variable can be used in a LOOP or IF statement to check a variable's value to decide which step to perform:

```
If $amount_owed > 0 print('$invoice.doc');
```

If you define variables in a job or work flow, Data Integrator typically uses them in a script, catch, or conditional process.



You can use variables inside data flows. For example, use them in a custom function or in the WHERE clause of a query transform.

In Data Integrator, *local variables* are restricted to the object in which they are created (job or work flow). You must use parameters to pass local variables to child objects (work flows and data flows).

Global variables are restricted to the job in which they are created; however, they do not require parameters to be passed to work flows and data flows.

Parameters are expressions that pass to a work flow or data flow when they are called in a job.

You create local variables, parameters, and global variables using the Variables and Parameters window in the Designer.

You can set values for local or global variables in script objects. You can also set global variable values using external job, execution, or schedule properties.

Using global variables provides you with maximum flexibility. For example, during production you can change values for default global variables at runtime from a job's schedule or SOAP call without having to open a job in the Designer. For more information about setting global variable values in SOAP calls, see "Support for Web Services" on page 115 of the *Data Integrator Administrator Guide*.

Variables can be used as file names for:

- Flat file sources and targets
- XML file sources and targets
- XML message targets (executed in the Designer in test mode)
- IDoc file sources and targets (in an SAP R/3 environment)
- IDoc message sources and targets (SAP R/3 environment)

The Variables and Parameters window

Data Integrator displays the variables and parameters defined for an object in the Variables and Parameters window.

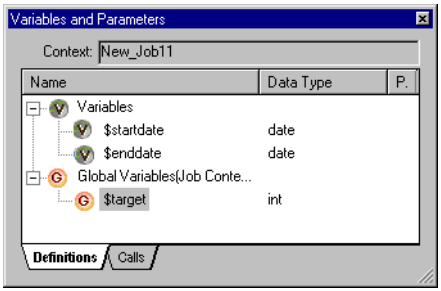
➢ To view the variables and parameters in each job, work flow, or data flow

1. In the **Tools** menu, select **Variables**.

   The Variables and Parameters window opens.

2. From the object library, double-click an object, or from the project area click an object to open it in the workspace.

   The **Context** box in the window changes to show the object you are viewing. If there is no object selected, the window does not indicate a context.



The Variables and Parameters window contains two tabs.

The **Definitions** tab allows you to create and view variables (name and data type) and parameters (name, data type, and parameter type) for an object type. Local variable parameters can only be set at the work flow and data flow level. Global variables can only be set at the job level.

The following table lists what type of variables and parameters you can create using the Variables and Parameters window when you select different objects.

Object Type	What you can create for the object	Used by
Job	Local variables	A script or conditional in the job
	Global variables	Any object in the job
Work flow	Local variables	This work flow or passed down to other work flows or data flows using a parameter.
	Parameters	Parent objects to pass local variables. Work flows may also return variables or parameters to parent objects.
Data flow	Parameters	A WHERE clause, column mapping, or a function in the data flow. Data flows cannot return output values.

The **Calls** tab allows you to view the name of each parameter defined for all objects in a parent object's definition. You can also enter values for each parameter by right-clicking a parameter and clicking **Properties**.

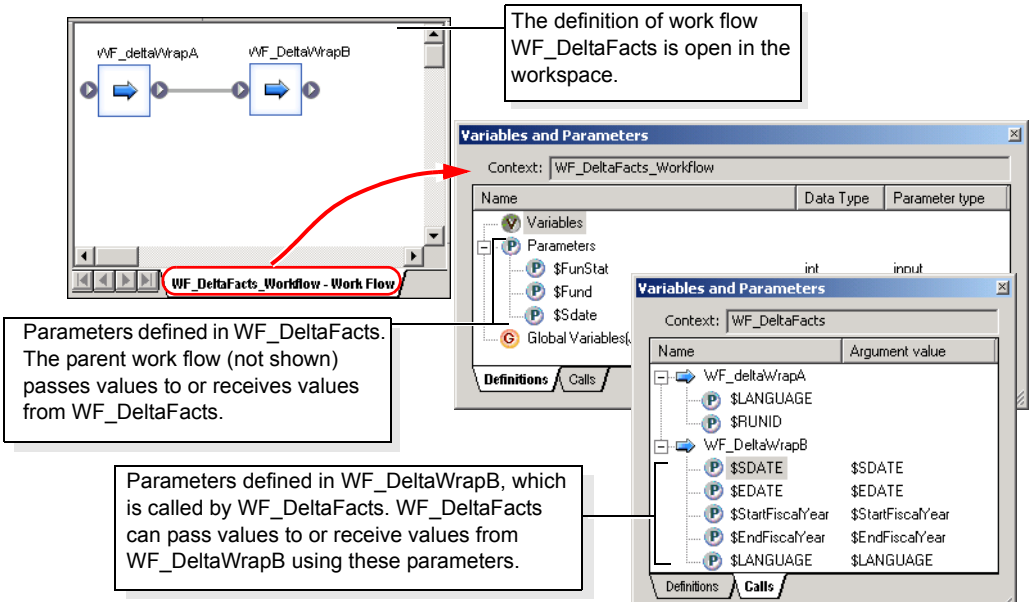
For the input parameter type, values in the **Calls** tab can be constants, variables, or another parameter.

For the output or input/output parameter type, values in the **Calls** tab can be variables or parameters.

Values in the **Calls** tab must also use:

- The same data type as the variable if they are placed inside an input or input/output parameter type, and a compatible data type if they are placed inside an output parameter type.
- Data Integrator scripting language rules and syntax

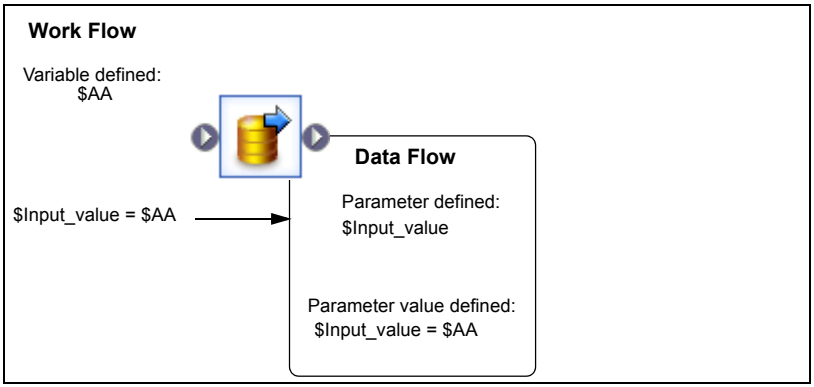
The following illustration shows the relationship between an open work flow called DeltaFacts, the **Context** box in the Variables and Parameters window, and the content in the **Definition** and **Calls** tabs.



Using local variables and parameters

To pass a local variable to another object, define the local variable, then from the calling object, create a parameter and map the parameter to the local variable by entering a parameter value.

For example, to use a local variable inside a data flow, define the variable in a parent work flow and then pass the value of the variable as a parameter of the data flow.



Parameters

Parameters can be defined to:

- Pass their values into and out of work flows
- Pass their values into data flows

Each parameter is assigned a type: input, output, or input/output. The value passed by the parameter can be used by any object called by the work flow or data flow.

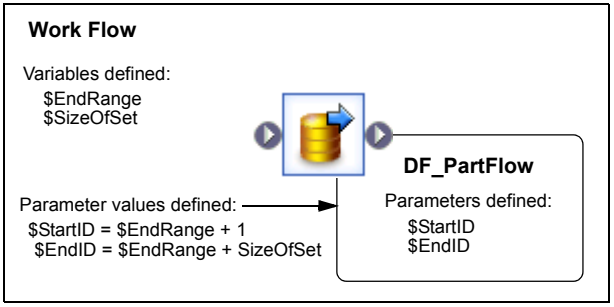
NOTE: You can also create local variables and parameters for use in custom functions. For more information see "Custom functions" on page 333 of the *Data Integrator Reference Guide*.

Passing values into data flows

You can use a value passed as a parameter into a data flow to control the data transformed in the data flow. For example, the data flow DF_PartFlow processes daily inventory values. It can process all of the part numbers in use or a range of part numbers based on external requirements such as the range of numbers processed most recently.

If the work flow that calls DF_PartFlow records the range of numbers processed, it can pass the end value of the range `$EndRange` as a parameter to the data flow to indicate the start value of the range to process next.

Data Integrator can calculate a new end value based on a stored number of parts to process each time, such as `$SizeOfSet`, and pass that value to the data flow as the end value. A query transform in the data flow uses the parameters passed in to filter the part numbers extracted from the source.



The data flow could be used by multiple calls contained in one or more work flows to perform the same task on different part number ranges by specifying different parameters for the particular calls.

Defining local variables

Variables are defined in the Variables and Parameter window.

➢ To define a local variable

1. Click the name of the job or work flow in the project area or workspace, or double-click one from the object library.
2. Choose **Tools** > **Variables** to open the Variables and Parameters window.
3. Go the **Definition** tab.
4. Select **Variables**.
5. Right-click and choose **Insert**.
6. Select the new variable (for example, `$NewVariable0`).
7. Right-click and choose **Properties**.
8. Enter the name of the new variable.

   The name can include any alpha or numeric character or underscores (_), but cannot contain blank spaces. Always begin the name with a dollar sign ($).
9. Select the data type for the variable.
10. Click **OK**.

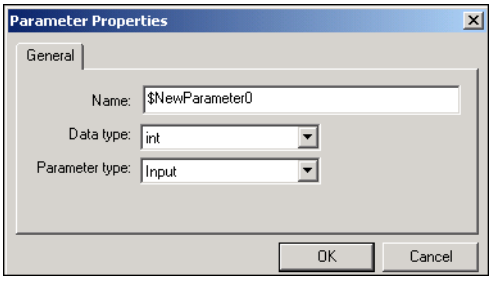
Defining parameters

There are two steps for setting up a parameter for a work flow or data flow:

- Add the parameter definition to the flow.
- Set the value of the parameter in the flow call.

➢ To add the parameter to the flow definition

1. Click the name of the work flow or data flow.
2. Open the Variables and Parameters window.
3. Go to the **Definition** tab.
4. Select **Parameters**.
5. Right-click and choose **Insert**.
6. Select the new parameter (for example, `$NewArgument1`).
7. Right-click and choose **Properties**.



8. Enter the name of the parameter using alphanumeric characters with no blank spaces.
9. Select the data type for the parameter.

   The parameter must have the same data type as the variable if it is an input or input/output parameter; it must have a compatible data type if it is an output parameter type.
10. Select the parameter type (input, output, or input/output).
11. Click **OK**.

➢ To set the value of the parameter in the flow call

1. Open the calling job, work flow, or data flow.
2. In the Variables and Parameters window, select the **Calls** tab.

The **Calls** tab shows all the objects that are called from the open job, work flow, or data flow.

3. Click the plus sign (+) next to the object that contains the parameter you want to set.

   A list of parameters passed to that object appears.

4. Select the parameter, right-click, and choose **Properties**.

5. Enter the expression the parameter will pass in the **Value** box.

   If the parameter type is input, then its value can be an expression that contains a constant (for example 0, 3, or 'string1'), a variable, or another parameter (for example `$startID` or `$parm1`).

   If the parameter type is output or input/output, then the value is a variable or parameter.

   Use the following syntax to indicate special values:

Value type	Special syntax
Variable	*$variable_name*
String	*'string'*

6. Click **OK**.

Using global variables

Global variables are global within a job. Setting parameters is not necessary when you use global variables. However, once you use a name for a global variable in a job, that name becomes reserved for the job. Global variables are exclusive within the context of the job in which they are created.

This section discusses:

- Creating global variables
- Viewing global variables
- Setting global variable values

Creating global variables

Define variables in the Variables and Parameter window.

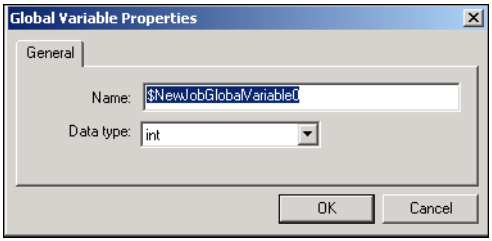
➢ To create a global variable

1. Click the name of a job in the project area or double-click a job from the object library.
2. Choose **Tools** > **Variables** to open the Variables and Parameters window.
3. Go the **Definition** tab.
4. Right-click **Global Variables (job Context_)** to open a shortcut menu.
5. From the shortcut menu, click **Insert**.

   $NewJobGlobalVariable appears inside the global variables tree:

   $NewJobGlobalVariable
6. Right-click **$NewJobGlobalVariable** and select **Properties** from the shortcut menu.

The Global Variable Properties window opens:



7. Rename the variable and select a data type.
8. Click **OK**.

   The Variables and Parameters window displays the renamed global variable.

Viewing global variables

Global variables, defined in a job, are visible to those objects relative to that job. A global variable defined in one job is not available for modification or viewing from another job.

You can view global variables from the Variables and Parameters window (with an open job in the work space) or from the Properties dialog of a selected job.

➢ To view global variables in a job from the Properties dialog

1. In the object library, select the **Jobs** tab.
2. Right-click and select **Properties**.
3. Click the **Global Variables** tab.

   Global variables appear on this tab.

Setting global variable values

In addition to setting a variable inside a job using an initialization script, you can set and maintain global variable values outside a job. Values set outside a job are processed the same way as those set in an initialization script. However, if you set a value for the same variable both inside and outside a job, the internal value will override the external job value.

Values for global variables can be set outside a job:

- As a job property
- As an execution or schedule property

Global variables without defined values are also allowed. They are read as NULL.

All values defined as job properties are shown in the Properties and the Execution Properties dialogs of the Designer and in the Execution Options and Schedule pages of the Administrator. By setting values outside a job, you can rely on these dialogs for viewing values set for global variables and easily edit values when testing or scheduling a job.

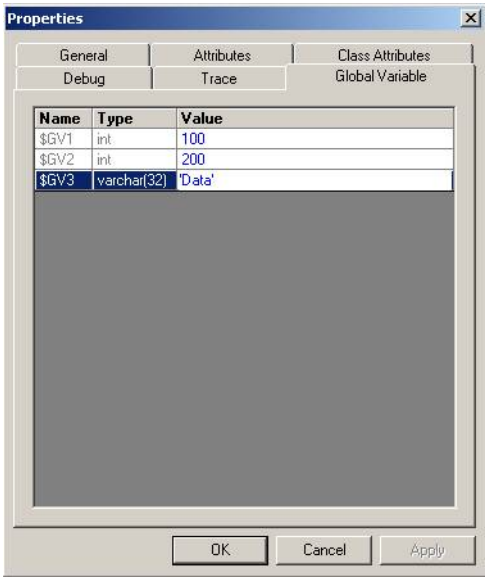
NOTE: You cannot pass global variables as command line arguments for real-time jobs.

➢ To set a global variable value as a job property

1. Right-click a job in the object library or project area.
2. Click **Properties**.
3. Click the **Global Variable** tab.

   All global variables created in the job appear.
4. Enter values for the global variables in this job.

You can use any statement used in a script with this option.



See "Data Integrator Scripting Language" on page 511 of the *Data Integrator Reference Guide* for syntax information and example scripts.

5. Click **OK**.

   Data Integrator saves values in the repository as job properties.

   You can also view and edit these default values in the Execution Properties dialog of the Designer and in the Execution Options and Schedule pages of the Administrator. This allows you to override job property values at runtime.

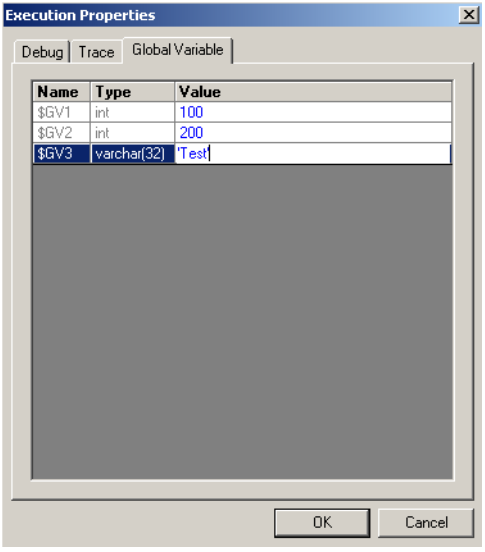
➢ To set a global variable value as an execution property

1. Execute a job from the Designer, or from the Administrator execute or schedule a batch job.

   NOTE: For testing purposes, you can execute real-time jobs from the Designer in test mode. Make sure to the execution properties for a real-time job.

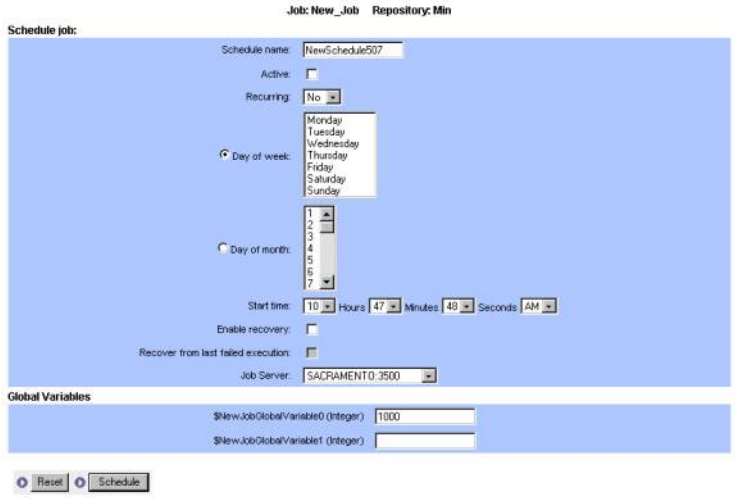
2. View the global variables in the job and their default values (if available).

Global Variables displayed on the Execution Properties dialog



Global Variables displayed on the Execution Properties page

Global Variables displayed on the Schedule Jobs page



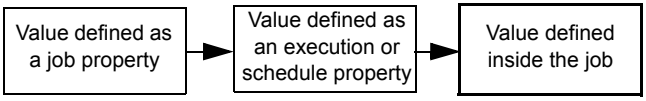
If no global variables exist in a job, the **Global Variable** sections in these windows do not appear.

3. Edit values for global variables as desired.
4. If you are using the Designer, click **OK**. If you are using the Administrator, click **Execute** or **Schedule**.

The job runs using the values you enter. Values entered as execution properties are not saved. Values entered as schedule properties are saved but can only be accessed from within the Administrator.

Automatic ranking of global variable values in a job

Using the methods described in the previous section, if you enter different values for a single global variable, Data Integrator selects the highest ranking value for use in the job. A value entered as a job property has the lowest rank. A value defined inside a job has the highest rank.



- If you set a global variable value as both a job and an execution property, the execution property value overrides the job property value and becomes the default value for the current job run. You cannot save execution property global variable values.

  For example, assume that a job, `JOB_Test1`, has three global variables declared: `$YEAR`, `$MONTH`, and `$DAY`. Variable `$YEAR` is set as a job property with a value of `2003`.

  For your the job run, you set variables `$MONTH` and `$DAY` as execution properties to values '`JANUARY`' and `31` respectively. Data Integrator executes a list of statements which includes default values for `JOB_Test1`:

```
$YEAR=2003;
$MONTH='JANUARY';
$DAY=31;
```

For the second job run, if you set variables $YEAR and $MONTH as execution properties to values 2002 and 'JANUARY' respectively, then the statement $YEAR=2002 will replace $YEAR=2003. Data Integrator executes the following list of statements:

```
$YEAR=2002;
$MONTH='JANUARY';
```

NOTE: In this scenario, $DAY is not defined and Data Integrator reads it as NULL. You set $DAY to 31 during the first job run; however, execution properties for global variable values are not saved.

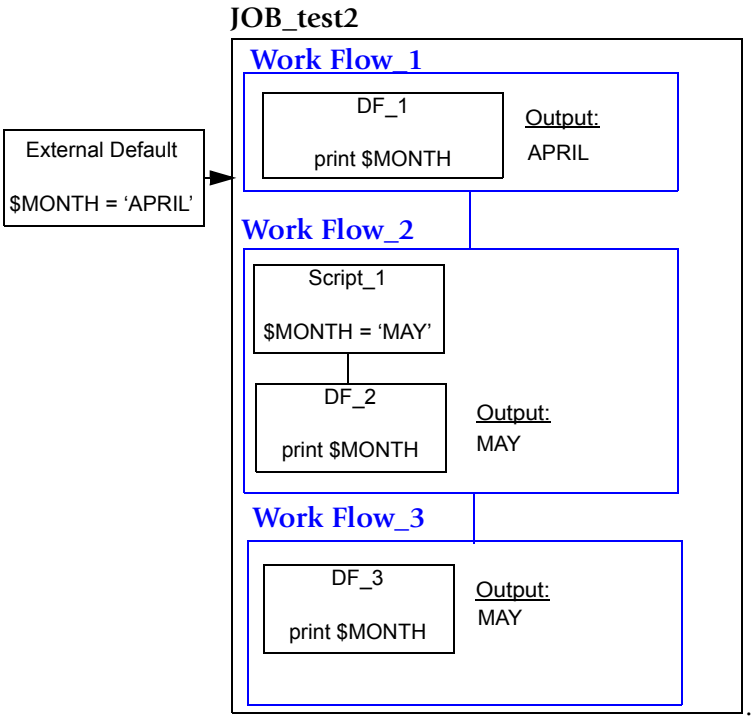
- If you set a global variable value for both a job property and a schedule property, the schedule property value overrides the job property value and becomes the external, default value for the current job run.

  Data Integrator saves schedule property values in the repository. However, these values are only associated with a job schedule, not the job itself. Consequently, these values are viewed and edited from within the Administrator.

- A global variable value defined inside a job always overrides any external values. However, the override does not occur until Data Integrator attempts to apply the external values to the job being processed with the internal value. Up until that point, Data Integrator processes execution, schedule, or job property values as default values.

  For example, suppose you have a job called JOB_Test2 that has three work flows, each containing a data flow. The second data flow is inside a work flow that is preceded by a script in which $MONTH is defined as 'MAY'. The first and third data flows have the same global variable with no value defined. The execution property $MONTH = 'APRIL' is the global variable value.

In this scenario, 'APRIL' becomes the default value for the job. 'APRIL' remains the value for the global variable until it encounters the other value for the same variable in the second work flow. Since the value in the script is inside the job, 'MAY' overrides 'APRIL' for the variable $MONTH. Data Integrator continues the processing the job with this new value.



Advantages to setting values outside a job

While you can set values inside jobs, there are advantages to defining values for global variables outside a job.

For example, values defined as job properties are shown in the Properties and the Execution Properties dialogs of the Designer and in the Execution Options and Schedule pages of the Administrator. By setting values outside a job, you can rely on these dialogs for viewing all global variables and their values. You can also easily edit them for testing and scheduling.

In the Administrator, you can set global variable values when creating or editing a schedule without opening the Designer. For example, use global variables as file names and start and end dates.

Local and global variable rules

When defining local or global variables, consider rules for:

- Naming
- Replicating jobs and work flows
- Importing and exporting

Naming

- Local and global variables must have unique names within their job context.
- Any name modification to a global variable can only be performed at the job level.

Replicating jobs and work flows

- When you replicate all objects, the local and global variables defined in that job context are also replicated.
- When you replicate a data flow or work flow, all parameters and local and global variables are also replicated. However, you must validate these local and global variables within the job context in which they were created. If you attempt to validate a data flow or work flow containing global variables without a job, Data Integrator reports an error.

Importing and exporting

- When you export a job object, you also export all local and global variables defined for that job.
- When you export a lower-level object (such as a data flow) without the parent job, the global variable is not exported. Only the call to that global variable is exported. If you use this object in another job without defining the global variable in the new job, a validation error will occur.

Environment variables

You can use system-environment variables inside Data Integrator jobs, work flows, or data flows. The *get_env*, *set_env*, and *is_set_env* functions provide access to underlying operating system variables that behave as the operating system allows.

You can temporarily set the value of an environment variable inside a job, work flow or data flow. Once set, the value is visible to all objects in that job.

Use the *get_env*, *set_env*, and *is_set_env* functions to set, retrieve, and test the values of environment variables. For more information about these functions, see Chapter 6, "Functions and Procedures," in the *Data Integrator Reference Guide*.

Setting file names at run-time using variables

You can set file names at runtime by specifying a variable as the file name.

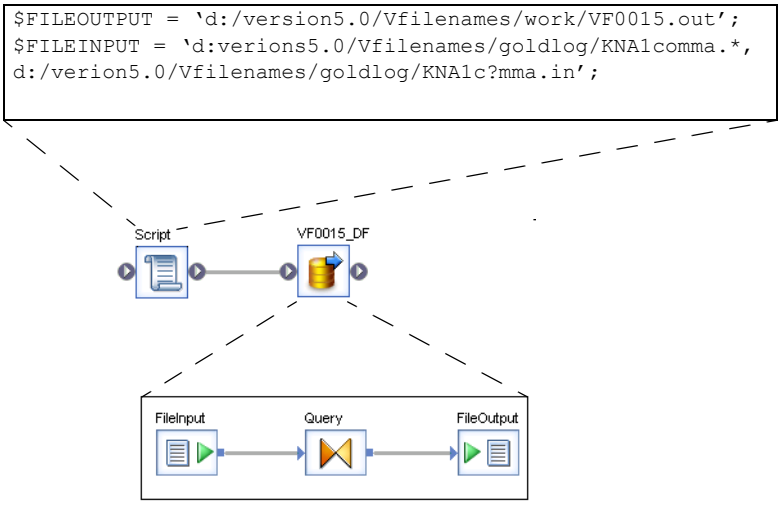
Variables can be used as file names for:

- The following sources and targets:
  - Flat files
  - XML files and messages
  - IDoc files and messages (in an SAP R/3 environment)
- The `lookup_ext` function (for a flat file used as a translate table parameter)

➢ To use a variable in a flat file name

1. Create a local or global variable using the Variables and Parameters window.
2. Create a script to set the value of a local or global variable, or call a system environment variable.
3. Declare the variable in the file format editor or in the Function editor as a `lookup_ext` parameter.
   - When you set a variable value for a flat file, specify both the file name and the directory name. Enter the variable in the **File(s)** property under **Data File(s)** in the File Format Editor. You cannot enter a variable in the **Root directory** property.
   - For lookups, substitute the path and file name in the **Translate table** box in the `lookup_ext` function editor with the variable name. For more information, see "lookup_ext" on page 411 of the *Data Integrator Reference Guide*.

The following figure shows how you can set values for variables in flat file sources and targets in a script.



When you use variables as sources and targets, you can also use multiple file names and wild cards. Neither is supported when using variables in the `lookup_ext` function.

The figure above provides an example of how to use multiple variable names and wild cards. Notice that the `$FILEINPUT` variable includes two file names (separated by a comma). The two names (`KNA1comma.*` and `KNA1c?mma.in`) also make use of the wild cards (`*` and `?`) supported by Data Integrator. See Chapter 7, "Data Integrator Scripting Language," in the *Data Integrator Reference Guide* for more information about creating scripts.

13 Executing Jobs

This chapter contains the following sections:

Overview of Data Integrator job execution

You can run Data Integrator jobs in three different ways. Depending on your needs, you can configure:

- **Immediate jobs**

  Data Integrator initiates both batch and real-time jobs and runs them immediately from within the Data Integrator Designer. For these jobs, both the Designer and designated Job Server (where the job executes, usually many times on the same machine) must be running. You will most likely run immediate jobs only during the development cycle.

- **Scheduled jobs**

  Batch jobs are scheduled. To schedule a job, use the Data Integrator Administrator or use a third-party scheduler.

  When jobs are scheduled by third-party software:

  - The job initiates outside of Data Integrator.
  - The job operates on a batch job (or shell script for UNIX) that has been exported from Data Integrator.

  When a job is invoked by a third-party scheduler:

  - The corresponding Job Server must be running.
  - The Data Integrator Designer does not need to be running.

- **Services**

  Real-time jobs are set up as services that continuously listen for requests from an Access Server and process requests on-demand as they are received. Use the Data Integrator Administrator to create a service from a real-time job.

Preparing for job execution

Follow these preparation procedures before you execute, schedule, or export a job to be executed as a scheduled task:

- Validating jobs and job components
- Ensuring that the Job Server is running
- Setting job execution options

Validating jobs and job components

You can also explicitly validate jobs and their components as you create them by:

Clicking the **Validate All** button from the toolbar (or choosing **Validate** > **All Objects in View** from the **Debug** menu). This command checks the syntax of the object definition for the active workspace and for all objects that are called from the active workspace view recursively.

Clicking the **Validate Current View** button from the toolbar (or choosing **Validate** > **Current View** from the **Debug** menu). This command checks the syntax of the object definition for the active workspace.

You can set the Designer options (**Tools** > **Options** > **Designer** > **General**) to validate jobs started in Designer before job execution. The default is not to validate.

Data Integrator also validates jobs before exporting them.

If during validation Data Integrator discovers an error in an object definition, it opens a dialog box indicating that an error exists, then opens the Output window to display the error.

If there are errors, double-click the error in the Output window to open the editor of the object containing the error.

If you are unable to read the complete error text in the window, you can access additional information by right-clicking the error listing and selecting **View** from the context menu.

Error messages have these levels of severity:

Severity	Description
Information 	Informative message only—does not prevent the job from running. No action is required.
Warning 	The error is not severe enough to stop job execution, but you might get unexpected results. For example, if the data type of a source column in a transform within a data flow does not match the data type of the target column in the transform, Data Integrator alerts you with a warning message.
Error 	The error is severe enough to stop job execution. You must fix the error before the job will execute.

Ensuring that the Job Server is running

Before you execute a job (either as an immediate or scheduled task), ensure that the Job Server is associated with the repository where the client is running.

When the Designer starts, it displays the status of the Job Server for the repository to which you are connected.

 Job Server is running

 Job Server is inactive

The name of the active Job Server and port number displays in the status bar when the cursor is over the icon.

The name of the active Job Server and port number display when you roll-over the Job Server icon.

Setting job execution options

Options for jobs include Debug and Trace. Although these are object options—they affect the function of the object—they are located in either the Property or the Execution window associated with the job.

Execution options for jobs can either be set for a single instance or as a default value.

- The right-click **Execute** menu sets the options for a single execution only and overrides the default settings
- The right-click **Properties** menu sets the default settings

➢ To set execution options for every execution of the job

1. From the **Project** area, right-click the job name and choose **Properties**.
2. Select options on the Properties window:
   - For an introduction to object properties, see "Viewing and changing object properties" on page 48
   - For information about **Debug** and **Trace** properties, see "Parameters" on page 17 of the *Data Integrator Reference Guide* and "Trace properties" on page 20 of the *Data Integrator Reference Guide*.
   - For more information about using the **Global Variable** tab, see "Setting global variable values" on page 303 of the *Data Integrator Designer Guide*.

Executing jobs as immediate tasks

Immediate or "on demand" tasks are initiated from the Data Integrator Designer. Both the Designer and Job Server must be running for the job to execute.

➢ To execute a job as an immediate task

1. In the project area, select the job name.
2. Right-click and choose **Execute**.

   Data Integrator prompts you to save any objects that have changes that have not been saved.
3. The next step depends on whether you selected the **Perform complete validation before job execution** check box in the Designer Options (see "Designer — General" on page 65):
   - If you have *not* selected this check box, a window opens showing execution properties (debug and trace) for the job. Proceed to the next step.
   - If you *have* selected this check box, Data Integrator validates the job before it runs. You must correct any serious errors before the job will run. There might also be warning messages—for example, messages indicating that date values will be converted to datetime values. Correct them if you want (they will not prevent job execution) or click **OK** to continue. After the job validates, a window opens showing the execution properties (debug and trace) for the job.
4. Set the execution properties.

   You can choose the Job Server that you want to process this job, datastore profiles for sources and targets if applicable, enable automatic recovery, override the default trace properties, or select global variables at runtime.

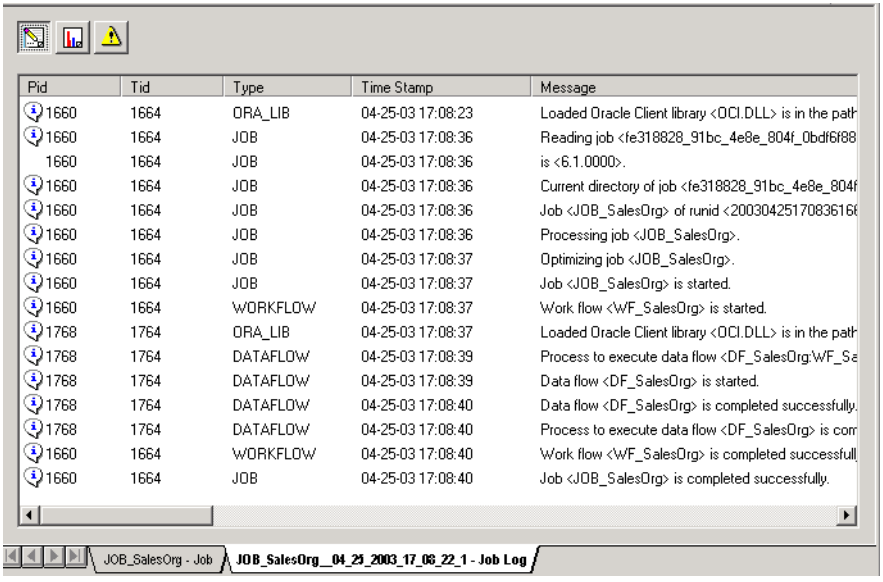
   For more information, see:

- "Parameters" on page 17 of the *Data Integrator Reference Guide*
- "Trace properties" on page 20 of the *Data Integrator Reference Guide*.
- "Setting global variable values" on page 303.

NOTE: Setting execution properties here affects a temporary change for the current execution only.

5. Click **OK**.

   As Data Integrator begins execution, the execution window opens with the trace log button active.



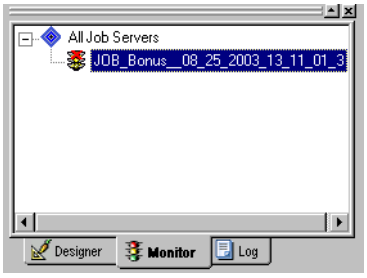
Use the buttons at the top of the log window to display the trace log, statistics log, and error log (if there are any errors).

For more information about execution logs, see "Debugging execution errors" on page 324.

After the job is complete, use an RDBMS query tool to check the contents of the target table or file. See "Examining target data" on page 329.

Monitor tab

The **Monitor** tab lists the trace logs of all current or most recent executions of a job.



The traffic-light icons in the **Monitor** tab have the following meanings:

- A green light indicates that the job is running

  You can right-click and select **Kill Job** to stop a job that is still running.

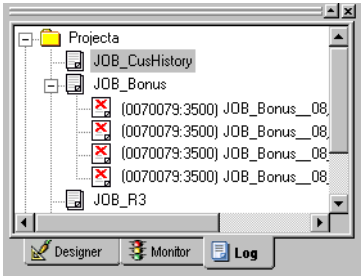
- A red light indicates that the job has stopped

  You can right-click and select **Properties** to add a description for a specific trace log. This description is saved with the log which can be accessed later from the **Log** tab.

- A red cross indicates that the job encountered an error

Log tab

You can also select the **Log** tab to view a job's trace log history.



Click on a trace log to open it in the workspace.

Use the trace, statistics, and error log icons (left to right at the top of the job execution window in the workspace) to view each type of available log for the date and time that the job was run.

Debugging execution errors

The following tables lists tools that can help you understand execution errors:

Tool	Definition
Trace log	Itemizes the steps executed in the job and the time execution began and ended.
Statistics log	Displays each step of each data flow in the job, the number of rows streamed through each step, and the duration of each step.
Error log	Displays the name of the object being executed when an Data Integrator error occurred and the text of the resulting error message. If the job ran against SAP data, some of the ABAP errors are also available in the Data Integrator error log.
Target data	Always examine your target data to see if your job produced the results you expected.

The following sections describe how to use these tools:

- Using Data Integrator logs
  - Examining trace logs
  - Examining statistics logs
  - Examining error logs
- Examining target data

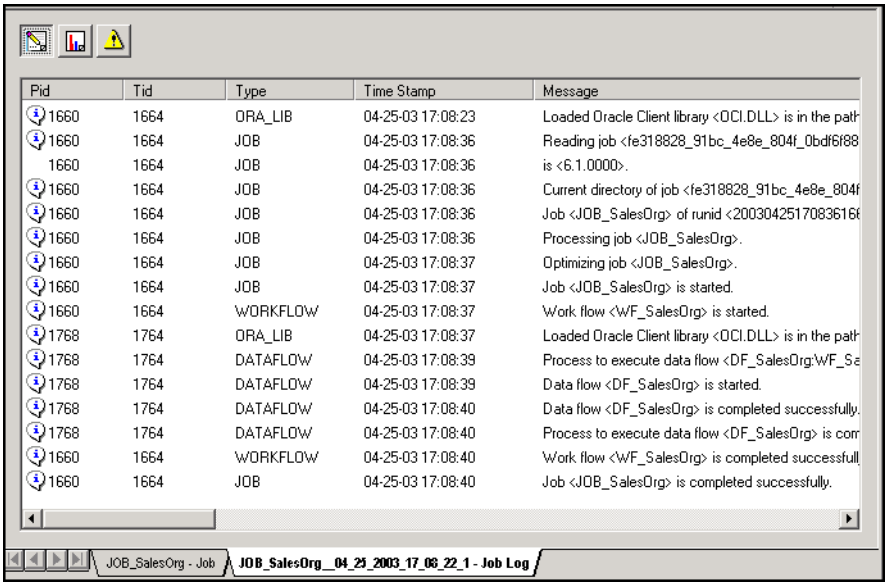
Using Data Integrator logs

This section describes how to use Data Integrator logs in the Designer. For information about administering logs from the Administrator, see the *Data Integrator Administrator Guide*.

- To open the trace log on job execution, select **Tools** > **Options** > **Designer** > **General** > **Open monitor on job execution**.
- To copy log content from an open log, select one or multiple lines and use the key commands [Ctrl+C].

➢ To access a log during job execution

If your Designer is running when job execution begins, the execution window opens automatically, displaying the trace log information.



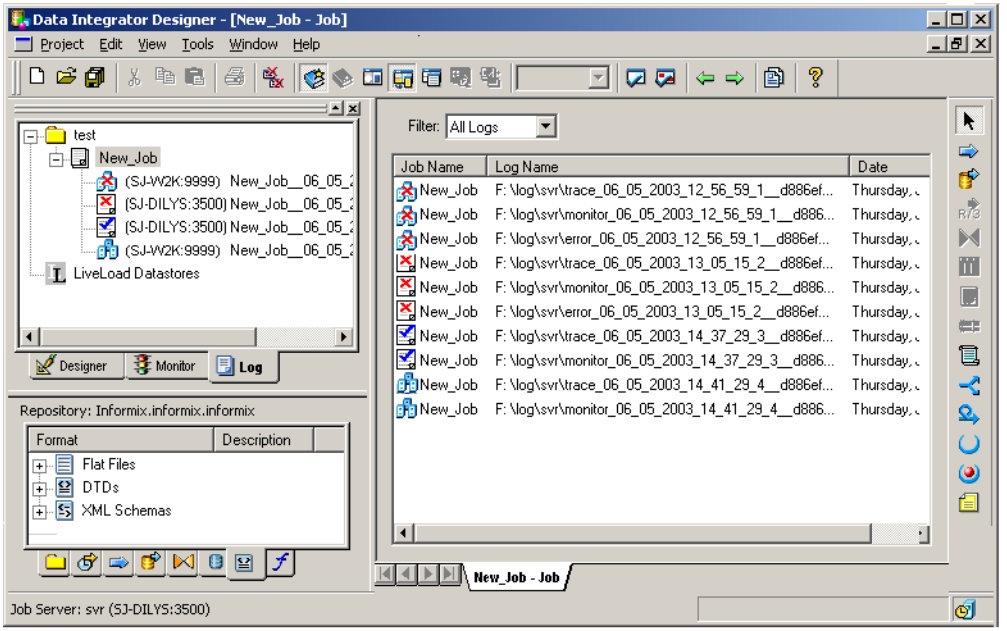
Use the statistics and error log icons (middle and right icons at the top of the execution window) to view these logs.

The execution window stays open until you explicitly close it.

➢ To access a log after the execution window has been closed

1. In the project area, click the **Log** tab.

2. Click a job name to view all trace, statistics (monitor), and error log files in the workspace. Alternatively, expand the job you are interested in to view the list of trace log files and click one.



Log indicators signify the following:

Job Log Indicator	Description
	Indicates that the job executed successfully on this explicitly selected Job Server.
	Indicates that the was job executed successfully by a server group. The Job Server listed executed the job.
	Indicates that the job encountered an error on this explicitly selected Job Server.
	Indicates that the job encountered an error while being executed by a server group. The Job Server listed executed the job.

3 Click the log icon for the execution of the job you are interested in. (Identify the execution from the position in sequence or datetime stamp.)

4 Use the list box to switch between log types or to view **No logs** or **All logs**.

➢ To delete a log

You can set how long to keep logs in Data Integrator Administrator. For more information, see "Setting the log retention period" on page 25 of the *Data Integrator Administrator Guide*.

If want to delete logs from the Designer manually:

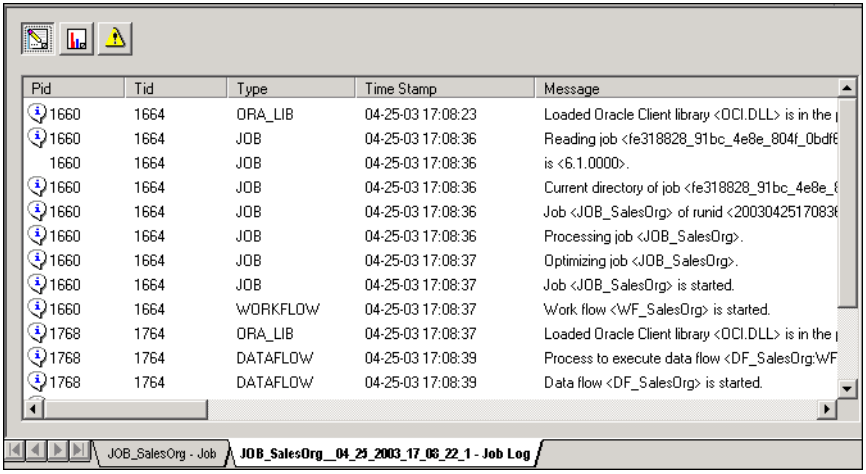
1. In the project area, click the **Log** tab.
2. Right-click the log you want to delete and select **Delete Log**.

Examining trace logs

Use the trace logs to determine where an execution failed, whether the execution steps occur in the order you expect, and which parts of the execution are the most time consuming.

For information about examining trace logs from the Administrator, see the *Data Integrator Administrator Guide*.

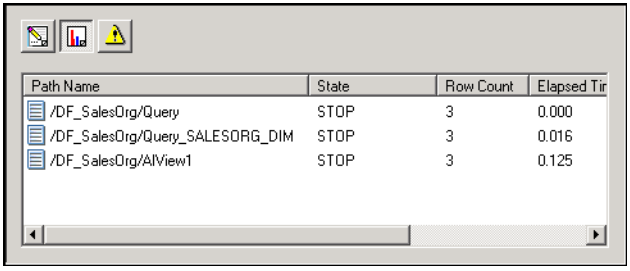
The following figure shows an example of a trace log.



Examining statistics logs

The statistics log (also known as the *monitor* log) quantifies the activities of the components of the job. It lists the time spent in a given component of a job and the number of data rows that streamed through the component.

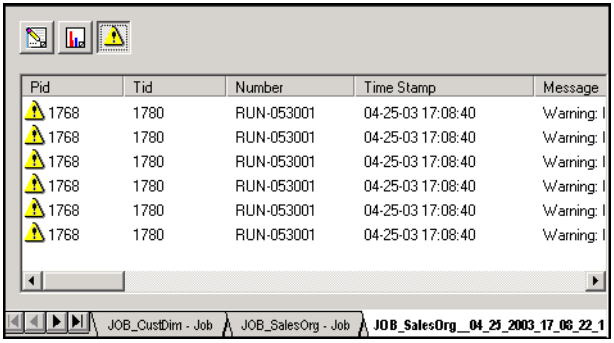
The following screen shows an example of a statistics log.



Examining error logs

Data Integrator produces an error log for every job execution. Use the error logs to determine how an execution failed. If the execution completed without error, the error log is blank.

The following screen shows an example of an error log.



Examining target data

The best measure of the success of a job is the state of the target data. Always examine your data to make sure the data movement operation produced the results you expect. Be sure that:

- Data was not converted to incompatible types or truncated.
- Data was not duplicated in the target.
- Data was not lost between updates of the target.
- Generated keys have been properly incremented.
- Updated values were handled properly.

Changing Job Server options

There are many options available in Data Integrator for troubleshooting and tuning a job. After you familiarize yourself with the more technical aspects of how Data Integrator handles data (using the *Data Integrator Reference Guide* and the *Data Integrator Performance Optimization Guide*) and some of its interfaces like those for adapters and SAP R/3, you might want to return to the Designer and change values for the following Job Server options:

Job Server Options

Option	Option Description	Default Value
Adapter Data Exchange Timeout	(For adapters) Defines the time a function call or outbound message will wait for the response from the adapter operation.	10800000 (3 hours)
Adapter Start Timeout	(For adapters) Defines the time that the Administrator or Designer will wait for a response from the Job Server that manages adapters (start/stop/status).	90000 (90 seconds)
AL_JobServerLoad BalanceDebug	Enables a Job Server to log server group information if the value is set to TRUE. Information is saved in: *$LINK_DIR*/log/<*JobServerName*>/server_eventlog.txt	FALSE
AL_JobServerLoad OSPolling	Sets the polling interval (in seconds) that Data Integrator uses to get status information used to calculate the load balancing index. This index is used by server groups.	60
FTP Number of Retry	Sets the number of retries for an FTP connection that initially fails.	0
FTP Retry Interval	Sets the FTP connection retry interval in milliseconds.	1000
Global_DOP	Sets the Degree of Parallelism for all data flows run by a given Job Server. You can also set the **Degree of parallelism** for individual data flows from each data flow's Properties window. If a data flow's **Degree of parallelism** value is 0, then the Job Server will use the Global_DOP value. The Job Server will use the data flow's **Degree of parallelism** value if it is set to any value except zero because it overrides the Global_DOP value. For more information, see "Degree of parallelism" on page 67 of the *Data Integrator Performance Optimization Guide*.	1
Ignore Reduced Msg Type	(For SAP R/3) Disables IDoc reduced message type processing for all message types if the value is set to TRUE.	FALSE
Ignore Reduced Msg Type_foo	(For SAP R/3) Disables IDoc reduced message type processing for a specific message type (such as *foo*) if the value is set to TRUE.	FALSE
OCI Server Attach Retry	The engine calls the Oracle OCIServerAttach function each time it makes a connection to Oracle. If the engine calls this function too fast (processing parallel data flows for example), the function may fail. To correct this, increase the retry value to 5.	3
Splitter Optimization	If you create a job in which a file source feeds into two queries Data Integrator might hang. If this option is set to TRUE, the engine internally creates two source files that feed the two queries instead of a splitter that feeds the two queries.	FALSE

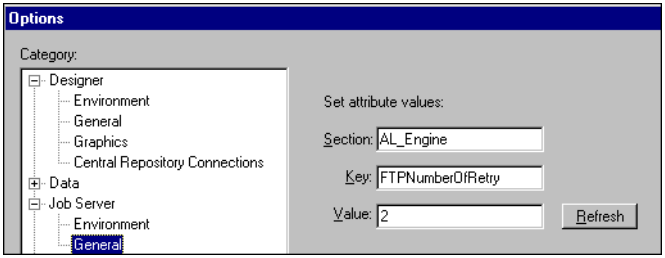
➢ To change option values for a given Job Server

1. Select the Job Server you want to work with by making it your default Job Server.
   a. Select **Tools** > **Options** > **Designer** > **Environment**.
   b. Select a Job Server from the **Default Job Server** section.
   c. Click **OK**.
2. Select **Tools** > **Options** > **Job Server** > **General**.
3. Enter the section and key you want to use from the following list of value pairs:

Section	Key
int	AdapterDataExchangeTimeout
int	AdapterStartTimeout
AL_Engine	FTPNumberOfRetry
AL_Engine	FTPRetryInterval
AL_Engine	Global_DOP
AL_Engine	IgnoreReducedMsgType
AL_Engine	IgnoreReducedMsgType_foo
AL_Engine	OCIServerAttach_Retry
AL_Engine	SPLITTER_OPTIMIZATION
AL_JobServer	AL_JobServerLoadBalanceDebug
AL_JobServer	AL_JobServerLoadOSPolling

4. Enter a value.

For example, enter the following to change the default value for the number of times a Job Server will retry to make an FTP connection if it initially fails:



These settings will change the default value for the **FTPNumberOfRetry** option from zero to two.

5. To save the settings and close the Options window, click **OK**.
6. Re-select a default Job Server by repeating step 1, as needed.

14 Migration and Repositories

This chapter contains the following sections:

- Development process and migration
- Exporting/importing objects in Data Integrator
- Removing obsolete repository contents
- Backing up repositories
- Maintaining Job Server performance

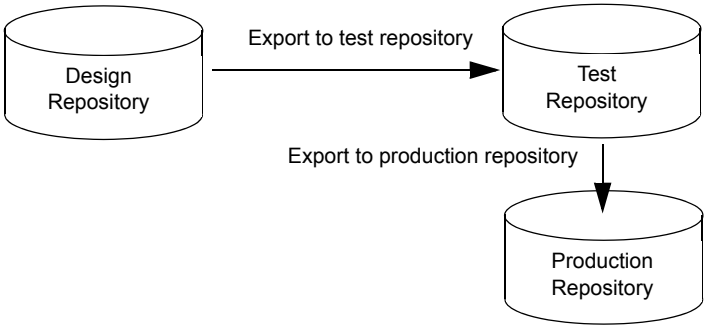
"Metadata Reporting" on page 415 lists the repository tables and describes how to retrieve metadata from them.

Development process and migration

The development process you use to create your ETL application involves three distinct phases: design, test, and production.

Each phase might require a different computer in a different environment and different security settings for each. For example, design and initial test might only require limited sample data and low security, while final testing might require a full emulation of the production environment including strict security.

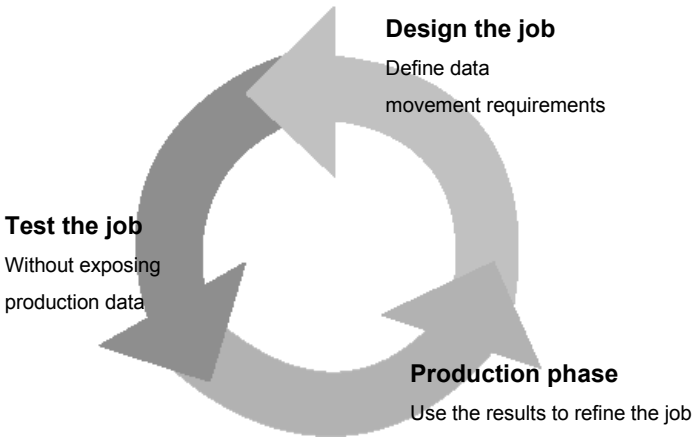
To control the environmental differences, each phase might require a different repository. Objects created in the design environment must be moved to the testing environment, and finally to the production environment, in a controlled way.



For example, when moving from design phase to testing phase, export jobs from your design repository to either a file (.atl or .xml) or a database, then import them into your test repository. For more information on exporting, see "Exporting/importing objects in Data Integrator" on page 340. Application errors discovered through testing must be corrected and retested.

Use Data Integrator in all three phases:

- In the *design phase*, you define objects and build diagrams that instruct Data Integrator in your data movement requirements. Data Integrator stores these specifications so you can reuse them or modify them as your system evolves.
- In the *testing phase*, you use Data Integrator to test the execution of your application. At this point, you can test for errors and trace the flow of execution without exposing production data to any risk. If errors appear in this phase, return the application to the design phase for correction and test again.
- In the *production phase*, you set up a schedule in Data Integrator to run your application as a job. You can evaluate results from production runs and when necessary, return to the design phase to optimize performance and refine your target requirements.



Design phase

Design your project with testing in mind. Consider these basic guidelines as you design your project:

- Construct the steps of the design as independent, testable modules.
- Use meaningful names for each step you construct.
- Make independent modules that you can use repeatedly to handle common operations.
- Use test data that reflects all the variations in your production data.

Testing phase

Testing happens in two sub-phases:

- The first sub-phase occurs when you are designing data movement using your local repository.
- The second sub-phase occurs when you fully emulate your production environment.

Data Integrator provides feedback during these phases through trace, error, and statistics logs.

The testing repository should emulate your production environment as closely as possible. For instance, include scheduled jobs rather than manually starting them.

Production phase

After moving a Data Integrator application into production, monitor it for performance and results. During production you:

- Monitor your jobs and the time it takes for each job to complete. The trace and monitoring logs provide information about jobs as well as the work flows and data flows contained within each job. You can customize log detail; however, the more information you request in the logs, the longer your job runs. Balance job runtime against information you require for job performance analysis. For more information see the *Data Integrator Performance Optimization Guide*.
- Check the accuracy of the data.

If you need enhancements or require corrections:

- Make corrections in your design environment.
- Repeat object testing.
- Move objects back into production.

Exporting/importing objects in Data Integrator

The export feature gives you the flexibility to manage and migrate projects involving multiple developers and different execution environments. When you export a job from a development repository to a production repository, you can change the properties of objects being exported to match your production environment.

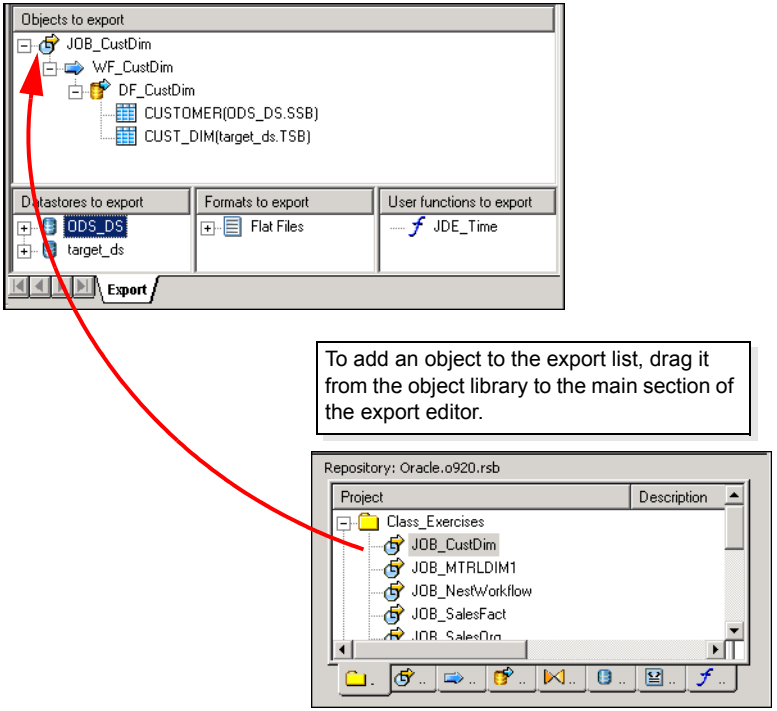
In particular, you can change datastore definitions—application and database locations and login information—to reflect production sources and targets.

You can export objects to another repository or a flat file (.atl or .xml). If the destination is another repository, you must be able to connect to and have write permission for that repository, and your repository versions must match.

This section discusses:

The Export editor

In the Export editor, you specify which objects you want to export and to where you want to export them. Choose **Tools** > **Export** or select an object and right-click **Export** to open the export editor. To specify an object to export, drag the object from the object library into the **Objects to Export** window.



The Object to Export window shows the final list of objects to be exported. When you drag any object from the object library, the datastores, file formats, and custom functions included in the object definition are automatically added to the other export sections. Each object in an export window opens to show objects called by this object.

You can control which associated objects to exclude or include. For example, you can export a work flow and all tables contained in the work flow without exporting an associated data flow.

To control which objects to export, either select an object, right-click, and choose a shortcut menu option, or select the white space in the Export editor, right-click, and choose a shortcut menu option:

- **Exclude**

  Removes only the selected object from the list of objects to be exported. The object remains in the list, but its exclusion is indicated by a red "x" on the object icon.

  All occurrences of the object are excluded.

  When you export the list, excluded objects are not copied to the destination. Objects called by this object are not removed from the list of objects to be exported, unless they are specifically excluded.

- **Include**

  Adds an excluded object to the export plan. The red "**X**" on the icon disappears. All occurrences of the object are included.

  When you export, the included objects are copied to the destination.

- **Exclude Tree**

  Removes the selected object and all objects called by this object from the export. The objects remain in the list, but their exclusion is indicated by a red "x" on the icons—the selected object and any objects it calls are excluded.

  When you export the list, the excluded objects are not copied to the destination.

- **Include Tree**

  Add the selected excluded object and the objects it calls to the export list. The red x on the selected object and dependents disappears. When you export the list, the included objects are copied to the destination.

- **Exclude environmental information**

  Removes all connections (datastores and formats) and their dependent content (tables, files, functions) from the objects in the Export editor.

  From the white space in the Export editor, you can right-click and select this option. Using this option you can export jobs without connections so that you avoid connection errors. Business Objects recommends you configure datastores and formats for the new environment separately.

  When you export, excluded objects are not copied to the destination.

- **Include environmental information**

  Adds all connections (datastores and formats) and their dependent content (tables, files, functions) to the objects you want to export.

- **Clear All**

  Removes all objects from all sections of the editor.

- **Delete**

  Removes the selected object and objects it calls from the Export editor. Only the selected occurrence is deleted; if any of the effected objects appear in another place in the export plan, the objects are still exported.

  This option is available only at the top level. You cannot delete other objects; you can only exclude them.

- **Export**

  Starts the export process.

Exporting objects to a database

You can export objects from the current repository to another repository. However, the other repository must be the same version as the current one. The export process allows you to change environment-specific information defined in datastores and file formats to match the new environment.

➢ To export a repository object to a database

1. In the object library, choose an object to export.

   Right-click and choose **Export**.

   The Export editor opens in the workspace. To add more objects to the list of objects to export, drag the objects from the object library into the Objects to Export section of the editor.

2. Refine the list of objects to export.

   You can use the options available in the right-click menu for each object to include or exclude the object from the export list.

3. When your list is complete, right-click and choose **Export**.

4. In the Export Destination window, add the destination database connection information.

5. In Export Confirmation window, verify the components to export.

   The Destination status column shows the status of the component in the target database and the proposed action.

Destination Status	Action
Does not exist	Create/Exclude
Exists	Replace/Exclude

   To edit an action, select any number of objects (using SHIFT and CTRL keys) and select either **Create**, **Exclude**, or **Replace** from the **Target Status** list box.

6. Click **Next**.

7. In the Datastore Export Options window, select the datastore.

   You can change the owner of a table or the connection properties of the datastore.

   Click **Advanced**.

8. Change the database connection information as required by the target database.

   Click **Next**.

9. In the File Format Mapping dialog, select a file and change the Destination Root Path if necessary.

   You can change the **Destination Root Path** for any file formats to match the new destination.

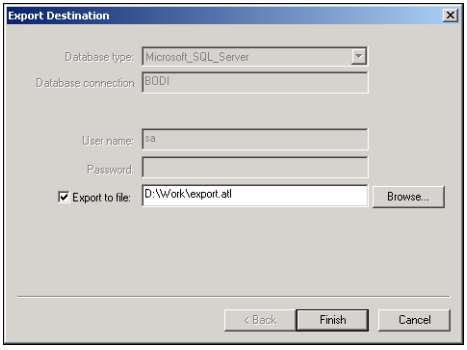
10. Click **Finish**.

Data Integrator copies objects in the Export editor to the target destination. When copying is complete, the objects display in the Output window. The Output window shows the number of objects exported as well as a list of any errors.

Exporting objects to a file

You can also export objects to a file. If you choose a file as the export destination, Data Integrator does not provide options to change environment-specific information.

NOTE: Objects in a repository are exported in the .atl format, while whole repositories can be exported in either the .atl or .xml format. ATL is Data Integrator's scripting language format. Using the .xml file format might make repository content easier for you to read. It also allows you to export Data Integrator to other products.



Exporting a repository to a file

You can also export an entire repository to a file. When you export or import a repository, jobs and their schedules (created in Data Integrator) are automatically exported or imported as well. Schedules cannot be exported or imported without an associated job and its repository.

If you choose a file as the export destination, Data Integrator does not provide options to change environment-specific information.

➢ To export a repository to a file

1. From the object library, right-click and choose **Repository > Export To File**.

   A window opens to prompt you for the destination of the export file. You can browse the directory to change the location, set the file type (.xml or .atl), and enter a name for the file.

2. Click **Open**.

   The repository is exported to the file.

Importing from a file

Importing objects or an entire repository from a file overwrites existing objects with the same names in the destination repository. You must restart Data Integrator after the import process completes.

➢ To import a repository from a file

1. There are two ways to import Data Integrator repository files into another repository. Use **Tools > Import from file**, or in the object library, right-click and choose **Repository > Import from File**.

   A window opens for you to specify the file to import. You can import individual files or the whole repository using either an XML or ATL file type. (ATL is Data Integrator's internal scripting language.)

2. Select the file to import and click **Open**.

Data Integrator displays this warning:

```
Warning: You are about to import using the
path filename file. This will create new
versions of existing objects and requires
shutting down the application. This may take
a long time. Are you sure?
```

3. Click **OK**.

   Data Integrator imports the repository file and shuts down.

4. Choose **Programs > BusinessObjects Data Integrator 6.1 > Data Integrator Designer** from the **Start** menu.
5. Log in to the repository where the file was imported.
6. Verify that the file or repository was imported.

Removing obsolete repository contents

Data Integrator saves a version of each object every time you save the object. Repeatedly modified object definitions can use a substantial amount of space. If you notice your repository performance degrading, consider compacting your repository.

You can access the **Compact Repository** command by selecting **Project > Compact Repository** from the menu bar. This command removes the previous object versions maintained by Data Integrator.

You can also compact your repository manually. If you have never compacted the repository, the majority of space in the repository could be occupied by old versions of Data Integrator objects. In this case, the **Compact Repository** command might be too slow and tedious. Instead, you can export the latest versions of the respository object definitions to a file, clear the repository database by creating a new repository, then reimport the object definitions.

➢ To compact your repository by creating a new repository

1. Export the repository to a file.

   The file type can be either XML or ATL. The latest version of each object is exported.

2. Choose **Repository Manager** from the **Start > Programs > Data Integrator** menu.

3. From the Repository Manager, add the database connection information for the repository.

4. Click **Create**.

   Data Integrator warns that a valid repository already exists.

5. Click **Yes** to overwrite the old repository.

   The Repository Manager creates a new repository, removing all of the old objects.

6. Import the previously exported repository.

Backing up repositories

Use your DBMS utilities to back up your repositories regularly. For information, refer to your DBMS documentation.

Maintaining Job Server performance

If you are designing jobs, typically you might use the same computer for your Designer, repository, and Job Server. In addition, you might use the same datastore for both your repository and your target database.

However, when you move your jobs into a test environment, the Job Server might move to a separate computer (typically from a Windows to a UNIX platform). The Data Integrator Job Server computer uses source, target, and repository database client libraries to extract, transform, and load data according to a job's design. Therefore, the Job Server computer must have a database client installed for each database you are using to run a Data Integrator job.

In addition, you must set a locale for each Job Server, its datastores and file formats, as well as its connected databases and their clients. For more information about how to set locale values for a job and how to minimize the effect code page settings have on job performance, see "Locales and Multi-Byte Functionality" on page 549 of the *Data Integrator Reference Guide*.

15 Design and Debug

This chapter covers three Designer features that you can use to design and debug jobs:

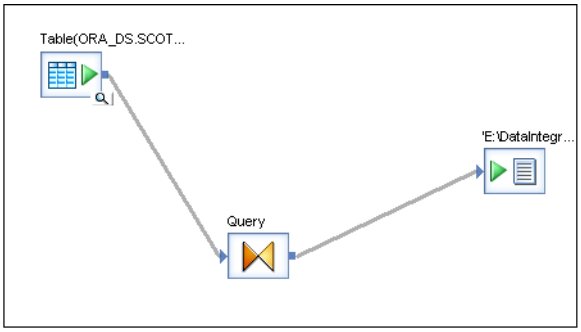
- Use the View Where Used feature to determine the impact of editing a metadata object (for example, at table). See which data flows use the same object.
- Use the View Data feature to view sample source, transform, and target data in a data flow after a job executes.
- Use the Interactive Debugger to set breakpoints and filters between transforms within a data flow and view job data row-by-row during a job execution.

This chapter contains the following sections:

Using View Where Used

When you save a job, work flow, or data flow Data Integrator also saves the list of objects used in them in your repository. Parent/child relationship data is preserved. For example, when the following parent data flow is saved, Data Integrator also saves pointers between it and its three children:

- a table source
- a query transform
- a file target



You can use this parent/child relationship data to determine what impact a table change, for example, will have on other data flows that are using the same table. The data can be accessed using the **View Where Used** option.

For example, while maintaining a data flow, you may need to delete a source table definition and re-import the table (or edit the table schema). Before doing this, find all the data flows that are also using the table and update them as needed.

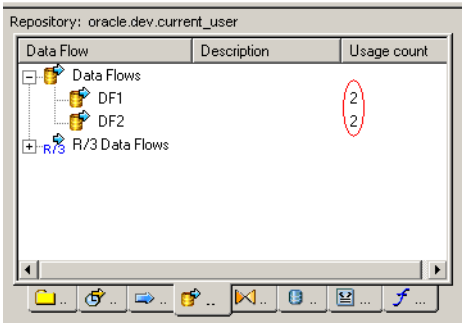
To access the **View Where Used** option in the Designer you can work from the object library or the workspace.

From the object library

You can view how many times an object is used and then view where it is used.

➢ To access parent/child relationship information from the object library

1. View an object in the object library to see the number of times that it has been used.

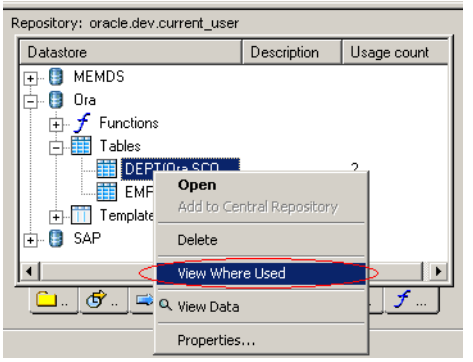


The **Usage Count** column is displayed on all object library tabs except:

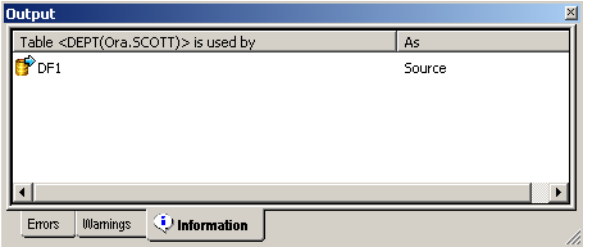
- Projects
- Jobs
- Transforms

Click the **Usage Count** column heading to sort values. For example, to find objects that are not used.

2. If the **Usage count** is greater than zero, right-click the object and select **View Where Used.**



The Output window opens. The **Information** tab displays rows for each parent of the object you selected. The type and name of the selected object is displayed in the first column's heading. Table `DEPT` is used by data flow `DF1`, in the following example:



The **As** column provides additional context. The **As** column tells you how the selected object is used by the parent. For example, in data flow `DF1`, table `DEPT` is used as a `Source`.

Other possible values for the **As** column are:

- For XML files and messages, tables, flat files, etc.:
  - Source
  - Target

- For flat files and tables only:

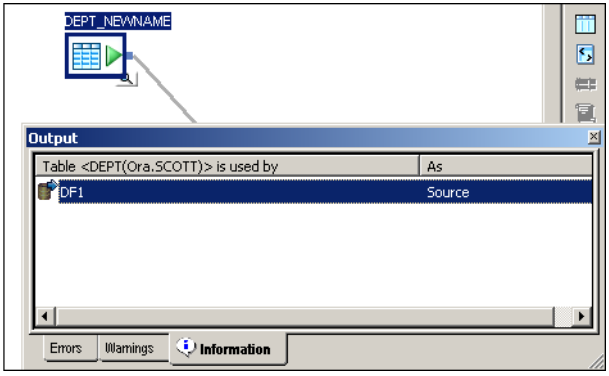
As	Description
Lookup()	Translate table/file used in a `lookup` function
Lookup_ext()	Translate table/file used in a `lookup_ext` function
Lookup_seq()	Translate table/file used in a `lookup_seq` function

- For tables only:

As	Description
Comparison	Table used in the Table Comparison transform
Key Generation	Table used in the Key Generation transform

3. From the Output window, double-click a parent object.

   The workspace diagram opens highlighting the child object the parent is using.



   Once a parent is open in the workspace, you can double-click a row in the output window again.

   - If the row represents a different parent, the workspace diagram for that object opens.
   - If the row represents a child object in the same parent, this object is simply highlighted in the open diagram.

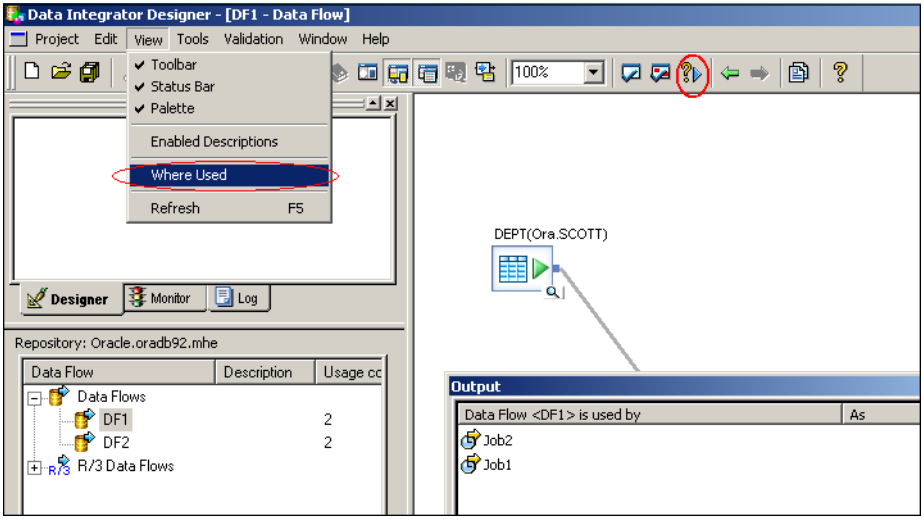
This is an important option because a child object in the Output window might not match the name used in its parent. You can customize workspace object names for sources and targets.

Data Integrator saves both the name used in each parent and the name used in the object library. The **Information** tab on the Output window displays the name used in the object library. The names of objects used in parents can only be seen by opening the parent in the workspace.

From the workspace

From an open diagram of an object in the workspace (such as a data flow), you can view where a parent or child object is used:

- To view information for the open (parent) object, select **View** > **Where Used**, or from the tool bar, select the **View Where Used** button.



In this example, the Output window opens with a list of jobs (parent objects) that use the open data flow.

- To view information for a child object, right-click an object in the workspace diagram and select the **View Where Used** option.

  The Output window opens with a list of parent objects that use the selected object. For example, if you select a table, the Output window displays a list of data flows that use the table.

Limitations

- This feature is not supported in central repositories.
- Only parent and child pairs are shown in the **Information** tab of the Output window.

  For example, for a table, a data flow is the parent. If the table is also used by a grandparent (a work flow for example), these are not listed in the Output window display for a table. To see the relationship between a data flow and a work flow, open the work flow in the workspace, then right-click a data flow and select the **View Where Used** option. You can also use the Metadata Reports tool to run a **Where Used** dependency report for any object. This report lists all related objects not just parent/child pairs. For more information see, "Where Used" on page 451.
- Data Integrator does not save parent/child relationships between functions.
  - If function `A` calls function `B`, and function `A` is not in any data flows or scripts, the **Usage count** in the object library will be zero for both functions. The fact that function `B` is used once in function `A` is not counted.
  - If function `A` is saved in one data flow, the usage count in the object library will be `1` for both functions `A` and `B`.
- Transforms are not supported. This includes custom ABAP transforms that you might create to support an SAP R/3 environment.

- The Designer counts an object's usage as the number of times it is used for a unique purpose. For example, in data flow `DF1` if table `DEPT` is used as a source twice and a target once the object library displays its **Usage count** as `2`. This occurrence should be rare. For example, a table is not often joined to itself in a job design.

Using View Data

View Data provides a way to scan and capture a sample of the data produced by each step in a job—even when the job does not execute successfully. View imported source data, changed data from transformations, and ending data at your targets. At any point after you import a data source, you can check on the status of that data—before and after processing your data flows.

Use View Data to check the data while designing and testing jobs to ensure that your design returns the results you expect. Using one or more View Data panes, you can view and compare sample data from different steps. View Data information is displayed in embedded panels for easy navigation between your flows and the data.

NOTE: Do not run View Data in a production environment because it returns only a sample of the data.

Use View Data to look at:

- Sources and targets

  View Data allows you to see data before you execute a job. Armed with data details,you can create higher quality job designs. You can scan and analyze imported table and file data from the object library as well as see the data for those same objects within existing jobs. Of course after you execute the job, you can refer back to the source data again.

  With View Data, you can check on your target data before executing your job, then look at the changed data after the job executes. In a data flow, you can use one or more View Data panels to compare data between transforms and within source and target objects.

- Transforms

  When you execute a job using the **Capture data for viewing** option, View Data can temporarily capture the output data so you can see the data changes produced by each transform in that job—even when the job does not execute successfully. For example, if a target table has four rows before a job executes and six rows after, you might open View Data for the query transform in that job and see that in this case, the delta change consisted of two rows.

- lines (For more information, see "Using the interactive debugger" on page 382).

View Data allows you to see data in imported flat files and XML files. View Data also allows you to see data for test files of XML messages, tables (in natively supported databases), and the following application interfaces:

- Oracle Applications
- PeopleSoft
- SAP R/3
- Siebel

NOTE: View Data is not supported for SAP R/3 IDocs. For SAP R/3 and PeopleSoft, the **Table Profile** tab and **Column Profile** tab options are not supported for hierarchies.

The topics in this section include:

- Accessing View Data
- Viewing data in the workspace
- View Data properties
- View Data tabs
- Generating sample data from executed jobs

Accessing View Data

There are multiple places throughout Designer where you can open a View Data pane.

Sources and targets

You can view data for sources and targets from two different locations:

- **View Data button**

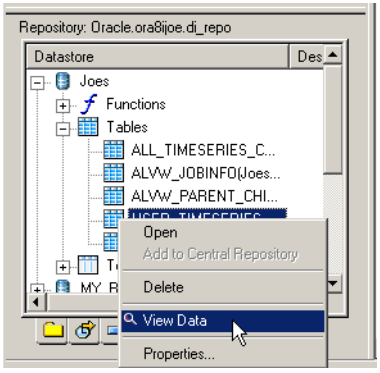
  View Data buttons appear on source and target objects when you drag them into the workspace. Click the View data button (magnifying glass icon) to open a View Data pane for that source or target object. (See "Viewing data in the workspace" on page 364 for more information.)

- **Object library**

  View Data in potential source or target objects from the Datastores or Formats tabs.

  There are two ways to open a View Data pane from the object library:

  - Right-click a table object and select **View Data**.



  - Right-click a table and select **Open** or **Properties**.

The Table Metadata, XML Format Editor, or Properties window opens. From any of these windows, you can select the **View Data** tab.

To view data for a file, the file must physically exist and be available from your computer's operating system. To view data for a table, the table must be from a supported database.

Transforms

You can access View Data for transform objects from the workspace and project tree. Furthermore those transform objects must be part of a job executed with the **Capture data for viewing** option selected during that execution session.

When you execute a job with the **Capture data for viewing** option, Data Integrator creates a temporary data file for viewing purposes only. When you end the Designer session, those files are deleted. So, to view data for those transforms during another Designer session, you must execute the job again with **Capture data for viewing**.

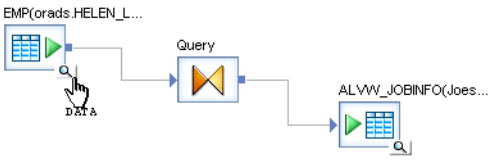
For more information, see "Generating sample data from executed jobs" on page 379.

Viewing data in the workspace

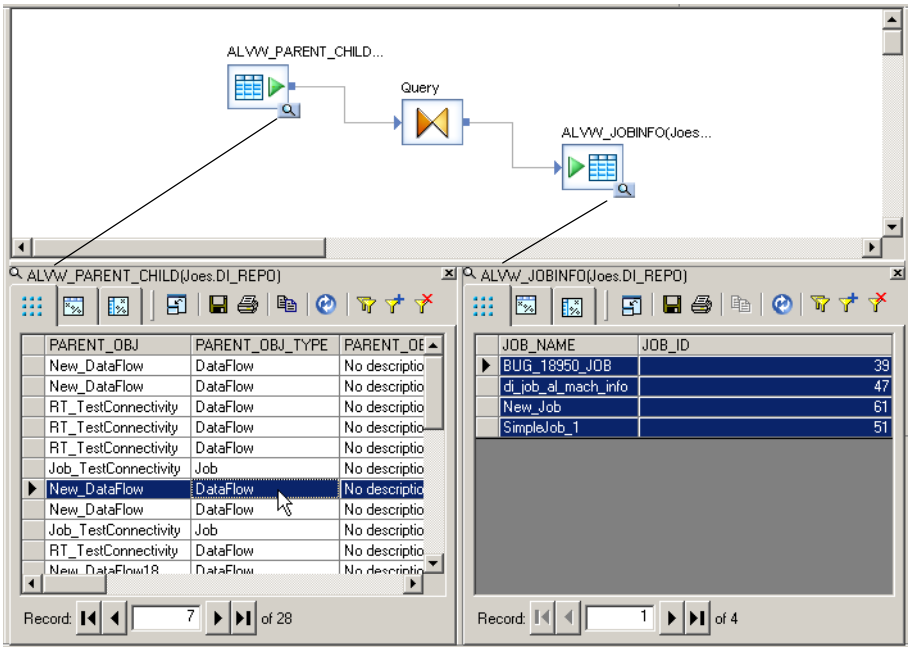
View Data can be accessed from the workspace when magnifying glass buttons appear over qualified objects in a data flow. This means:

- For sources and targets, files must physically exist and be accessible, and tables must be from a supported database.
- For transforms, the object must be part of a the last job executed with **Capture data for viewing** during the current session.

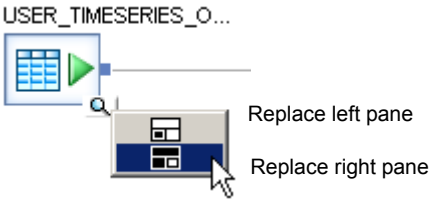
To open a View Data pane in the Designer workspace, click the magnifying glass button on a data flow object.



A large View Data pane appears beneath the current workspace area. Click the magnifying glass button for another object and a second pane appears below the workspace area (Note that the first pane area shrinks to accommodate the presence of the second pane).



You can open two View Data panes for simultaneous viewing. When both panes are filled and you click another View Data button, a small menu appears containing window placement icons. The black area in each icon indicates the pane you want to replace with a new set of data. Click a menu option and the data from the latest selected object replaces the data in the corresponding pane.



The path for the selected object displays at the top of the pane.

- For sources and targets, this consists of the full object name:
  - `ObjectName(Datastore.Owner)` for tables
  - `FileName(File Format Name)` for files
- For transforms, this consists of the full execution path name.

  For example, if a query with the object name `PRICE` is located inside an embedded data flow (`EDF1`), inside a data flow (`DF1`), inside a work flow (`WF1`), inside a job (`JOB1`), you would see the following:

  ```
  PRICE(JOB1.WF1.DF1.EDF1)
  ```

  Execution path name information can also contain a transform, if applicable (for example, if one or more embedded data flows are output from a Case Transform).

  NOTE: Embedded data flows with an output port are displayed with names of the form *`EDFname`*. Embedded data flows without an output port display names of the form *`EDFname-sourcename`*.

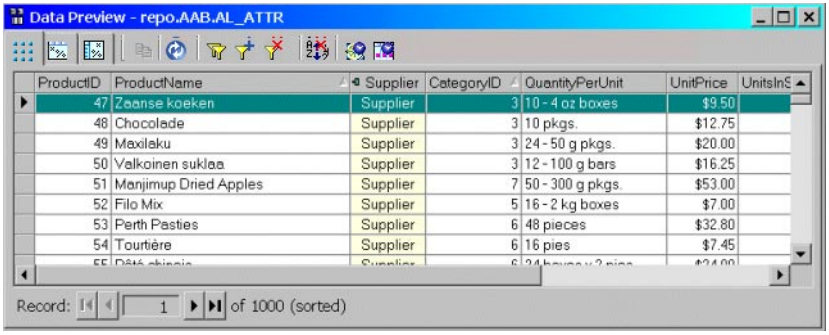
You can also find the View Data pane that is associated with an object or line by:

- Rolling your cursor over a View Data button on an object or line. The Designer highlights the View Data pane for the object.

- Looking for grey View Data buttons on objects and lines. The Designer displays View Data buttons on open objects with grey rather than white backgrounds.

View Data properties

You can access View Data properties from tool bar buttons or the right-click menu.



View Data displays your data in the rows and columns of a data grid. The number of rows displayed is determined by a combination of several conditions:

- Sample size — The number of rows sampled in memory. Default sample size is 1000 rows for imported source and target objects and 100 for transforms. Maximum sample size is 5000 rows.

  NOTE: If you use the interactive debugger, Data Integrator uses the **Data sample rate** option instead of sample size. For more information, see "Starting and stopping the interactive debugger" on page 389.

- Filtering
- Sorting

If your original data set is smaller or if you use filters, the number of returned rows could be less than the default. For some transforms, the number of returned rows can be more.

- Set sample size for sources and targets from **Tools** > **Options** > **Designer** > **General** > **View Data sampling size**.
- Set sample size for transforms from the Execution Properties window under **Capture data for viewing**. For more information, see "Generating sample data from executed jobs" on page 379.

View applied conditions in the **navigation bar**.

Filtering

You can focus on different sets of rows in a local or new data sample by placing fetch conditions on columns.

➢ To view and add filters

1. In the View Data tool bar, click the Filters button, or right-click the grid and select **Filters**.

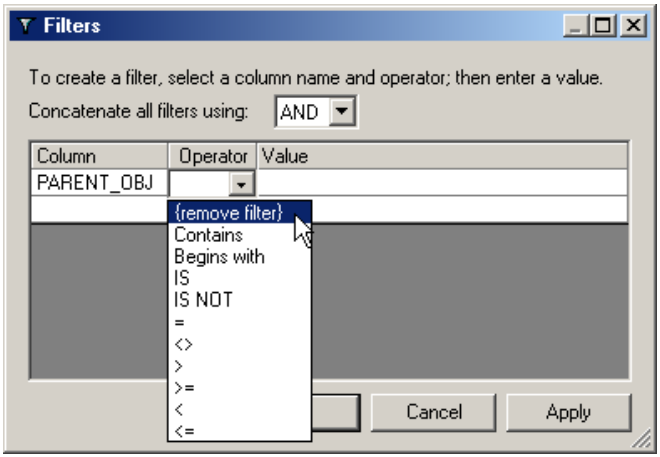
   The Filters window opens.

2. Create filters.

   The Filters window has three columns:

   a. Column—Select a name from the first column. Select **{remove filter}** to delete the filter.

b. Operator—Select an operator from the second column.

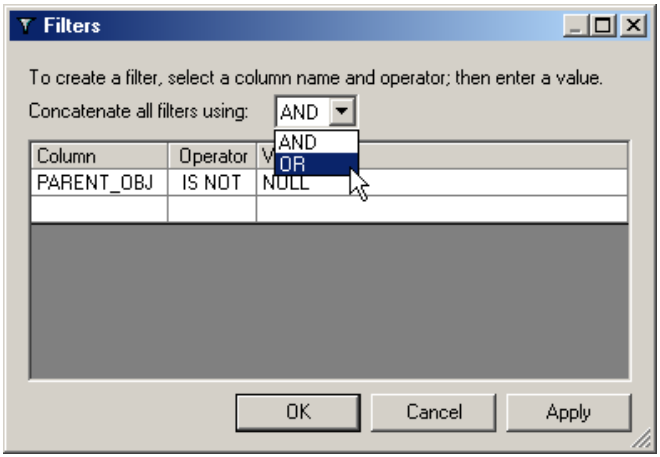


c. Value—Enter a value in the third column that uses one of the following data type formats:

Data Type	Format
Integer, double, real	standard
date	yyyy.mm.dd
time	hh24:mm:ss
datetime	yyyy.mm.dd hh24:mm.ss
varchar	'abc'

3. In the **Concatenate all filters using** list box, select an operator (**AND**, **OR**) for the engine to use in concatenating filters.

Each row in this window is considered a filter.



4. To see how the filter affects the current set of returned rows, click **Apply**.
5. To save filters and close the Filters window, click **OK**.

   Your filters are saved for the current object and the local sample updates to show the data filtered as specified in the Filters dialog. To use filters with a new sample, see "Using Refresh" on page 371.

➢ To add a filter for a selected cell

1. Select a cell from the sample data grid.
2. In the View Data tool bar, click the Add Filter button, or right-click the cell and select **Add Filter**.

   The **Add Filter** option adds the new filter condition, `<column> = <cell value>`, then opens the Filters window so you can view or edit the new filter.
3. When you are finished, click **OK**.

To remove filters from an object, go to the View Data tool bar and click the Remove Filters button, or right-click the grid and select **Remove Filters**. All filters are removed for the current object.

Sorting

You can click one or more column headings in the data grid to sort your data. An arrow appears on the heading to indicate sort order: ascending (up arrow) or descending (down arrow).

To change sort order, click the column heading again. The priority of a sort is from left to right on the grid. To use sorts with a new sample, see "Using Refresh" on page 371.

To remove sorting for an object, from the tool bar click the Remove Sort button, or right-click the grid and select **Remove Sort**.

Using Refresh

To fetch another data sample from the database using new filter and sort settings, use the **Refresh** command. After you edit filtering and sorting, in the tool bar click the Refresh button in the tool bar, or right-click the data grid and select **Refresh**.

To stop a refresh operation, click the Stop button. While Data Integrator is refreshing the data, all View Data controls except the **Stop** button are disabled.

Using Show/Hide Columns

You can limit the number of columns displayed in View Data by using the **Show/Hide Columns** option from:

- The tool bar.
- The right-click menu.

- The arrow shortcut menu, located to the right of the **Show/Hide Columns** tool bar button. This option is only available if the total number of columns in the table is ten or fewer. Select a column to display it.

You can also "quick hide" a column by right-clicking the column heading and selecting **Hide** from the menu.

➢ To show or hide columns

1. Click the Show/Hide columns tool bar button, or right-click the data grid and select **Show/Hide Columns**.

   The Column Settings window opens.

2. Select the columns that you want to display or click one of the following buttons: **Show**, **Show All**, **Hide**, or **Hide All**.

3. Click **OK**.

Opening a new window

To see more of the data sample that you are viewing in a View Data pane, open a full-sized View Data window. From any View Data pane, click the Open Window tool bar button to activate a separate, full-sized View Data window. Alternatively, you can right-click and select **Open in new window** from the menu.

View Data tool bar options

The following options are available on View Data panes.

View Data pane tool bar options

	Option	Description
	Open in new window	Opens the View Data pane in a larger window. See "Opening a new window" on page 372.
	Save As	Saves the data in the View Data pane.
	Print	Prints View Data pane data.
	Copy Cell	Copies View Data pane cell data.
	Refresh data	Fetches another data sample from existing data in the View Data pane using new filter and sort settings. See "Using Refresh" on page 371.
	Open Filters window	Opens the Filters window. See "Filtering" on page 368.
	Add a Filter	See "To add a filter for a selected cell" on page 370.
	Remove Filter	Removes all filters in the View Data pane.
	Remove Sort	Removes sort settings for the object you select. See "Sorting" on page 371.
	Show/hide navigation	Shows or hides the navigation bar which appears below the data table.
	Show/hide columns	See "Using Show/Hide Columns" on page 371

View Data tabs

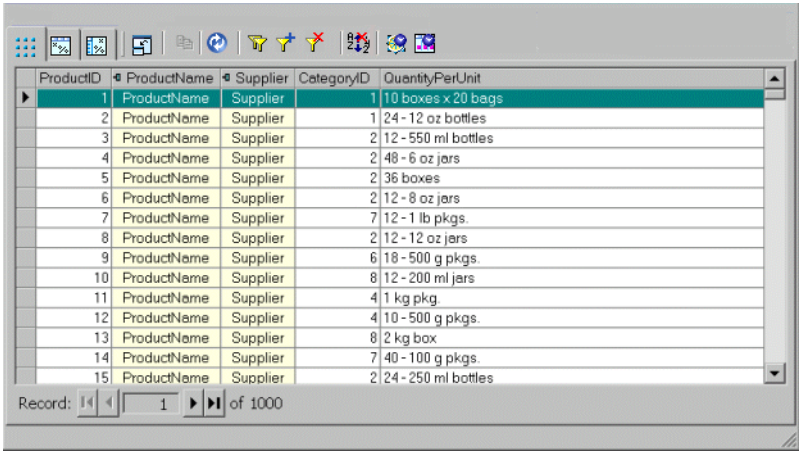
The View Data panel contains three tabs:

- Data tab
- Profile tab
- Column Profile tab

Use tab options to give you a complete profile of a source or target object. The Data tab is always available. The Profile and Column Profile tabs are supported for some sources and targets. See release notes for more information.

Data tab

The Data tab allows you to use the properties described in "View Data properties" on page 367. It also indicates nested schemas such as those used in XML files and messages. When a column references nested schemas, that column is shaded yellow and a small table icon appears in the column heading.



➢ To view a nested schema

1. Double-click a cell.

Alternatively,

a. Click a cell in a marked column.

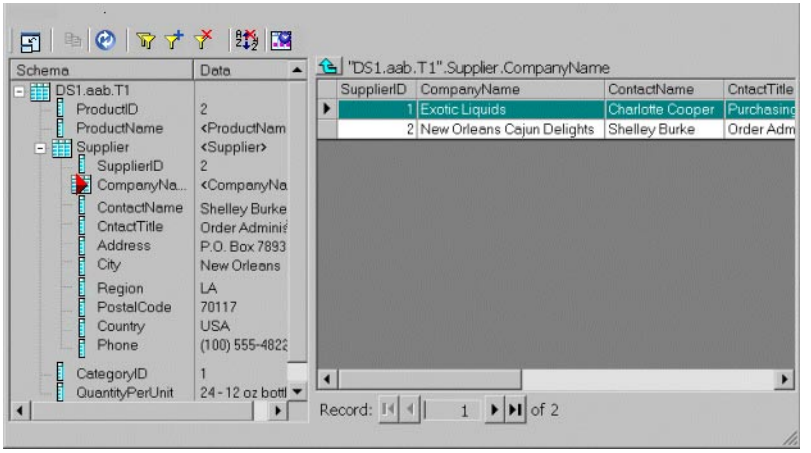
b. The **Drill Down** button (an ellipses) appears in the cell.

c. Click the **Drill Down button**.

The data grid updates to show the data in the selected cell or nested table.

In the Schema area, the selected cell value is marked by a special icon. Also, tables and columns in the selected path are displayed in blue, while nested schema references are displayed in grey.

In the Data area, data is shown for columns. Nested schema references are shown in angle brackets, for example `<CompanyName>`.



Continue to use the data grid side of the panel to navigate. For example:

- Select a lower-level nested column and double click a cell to update the data grid.

- Click the **Drill Up** button at the top of the data grid to move up in the hierarchy.

  See the entire path to the selected column or table displayed to the right of the **Drill Up** button. Use the path and the data grid to navigate through nested schemas.

Profile tab

The Profile tab allows you to calculate statistical information for any set of columns you choose. This optional feature is not available for columns with nested schemas or for the LONG data type.

➢ To use the Profile tab

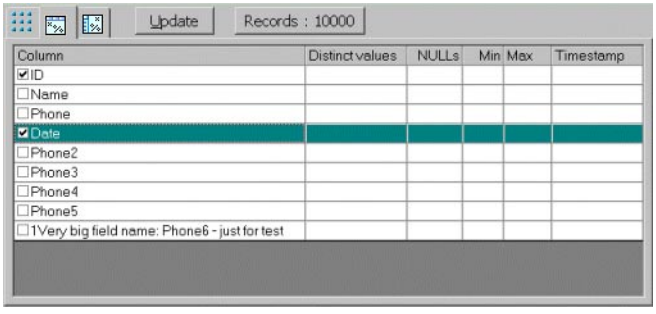
1. Select one or more columns.

   Select only the column names you need for this profiling operation because **Update** calculations impact performance.

   You can also right-click to use the **Select All** and **Deselect All** menu options.

2. Click **Update**.

3. The statistics display in the Profile grid.



The grid contains six columns:

Column	Description
Column	Names of columns in the current table. Select names from this column, then click **Update** to populate the profile grid.
Distinct Values	The total number of distinct values in this column.
NULLs	The total number of NULL values in this column.
Min	Of all values, the minimum value in this column.
Max	Of all values, the maximum value in this column.
Last Updated	The time that this statistic was calculated.

Sort values in this grid by clicking the column headings. Note that **Min** and **Max** columns are not sortable.

In addition to updating statistics, you can click the **Records** button on the Profile tab to count the total number of physical records in the object you are profiling.

Data Integrator saves previously calculated values in the repository and displays them until the next update.

Column Profile tab

The Column Profile tab allows you to calculate statistical information for a single column.

NOTE: This optional feature is not available for columns with nested schemas or the LONG data type.

➢ To calculate value usage statistics for a column

1. Enter a number in the **Top** box.

   This number is used to find the most frequently used values in the column. The default is 10, which means that Data Integrator returns the top 10 most frequently used values.

2. Select a column name in the list box.

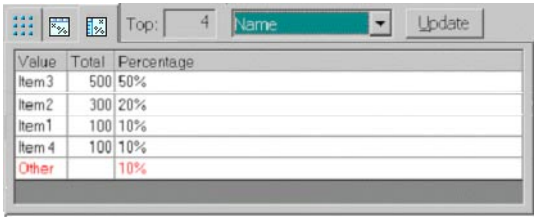
3. Click **Update**.

The Column Profile grid displays statistics for the specified column. The grid contains three columns:

Column	Description
Value	A "top" (most frequently used) value found in your specified column, or "Other" (remaining values that are not used as frequently).
Total	The total number of rows in the specified column that contain this value.
Percentage	The percentage of rows in the specified column that have this value compared to the total number of values in the column.

Data Integrator returns a number of values up to the number specified in the **Top** box, plus an additional value called "Other."

So, if you enter 5 in the **Top** box, you will get up to 6 returned values (the top 5 used values in the specified column, plus the "Other" category). Results are saved in the repository and displayed until you perform a new update.



For example, statistical results in the preceding table indicate that of the four most frequently used values in the **Name** column, 50 percent use the value, `Item3`, 20 percent use the value, `Item2`, and so on. You can also see that the four most frequently used values (the "top four") are used in 90 percent of all cases, as only 10 percent is shown in the Other category. For this example, the total number of rows counted during the calculation for each top value is 1000.

Generating sample data from executed jobs

To view the data passed by transforms, you need to first execute the job. Data Integrator captures a sample number of rows for each transform in the job.

The sampling process works as follows:

1. Execute a job with the **Capture data for viewing** option selected. Before executing, set the number of output rows that you want captured for the sample (default is 100 rows).
   - Jobs execute with optimization and real-time jobs are automatically executed in test mode.
   - Job execution stops when the specified number of rows has been collected or when the job completes (whichever occurs first).
   - If a job fails, you can still view data for the transforms from which the requisite number of rows were captured before the job failed.
   - In data flows and embedded data flows, data is captured for transforms. In R/3 data flows, data is captured for the entire data flow.
2. View and analyze output data using View Data:
   - Open a data flow that was used in the executed job (or open a job or workflow in the case of an R/3 data flow) and click the View Data button on any transform.
   - The View Data pane opens below the data flow in the same window.
3. Compare different data samples using a second View Data pane.
4. Execute another job or exit the Designer.

Data Integrator retains the data from this capture until you execute another job with the **Capture data for viewing** option selected (each data capture overwrites the previous data capture run) or until you exit the Designer. Data Integrator erases all View Data samples when you exit the Designer.

➢ To generate sample data using View Data

1. In the project area, right-click a job and select **Execute.**

   The Execution Properties window opens.

2. Select the **Capture data for viewing** check box.

   An option to set the number of output rows appears.

3. Enter a value in the **Capture rows** text box or leave the default value. The default number of return rows is 100. This value limits the number of data rows returned for each data flow object.

   When working with nested data, the limit applies only to the first-level data; there is no limit to the number of rows returned in the lower levels.

   For example, if a job contains a query transform with a restrictive WHERE clause, 100 rows may be counted but only 10 of these might be output. A Pivot transform acting on a row with 200 columns has only one row that can be counted but it will emit 200 rows.

4. Click **OK**.

   The job runs until the set number of rows has been collected or until the job completes (whichever occurs first).

   NOTE: To avoid premature termination of production jobs, do not activate **Capture data for viewing** in a production environment.

➢ To view sample data using View Data

1. In the workspace, drill into the last job you executed (with **Capture data for viewing** enabled).
2. To view captured data, click the View Data button displayed over a transform in a data flow.

   NOTE: To see sample data for a Case Transform, click the View Data button on this transform. The output of each case is displayed by name in a popup list box. From the three cases displayed, click the output you want to view. The output is loaded into a View Data pane.

   For each viewed object, the View Data pane displays:

   - The View Data icon, the object name and path (located in the upper left corner of the pane)
   - Data capture date and timestamp
   - The data grid
3. Navigate through the data to analyze it.

   While actively viewing data for an object in a View Data pane, you can use properties (see "View Data properties" on page 367) and tab options (see "Data tab" on page 374).

Using the interactive debugger

The Designer includes an interactive debugger that allows you to examine and modify data row-by-row (during a debug mode job execution) by placing filters and breakpoints on lines in a data flow diagram. The interactive debugger provides powerful options to debug a job.

Topics in this section are:

- Before starting the interactive debugger
- Starting and stopping the interactive debugger
- Windows
- Menu options and tool bar
- Push-down optimizer
- Limitations

Before starting the interactive debugger

Like executing a job, you can start the interactive debugger from the **Debug** menu when a job is active in the workspace. Select **Start Debug**, set properties for the execution, then click **OK**. The debug mode begins. The Debug mode provides the interactive debugger's windows, menus, and tool bar buttons that you can use to control the pace of the job and view data by pausing the job exeuction using filters and breakpoints.

While in debug mode, all other Designer features are set to read-only. To exit the debug mode and return other Designer features to read/write, click the **Stop Debug** button on the interactive debugger toolbar.

All interactive debugger commands are listed in the Designer's **Debug** menu. The Designer enables the appropriate commands as you progress thougth an interactive debugging session.

Before you start a debugging session, however, you might want to set the following:

- Filters and breakpoints
- Interactive debugger port between the Designer and an engine.

Setting filters and breakpoints

You can set any combination of filters and breakpoints in a data flow before you start the interactive debugger. The debugger uses the filters and pauses at the breakpoints you set.

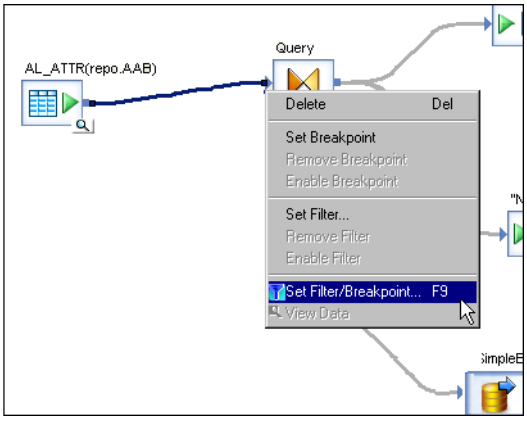
If you do not set pre-defined filters or breakpoints:

- The Designer will optimize the debug job execution. This often means that the first transform in each data flow of a job is pushed down to the source database. Consequently, you cannot view the data in a job between its source and the first transform unless you set a pre-defined breakpoint on that line. For more information, see "Push-down optimizer" on page 402.
- You can pause a job manually by using a debug option called **Pause Debug** (the job pauses before it encounters the next transform).

➢ To set a filter or breakpoint

1. In the workspace, open the job that you want to debug.
2. Open one of its data flows.
3. Right-click the line that you want to examine and select **Set Filter/Breakpoint**.

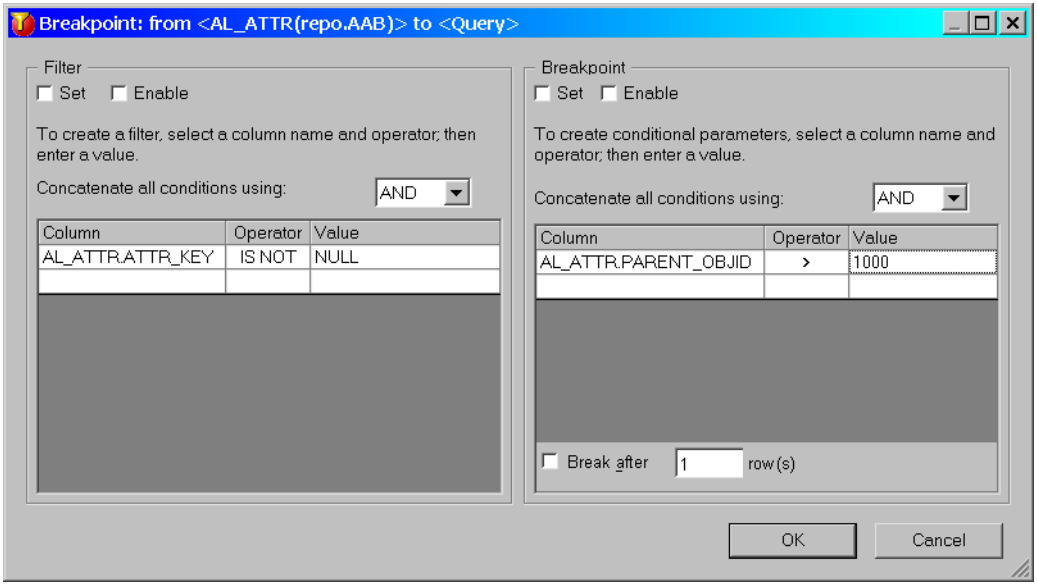
A line is a line between two objects in a workspace diagram.



Alternatively, you can double-click the line or click the line then:

- Press **F9**
- Click the **Set filter/breakpoint** button on the tool bar
- Select **Debug** > **Set Filter/Breakpoint**.

The Breakpoint window opens. Its title bar displays the objects to which the line connects. For example, the following window represents the line between `AL_ATTR` (a source table) and Query (a Query transform).



3. Set and enable a filter or a breakpoint using the options in this window.

   A debug filter functions as a simple Query transform with a `WHERE` clause. Use a filter to reduce a data set in a debug job execution. Note that complex expressions are not supported in a debug filter.

   Place a debug filter on a line between a source and a transform or two transforms. If you set a filter and a breakpoint on the same line, Data Integrator applies the filter first. The breakpoint can only see the filtered rows.

   Like a filter, you can set a breakpoint between a source and transform or two transforms. A breakpoint is the location where a debug job execution pauses and returns control to you.

Choose to use a breakpoint with or without conditions.

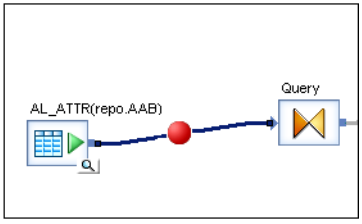
- If you use a breakpoint without a condition, the job execution pauses for the first row passed to a breakpoint.
- If you use a breakpoint with a condition, the job execution pauses for the first row passed to the breakpoint that meets the condition.

A breakpoint condition applies to the *after* image for UPDATE, NORMAL and INSERT row types and to the *before* image for a DELETE row type.

Instead of selecting a conditional or unconditional breakpoint, you can also use the **Break after 'n' row(s)** option. In this case, the execution pauses when the number of rows you specify pass through the breakpoint.

4. Click **OK**.

The appropriate icon appears on the selected line.

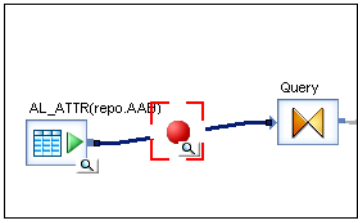


In the preceeding diagram, the red icon indicates that a breakpoint is set and enabled.

Data Integrator supports the following filter and breakpoint conditions:

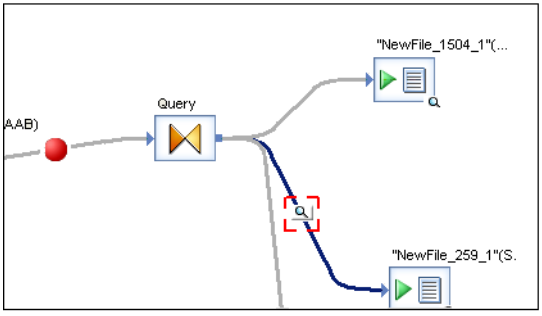
Icon	Description
	Breakpoint disabled
	Breakpoint enabled
	Filter disabled
	Filter enabled
	Filter and breakpoint disabled
	Filter and breakpoint enabled
	Filter enabled and breakpoint disabled
	Filter disabled and breakpoint enabled

In addition to the filter and breakpoint icons that can appear on a line, the debugger highlights a line when it pauses there. A red locator box also indicates your current location in the data flow. For example, when you start the interactive debugger, the job pauses at your breakpoint. The locator box appears over the breakpoint icon as shown in the following diagram:



A **View Data** button also appears over the breakpoint. You can use this button to open and close the View Data panes. For more information, see "Windows" on page 393.

As the debugger steps though your job's data flow logic, it highlights subsequent lines and displays the locator box at your current position.

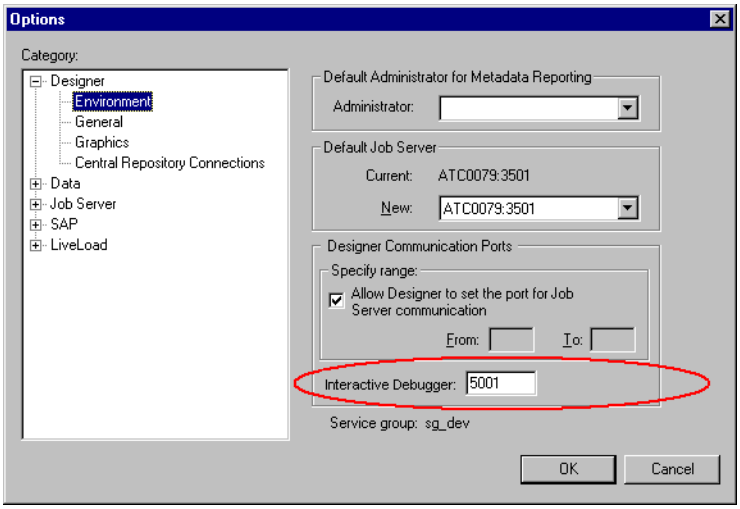


Changing the interactive debugger port

The Designer uses a port to an engine to start and stop the interactive debugger. The interactive debugger port is set to `5001` by default.

➢ To change the interactive debugger port setting

1. Select **Tools** > **Options** > **Designer** > **Environment.**
2. Enter a value in the **Interactive Debugger** box.



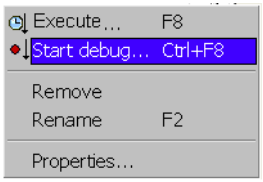
3. Click **OK**.

Starting and stopping the interactive debugger

A job must be active in the workspace before you can start the interactive debugger. You can select a job from the object library or from the project area to activate it in the workspace. Once a job is active, the Designer enables the **Start Debug** option on the **Debug** menu and tool bar.

➢ To start the interactive debugger

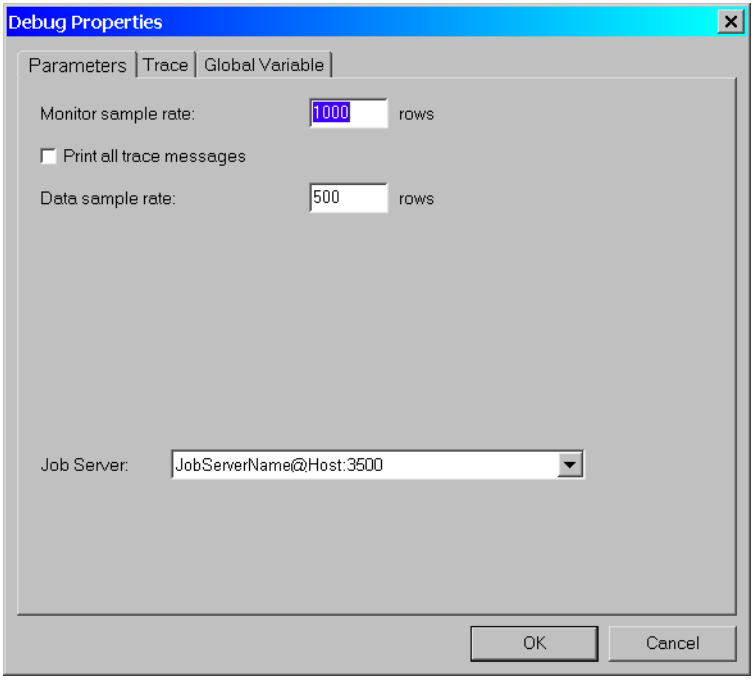
1. In the project area, right-click a job and select **Start debug**.



Alternatively, in the project area you can click a job and then:

- Press **Ctrl+F8**
- Select **Debug** > **Start Debug**.
- Click the **Start Debug** button on the tool bar.

The Debug Properties window opens.

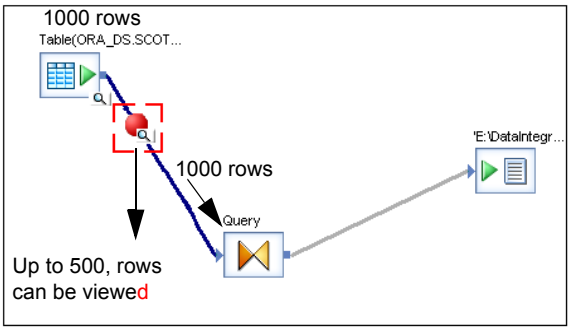


The Debug Properties window includes three parameters similar to the Execution Properties window (used when you just want to run a job). See "Parameters" on page 17 of the *Data Integrator Reference Guide* for a description of the **Monitor sample rate**, **Print all trace messages**, and **Job Server** options. You will also find more information about the **Trace** and **Global Variable** options.

The option unique to the Debug Properties window is **Data sample rate**. The **Data sample rate** is the number of rows cached for each line when a job executes using the interactive debugger.

For example, in the following data flow diagram, if the source table has 1000 rows and you set the **Data sample rate** to `500`, then the Designer displays up to 500 of the last rows that pass through a selected line. The debugger displays the last row processed when it reaches a breakpoint.

You can add to the number of rows cached and displayed by interacting with the debugger. For more information, see "View Data pane" on page 395.



2. Enter the debug properties that you want to use.
3. Click **OK**.

   The job you selected from the project area starts to run in debug mode. The Designer:

   - Displays the interactive debugger windows.

- Adds `Debugging Job <JobName>` to its title bar.
- Enables the appropriate **Debug** menu and tool bar options.

- Displays the debug icon in the status bar.
- Sets the user interface to read-only.

  NOTE: You cannot perform any operations that affect your repository (such as dropping objects into a data flow) when you execute a job in debug mode.

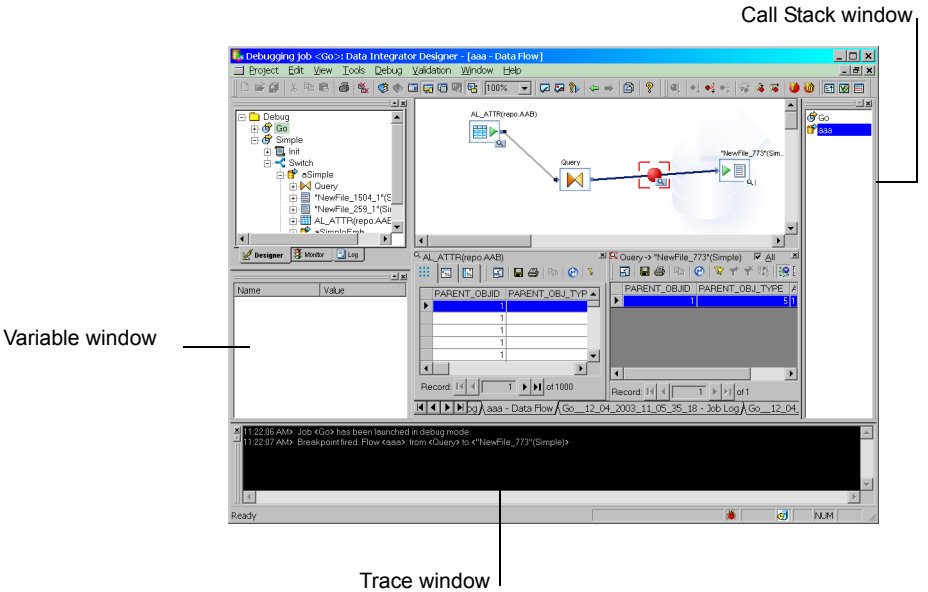
When the debugger encounters a breakpoint, it pauses the job run. You now have control of the job run. The interactive debugger windows display information about the job run up to this point. They also update as you manually step through the job or allow the debugger to continue the run.

➢ To stop a job in debug mode and exit the interactive debugger

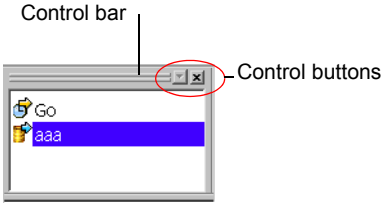
Select the **Stop Debug** button on the tool bar, press **Shift+F8**, or select **Debug** > **Stop Debug**.

Windows

When you start a job in the interactive debugger, the Designer displays three additional windows as well as the View Data panes beneath the work space. The following diagram shows the default locations for these windows.



Each window is docked in the Designer's window. To move a debugger window, double-click on the window's control bar to release it, then click and drag its title bar to re-dock it.



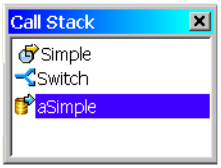
The Designer saves the layout you create when you stop the interactive debugger. Your layout is preserved for your next Designer session.

You can resize or hide a debugger window using its control buttons. To show or hide a debugger window manually, use the **Debug** menu or the tool bar. See "Menu options and tool bar" on page 400.

Call stack window

The Call Stack window lists the objects in the path encountered so far (before either the job completes, encounters a breakpoint, or you pause it).

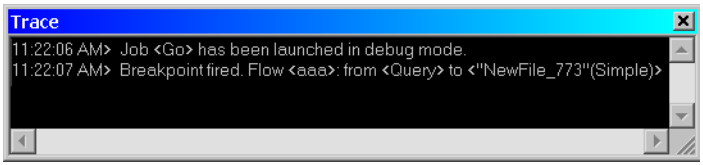
For example, the following Call Stack window indicates that the data you are currently viewing is in a data flow called `aSimple` and shows that the path taken began with a job called `Simple` and passed through a condition called `Switch` before it entered the data flow.



You can double-click an object in the Call Stack window to open it in the workspace. Similarly, if you click an object in a diagram, the Call Stack window highlights the object.

Trace window

The Trace window displays the debugger's output status and errors.



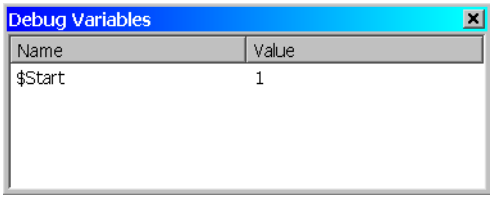
When the job completes, this window displays the following:

`Job <JobName> finished. Stop debugger.`

When the job completes the debugger gives you a final opportunity to examine data. When you must exit the debugger, select the **Stop Debug** button on the tool bar, press **shift**+**F8**, or select **Debug** > **Stop Debug**.

Debug Variables window

The Debug Variables window displays global variables in use by the job at each breakpoint.



View Data pane

The View Data pane for lines uses the same tool bar and navigation options described for the View Data feature. For more information, see "Using View Data" on page 361.

The following View Data pane options are unique to the interactive debugger:

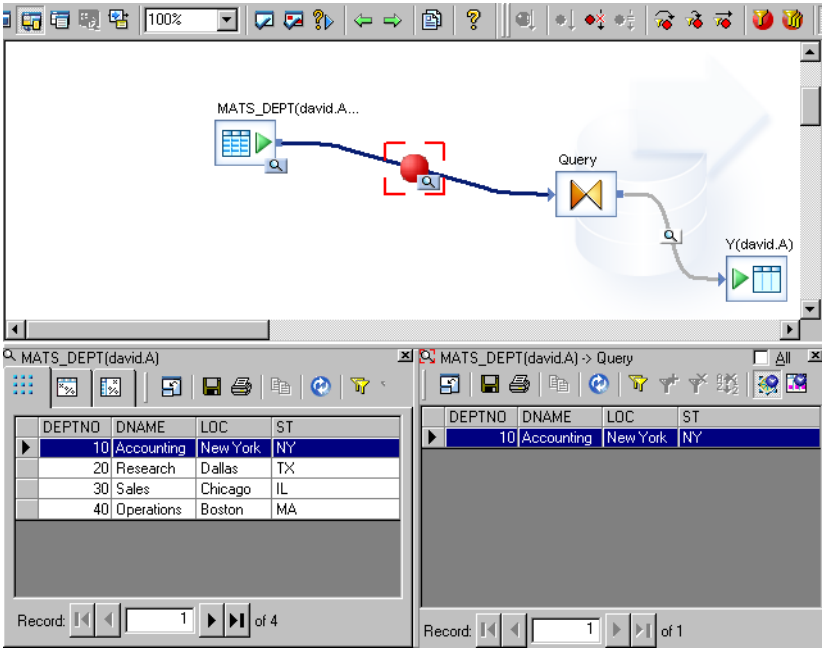
- Allows you to view data that passes through lines.
- Displays (above the View Data tool bar) the names of objects to which a line connects using the format: *TableName*(*DatastoreName*.*TableOwnerName*)-> *QueryName*.
- Displays data one row at a time by default.
- Provides the **All** check box which allows you to see more than one row of processed data.
- Allows you to edit data in a cell.

  For example, You might want to fix an error temporarily to continue with a debugger run. You can fix the job design later to eliminate the error permanently.

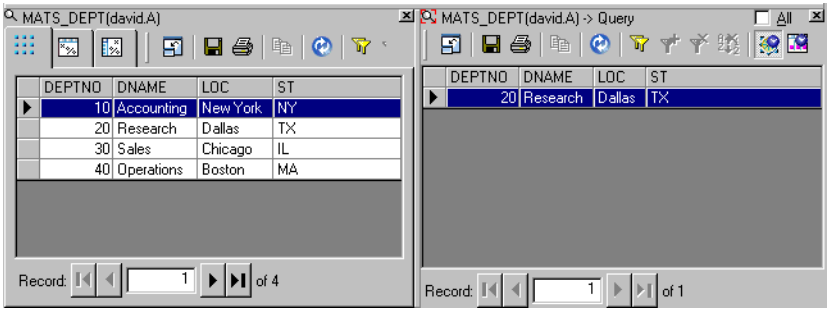
  To edit cell data:

  - Deselect the **All** check box so that only has one row displayed.
  - Double-click a cell or right-click it and select **Edit cell**.
- Uses a property called the **Data sample rate**.

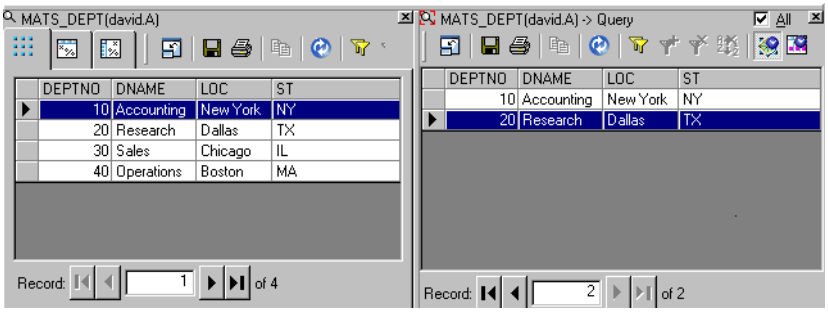
For example, if a source in a data flow has four rows and you set the **Data sample rate** to 2 when you start the debugger, it displays the first row processed at a pre-defined breakpoint.



If you use the **Get Next Row** option, then the next row at the same breakpoint is displayed:

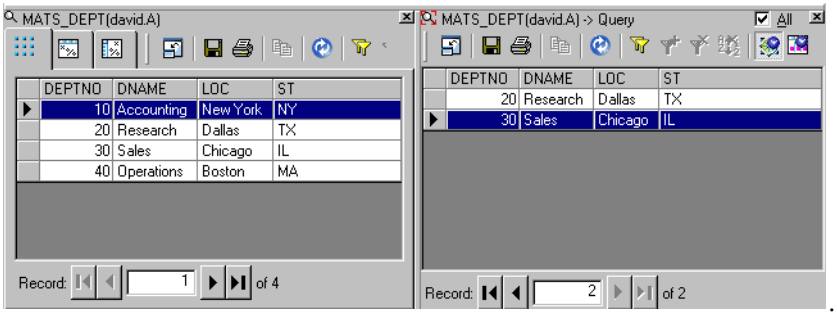


If you want to see both rows, select the **All** check box on the upper-right corner of this pane. The row displayed at the bottom of the table is the last row processed.



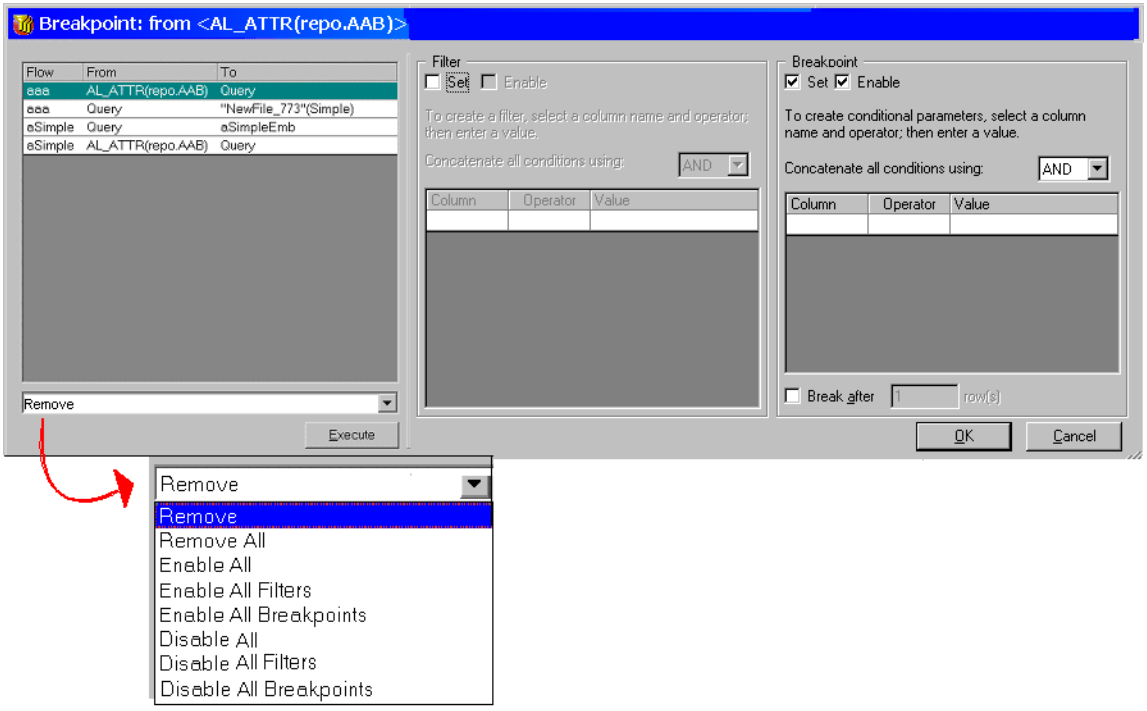
At this point, you have viewed two rows that have passed through a line.

If you click **Get Next Row** again, only the last two rows processed are displayed because you set the sample size to 2.



Filters and Breakpoints window

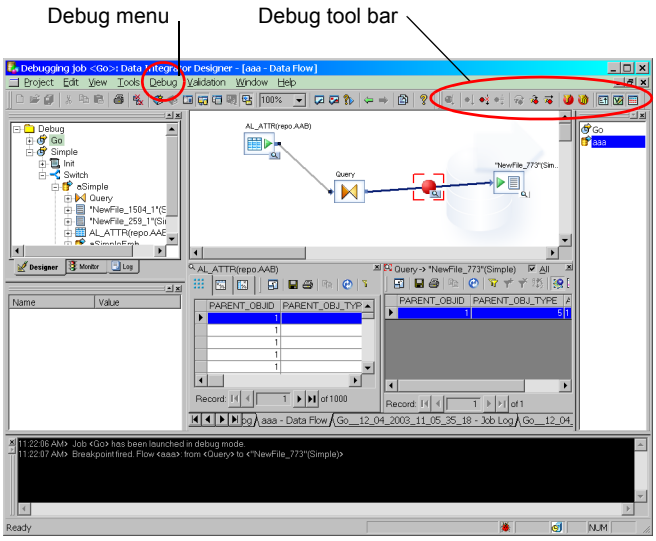
You can manage interactive debugger filters and breakpoints using the Filters/Breakpoints window. You can open this window from the **Debug** menu or tool bar.



Lines that contain filters or breakpoints are listed in the far-left side of the Filters/Breakpoints window. To manage these, select the line(s) that you want to edit, select a command from the list and click **Execute**. You can also select a single line on the left and view/edit its filters and breakpoints on the right side of this window. When you are finished using the Filters/Breakpoints window, click **OK**.

Menu options and tool bar

Once you start the interactive debugger, you can access appropriate options from the Designer's **Debug** menu and tool bar.



Debug menu and tool bar options

	Option	Description	Key Commands
	Execute	Opens the Execution Properties window from which you can select job properties then execute a job outside the debug mode. Available when a job is active in the workspace.	F8
	Start debug	Opens the Debug Properties window from which you can select job properties then execute a job in debug mode (start the debugger). Other Designer operations are set to read-only until you stop the debugger. Available when a job is active in the workspace.	Ctrl+F8
	Stop debug	Stops a debug mode execution and exits the debugger. All Designer operations are reset to read/write.	Shift+F8
	Pause debug	Allows you to manually pause the debugger. You can use this option instead of a breakpoint.	None
	Step over	Allows you to manually move to the next line in a data flow by stepping over a transform in the workspace. Use this option to see the first row in a data set after it is transformed. The workspace displays a red square on the line to indicate the path you are using. If the transform you step over has multiple outputs, the Designer provides a popup menu from which you can select the logic branch you want to take.	F10
	Get next row	Allows you to stay at the current breakpoint and view the next row of data in the data set.	F11
	Continue	Allows you to give control of the job back to the Designer. The debugger continues until: • You use the Pause debug option • Another breakpoint is encountered • The job completes	Ctrl+F10
	Show Filters/Breakpoints	Shows all filters and breakpoints that exist in a job. When not selected, all filters and breakpoints are hidden from view. This option is always available in the Designer.	None

Debug menu and tool bar options (Continued)

	Option	Description	Key Commands
	Set Filter/Breakpoints...	Opens a dialog from which you can set, remove, edit, enable or disable filters and breakpoints. You can also set conditions for breakpoints. From the workspace, you can right-click a line and select the same option from a short cut menu. Available when a data flow is active in the workspace.	F9
	Filters/Breakpoints...	Opens a dialog with which you can manage multiple filters and breakpoints in a data flow. Also offers the same functionality as the Set Filters/Breakpoints window. This option is always available in the Designer.	Alt+F9
	Call Stack	Shows or hides the Call Stack window	None
	Variables	Shows or hides the Debug Variables window	None
	Trace	Shows or hides the Trace window	None

Push-down optimizer

When Data Integrator executes a job, it normally pushes down as many operations as possible to the source database to maximize performance. Because the interactive debugger requires a job execution, the following push-down rules apply:

- Query transforms

The first transform after a source object in a data flow is optimized in the interactive debugger and pushed down to the source database if both objects meet the push-down criteria and if you have not placed a breakpoint on the line before the first transform. For more information about push-down criteria, see "Maximizing the number of pushed-down operations" on page 31 of the *Data Integrator Performance Optimization Guide*.

- Breakpoints

  Data Integrator does not push down any operations if you set a pre-defined breakpoint. Pre-defined breakpoints are breakpoints defined before you start the interactive debugger.

  After the interactive debugger is started, if the first transform is pushed down, the line is disabled during the debugging session. You cannot place a breakpoint on this line and you cannot use the View Data pane.

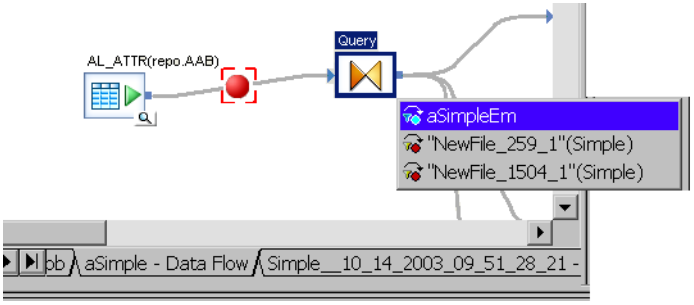
- Filters

  If the input of a pre-defined filter is a database source, it is push-down. Pre-defined filters are interactive debugger filters defined before you start the interactive debugger.

Limitations

- The interactive debugger can be used to examine data flows. Debug options are not available at the work flow level.
- A repository upgrade is required to use this feature.
- The debugger cannot be used with SAP R/3 data flows.

- All objects in a data flow must have a unique name. For example, if there are several outputs for a transform you can choose which path to use. If any of these objects have the same name, the result of your selection is unpredictable.



16 Exchanging metadata

Data Integrator allows you to import metadata from AllFusion ERwin Data Modeler (ERwin) from Computer Associates and export metadata for use with reporting tools like those available with the BusinessObjects 2000 BI Platform.

You can use the Metadata Exchange option or the Business Objects Universes option to export metadata.

- Using the Metadata Exchange option, you can export metadata into an XML file. After you create the file, you must manually import it into another tool.
- Using the Business Objects Universes option, you can export metadata directly from a repository into a universe using the **Create** or **Update** data mode.

This chapter discusses these topics:

Metadata exchange

You can exchange metadata between Data Integrator and third-party tools using XML files and the Metadata Exchange option.

Data Integrator supports two built-in metadata exchange formats:

- CWM 1.0 XML/XMI 1.1

  CWM (the Common Warehouse Metamodel)— is a specification that enables easy interchange of data warehouse metadata between tools, platforms, and repositories in distributed heterogeneous environments.

- ERwin 4.x XML

Data Integrator can also use:

- MIMB (the *Meta Integration® Model Bridge*)

  MIMB is a Windows stand-alone utility that converts metadata models among design tool formats. By using MIMB with Data Integrator, you can exchange metadata with all formats that MIMB supports. If MIMB is installed, the additional formats it supports are listed in the Metadata Exchange window.

- BusinessObjects Universal Metadata Bridge

  Converts Data Integrator repository metadata to Business Objects universe metadata. See "Creating Business Objects universes" on page 410.

This section discusses:

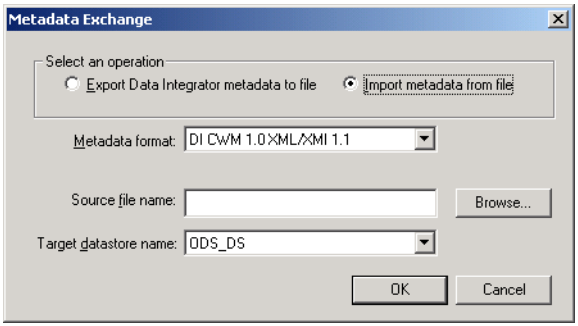
- Creating Business Objects universes
- Exporting metadata files from Data Integrator

Importing metadata files into Data Integrator

You can import metadata from ERwin Data Modeler 4.x XML into a Data Integrator datastore.

➢ To import metadata into Data Integrator using Metadata Exchange

1. From the **Tools** menu, select **Metadata Exchange**.
2. In the Metadata Exchange window, select **Import metadata from file**.



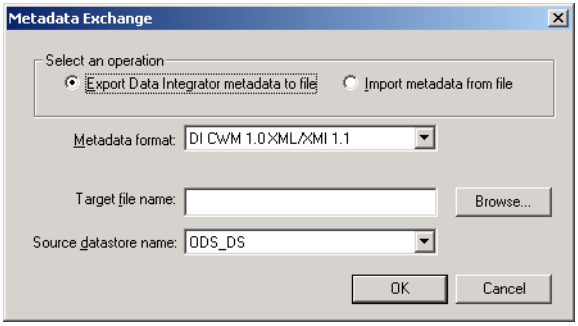
3. In the **Metadata format** box, select `ERwin 4.x XML` from the list of available formats.
4. Specify the **Source file name** (enter directly or click **Browse** to search).
5. Select the **Target datastore name** from the list of Data Integrator datastores.
6. Click **OK** to complete the import.

Exporting metadata files from Data Integrator

You can export Data Integrator metadata into a file that other tools can read.

➢ To export metadata from Data Integrator using Metadata Exchange

1. From the **Tools** menu, select **Metadata Exchange**.
2. In the Metadata Exchange window, select **Export Data Integrator metadata to file**.



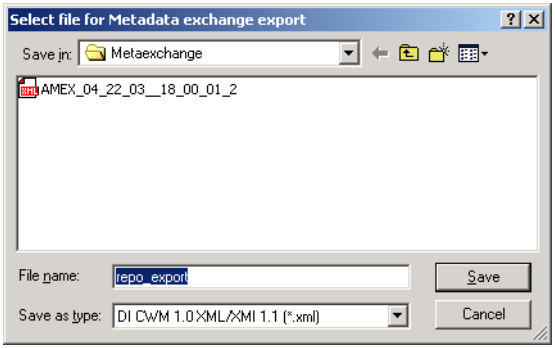
3. Select a **Metadata format** for the target from the list of available formats.

   If you have MIMB installed and you select an MIMB-supported format, select the **Visual** check box to open the MIMB application when completing the export process.

   If you do not select the **Visual** check box, the metadata is exported without opening the MIMB application. Using the MIMB application provides more configuration options for structuring the metadata in the exported file.

4. Specify the target file name (enter directly or click **Browse** to search).

When you search for the file, you open a typical browse window, like this:



Find any of the following file formats/types:

Format	File type
DI CWM 1.0 XML/XMI 1.1	XML
DI ERwin 4.x XML	XML
MIMB format (only if installed)	All

After you select a file, click **Open**.

5. Select the **Source datastore name** from the list of Data Integrator datastores.
6. Click **OK** to complete the export.

Creating Business Objects universes

Data Integrator allows you to easily export its metadata to Business Objects universes for use with business intelligence tools. A universe is a layer of metadata used to translate physical metadata into logical metadata. For example the physical column name `deptno` might become `Department Number` according to a given universe design.

NOTE: To use this export option, first install BusinessObjects Universal Metadata Bridge on the same machine as BusinessObjects Designer and Data Integrator Designer. You can install Universal Metadata Bridge using the installer for Data Integrator Designer or using the separate Universal Metadata Bridge CD.

You can create Business Objects universes using the Tools menu or the object library.

➢ To create universes using the Tools menu

1. Select **Tools** > **Business Objects Universes**.
2. Select either **Create** or **Update**.

   The Create Universe or Update Universe window opens.
3. Select the datastore(s) that contain the tables and columns to export and click **OK**.

   Data Integrator launches the Universal Metadata Bridge application and provides repository information for the selected datastores.

   For more information, refer to the *BusinessObjects Universal Metadata Bridge Guide*.

➢ To create universes using the object library

1. Select the **Datastores** tab.

2. Right-click a datastore and select **Business Objects Universes**.
3. Select either **Create** or **Update**.

   Data Integrator launches the Universal Metadata Bridge application and provides repository information for the selected datastores.

   For more information, refer to your *BusinessObjects Universal Metadata Bridge* documentation.

Mappings between repository and universe metadata

Data Integrator metadata maps to BusinessObjects Universe metadata as follows.

Data Integrator	BusinessObjects Universe
Table	Class, table
Column	Object, column
Owner	Schema
Column data type (see next table)	Object data type
Primary key/foreign key relationship	Join expression
Table description	Class description
Table Business Description	Class description
Table Business Name	Class name
Column description	Object description
Column Business description	Object description
Column Business Name	Object name
Column mapping	Object description
Column source information (lineage)	Object description

Data types also map:

Data Integrator Data type	BusinessObjects Type
Date/Datetime/Time	Date
Decimal	Number
Int	Number
Double/Real	Number
Interval	Number
Varchar	Character
Long	Long Text

Attributes that support metadata exchange

The attributes **Business_Name** and **Business_Description** exist in Data Integrator for both tables and columns. These attributes support metadata exchanged between Data Integrator and BusinessObjects through the Universal Metadata Bridge (UMB) 1.1.

- A Business_Name is a logical field. Data Integrator stores it as a separate and distinct field from physical table or column names. Use this attribute to define and run jobs that extract, transform, and load physical data while the Business Name data remains intact.
- A Business_Description is a business-level description of a table or column. Data Integrator transfers this information separately and adds it to a BusinessObjects Class description.

Data Integrator includes two additional column attributes that support metadata exchanged between Data Integrator and BusinessObjects:

- Column_Usage
- Associated_Dimension

For more information see "Column attributes for tables" on page 101 of the *Data Integrator Reference Guide*.

17 Metadata Reporting

Data Integrator provides full access to the metadata of applications you create. You can use this metadata to analyze your applications and you can exchange this metadata with other tools.

This chapter discusses:

- Repository reporting tables and views
- Metadata reporting tool
- Metadata analysis categories

Repository reporting tables and views

The Data Integrator repository is a database that stores your application components and the built-in Data Integrator design components and their properties. The open architecture of the repository allows for metadata sharing with other enterprise tools.

Within your repository, Data Integrator populates a special set of reporting tables with metadata describing the objects in your repository. When you query these tables, you can perform analyses on your Data Integrator applications. You can:

- Analyze dependencies between Data Integrator objects

  For example, you can determine which jobs call a particular data flow, and in turn, which data flows are called by a particular job. With this kind of information, you can analyze the impact of changing or deleting an object.

- Determine which sources and columns are used to populate a given target column

  With this kind of information you can answer questions such as "Which targets are affected if source column Region is dropped from source Customer?"

- Determine the expressions used to populate columns in a target data warehouse or data mart

  With this kind of information you can answer questions such as "What expression is used to populate column Unit_Cost in target Inventory?"

  Determining this kind of information from working backwards through a complex series of data transformations can be difficult and time-consuming. With Data Integrator metadata reporting, a complex transformation sequence is represented by a single mathematical expression.

- Exchange metadata with another application

  Some reporting tables contain information that can be used to exchange metadata with other applications, such as an OLAP or BI tool.

Data Integrator metadata reporting tables and views are listed below and described in Chapter 8, "Metadata in repository tables and views," in the *Data Integrator Reference Guide*. Note that this is not the complete list because some repository tables and views are for internal use.

Name	Contains
AL_ATTR	Attribute information about native objects
AL_HISTORY	Execution statistics about jobs and data flows
AL_INDEX	Index information about imported tables
AL_LANG	Information about native Data Integrator objects
AL_PCOLUMN	Column information about imported table partitions
AL_PKEY	Primary key information about imported tables
AL_USAGE	All ancestor-descendant relationships between Data Integrator objects
ALVW_COLUMNATTR	Attribute information about imported columns.
ALVW_COLUMNINFO	Information about imported column.
ALVW_FKREL	Primary-foreign key relationships among imported tables.
ALVW_FLOW_STAT	Execution statistics about individual transforms within data flows.
ALVW_FUNCINFO	Information about both native functions and functions imported from external systems.
ALVW_MAPPING	Mapping and lineage information for target tables.
ALVW_PARENT_CHILD	Direct parent-child relationships between Data Integrator objects.
ALVW_TABLEATTR	Attribute information about imported (external) tables.
ALVW_TABLEINFO	Information about imported tables

Reporting tables are automatically created for each new or upgraded Data Integrator repository. Except for AL_USAGE, all reporting tables are updated automatically by Data Integrator.

Metadata reporting tool

Using the metadata reporting tool, you can easily browse and report on metadata associated with a Data Integrator application. The metadata reporting tool is a Web-based application. You can access the metadata reporting tool from the Designer, which launches a default browser and points it to the URL for the metadata reporting tool, or you can access the metadata reporting tool directly from a Web browser.

This section discusses:

- Configuration
- Viewing metadata reports

Configuration

The metadata reporting tool uses:

- The Web Server service (which uses the Tomcat servlet engine)
- JDBC drivers to connect to a repository

Use the Data Integrator Administrator to configure repositories the metadata reporting tool will access. See the *Data Integrator Administrator Guide* for detailed information.

Viewing metadata reports

You can open the metadata reporting tool and generate metadata reports in several ways:

- Open a Web browser and enter the URL for the metadata reporting tool:

  `http://computer_name:port/reports`

  The default port is the same as the Web Server (28080).

Use the same information you use to log in to the Administrator. Default user name is **admin** and default password is **admin**. Only users with valid Administrator login privileges can log in to the Metadata Reports tool.

- From the Designer, select **Tools** > **Metadata Reports** or click the **Metadata Reports** tool bar icon.

  Data Integrator launches a report browser, and logs in to the same repository as the Designer. You can select a report or switch to a different repository.

- In the object library, right-click any object and select **Metadata Reports**.

  Data Integrator launches a report browser, logs in to the same repository as the Designer, and opens the Metadata Report application.

NOTE: Certain reports are only current after you calculate usage dependencies and column mappings. For more information, see"Refresh report content" on page 456.

Metadata analysis categories

Data Integrator Metadata Reporting provides you with object summary reports, relationship analysis reports, job and data flow statistics, and tools to manage database connections and update your repository information. Metadata analysis categories include:

- Repository summary
- Datastore analysis
- Operational statistics
- Universe analysis
- Dependency analysis
- Tools

NOTE: When you see up/down arrows on a report column header, you can click the arrows to sort the column in ascending or descending order. After a manual sort, sort order is indicated by arrow direction. Default sort order is alphabetical ascending.

Repository summary

The repository summary is the home page that appears when you open Data Integrator Metadata Reporting. Using reports in this category, you can:

- See all jobs in a repository.
- Select a job to find out more details about that job—the properties of the job and the objects in the job.
- Select objects, such as work flows, to find out more information about those objects.
- Drill down to view data flow level information.
- Click a Report Shortcuts icon to view associated reports.

This page provides quick access to the following repository reports:

Overview

This is the default report and shows repository name, type, version, number of objects in use, and total objects in the repository. The overview provides a "big picture" view of the repository, including size. **Linked reports**: None.

Projects

This report lists all projects in the repository. **Report Shortcuts**: None.

Jobs

Lists all jobs in the repository. **Report Shortcuts**: Component Analysis, Job execution statistics (all).

- Click the **Job Component Analysis** icon for any job to view all first-level objects within that job. First-level objects that contain nested objects display as links. Click a link to view the Component Analysis report for that object.

- Click the **Job Execution Statistics** icon to view Job execution statistics (all) for that job.

Work Flows

The work flows report is similar to the Jobs report but lists all work flows in the repository. **Report Shortcuts**: Component Analysis, Where Used.

- Click the **Work Flow Component Analysis** icon for any work flow to view all first level objects within that work flow. First level objects that contain nested objects are displayed as links. Click a link to view the Component Analysis for that object.

- Click the **Where Used** icon to search for where this work flow is used within all jobs or a specific job.

Data Flows

This report lists all data flows in the repository. **Report Shortcuts**: Data flow execution statistics (all), Component Analysis, Where Used.

- Click the **Data Flow Component Analysis** icon for any data flow to view all first-level objects within that data flow. First-level objects that contain nested objects appear as links. Click a link to view the Component Analysis for that object.

- Click the **Data Flow Execution Statistics** icon to view Data flow execution statistics (all) for that data flow.

- Click the **Where Used** icon to search for where this data flow is used within one job or all jobs.

Datastores

This report lists all datastores defined in this repository. **Report Shortcuts**: Datastore Overview Report, Tables Report, Functions Report, Hierarchies Report.

- Click the **Datastore Overview Report** icon to view Overview information for that datastore.

- Click the **Tables Report** icon to view Tables information for that datastore.

- Click the **Functions Report** icon to view Functions information for that datastore.

- Click the **Hierarchies Report** icon to view Hierarchies information for that datastore.

Transforms

This report lists all user-defined (custom) transforms in the repository. **Report Shortcuts**: Where Used.

Click the **Where Used** icon to search for where a transform is used within all jobs or a specific job.

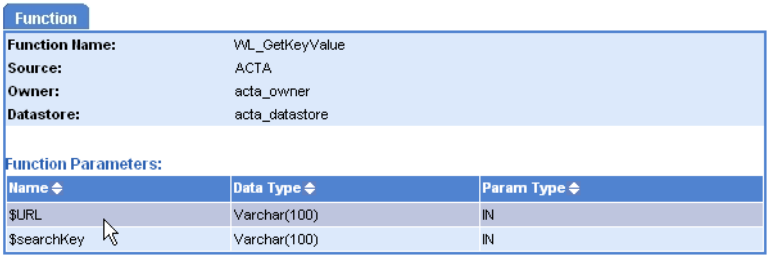
Formats

This report lists all File Formats (Flat Files, DTDs, and XML Schemas) available within the repository. **Report Shortcuts**: None.

Custom Functions

This report lists all user-defined (custom) functions in the repository. **Linked reports**: Function Details Report, Where Used.

- Click the **Function Report** icon to view the Function Information report for that custom function.



- Click the **Where Used** icon to search for where this custom function is used within all jobs or a specific job.

Any time you see the **Generate Report** button, you can click it to generate a full report containing summarized information for the objects you select. Each report appears in a separate window and you can print a report at any time before closing its window. With separate windows, you can compare and analyze multiple reports before closing them. Note that when you click **Generate Report** while viewing the Repository Summary, your report will list all objects on each tab of the Repository Summary node. After closing a given report window, you cannot reopen it; instead, you must generate a new report.

Component Analysis

This report type is accessible only from Jobs, Work Flows, and Data Flows reports when you click the corresponding Report Shortcut icon:

- Job component analysis

- Work flows component analysis

- Data flows component analysis

Click a component-analysis icon to view first-level objects for a selected job, work flow, or data flow. Note that this report does not display the entire nested structure. However, if more objects are nested within a displayed first-level object, you can click the link for that object to view a component analysis for the selected object.

So, if you click the component-analysis icon for a job called "Sales_Region" and see three work flows and two data flows all displayed as links, you can click any of those links to view the next-level analysis. For instance, if you clicked the "Sales_MonthlyWF" work flow, you might see more linked work flows and/or linked/unlinked data flows as first-level objects, and so forth.

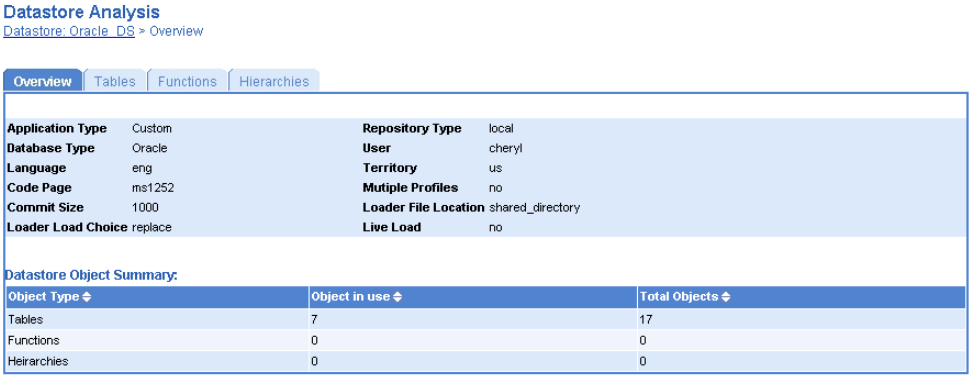
Datastore analysis

This category contains a set of reports for each datastore in the repository. Some reports include more detailed sub-reports. Under each datastore node, find the following reports:

- Overview
- Tables
- Functions
- Hierarchies

Overview

Lists datastore attributes, number of objects in-use, and total number of objects in the datastore.



Tables

Lists all tables in the repository, specifying associated Business Name, Object Type, Owner name, Description, and Report Shortcuts. Click a table name link or one of the Report Shortcuts to access table sub-report tabs. Sub-reports include:

- Table Information
- Table Lineage
- Table Impact
- Column Mapping
- Column Lineage
- Column Impact
- Universe Impact

Click a table link to see the Table Information report. Data Integrator also provides report shortcuts to Table Impact, Column Mapping, and Business Objects Universes (when applicable). You can view and print a custom report by selecting one or more tables and clicking **Generate Report**.

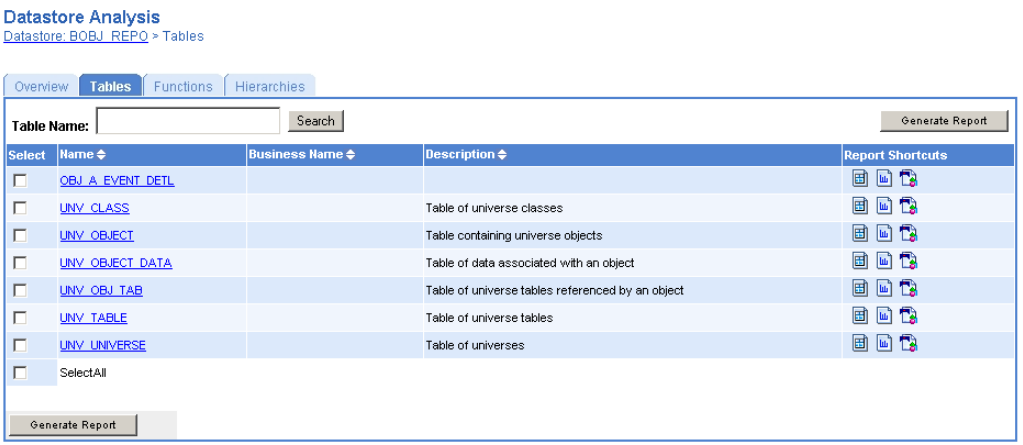
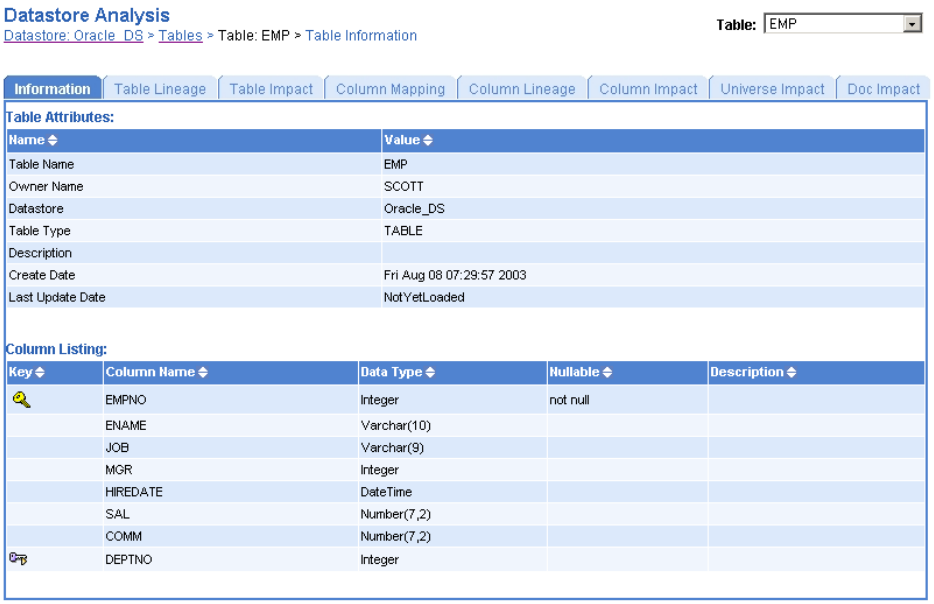


Table Information

The table information sub-report is split into multiple sections: Table Attributes, Column Listing, and Partition Information (when applicable). Table attributes are listed with attribute values. Column Listing includes each column name in the table, its data type, key (if applicable), null setting, and description.



Select any table in the current repository from the **Table** list to instantly view information for that table.

Table Lineage

The table lineage sub-report provides a graphical "map" to show what sources feed into the selected table including the name of the source, source type, and associated data flow. When no lineage is identified, the report indicates that the table is an original source.

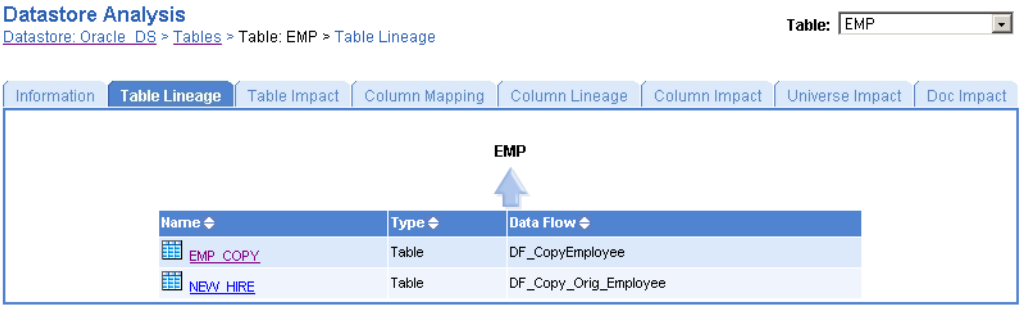
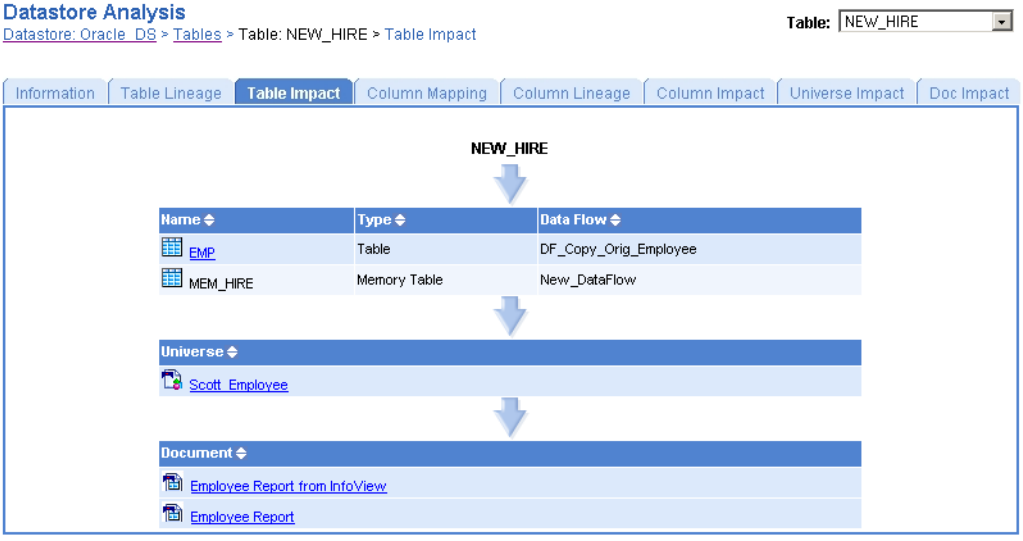


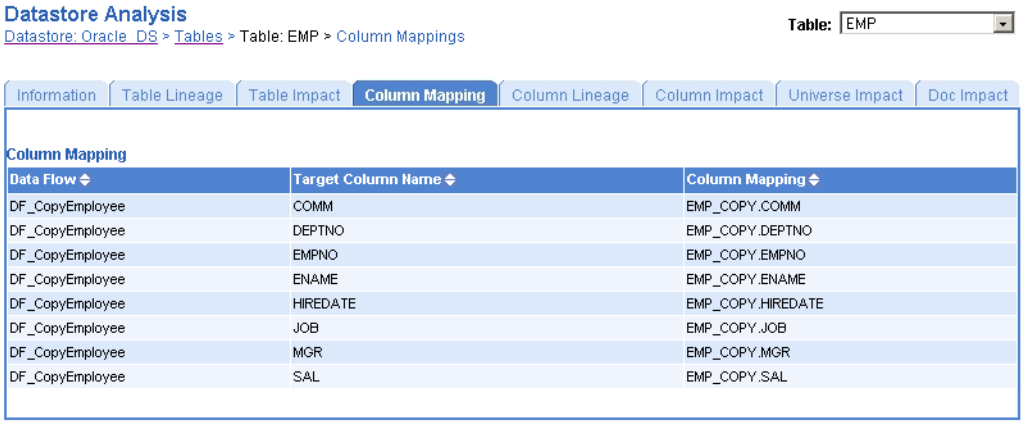
Table Impact

The table impact sub-report provides a graphical “map” to show what targets this table feeds including the name of the target, target type, and associated data flow. When no impact is identified, the report indicates the table is a final target.



Column Mapping

For a specified table, the column mapping report shows target column name, associated data flow, and mapping.



Datastore Analysis

Datastore: Oracle_DS > Tables > Table: EMP > Column Mappings

Table: EMP

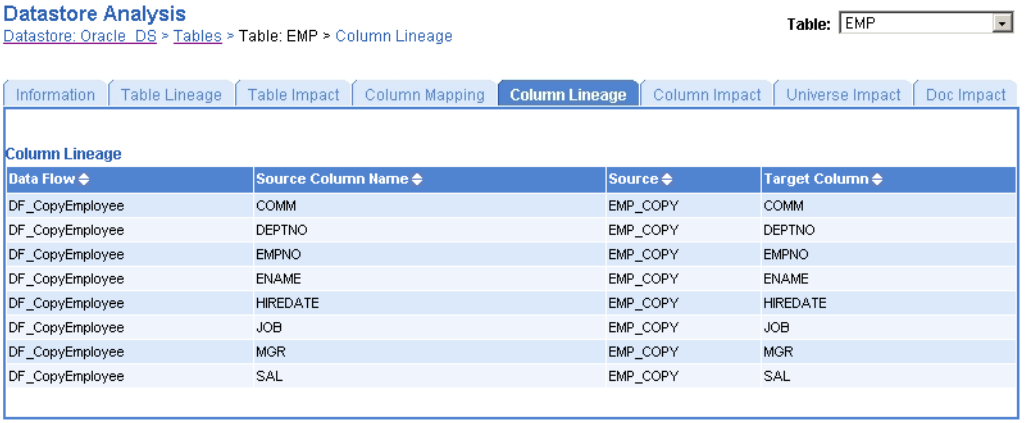
Information | Table Lineage | Table Impact | **Column Mapping** | Column Lineage | Column Impact | Universe Impact | Doc Impact

Column Mapping

Data Flow	Target Column Name	Column Mapping
DF_CopyEmployee	COMM	EMP_COPY.COMM
DF_CopyEmployee	DEPTNO	EMP_COPY.DEPTNO
DF_CopyEmployee	EMPNO	EMP_COPY.EMPNO
DF_CopyEmployee	ENAME	EMP_COPY.ENAME
DF_CopyEmployee	HIREDATE	EMP_COPY.HIREDATE
DF_CopyEmployee	JOB	EMP_COPY.JOB
DF_CopyEmployee	MGR	EMP_COPY.MGR
DF_CopyEmployee	SAL	EMP_COPY.SAL

Column Lineage

For a specified column name, the column lineage sub-report shows all data flows that use this column as a target. The report shows data flow, source column name, source object, and target column name.



Datastore Analysis

Datastore: Oracle_DS > Tables > Table: EMP > Column Lineage

Table: EMP

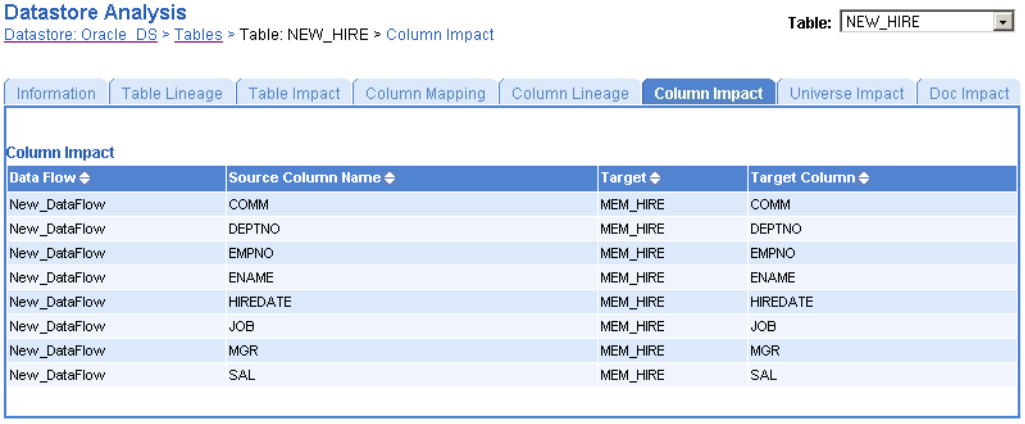
Information | Table Lineage | Table Impact | Column Mapping | **Column Lineage** | Column Impact | Universe Impact | Doc Impact

Column Lineage

Data Flow	Source Column Name	Source	Target Column
DF_CopyEmployee	COMM	EMP_COPY	COMM
DF_CopyEmployee	DEPTNO	EMP_COPY	DEPTNO
DF_CopyEmployee	EMPNO	EMP_COPY	EMPNO
DF_CopyEmployee	ENAME	EMP_COPY	ENAME
DF_CopyEmployee	HIREDATE	EMP_COPY	HIREDATE
DF_CopyEmployee	JOB	EMP_COPY	JOB
DF_CopyEmployee	MGR	EMP_COPY	MGR
DF_CopyEmployee	SAL	EMP_COPY	SAL

Column Impact

This report shows the data flows that this column impacts, including source column name, target object, and target column.



Datastore Analysis

Datastore: Oracle_DS > Tables > Table: NEW_HIRE > Column Impact

Table: NEW_HIRE

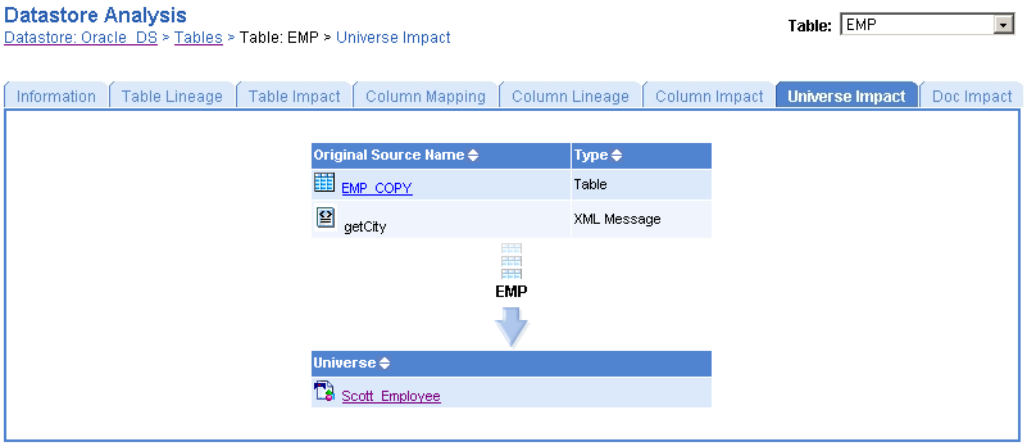
Information | Table Lineage | Table Impact | Column Mapping | Column Lineage | **Column Impact** | Universe Impact | Doc Impact

Column Impact

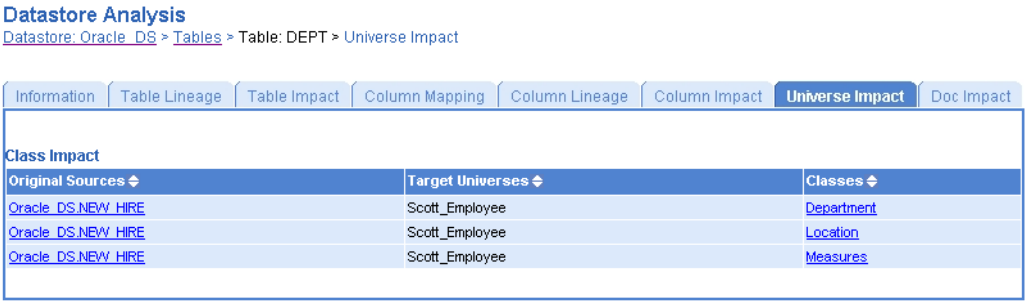
Data Flow	Source Column Name	Target	Target Column
New_DataFlow	COMM	MEM_HIRE	COMM
New_DataFlow	DEPTNO	MEM_HIRE	DEPTNO
New_DataFlow	EMPNO	MEM_HIRE	EMPNO
New_DataFlow	ENAME	MEM_HIRE	ENAME
New_DataFlow	HIREDATE	MEM_HIRE	HIREDATE
New_DataFlow	JOB	MEM_HIRE	JOB
New_DataFlow	MGR	MEM_HIRE	MGR
New_DataFlow	SAL	MEM_HIRE	SAL

Universe Impact

The Universe impact report only applies if you have configured a Business Objects Universe. The report provides a graphical "map" to show the original source (and source type) feeding the table that impacts the Universe as well as the staging tables in the table being reported on, and all Universes that use that table.



When you select a Universe, a sub-report shows a list of all configured Business Objects Universes and Classes where this table is used. In addition, this tab shows the original sources for the table.



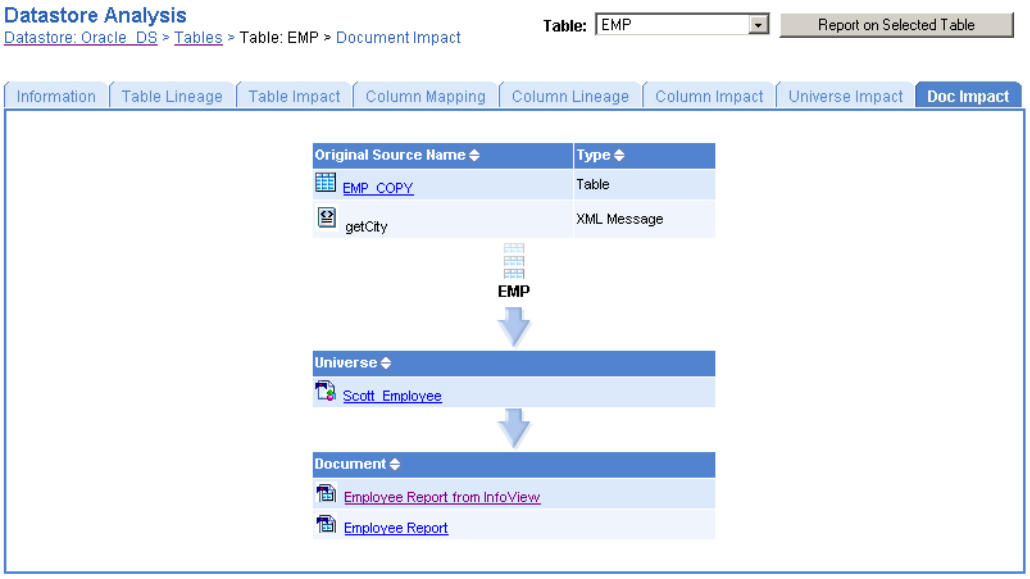
Click a class link to see the objects in that class. The object-level report shows:

- Original Sources (listed as datastore.owner.source_name) can be a table, flat file, message, function, and so forth.
- Target Universes (that use this table)
- Classes (class name)
- Objects (in the class that use this table)
- Universe Columns (used by the class, listed as datastore.owner.table_name)

For more Universe-related reports, see Universe analysis.

Document Impact

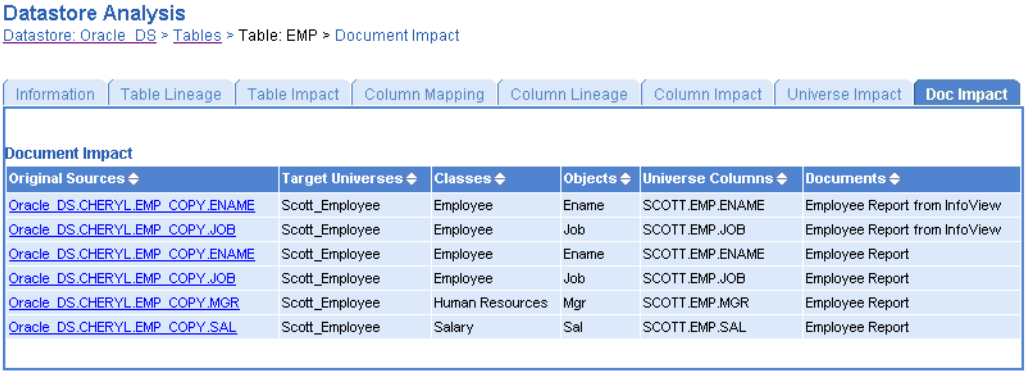
The Document Impact report only applies only if you have configured a Business Objects Auditor database. The report provides a graphical "map" to show the original source (and source type) feeding your table as well as the Universes and Documents impacted by the table.



If instead of an arrow, you see a progressive table illustration between the original source and your table, this means staging tables exist. Click the illustration to review staging table details on the Table Lineage tab. You can also select any Document to view a sub-report with details that include:

- Original Sources (can be a table, flat file, message, function, and so forth).
- Target Universes (that use your selected table)
- Classes (class name)
- Objects (in the class that use this table)
- Universe Columns (used by the object)

- Documents (that use the identified object)



Datastore Analysis

Datastore: Oracle_DS > Tables > Table: EMP > Document Impact

Information | Table Lineage | Table Impact | Column Mapping | Column Lineage | Column Impact | Universe Impact | Doc Impact

Document Impact

Original Sources	Target Universes	Classes	Objects	Universe Columns	Documents
Oracle_DS.CHERYL.EMP_COPY.ENAME	Scott_Employee	Employee	Ename	SCOTT.EMP.ENAME	Employee Report from InfoView
Oracle_DS.CHERYL.EMP_COPY.JOB	Scott_Employee	Employee	Job	SCOTT.EMP.JOB	Employee Report from InfoView
Oracle_DS.CHERYL.EMP_COPY.ENAME	Scott_Employee	Employee	Ename	SCOTT.EMP.ENAME	Employee Report
Oracle_DS.CHERYL.EMP_COPY.JOB	Scott_Employee	Employee	Job	SCOTT.EMP.JOB	Employee Report
Oracle_DS.CHERYL.EMP_COPY.MGR	Scott_Employee	Human Resources	Mgr	SCOTT.EMP.MGR	Employee Report
Oracle_DS.CHERYL.EMP_COPY.SAL	Scott_Employee	Salary	Sal	SCOTT.EMP.SAL	Employee Report

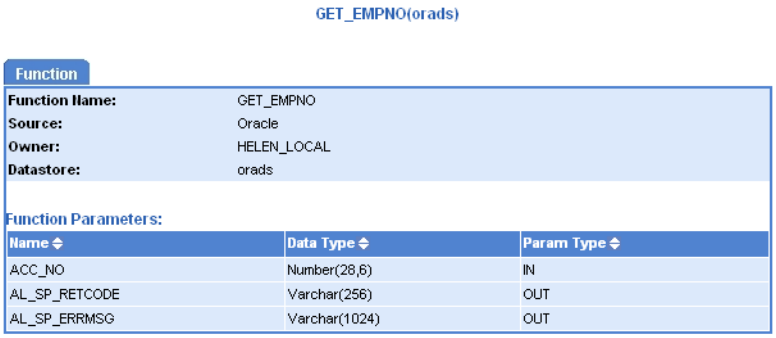
NOTE: This report also shows column name, partition name, partition type, and range of partitioned tables.

For more Business Objects Auditor-related reports, see "Universe analysis" on page 440, "Document Table Lineage" on page 445 and "Document Column Lineage" on page 446.

Functions

Lists the name, source, and description of each function available in the datastore. Select a function name link to view the information sub-report for that function.

- **Information** — This sub-report shows function attributes as well as function parameters (name, data type, parameter type).



Hierarchies

Lists all hierarchies in the datastore with owner name and description. You can search for hierarchies by name.

Operational statistics

This category contains performance-related reports including:

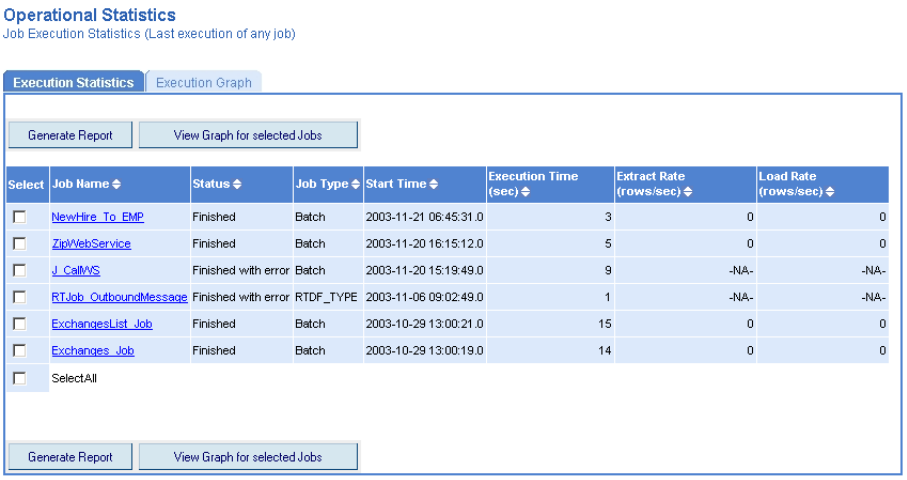
- Job execution statistics (last)
- Job execution statistics (all)
- Data flow execution statistics (last)
- Data flow execution statistics (all)

You can use these reports to answer some of the following questions:

Which jobs have consistent performance characteristics? Which jobs have degraded performance over time? Which data flows are bottlenecks? What is my data throughput?

Job execution statistics (last)

Execution Statistics—This report displays information for the last execution of the most currently run jobs. Data Integrator initially sorts this report by start time, but you can sort the list by job name, execution time, and so forth.



Click a job name to show execution details for that job including job execution summary, dataflow execution summary, and specific execution information. Select up to 10 specific jobs, then click **View Graph for selected jobs** to see a visual representation of the last 15 executions within the last 15 days for those jobs. The graph plots job start time and execution time (in seconds).

Execution Graph—Click this tab to view a graphical representation of job performance. The graph shows the last 15 executions within the last 15 days for up to10 jobs. To view other jobs, select them, then click **View Graph for selected jobs**. The graph plots job start time and date and execution time (in seconds).

Job execution statistics (all)

Execution Statistics—This report shows performance statistics for the last 15 days or the last 15 executions of a selected job. Job names are listed in alphabetical order and statistics for the first job on the list are displayed as the default. To filter results for another job, choose another Job Name from the list, then click **Search**. Default sort is by Start Time; however, you can re-sort the information by Start Time or sort by Execution Time.

Execution Graph—Click this tab to view a graphical representation of how this job performed over the last 15 executions within the last 15 days. The graph also displays the five data flows that consumed the most processing time during these executions.

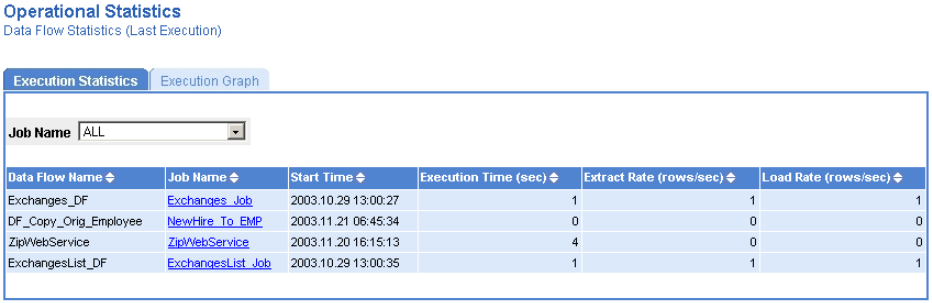
Data flow execution statistics (last)

Execution Statistics—This report shows performance statistics for data flows within a specified job. Filter data flows by job name. Similar to the job execution graph, you can view the last 15 executions of a selected job.

Execution Graph—Click this tab to view a graphical representation of how this these data flows performed over the last 15 executions.

Data flow execution statistics (all)

Execution Statistics—This report shows performance statistics for data flows within indicated jobs. You can filter data flows by job name. Similar to the previous graph, you can view the last 15 executions of a selected job.


Operational Statistics
Data Flow Statistics (Last Execution)

Execution Statistics | Execution Graph

Job Name: ALL

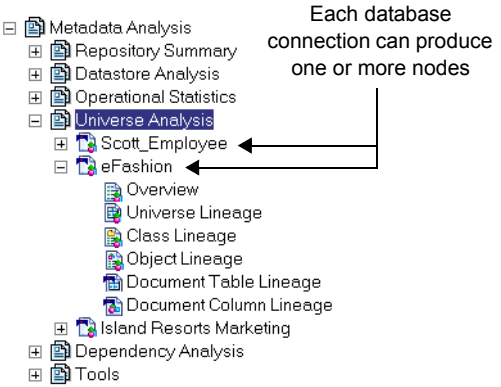
Data Flow Name	Job Name	Start Time	Execution Time (sec)	Extract Rate (rows/sec)	Load Rate (rows/sec)
Exchanges_DF	Exchanges_Job	2003.10.29 13:00:27	1	1	1
DF_Copy_Orig_Employee	NewHire_To_EMP	2003.11.21 06:45:34	0	0	0
ZipWebService	ZipWebService	2003.11.20 16:15:13	4	0	0
ExchangesList_DF	ExchangesList_Job	2003.10.29 13:00:35	1	1	1

Execution Graph—Click this tab to view a graphical representation of how this these data flows performed over the last 15 executions.

Universe analysis

Use Universe analysis to immediately “see into” external repositories and view metadata relationships between your Data Integrator repository and external Business Objects repositories such as Universes and Auditor databases. Similar to Datastore analysis, you configure the database connections you want and each logical data grouping associated with that connection is represented as one or more nodes under Universe Analysis. For each node, Data Integrator produces a series of reports that shows the relationship between Data Integrator sources and Business Objects Universes.

To configure connections, see Business Objects Connections in the "Tools" node of the navigation tree.



Business Objects Universe reports

For each Business Objects repository connection you create, one or more Universe nodes appears under Universe Analysis in the navigation tree. For each node, reports include:

- Overview
- Universe Lineage
- Class Lineage
- Object Lineage
- Document Table Lineage
- Document Column Lineage

Within several reports, you can filter information, limiting the scope of the report. Filters include:

- **Class** — Lists every class in the Universe. To limit report scope to a single class, select the class name and click **Search**. To see all classes, select **ALL** and click **Search**.

- **Document**—Lists every document in the Business Objects Universe. To limit report scope to a single document, select the class name and click **Search**. To see all documents, select **ALL** and click **Search**.
- **Object**—Lists every object in the Business Objects Universe. To limit report scope to a single object, select the object name and click **Search**. To see all objects, select **ALL** and click **Search**.
- **Table**—Lists every table in the Business Objects Universe. To limit report scope to a single table, select the class name and click **Search**. To see all tables (including those not referenced by Data Integrator), select **ALL** and click **Search**.

NOTE: Filters can only be applied for a single category (class, object, table, or document). If you select an entry from one filter category, all other filters default to "ALL."

If you cannot see your Business Objects Documents within the Metadata Reports window, see "If Business Objects Documents do not appear" on page 447.

Overview

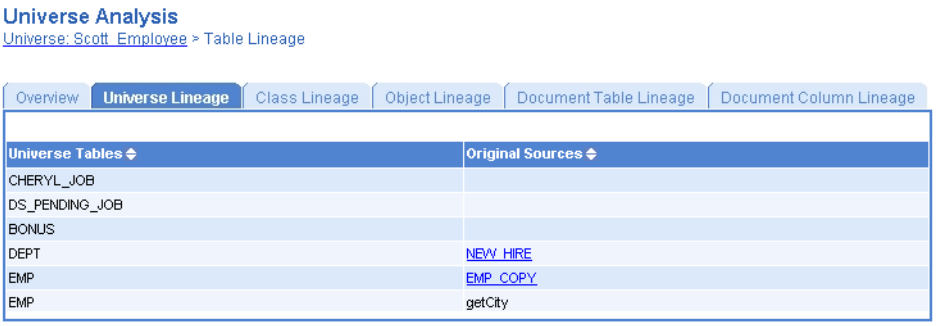
This report lists Universe attributes.

- Universe name
- Universe filename
- Universe ID (internal ID assigned to the Universe)
- Database connection name (you defined in Database connections page)

Universe Lineage

This report shows all tables in the selected Universe, listing every table used by that Universe. If a Data Integrator data flow uses a table, this report identifies the original source of the data. In this report, view:

- Universe Table names
- Original Sources (listed by source name) can be a table, flat file, message, function, and so forth. Note that if the source is a table or function, it is linked to the associated table or function report under Datastore analysis.

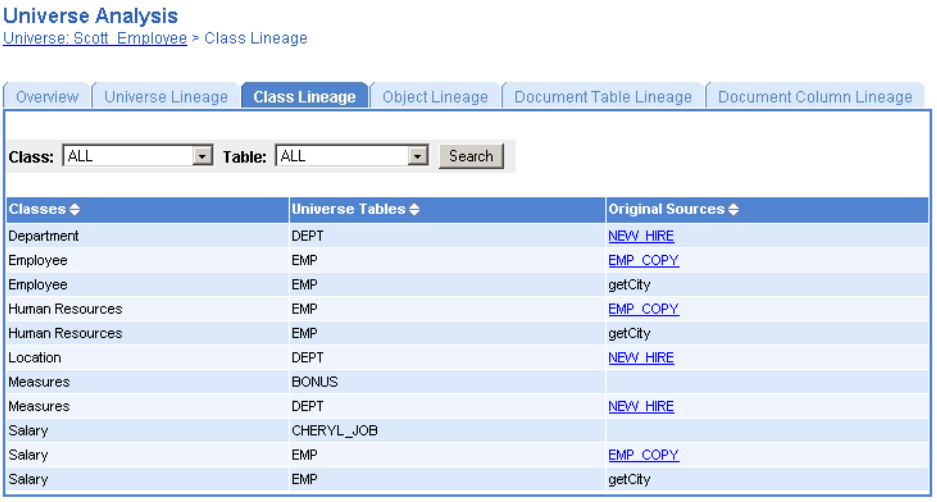


Class Lineage

This report shows all classes in the selected Universe and the Data Integrator tables that they use, listing every class and associated table. If a Data Integrator data flow uses a table, this report identifies the source of the data.

- Class (class name)
- Universe table (used by the class, listed by table name)
- Original Sources (listed by source name) can be a table, flat file, message, function, and so forth. Note that if the source is a table or function, it is linked to the associated table or function report under Datastore analysis.

Data Integrator initially sorts this report by the first class in the list. You can filter this report by class or Universe table.



Universe Analysis

Universe: Scott_Employee > Class Lineage

Overview | Universe Lineage | Class Lineage | Object Lineage | Document Table Lineage | Document Column Lineage

Class: ALL Table: ALL Search

Classes	Universe Tables	Original Sources
Department	DEPT	NEW_HIRE
Employee	EMP	EMP_COPY
Employee	EMP	getCity
Human Resources	EMP	EMP_COPY
Human Resources	EMP	getCity
Location	DEPT	NEW_HIRE
Measures	BONUS	
Measures	DEPT	NEW_HIRE
Salary	CHERYL_JOB	
Salary	EMP	EMP_COPY
Salary	EMP	getCity

Object Lineage

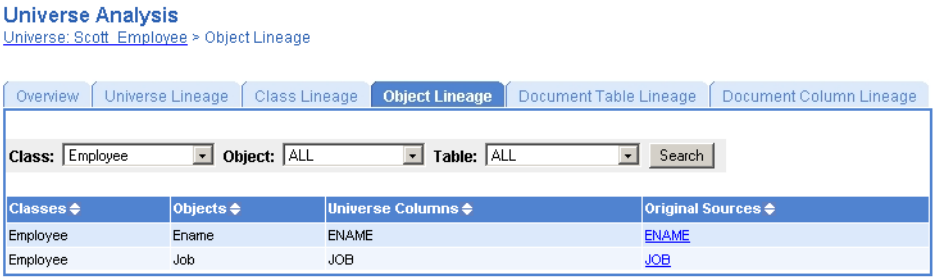
This report shows all objects in a selected Universe and the Universe columns that they use. Data Integrator shows every object and associated column used by the Universe. If a Data Integrator flow uses a column, this report identifies the source.

- Classes
- Objects (object name)
- Universe Columns (listed by column name)
- Original Sources (listed by column_name) can be a table, flat file, message, function, and so on. If the source is a table or function, it is linked to the associated table or function report under Datastore analysis.

Data Integrator initially sorts this report by the first object name in the list. You can also filter this report by:

- Class
- Original Source

- Universe Column



Document Table Lineage

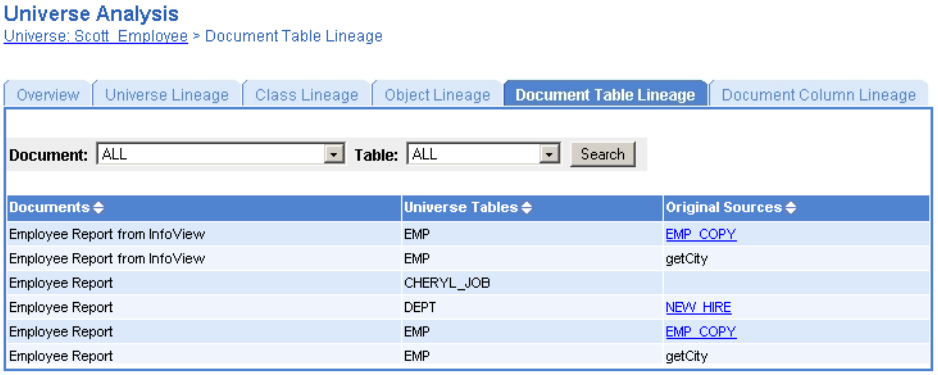
This report appears only with a Business Objects Auditor database configuration. The report shows all documents that use this Universe and the tables used by each document.

- Documents (that use the selected Universe)
- Universe Tables (used by the document, listed by table name)
- Original Sources (listed by source name) can be a table, flat file, message, function, and so forth. Note that if the source is a table or function, it is linked to the associated table or function report under Datastore analysis.

Data Integrator initially sorts this report by the first document name in the list. You can filter this report by:

- Document

- Table



Document Column Lineage

This report appears only with a Business Objects Auditor database configuration and shows all documents that use this Universe as well as the column used by each document.

- Documents (that uses the selected Universe)
- Classes (used by the document)
- Objects (used by the document)
- Universe Columns (used by the document, listed by column name)
- Original Sources (listed by source column name) can be a table, flat file, message, function, and so forth. Note that if the source is a table or function, it is linked to the associated table or function report under Datastore analysis.

Data Integrator initially sorts this report by the first document name in the list. You can filter this report by:

- Document
- Object
- Table

If Business Objects Documents do not appear

Data Integrator Metadata Reports list and describe tables and columns used for Business Objects Documents (reports from a Universe). However, if Business Objects Documents do not appear in the Metadata Reporting area, you should:

1. Set WEBI logging to record these details to a database.
2. Enable WEBI logging.
3. Refresh Business Objects Documents using WEBI InfoView.

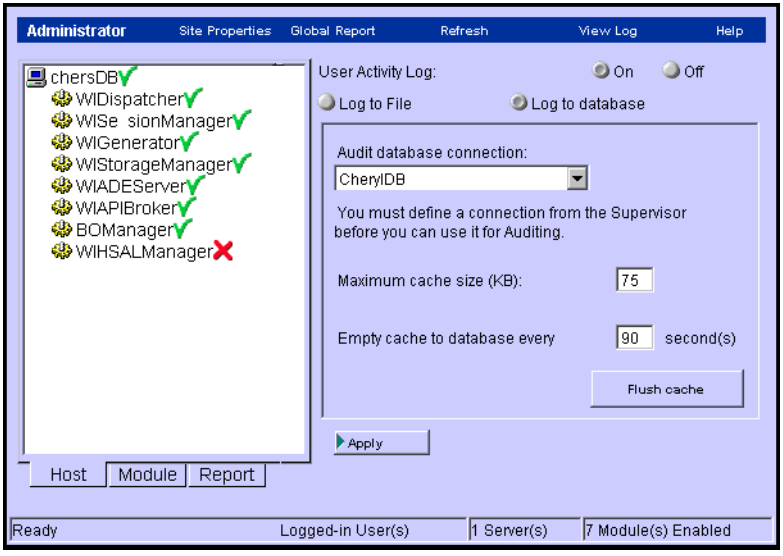
If Document information still does not display, perform the following steps to view the WEBI logging state:

1. Enable diagnostic tracing for Data Integrator Metadata Reports:
   a. Open: `<linkdir>\ext\webserver\webapps\acta_metadata_reports\WEB-INF\ log4j.properties`
   b. In the first line of the properties file, set the trace level to `DEBUG`. The first line of the properties file indicates the level of diagnostics recorded to the log file. The default is `'log4j.rootLogger=INFO, A'`. For `DEBUG` tracing, the line should read `'log4j.rootLogger=DEBUG, A'`.
   c. Restart the service for Data Integrator Web Server so the Properties file can recognize your changes.
2. After enabling diagnostic tracing, open Data Integrator Metadata Reports and open any Document report.
3. Review the Data Integrator trace file and search for diagnostic information indicating the last event Data Integrator found recorded in the WEBI database.
   a. Open `<linkdir>\log\reports.log`
   b. Look for the following entry where `<nn>` is the last event identifier found in the WEBI database.

```
[Metadata Reports] last event recorded in
WEBI log is <nn>
```

4. Go to WEBI InfoView and refresh a report.
5. Return to the Data Integrator Metadata Reports window and reopen the Document report.
6. Review the Data Integrator trace file and search for the `[Metadata Reports] last event recorded in WEBI log is <nn>` entry. The sequence number should have increased.

   If the sequence number did not increase, WEBI logging is not recording information in the database.
7. Verify the WEBI logging state by opening WEBI Administrator.
   a. Open the log page. (The log page is named **View Log** in pre-E6 releases. It is named **Audit** in E6.)



   b. Set logging to **On** and enable the **Log to database** option.

The database listed as the **Audit** database connection must be the same connection configured in Data Integrator Metadata Reports for **Auditor**.

Dependency analysis

This category produces reports based on search criteria you specify to find specific objects in the repository. Furthermore, you can use dependency analysis reports to see past the repository and into external repositories that you configure under Universe analysis.

These reports can help you answer questions like:

- What is the ripple effect if I change this table?
- What is the ripple effect if I change this column?
- How is this column mapped to other columns?
- Where am I using this object?

Dependency analysis tools include:

- Table
- Column
- Where Used
- Contains

Table

Use this the table dependency tool to search for one or more specific tables. To view the Table Lineage or Table Impact information for any table returned by the search, click the appropriate Report Shortcut link.

➢ To perform a table dependency search

1. Choose a repository to search.

2. Enter the table name or a subset of context-sensitive characters as search criteria, for example %SALES%.
3. Choose a datastore to search. If you want to search through all configured datastores, select **ALL**.
4. Click **Search**.

   Data Integrator displays search results as a list of table links with associated Datastore and Owner name, as well as Table Lineage and Table Impact Report Shortcuts. You can also click the table name link to view the Table Impact report for that table.

Column

Use this tool to search for a specific column. For each column returned from the search, you can click an associated link to see its Column Mapping, Column Lineage, or Column Impact report.

➢ To perform a column dependency analysis search

1. Choose a repository to search.
2. Enter the column name or a subset of context-sensitive characters as search criteria, for example %EMP%.
3. Choose a datastore to search. If you want to search through all configured datastores, select **ALL**.
4. Click **Search**.

   Data Integrator displays search results as a list of column names with associated table name links as well as repository Owner name, Datastore name, and Report Shortcuts. Click the table link for a column to view the Column Mapping report associated with that column. Click the Column Mapping, Column Lineage, or Column Impact Report Shortcuts to view those reports.

Where Used

Use this powerful search tool and associated report to find all locations of a specified object within your repository. You select the scope of the report using Object Type and Object Name filters.

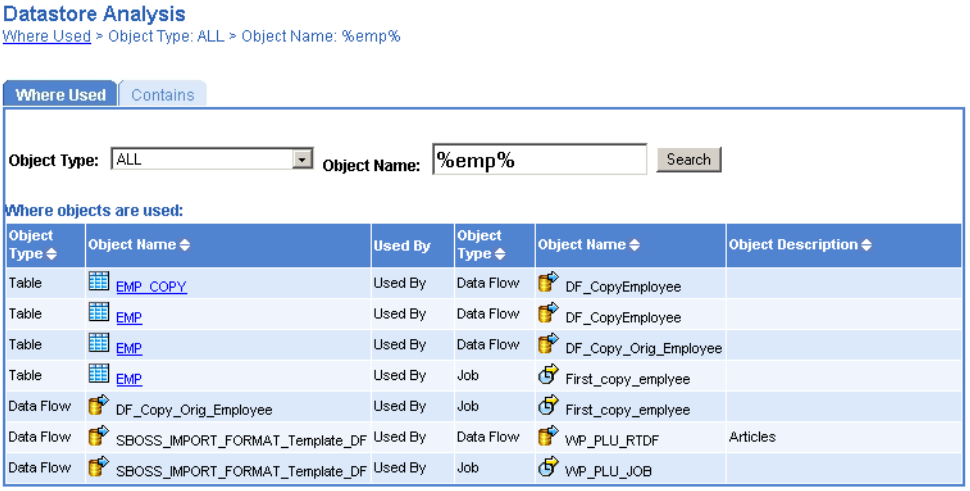
➢ To generate a Where Used report

1. Choose an object from the **Object Type** list, or select **ALL**.
2. Specify an **Object Name**.

   You can enter a subset of characters or the full name of the object you want to find. For example, you can search for "Work flows" that contain the character string "SALES_2000," or search for "Data flows" that contain the character "%."
3. Click **Search**.
4. Data Integrator displays search results in the following format:

   <*Object Type*> <*Object Name*> Used By <*Object Type*> <*Object Name*>

The first Object Type and Object Name columns represent the object for which you are searching, and the second Object Type and Object Name columns represent an object that uses your searched-for object.



Contains

Operating opposite of the Where Used report, this search tool and associated report finds all objects used by (or contained within) a specified object within your repository. You select the scope of the report using Object Type and Object Name filters.

➢ To generate a Contains report

1. Choose an object from the **Object Type** list, or select **ALL**.
2. Specify an **Object Name**.

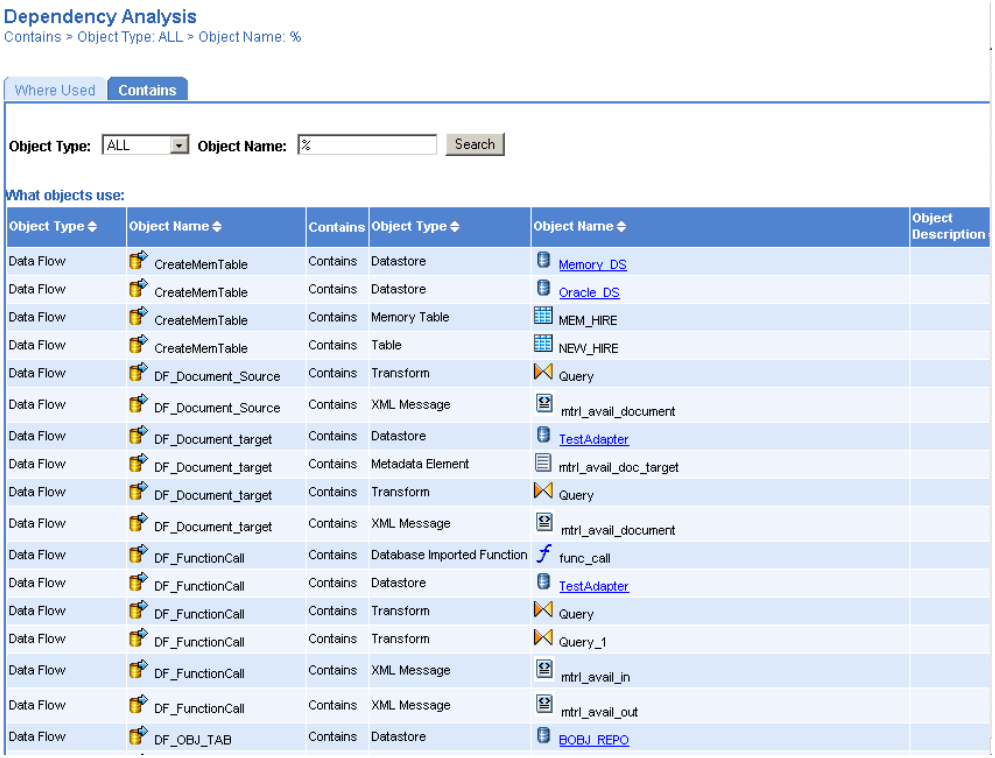
   You can enter a subset of characters or the full name of the object you want to find. For example, you can search for "Work flows" that contain the character string "SALES_2000," or search for "Data flows" that contain the character "%."
3. Click **Search**.

4. Data Integrator displays search results in the following format:

   <*Object Type*> <*Object Name*> Contains <*Object Type*> <*Object Name*>

   The first Object Type and Object Name columns represent the searched-for object, and the second Object Type and Object Name columns represent an object contained within (used by) your searched-for object.

   NOTE: To support reporting navigation, Data Integrator provides links for all datastores, functions, and tables shown in your Contains report.



Dependency Analysis

Contains > Object Type: ALL > Object Name: %

Where Used | Contains

Object Type: ALL Object Name: % Search

What objects use:

Object Type	Object Name	Contains	Object Type	Object Name	Object Description
Data Flow	CreateMemTable	Contains	Datastore	Memory_DS	
Data Flow	CreateMemTable	Contains	Datastore	Oracle_DS	
Data Flow	CreateMemTable	Contains	Memory Table	MEM_HIRE	
Data Flow	CreateMemTable	Contains	Table	NEW_HIRE	
Data Flow	DF_Document_Source	Contains	Transform	Query	
Data Flow	DF_Document_Source	Contains	XML Message	mtrl_avail_document	
Data Flow	DF_Document_target	Contains	Datastore	TestAdapter	
Data Flow	DF_Document_target	Contains	Metadata Element	mtrl_avail_doc_target	
Data Flow	DF_Document_target	Contains	Transform	Query	
Data Flow	DF_Document_target	Contains	XML Message	mtrl_avail_document	
Data Flow	DF_FunctionCall	Contains	Database Imported Function	func_call	
Data Flow	DF_FunctionCall	Contains	Datastore	TestAdapter	
Data Flow	DF_FunctionCall	Contains	Transform	Query	
Data Flow	DF_FunctionCall	Contains	Transform	Query_1	
Data Flow	DF_FunctionCall	Contains	XML Message	mtrl_avail_in	
Data Flow	DF_FunctionCall	Contains	XML Message	mtrl_avail_out	
Data Flow	DF_OBJ_TAB	Contains	Datastore	BOBJ_REPO	

Tools

This category provides tools that help you obtain the specific report information you need. Metadata analysis tools include:

- Business Objects Connections—to configure and manage connections to Business Objects Universes
- Refresh report content—to update repository information and update column mappings

Business Objects Connections

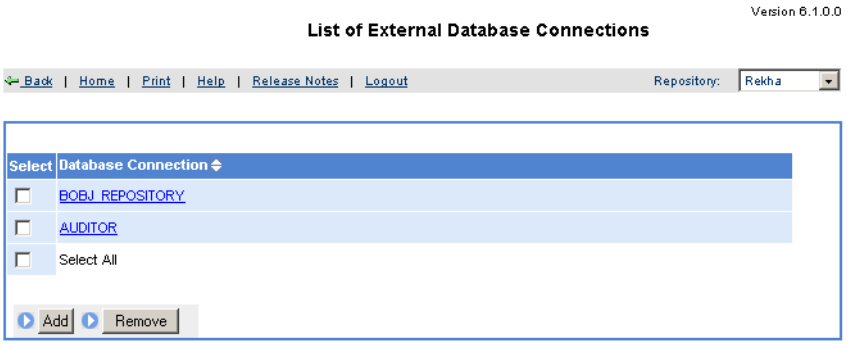
From this node, you manage your list of configured Business Objects database connections.

- Click a repository connection link to view database connection details (such as Business Objects repository).

  Each repository connection you create can contain several logical data groupings. Within each grouping, you will have several report options based on the external product to which you connect. After you define a connection, each grouping appears as a node under Universe analysis.

- To add a new repository connection, click **Add**, complete the Add/Edit page, and click **Apply**.

- To delete one or more repository connections from the list, mark the **Select** box next to each connection you want to delete and click **Remove**.



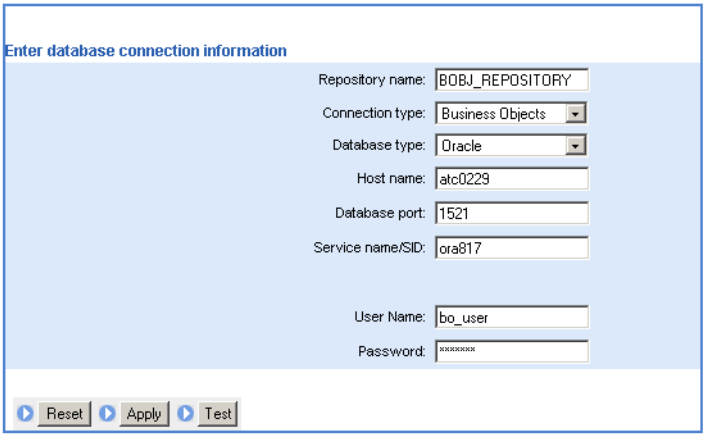
➢ To configure a database connection

1. In the Metadata Reports navigation tree, go to **Tools** > **Business Objects Connections**.

   NOTE: Only one Business Objects repository or auditor database at a time is supported. So, if you define multiple repositories or databases, only the first one will be used.

2. Click **Add** to view the Add/Edit screen.
3. Enter Repository type, Database type, Host name, Database port, Datasource, User name, and Password information.
   - Repository type can be either **BusinessObjects** (Business Objects repository) or **Auditor** (Business Objects Auditor database).

- All other configuration information is associated with the database to which you are connecting.



4. Click **Test** to verify that your connection information works.
5. If the test is successful, click **Apply**.

   If the test was unsuccessful, check to see that you have the correct connection information.

Refresh report content

You can calculate usage dependencies and column mappings from the Metadata Reports application or from the Designer at any time.

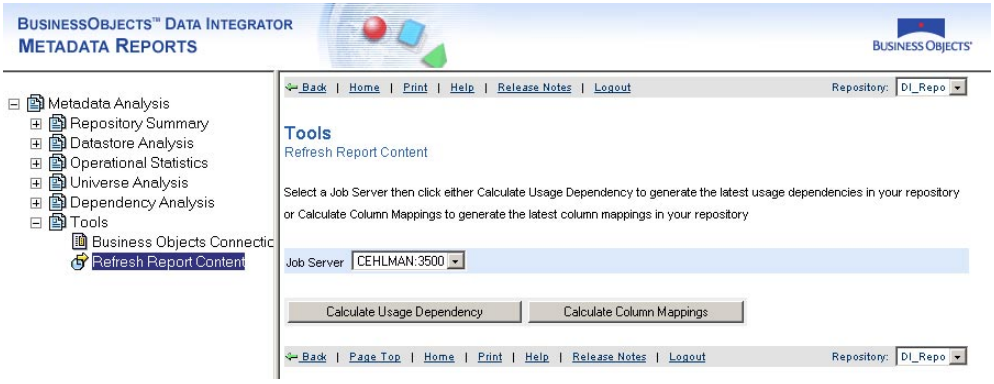
The **Calculate Usage Dependency** option populates the internal AL_USAGE table and ALVW_PARENT_CHILD view which provides current data to metadata reports.

The date and time of the last dependency calculation displays at the top of the metadata reports window. Dates and times are different from repository to repository depending on the last usage dependency calculation.

The **Calculate Column Mappings** option populates the internal AL_COLMAP table and ALVW_MAPPING view which provides current data to metadata reports.

Alternatively, you can calculate column mappings automatically when saving data flows in the Designer. Select **Tools** > **Options** > **Designer** > **General** > **Calculate column mappings when saving data flows**.

NOTE: If you do not select **Calculate column mappings when saving data flows**, you must select **Calculate Column Mappings** manually before generating reports to ensure that Data Integrator returns accurate mapping information. For example, assume you do not have **Calculate column mappings when saving data flows** selected. You click the **Calculate Column Mappings** button and you are able to view your column mappings. So far, everything is fine. Now, you unintentionally move an object in a data flow and when you try to execute the job Data Integrator prompts you to save the changes to the data flow. You comply. The next time you attempt to view mapping information for the data flow, the information is gone because the saved data flow is a new object in the repository. It does not have column mapping information until you manually calculate it.



➢ To calculate usage dependencies from the Metadata Reports application

1. In the Designer, open the Metadata Reports application by selecting **Tools** > **Metadata Reports**.
2. Select **Metadata Analysis** > **Tools** > **Calculate Usage Dependencies**.
3. Select the Job Server that is associated with the repository from which you want to get report data.
4. Click **Calculate usage dependency**.

   Data Integrator flattens the hierarchy of your selected repository, essentially caching the flattened hierarchy as a "data snapshot" table for use with the metadata reporting system. Each time you calculate usage dependencies, Data Integrator updates this table to ensure fast, accurate metadata reporting without compromising Data Integrator performance.

➢ To calculate usage dependencies from the Designer

Right-click in the object library of the current repository and select **Repository** > **Calculate Usage Dependencies**.

➢ To calculate column mappings in the Metadata Reports application

1. In the Designer, open the Metadata Reports application by selecting **Tools** > **Metadata Reports**.
2. Select **Metadata Analysis** > **Tools** > **Calculate Column Mappings**.
3. Select the Job Server that is associated with the repository you want to use.
4. Click **Calculate Column Mappings**.

➢ To calculate column mappings from the Designer

Right-click the object library and select **Repository** > **Calculate column mappings,** or

Select **Tools** > **Options** > **Designer** > **General** > **Calculate column mappings when saving data flows**.

18 Recovery Mechanisms

Recovery mechanisms are available in Data Integrator for batch jobs only.

This chapter contains the following sections:

Recovering from unsuccessful job execution

If an Data Integrator job does not complete properly, you must fix the problems that prevented the successful execution of the job and run the job again.

However, during the failed job execution, some data flows in the job may have completed and some tables may have been loaded, partially loaded, or altered. Therefore, you need to design your data movement jobs so that you can recover—that is, rerun the job and retrieve all the data without duplicate or missing data.

You can use various techniques to recover from unsuccessful job executions. This section discusses two techniques:

- **Automatically recovering jobs** — A Data Integrator feature that allows you to run unsuccessful jobs in recovery mode.
- **Manually recovering jobs using status tables** — A design technique that allows you to rerun jobs without regard to partial results in a previous run.

You might need to use a combination of these techniques depending on the relationships between data flows in your application.

If you do not use these techniques, you might need to roll back changes manually from target tables if interruptions occur during job execution.

Automatically recovering jobs

With automatic recovery, Data Integrator records the result of each successfully completed step in a job. If a job fails, you can choose to run the job again in recovery mode. During recovery mode, Data Integrator retrieves the results for successfully completed steps and reruns uncompleted or failed steps under the same conditions as the original job. For recovery purposes, Data Integrator considers steps that raise exceptions as failed steps, even if the step is caught in a try/catch block.

Enabling automated recovery

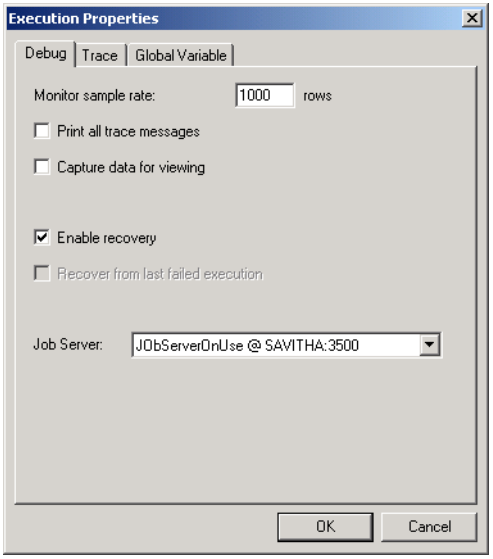
To use the automatic recover feature, you must enable the feature during initial execution of a job. Data Integrator saves the results from successfully completed steps when the automatic recovery feature is enabled.

➢ To run a job from Designer with recovery enabled

1. In the project area, select the job name.
2. Right-click and choose **Execute**.

   Data Integrator prompts you to save any changes.

3. Make sure that the **Enable Recovery** check box is selected on the Execution Properties window.



If this check box is not selected, Data Integrator does not record the results from the steps during the job and cannot recover the job if it fails. In that case, you must perform any recovery operations manually.

➢ To run a job with recovery enabled from the Administrator

When you schedule or execute a job from the Administrator, select the **Enable Recovery** check box. See the *Data Integrator Administrator Guide*.

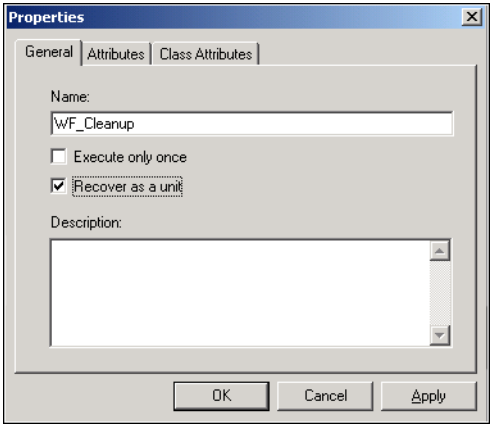
Marking recovery units

In some cases, steps in a work flow depend on each other and must be executed together. Because of the dependency, you should designate the work flow as a "recovery unit." When a work flow is a recovery unit, the entire work flow must complete successfully. If the work flow does not complete successfully, Data Integrator executes the entire work flow during recovery, even steps that executed successfully in prior work flow runs.

However, there are some exceptions to recovery unit processing. For example, when you specify that a work flow or a data flow should only execute once, a job will never re-execute that work flow or data flow after it completes successfully, even if that work flow or data flow is contained within a recovery unit work flow that re-executes. Therefore, Business Objects recommends that you not mark a work flow or data flow as **Execute only once** when the work flow or a parent work flow is a recovery unit.

➢ To specify a work flow as a recovery unit

1. In the project area, select the work flow.
2. Right-click and choose **Properties**.
3. Select the **Recover as a unit** check box, then click **OK**.



During recovery, Data Integrator considers this work flow a unit. If the entire work flow completes successfully—that is, without an error—during a previous execution, then Data Integrator retrieves the results from the previous execution. If any step in the work flow did not complete successfully, then the entire work flow re-executes during recovery.

On the workspace diagram, the black "**x**" and green arrow symbol indicate that a work flow is a recovery unit.

Running in recovery mode

If a job with automated recovery enabled fails during execution, you can re-execute the job in recovery mode.

As with any job execution failure, you need to determine and remove the cause of the failure and rerun the job in recovery mode. If you need to make any changes to the job itself to correct the failure, you cannot use automatic recovery but must run the job as if it is a first run.

In recovery mode, Data Integrator executes the steps or recovery units that did not complete successfully in a previous execution—this includes steps that failed and steps that threw an exception but completed successfully such as those in a try/catch block. As in normal job execution, Data Integrator executes the steps in parallel if they are not connected in the work flow diagrams and in serial if they are connected.

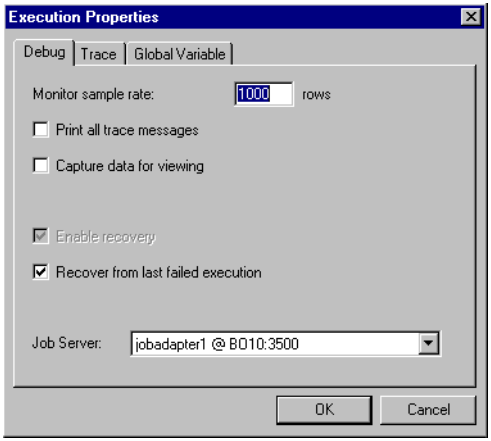
➢ To run a job in recovery mode from Designer

1. In the project area, select the (failed) job name.
2. Right-click and choose **Execute**.

   Data Integrator prompts you to save any objects that have unsaved changes.

3. Make sure that the **Recover from last failed execution** check box is selected.

   This option is not available when a job has not yet been executed, when the previous run succeeded, or when recovery mode was disabled during previous run.



   When you select **Recover from last failed execution**, Data Integrator retrieves the results from any steps that were previously executed successfully and executes or re-executes any other steps.

   If you clear this option, Data Integrator runs this job anew, performing all steps.

When you schedule or execute a (failed) job from the Administrator, select the **Recover from last failed execution** check box. See the *Data Integrator Administrator Guide*.

Ensuring proper execution path

The automated recovery system requires that a job in recovery mode runs again *exactly* as it ran previously.

It is important that the recovery job run exactly as the previous run. If the job was allowed to run under changed conditions—suppose a *sysdate* function returns a new date to control what data is extracted—then the new data loaded into the targets will no longer match data successfully loaded into the target during the first execution of the job.

For example, suppose a daily update job running overnight successfully loads dimension tables in a warehouse. However, while the job is running, the database log overflows and stops the job from loading fact tables. The next day, the administrator truncates the log file and runs the job again in recovery mode. The recovery job does not reload the dimension tables because the original, failed run successfully loaded them.

To ensure that the fact tables are loaded with the data that corresponds properly to the data already loaded in the dimension tables, the recovery job must use the same extraction criteria that the original job used when loading the dimension tables. If the recovery job used new extraction criteria—such as basing data extraction on the current system date—the data in the fact tables would not correspond to the data previously extracted into the dimension tables.

In addition, if the recovery job uses new values, then the job execution may follow a completely different path through conditional steps or try/catch blocks.

To ensure that the recovery job follows the exact execution path that the original job followed, Data Integrator records any external inputs to the job—return values for *systime* and *sysdate*, results from scripts, and so forth—and the recovery job uses the stored values.

When recovery is enabled, Data Integrator stores results from the following types of steps:

- Work flows
- Batch data flows

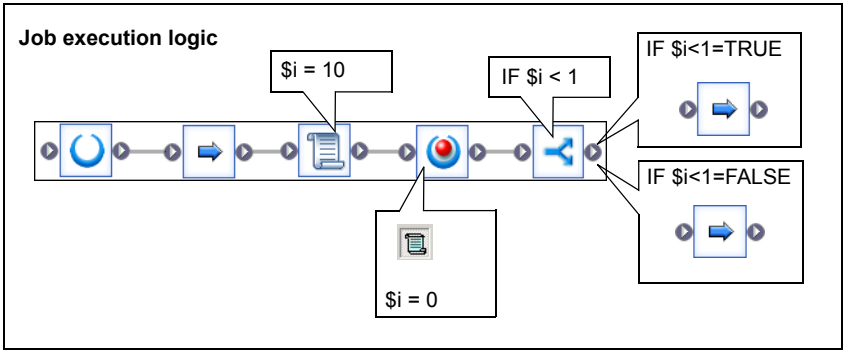
- Script statements
- Custom functions (stateless type only)
- SQL function
- exec function
- get_env function
- rand function
- sysdate function
- systime function

Using try/catch blocks with automatic recovery

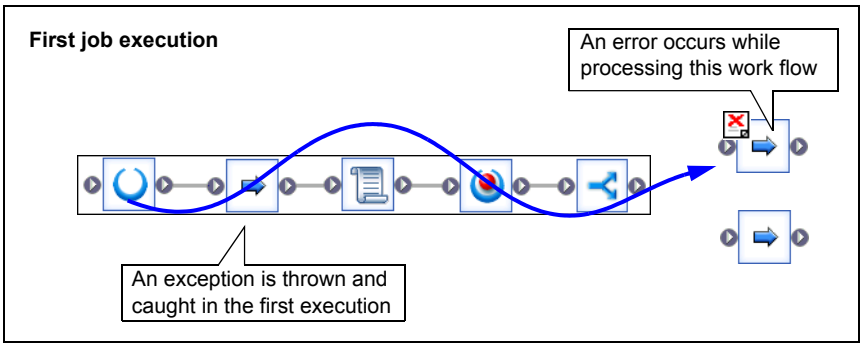
Data Integrator does not save the result of a try/catch block for reuse during recovery. If an exception is thrown inside a try/catch block, then during recovery Data Integrator executes the step that threw the exception and subsequent steps.

Because the execution path through the try/catch block might be different in the recovered job, using variables set in the try/catch block could alter the results during automatic recovery.

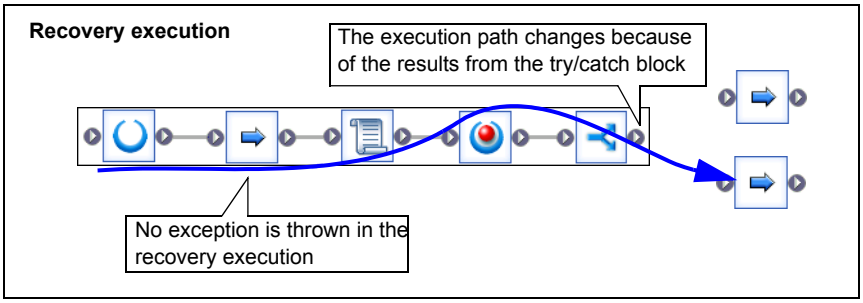
For example, suppose you create a job that defines a variable, `$i`, that you set within a try/catch block. If an exception occurs, you set an alternate value for `$i`. Subsequent steps are based on the value of `$i`.



During the first job execution, the first work flow contains an error that throws an exception, which is caught. However, the job fails in the subsequent work flow.



You fix the error and run the job in recovery mode. During the recovery execution, the first work flow no longer throws the exception. Thus the value of the variable, *$i*, is different, and the job selects a different subsequent work flow, producing different results.



To ensure proper results with automatic recovery when a job contains a try/catch block, do not use values set inside the try/catch block in any subsequent steps.

Ensuring that data is not duplicated in targets

Define work flows to allow jobs correct recovery. A data flow might be partially completed during an incomplete run. As a result, only some of the required rows could be inserted in a table. You do not want to insert duplicate rows during recovery when the data flow re-executes.

You can use several methods to ensure that you do not insert duplicate rows:

- Design the data flow to completely replace the target table during each execution

  This technique can be optimal when the changes to the target table are numerous compared to the size of the table. You can use tuning techniques such as bulk loading options to improve overall performance.

- Set the auto correct load option for the target table

  The auto correct load option checks the target table for existing rows before adding new rows to the table. Using the auto correct load option, however, can needlessly slow jobs executed in non-recovery mode.

  Consider this technique when the target table is large and the changes to the table are relatively few.

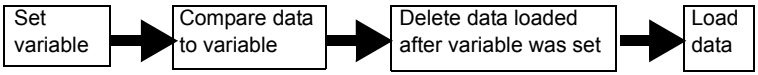
- Include a SQL command to execute before the table loads

  Preload SQL commands can remove partial database updates that occur during incomplete execution of a step in a job. Typically, the preload SQL command deletes rows based on a variable that is set before the partial insertion step began.

Using preload SQL to allow re-executable data flows

To use preload SQL commands to remove partial database updates, tables must contain a field that allows you to tell when a row was inserted. Create a preload SQL command that deletes rows based on the value in that field.

For example, suppose a table contains a column that records the time stamp of any row insertion. You can create a script with a variable that records the current time stamp before any new rows are inserted. In the target table options, add a preload SQL command that deletes any rows with a time-date stamp greater than that recorded by the variable.



During initial execution, no rows match the deletion criteria. During recovery, the variable value is not reset. (The variable value is set in a script, which is executed successfully during the initial run.) The rows inserted during the previous, partial database load would match this criteria, and the preload SQL command would delete them.

To use preload SQL commands properly, you must define variables and pass them to data flows correctly.

➢ To use preload SQL commands to ensure proper recovery

1. Determine appropriate values that you can use to track records inserted in your tables.

   For example, if each row in a table is marked with the insertion time stamp, then you can use the value from the *sysdate()* function to determine when a row was added to that table.

2. Create variables that can store the "tracking" values.

   Variables are either job or work-flow specific. If a work flow is a recovery unit, create the "tracking" variables for that work flow at the job level; otherwise, create your tracking variables at the work flow level. Generally, you do not want tracking variables reset during recovery because when they reset, the preload SQL command will not work properly.

   For information about creating a variable, see "Defining local variables" on page 298.

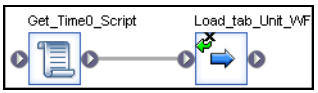
3. Create scripts that set the variables to the appropriate values.

   Scripts are unique steps in jobs or work flows. You need to create a separate script that sets the required variables before each data flow or work flow that loads a table. If a work flow is a recovery unit, create the scripts for that work flow at the job level; otherwise, create the scripts at the work flow level.

   For information about creating scripts, see "Scripts" on page 201.

4. Connect the scripts to the corresponding data flows or work flows.

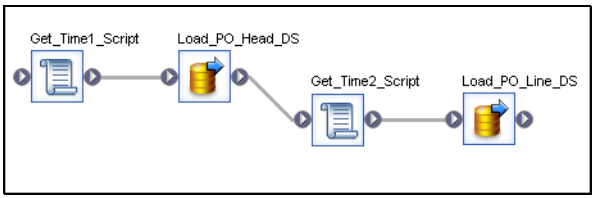
Job



Work Flow
(A recovery unit)

When a work flow is a recovery unit, create scripts that set tracking variables outside the work flow, typically at the job level. Connect the script to the work flow.

Work Flow (Not a recovery unit)



When a work flow is not a recovery unit, create scripts that set tracking variables inside the work flow before the data flow that requires the value. Connect the script directly to the appropriate data flow.

5. Create parameters to pass the variable information from the job or work flow where you created the variable to the data flow that uses the tracking variable in the preload SQL command.

   For information about creating variables, see "Defining parameters" on page 298.

6. Insert appropriate preload SQL commands that remove any records inserted during earlier unsuccessful runs.

   The preload SQL commands reference the parameter containing the tracking variable, deleting rows that were inserted after the variable was set.

   For example, suppose the PO_ITEM table records the date-time stamp in the TIMESTMP column. You created a variable *$load_time* that records the value from the *sysdate()* function before the load starts, and you passed that variable to the data flow that loads the PO_ITEM table in a parameter named *$load_time*. Then, your preload SQL command must delete any records in the table where the value in TIMESTMP is larger than the value in *$load_time*.

   ```
   delete from PO_ITEM where TIMESTMP >
   [$load_time]
   ```

   For information about creating preload SQL commands, see "Pre Load Commands" on page 112 of the *Data Integrator Reference Guide*.

Manually recovering jobs using status tables

You can design your jobs and work flows so that you can manually recover from an unsuccessful run. A job designed for manual recovery must have certain characteristics:

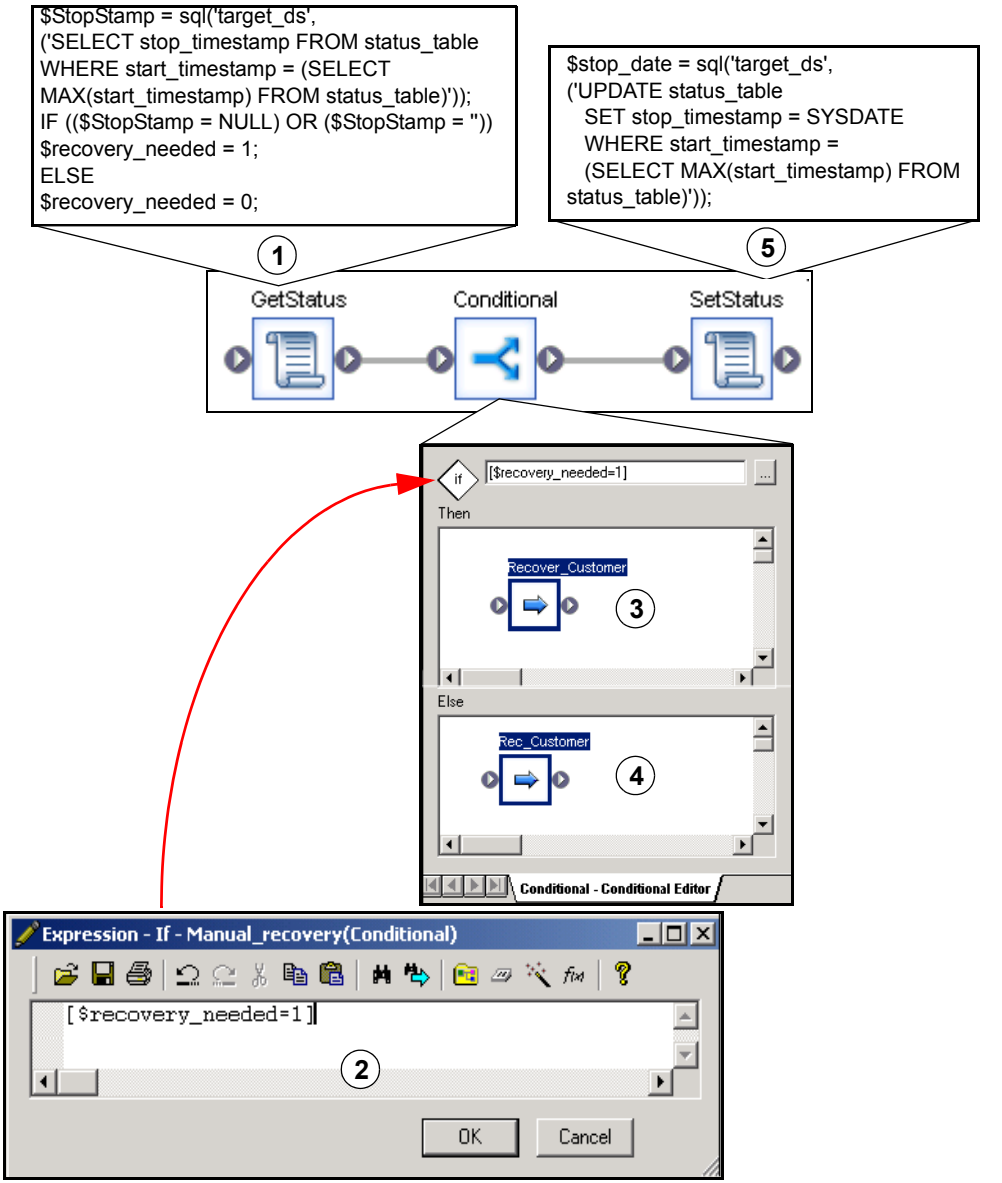
- You can run the job repeatedly.
- The job implements special steps to recover data when a step did not complete successfully during a previous run.

You can use an execution status table to produce jobs that can be run multiple times without duplicating target rows. The table records a job's execution status. A "failure" value signals Data Integrator to take a recovery execution path.

To implement a work flow with a recovery execution path:

- Define a flag that indicates when the work flow is running in recovery mode.
- Store the flag value in a status table.
- Check the flag value in the status table before executing a work flow to determine which path to execute in the work flow.
- Update the flag value when the work flow executes successfully.

For example, you could design a work flow that uses the auto correct load option when a previous run does not complete successfully. This work flow would have five steps, as illustrated:



1. Retrieve the flag value, which indicates the success or failure of the previous execution, from the status table. Store this value in a variable such as *$recovery_needed*.
2. In a conditional, evaluate the *$recovery_needed* variable.
3. If recovery is required, execute the recovery data flow recover_customer. This data flow loads the data using the auto correct load option. For more information about the auto correct load option, see "Target" on page 103 of the *Data Integrator Reference Guide*.
4. If recovery is not required, execute the non-recovery data flow load_customer. This data flow loads the data without the auto correct load option.
5. Update the flag value in the status table to indicate successful execution.

Processing data with problems

Jobs might not produce the results you expect because of problems with data. In some cases, Data Integrator is unable to insert a row. In other cases, Data Integrator might insert rows with missing information. You can design your data flows to anticipate and process these types of problems. For example, you might have a data flow write rows with missing information to a special file that you can inspect later.

This section describes mechanisms you can use to anticipate and process data problems. In particular, this section discusses three techniques:

- Using overflow files
- Filtering missing or bad values
- Handling facts with missing dimensions

Using overflow files

A row that cannot be inserted is a common data problem. Use the *overflow file* to process this type of data problem. When you specify an overflow file and Data Integrator cannot load a row into a table, Data Integrator writes the row to the overflow file instead. The trace log indicates the data flow in which the load failed and the location of the file.

For any table used as a target, you can set the option to use an overflow file in the **Options** tab. When you specify an overflow file, give a full path name to ensure that Data Integrator creates a unique file when more than one file is created in the same job. By default, the name of the overflow file is the target table name.

When you select the overflow file option, you choose what Data Integrator writes to the file about the rows that failed to load: either the data from the row or the SQL commands required to load the row. If you select data, you can use Data Integrator to read the data from the overflow file, cleanse it, and load it into the target table. If you select SQL commands, you can use the commands to load the target manually when the target is accessible.

There are many reasons for loading to fail, for example:

- Out of memory for the target
- Overflow column settings
- Duplicate key values

You can use the overflow information to identify invalid data in your source or problems introduced in the data movement. Every new run will overwrite the existing overflow file.

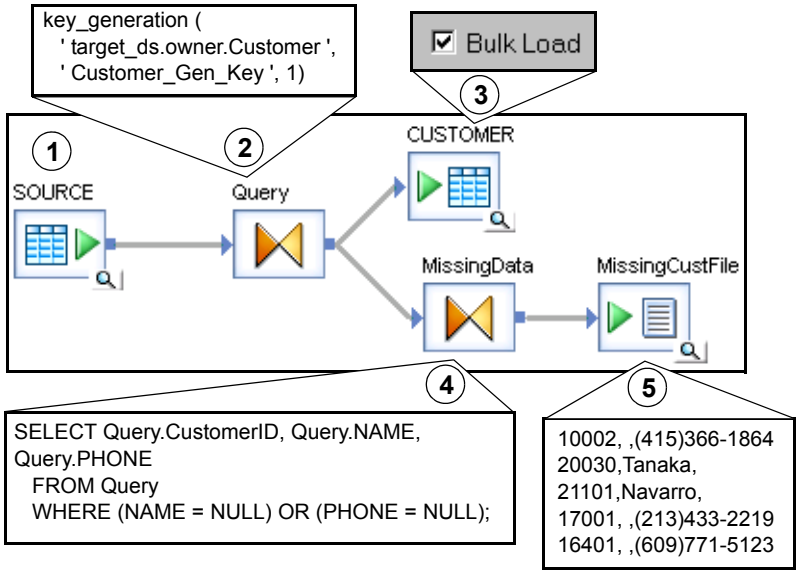
NOTE: You cannot use overflow files when loading to a BW Transfer Structure.

Filtering missing or bad values

A missing or invalid value in the source data is another common data problem. Using queries in data flows, you can identify missing or invalid values in source data. You can also choose to include this data in the target or to disregard it.

For example, suppose you are extracting data from a source and you know that some phone numbers and customer names are missing. You can use a data flow to extract data from the source, load the data into a target, and filter the NULL values into a file for your inspection.

This data flow has five steps, as illustrated.

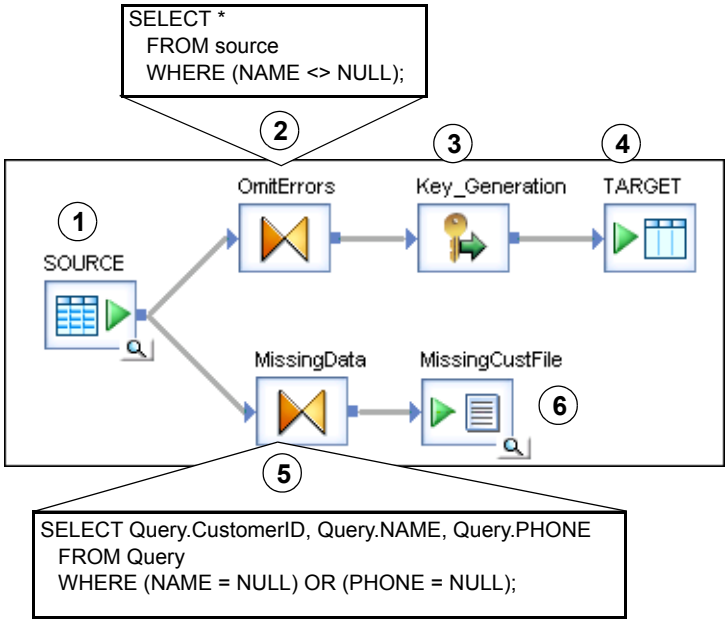


The data flow:

1. Extracts data from the source
2. Selects the data set to load into the target and applies new keys. (It does this by using the Key_Generation function.)
3. Loads the data set into the target, using the bulk load option for best performance
4. Uses the same data set for which new keys were generated in step 2, and select rows with missing customer names and phone numbers
5. Writes the customer IDs for the rows with missing data to a file

Now, suppose you do not want to load rows with missing customer names into your target. You can insert another query into the data flow to ensure that Data Integrator does not insert incomplete rows into the target. The new query filters the rows with missing customer names before loading any rows into the target. The missing data query still collects those rows along with the rows containing missing phone numbers. In this version of the example, the Key_Generation transform adds keys for new rows before inserting the filtered data set into the target.

The data flow now has six steps, as shown.



1. Extracts data from the source
2. Selects the data set to load into the target by filtering out rows with no customer name values
3. Generates keys for rows with customer names
4. Loads the valid data set (rows with customer names) into the target using the bulk load option for best performance

5. Uses a separate query transform to select rows from the source that have no names or phones

   Note that Data Integrator does not load rows with missing customer names into the target; however, Data Integrator does load rows with missing phone numbers.

6. Writes the customer IDs for the rows with missing data to a file.

You could add more queries into the data flow to select additional missing or invalid values for later inspection.

Handling facts with missing dimensions

Another data problem occurs when Data Integrator searches a dimension table and cannot find the values required to complete a fact table.

You can approach this problem in several ways:

- Leave the problem row out of the fact table.

  Typically, this is not a good idea because analysis done on the facts will be missing the contribution from this row.

- Note the row that generated the error, but load the row into the target table anyway.

  You can mark the row as having an error, or pass the row information to an error file as in the examples from "Filtering missing or bad values" on page 479.

- Fix the problem programmatically.

  Depending on the data missing, you can insert a new row in the dimension table, add information from a secondary source, or use some other method of providing data outside of the normal, high-performance path.

19 Techniques for Capturing Changed Data

This chapter contains the following sections:

Understanding changed-data capture

When you have a large amount of data to update regularly and a small amount of system down time for scheduled maintenance on a data warehouse, update data over time, or *delta load*. Two commonly used delta load methods are full refresh and changed-data capture (CDC).

Full refresh

Full refresh is easy to implement and easy to manage. This method ensures that no data will be overlooked or left out due to technical or programming errors. For an environment with a manageable amount of source data, full refresh is an easy method you can use to perform a delta load to a target system.

Capturing only changes

After an initial load is complete, you can choose to extract only new or modified data and update the target system. Identifying and loading only changed data is called *changed-data capture (CDC)*. This includes only incremental data that has changed since the last refresh cycle. Data Integrator acts as a mechanism to locate and extract only the incremental data that changed since the last refresh.

Improving performance and preserving history are the most important reasons for using changed-data capture.

- Performance improves because with less data to extract, transform, and load, the job typically takes less time.
- If the target system has to track the history of changes so that data can be correctly analyzed over time, the changed-data capture method can provide a record of these changes.

For example, if a customer moves from one sales region to another, simply updating the customer record to reflect the new region negatively affects any analysis by region over time because the purchases made by that customer before the move are attributed to the new region.

This chapter discusses both general concepts and specific procedures for performing changed-data capture in Data Integrator.

Source-based and target-based CDC

Changed-data capture can be either source-based or target-based.

Source-based CDC

Source-based changed-data capture extracts only the changed rows from the source. It is sometimes called *incremental extraction*.

This method is preferred for the performance gain of extracting the least rows.

Data Integrator offers three methods of source-based change data capture:

- For Oracle 9i, Data Integrator uses Oracle's CDC packages to create and manage CDC tables. These packages make use of a publish/subscribe model. You can create a CDC datastore for Oracle sources using Data Integrator Designer. You can also use the Designer to create CDC tables in Oracle then import them for use in Data Integrator jobs.
- For mainframes using DETAIL to connect to Data Integrator, you can use Striva DETAIL Change to track database logs for changed data. Then, create a CDC datastore, import changed-data tables, and create jobs.

- For other sources, you can use date and time fields to compare source-based changed-data capture job runs. This technique makes use of a creation and/or modification timestamp on every row. You can compare rows using the time of the last update as a reference. This method is called *timestamp-based CDC*.

Target-based CDC

Target-based changed-data capture extracts all the data from the source, but loads only the changed rows into the target.

Target-based changed-data capture is useful when you want to capture history but do not have the option to use source-based changed-data capture. Data Integrator offers *table comparison* to support this method.

System guidelines for source-based CDC

For Oracle (version 9i only), use the following system requirements on your Oracle source database server to track changes:

- Install Oracle's synchronous CDC packages. These packages are installed by default. However, if a CDC package needs to be re-installed, open Oracle's `Admin` directory, then find and run Oracle's SQL script `initcdc.sql`.
- Enable Oracle's system triggers.
- Enable Java.
- Set source table owner privileges so CDC tables can be created, purged, and dropped as needed.
- Give datastore owners the SELECT privilege for CDC tables and the SELECT_CATALOG_ROLE and EXECUTE_CATALOG_ROLE privileges.

Use mainframes accessed via Striva DETAIL Change to track changes using database logs. Configure this application before creating a CDC job in Data Integrator.

Use Timestamp-based CDC to track changes:

- If you are using sources other than Oracle 9i or mainframes accessed via Striva DETAIL and
- If the following conditions are true:
  - There are date and time fields in the tables being updated
  - You are updating a large table that has a small percentage of changes between extracts and an index on the date and time fields
  - You are not concerned about capturing intermediate results of each transaction between extracts (for example, if a customer changes regions twice in the same day).

Business Objects does not recommend using the Timestamp-based CDC when:

- You have a large table, a large percentage of it changes between extracts, and there is no index on the timestamps.
- You need to capture physical row deletes.
- You need to capture multiple events occurring on the same row between extracts.

Using CDC with Oracle sources

If your environment must keep large amounts of data current, the Oracle CDC feature is a simple solution to limiting the number or rows that Data Integrator reads on a regular basis. A source that reads only the most recent operations (INSERTS, UPDATES, DELETES), allows you to design smaller, faster delta loads.

With Oracle 9i, Data Integrator manages the CDC environment by accessing Oracle's CDC packages. Identify a source table then create a CDC table using these packages. An Oracle CDC table must reside on the same system as the original table.

Oracle's packages use the publish/subscribe model with its CDC tables. Oracle publishes changed data from the original table to its CDC table. After a CDC table receives published data, subscriptions can be created to access the data. Data Integrator Designer allows you to import CDC tables and create subscriptions for them. You can also enable check-points for subscriptions so that Data Integrator only reads the latest changes in the CDC table.

CDC datastores

To gain access to CDC tables, create a CDC datastore using the Designer. A CDC datastore is a read-only datastore that can only access tables. Like other datastores, you can create, edit, and access a CDC datastore from the **Datastores** tab of the object library.

➢ To create a CDC datastore for Oracle

1. Create a custom datastore with the **Database Type** option set to `Oracle`.
2. Select the **CDC** check box.
3. Select an **Oracle version**.

The Designer only allows you to select the Oracle versions that support CDC packages.

4. Complete the rest of the dialog and click **OK**.

   You can use this datastore to browse and import CDC tables.

Importing CDC data from Oracle

You must create a CDC table in Oracle for every source table you want to read from before you can import that CDC table using Data Integrator. There are two ways to achieve this:

- Use an Oracle utility to create CDC tables
- Use Data Integrator Designer to create CDC tables

➢ When CDC tables exist in Oracle

1. Import an Oracle CDC table by right-clicking the CDC datastore name in the object library and selecting **Open**, **Import by Name,** or **Search**.

   If you select **Open**, you can browse the datastore for existing CDC tables using the Datastore Explorer.

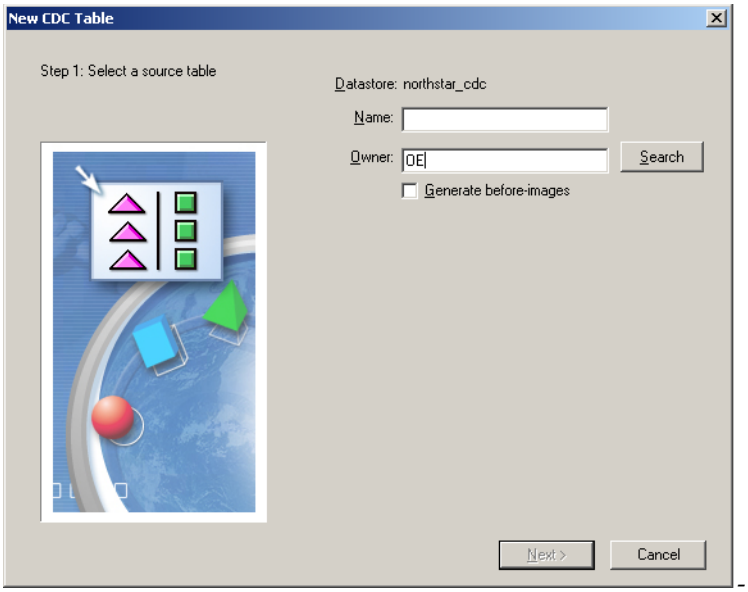
2. When you find the table that you want to import, right-click it and select **Import**.

➢ To create CDC tables in Oracle using the Designer

1. In the object library, right-click a CDC datastore and select **Open**.
2. In the Datastore Explorer, right-click the white space in the **External Metadata** section, and select **New**.

The New CDC table wizard opens. This wizard allows you to add a CDC table.

NOTE: If the Datastore Explorer opens and no CDC tables exist in your datastore, this wizard opens automatically.



3. Specify the source table information in the first page of the wizard.
   a. Click the **Search** button to see a list of non-CDC external tables available in this datastore.

      To filter a search, enter values for a table **Owner** and/or **Name**. You can use a wild-card character (%) to perform pattern matching for **Name** or **Owner** values.

   b. (Optional) Select **Generate before-images** if you want to track before- and after-images in the new CDC table. For more information about this option, see "Using before-images" on page 497.

c. Double-click the name in the list of returned tables (or click a name and click **Next)** to create a CDC table using the selected table as a source table.

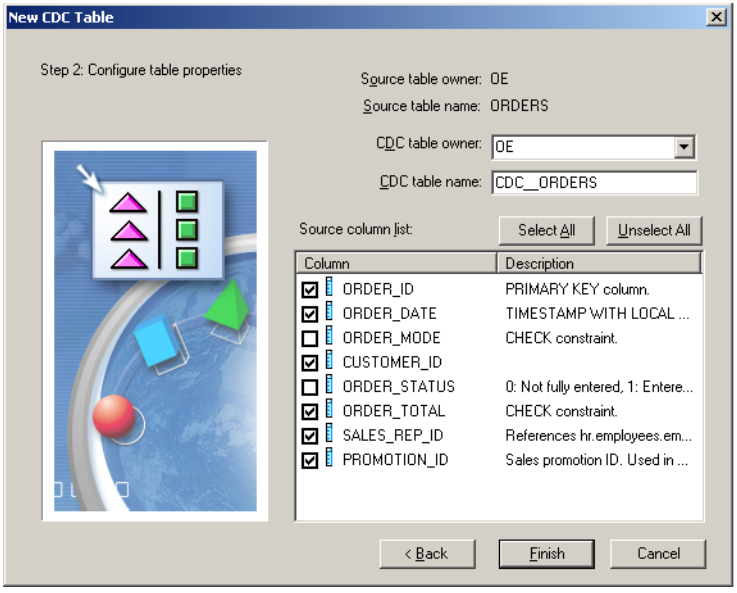
4. Specify the **CDC table owner** for the new CDC table.

   By default, the owner name of the new CDC table is the owner name of the datastore. The source table owner name is also displayed in the **CDC table owner** list box. If the owner name you want to use is not in the list, enter a different owner name.

5. Specify the **CDC table name** for the new CDC table.

   By default, Data Integrator generates a table name using the following convention: `CDC__SourceTableName`.

6. Specify which columns to include/exclude from the CDC table.



7. Click Finish.

The Designer connects to the Oracle instance, creates the CDC table on the Oracle server, and imports the table's metadata into Data Integrator's repository.

NOTE: All tables that Data Integrator imports through a CDC datastore contain a column that indicates which operation to perform for each row. For an Oracle CDC table, this column is called `Operation$`. In addition to this column, Oracle adds other columns when it creates a CDC table. These columns all use a dollar sign as a suffix.

8. Click **OK** on the information dialog.

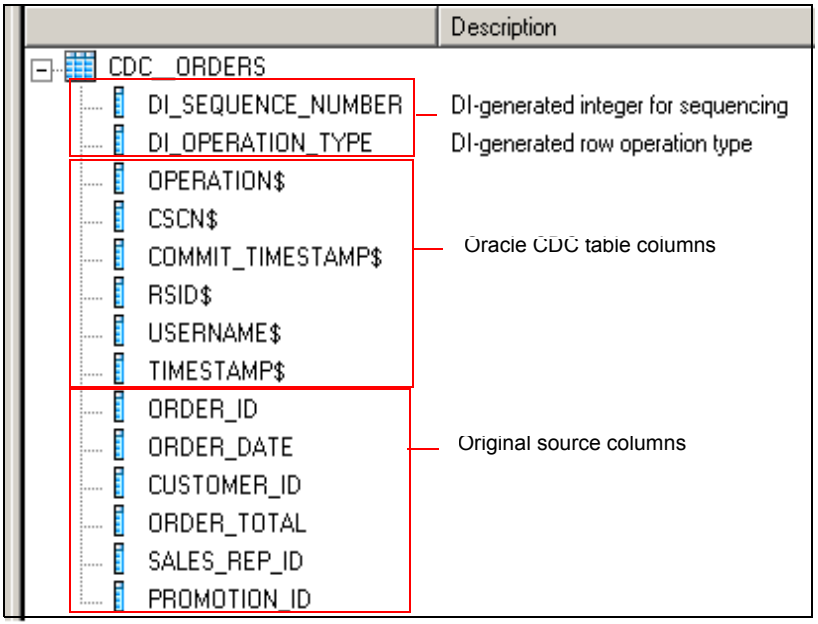
   This dialog confirms that Oracle created a new CDC table, then imported it successfully into Data Integrator.

➢ To view an imported CDC table

1. Find your CDC datastore in the object library.
2. Expand the **Tables** folder.
3. Double-click a table name or right-click and select **Open**.

When Data Integrator imports a CDC table, it also adds two columns to the table's schema: `DI_SEQUENCE_NUMBER` with the data type integer and `DI_OPERATION_TYPE` with the data type varchar(1).

An imported Oracle CDC table schema looks like the following:



This example has eight columns added to the original table:

- Two generated by Data Integrator
- Six supplied by Oracle

The DI_SEQUENCE_NUMBER column

The `DI_SEQUENCE_NUMBER` column starts with zero at the beginning of each extraction. This field increments by one each time Data Integrator reads a row except when it encounters a pair of before- and after-images for an UPDATE operation. Both the before- and after-images receive the same sequence number. This sequencing column provides a way to collate image pairs if they are separated as a result of the data flow design. For information about when to consider using before-images, see "Using before-images" on page 497.

The DI_OPERATION_TYPE column

The possible values for the `DI_OPERATION_TYPE` column are:

- `I` for INSERT
- `D` for DELETE
- `B` for before-image of an UPDATE
- `U` for after-image of an UPDATE

When Data Integrator reads rows from Oracle, it checks the values in column `Operation$` and translates them to Data Integrator values in the `DI_OPERATION_TYPE` column. The translation is as follows:

Operation$	DI_OPERATION_TYPE
I	I
D	D
UO, UU	B
UN	U

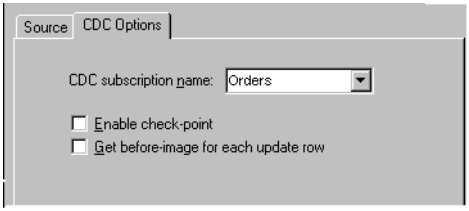
Configuring a CDC source

When you drag a CDC datastore table into a data flow, it automatically becomes a source object.

➢ To configure a CDC table

1. Drag a CDC datastore table into a data flow.
2. Click the name of this source object to open its Source Target Editor.

3. Click the **CDC Options** tab.



There are three CDC table options in the Source Target Editor's **CDC Options** tab:

Option Name	Description
CDC Subscription name	The name that marks a set of changed data in a continuously growing Oracle CDC table. Select from the list or create a new subscription. Subscriptions are created in Oracle and saved for each CDC table A subscription name is unique to a datastore, owner, and table name. For example, you can use the same subscription name (without conflict) with different tables in the same datastore if they have different owner names.
Enable check-point	Enables Data Integrator to restrict CDC subscription reads using check-points. Once a check-point is enabled, the next time the CDC job runs, it reads only the rows inserted into the CDC table since the last check-point. After a job completes successfully, Data Integrator moves the check-point forward to mark the last row read. Business Objects recommends that you enable check-pointing for a subscription name in a production environment. For more information, see "Using check-points".
Get before-image for each update row	Oracle allows a before-image and an after-image to be associated with an UPDATE row. If you want to read before-images (for a CDC table set to capture them), enable this option. By default, only after-images are retrieved. For more information, see "Using before-images" on page 497.

Using check-points

When a job in Data Integrator runs with check-pointing enabled, the Data Integrator uses the source table's subscription name to read the most recent set of appended rows. If you do not enable check-pointing, then the job reads all the rows in the table. This increases processing time.

To use check-points, on the Source Table Editor, enter a name in the **CDC Subscription name** box and select the **Enable check-point** option.

NOTE: To avoid data corruption problems, do not reuse data flows that use CDC datastores because each time a source table extracts data it uses the same subscription name. This means that identical jobs, depending upon when they run, can get different results and leave check-points in different locations in the table. When you migrate CDC jobs from test to production, for example, a best practice scenario would be to change the subscription name for the production job so that the test job, if ever runs again, will not affect the production job's results.

Using before-images

If you want to retrieve the before-images of UPDATE rows, prior to when the update operation is applied to the target, Data Integrator can expand the UPDATE row into two rows: one row for the *before-image* of the update, and one row for the *after-image* of the update. The before image of an update row is the image of the row before the row is changed, and the after image of an update row refers to the image of the row after the change is applied.

The default behavior is that a CDC reader retrieves after-images only. By not retrieving before-images, fewer rows pass through the engine which allows the job to execute in less time.

You can use before-images to:

- Update primary keys

However, under most circumstances, when source tables are updated, their primary keys do not need to be updated.

- Calculate change logic between data in columns

  For example, you can calculate the difference between an employee's new and old salary by looking at the difference between the values in salary fields.

When you want to capture before-images for update rows:

- At CDC table creation time, make sure the Oracle CDC table is also setup to retrieve full before-images. If you create an Oracle CDC table using Data Integrator Designer, you can select the **Generate before-images** check box to do this. For more information, see "To create CDC tables in Oracle using the Designer" on page 490.
- Select the **Get before-images for each update row** option in the CDC table's source editor.

  If the underlying, CDC table is not set-up properly, enabling the **Get before-images for each update row** option has no effect.

Once you select the **Get before-images for each update row** option, for every update, Data Integrator processes two rows. In addition to the performance impact of this data volume increase, the before- and after-image pairs may be separated or lost depending on the design of your data flow. This would cause data integrity issues.

The Map_CDC_Operation transform can resolve problems, but undesirable results may still occur due to programming errors. When using functions and transforms that re-order, re-direct, eliminate, and multiply the number of rows in a data flow (for example, due to the use of the group by or order by clauses in a query) be aware of the possible impact to targets.

The Map_CDC_Operation transform allows you to restore the original ordering of image pairs by using the DI_SEQUENCE_NUMBER column as its **Sequencing column**. For detailed information about the Map_CDC_Operation transform, see Chapter 5, "Transforms," in the *Data Integrator Reference Guide*.

Purging CDC tables

You need to purge CDC tables (truncate them) periodically so they do not grow indefinitely. This can be achieved by running the following program from your Oracle database utility (for example, SQL*Plus):

```
execute DBMS_LOGMNR_CDC_PUBLISH,PURGE();
```

Although this program affects all CDC tables in the system, it only purges data that is no longer needed by registered subscriptions.

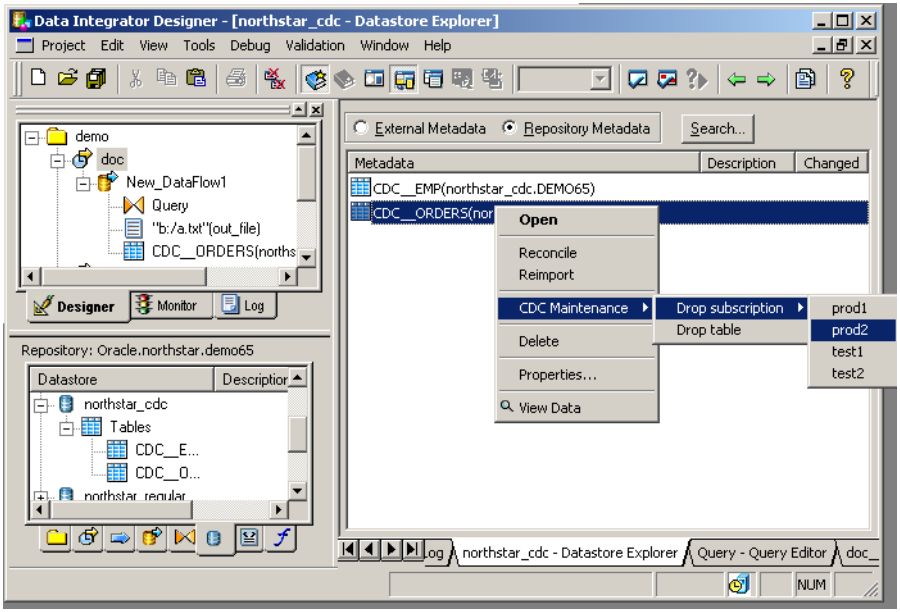
Dropping CDC subscriptions and tables

You can drop Oracle CDC tables and their subscriptions from the Datastore Explorer window in Data Integrator Designer.

➢ To drop Oracle CDC subscriptions or tables

1. From the object library, right-click a CDC datastore and select **Open**.
2. In the Datastore Explorer window, click **Repository Metadata**.

3. Right-click a table and select **CDC maintenance**.



4. Choose:

   - Drop Subscription

     This option opens the list of subscriptions you created in Data Integrator for the selected table. Oracle subscriptions are associated with these subscription names. Select each subscription name to drop it from Oracle and delete it from the Data Integrator repository.

   - Drop table

     This option drops the Oracle CDC table and also deletes it from the Data Integrator repository.

Limitations

- The table comparison transform and the `lookup` and `lookup_ext` functions cannot be used with a source table imported with a CDC datastore because of the existence of the Data Integrator generated columns for CDC tables. Data Integrator cannot compare or search these columns.
- Oracle CDC captures DML statements, including `INSERT`, `DELETE`, and `UPDATE`. However, Oracle CDC does not support the following operations because they disable all database triggers:
  - Direct-path `INSERT` statements
  - The `multi_table_insert` statement in parallel DML mode
- If you are using check-pointing and running your job in recovery mode, the recovered job will begin to review the job at the start of the CDC table. Check-points are ignored.

Using CDC with mainframe sources

If your environment must keep large amounts of data current, the mainframe CDC feature is a simple solution to limiting the number or rows that must be read on a regular basis. A source that reads only the most recent operations (INSERTS, UPDATES, DELETES) allows you to design smaller, faster delta loads.

Striva DETAIL Change

If you are extracting data from mainframe sources that are currently using Striva DETAIL to connect to Data Integrator, purchase a license for Striva's DETAIL Change to support changed-data capture.

Striva's DETAIL Change provides the facility to monitor a database table and provides a list of actions that are performed on the table in a separate log file. For example, if three rows are inserted into a database table, followed by two deletes, and one update, DETAIL Change appends six rows to a local CDC file. The first three rows are marked for INSERT, the next two rows are marked for DELETE, and the last row is marked for UPDATE. For configuration information, refer to Striva's documentation on DETAIL Change.

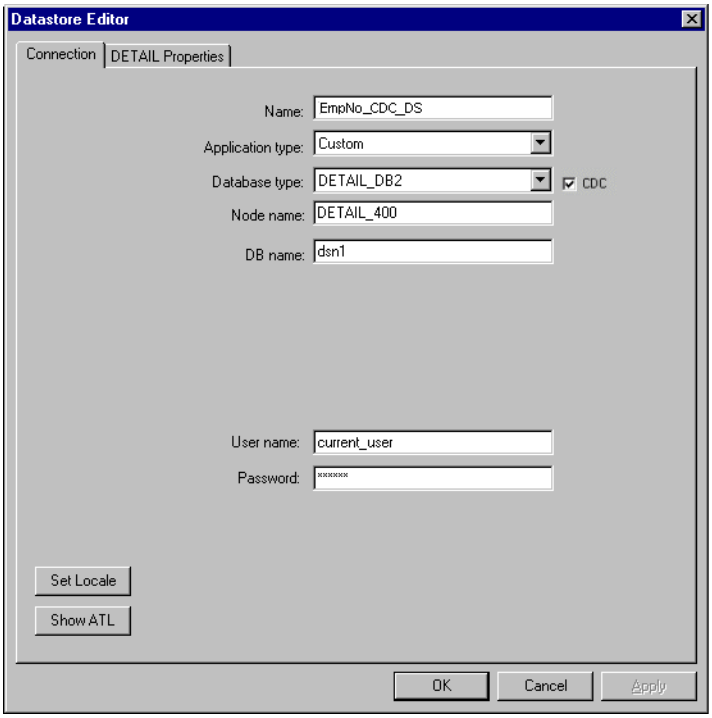
Once you configure Striva's DETAIL Change, you can create a batch job in Data Integrator to pick up changed data using a CDC datastore.

CDC datastores

The CDC datastore option is available for all mainframe interfaces to Data Integrator. Refer to "Mainframe Interfaces (DETAIL)" on page 80 of the *Data Integrator Designer Guide* for a list of mainframe data sources, the DETAIL client drivers used to connect to them, and an introduction to creating custom datastores.

➢ To create a CDC datastore for mainframe interfaces

1. Create a custom datastore with the **Database Type** option set to DETAIL_DB or DETAIL_NRDB.
2. Select the **CDC** check box.
3. Complete the rest of the dialog and click **OK**.



Once saved, this datastore becomes a CDC datastore. It can access data sources indicated by the client driver name selected in the **Database type** box.

Client Driver	Data Source
DETAIL_DB2	DB2 on OS/400 or MVS
DETAIL_NRDB	IMS, VSAM, ADABAS, IDMS

Importing mainframe CDC data

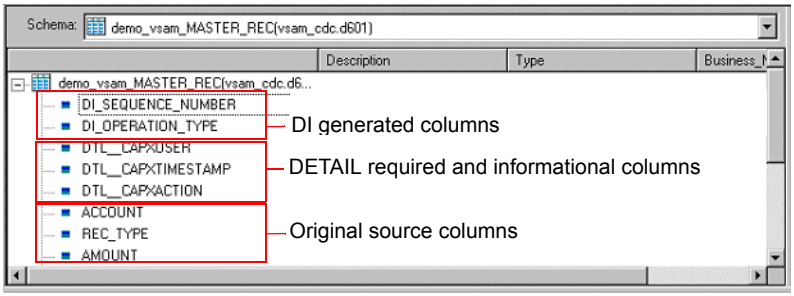
After you create a CDC datastore, you can use it to import CDC table metadata. In the object library, right-click the datastore name and select **Open**, **Import by Name,** or **Search**. For mainframe CDC, only the CDC tables that are registered with Striva DETAIL Change are visible when you browse external metadata.

All tables imported through a CDC datastore contain column(s) that tell Data Integrator which operation to perform for each row. For Striva DETAIL Change, the column `DTL_CAPXACTION` provides this information and is required. If this required column is missing during import, Data Integrator raises an error and does not import the table.

In addition to the required column(s), other columns are commonly added. For Striva DETAIL Change, these columns have a `DTL` prefix.

When Data Integrator imports a CDC table, it also adds two columns: `DI_SEQUENCE_NUMBER` with the data type integer and `DI_OPERATION_TYPE` with the data type varchar(1).

An imported CDC table schema might look like the following:



This example has five columns added to the original table:

- Three supplied by DETAIL (one required by Data Integrator)
- Two generated by Data Integrator

The DI_SEQUENCE_NUMBER column

The `DI_SEQUENCE_NUMBER` column starts with zero at the beginning of each extraction. This field increments by one each time Data Integrator reads a row except when it encounters a pair of before- and after-images. Both the before- and after-images receive the same sequence number. This sequencing column provides a way to collate image pairs if they become separated as a result of the data flow design.

You can configure Striva DETAIL Change to retrieve before-images of UPDATE rows before Data Integrator applies the UPDATE operation to the target. Note that if you do not configure Striva DETAIL Change to capture before- images in the database, Data Integrator will discard the rows. For information about when to consider using before-images, see "Using before-images" on page 497.

If during the course of a data flow the before- and after-images become separated or get multiplied into many rows (for example, using GROUP BY or ORDER BY clauses in a query), you can lose row order.

The Map_CDC_Operation transform allows you to restore the original ordering of image pairs by using the DI_SEQUENCE_NUMBER column as its **Sequencing column**. For detailed information about the Map_CDC_Operation transform, see Chapter 5, "Transforms," in the *Data Integrator Reference Guide*.

The DI_OPERATION_TYPE column

Data Integrator generates values in the DI_OPERATION_TYPE column. Valid values for this column are:

- `I` for INSERT
- `D` for DELETE
- `B` for before-image of an UPDATE
- `U` for after-image of an UPDATE

Data Integrator reads rows from DETAIL Change, checking and translating values in column `DTL_CAPXACTION` to Data Integrator values for the `DI_OPERATION_TYPE` column. The translation is as follows:

DTL_CAPXACTION	DI_OPERATION_TYPE
I	I
D	D
T	B
U	U

Configuring a mainframe CDC source

When you drag a CDC datastore table into a data flow, it automatically becomes a source object.

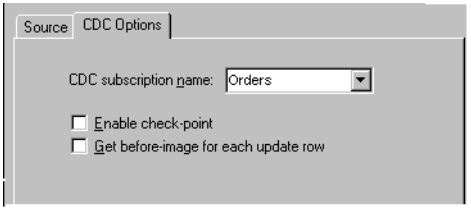
➢ To configure a mainframe CDC table

1. Drag a CDC datastore table into a data flow.

   The table automatically becomes a source object.

2. Click the name of this source object to open its Source Target Editor.

3. Click the **CDC Options** tab.



There are three CDC table options in the Source Target Editor's **CDC Options** tab:

Option Name	Description
CDC subscription name	A name that marks a set of changed data read from the continuously growing Striva CDC table.
Enable check-point	Enables Data Integrator to restrict CDC reads using check-points. Once a check-point is placed, the next time the CDC job runs, it reads only the rows inserted into the CDC table since the last check-point. For more information, see "Using mainframe check-points".
Get before-image for each update row	Some databases allow two images to be associated with an UPDATE row: a before-image and an after-image. If your source can log before-images and you want to read them during change-data capture jobs, enable this option. By default, only after-images are retrieved. For more information, see "Using before-images" on page 497.

Using mainframe check-points

Striva reads mainframe sources and loads changed data into its own CDC log file. Rows of changed data append to the previous load in this file.

When a job in Data Integrator runs with check-pointing enabled, the source table's subscription name is used to read the most recent set of appended rows and to mark the end of the read. If check-pointing is not enabled, then the CDC job reads all the rows in Striva's file, thus increasing processing time.

To use check-points, on the Source Table Editor enter the **CDC Subscription name** and select the **Enable check-point** option.

If you are using check-pointing and you are running your CDC job in recovery mode, the recovered job begins to review the job at the last check-point.

NOTE: To avoid data corruption problems, do not reuse data flows that use CDC datastores because each time a source table extracts data it uses the same subscription name. This means that identical jobs, depending upon when they run, can get different results and leave check-points in different locations in the file. When you migrate CDC jobs from test to production, for example, a best-practice scenario would be to change the subscription name for the production job so that the test job, if ever runs again, will not affect the production job's results.

Using before-images from mainframe sources

For an introduction to before- and after-images, see "Using before-images" on page 497.

When you must capture before-image update rows:

- Make sure DETAIL Change is set up to retrieve full before-images.
- Select the **Get before-images for each update row** option in the CDC table's source editor.

  The underlying, log-based CDC capture software must be set up properly, otherwise enabling the **Get before-images for each update row** option in Data Integrator will have no effect.

After you check the **Get before-images for each update row** option, for every update, Data Integrator processes two rows. In addition to the performance impact of this data volume increase, the before- and after-image pairs could be separated or lost depending on the design of your data flow, which would cause data integrity issues.

The Map_CDC_Operation transform can resolve problems, but undesirable results can still occur due to programming errors. When using functions and transforms that re-order, re-direct, eliminate, and multiply the number of rows in a data flow, be aware of the possible impact to targets.

For detailed information about the Map_CDC_Operation transform, see Chapter 5, "Transforms," in the *Data Integrator Reference Guide*.

Limitations

The table comparison transform and the `lookup` and `lookup_ext` functions cannot be used with a source table imported with a CDC datastore because of the existence of the Data Integrator generated columns. You cannot compare or search these columns.

Using CDC with timestamp-based sources

This section discusses what you need to consider when using source-based, timestamped, changed-data capture:

- Processing timestamps
- Overlaps
- Types of timestamps

In these sections, the term *timestamp* refers to date, time, or datetime values. The discussion in this section applies to cases where the source table has either `CREATE` or `UPDATE` timestamps for each row.

Timestamps can indicate whether a row was created or updated. Some tables have both create and update timestamps; some tables have just one. This section assumes that tables contain at least an update timestamp. For other situations, see "Types of timestamps" on page 522.

Some systems have timestamps with dates and times, some with just the dates, and some with monotonically generated increasing numbers. You can treat dates and generated numbers the same.

It is important to note that for the timestamps based on real time, time zones can become important. If you keep track of timestamps using the nomenclature of the source system (that is, using the source time or source-generated number), you can treat both temporal (specific time) and logical (time relative to another time or event) timestamps the same way.

Processing timestamps

The basic technique for using timestamps to determine changes and to save the highest timestamp loaded in a given job and start the next job with that timestamp.

To do this, create a status table that tracks the timestamps of rows loaded in a job. At the end of a job, `UPDATE` this table with the latest loaded timestamp. The next job then reads the timestamp from the status table and selects only the rows in the source for which the timestamp is later than the status table timestamp.

The following example illustrates the technique. Assume that the last load occurred at 2:00 PM on January 1, 1998. At that time, the source table had only one row (key=1) with a timestamp earlier than the previous load. Data Integrator loads this row into the target table and updates the status table with the highest timestamp loaded: 1:10 PM on January 1, 1998. After 2:00 PM Data Integrator adds more rows to the source table:

Source table

Key	Data	Update_Timestamp
1	Alvarez	01/01/98 01:10 PM
2	Tanaka	01/01/98 02:12 PM
3	Lani	01/01/98 02:39 PM

Target table

Key	Data	Update_Timestamp
1	Alvarez	01/01/98 01:10 PM

Status table

Last_Timestamp
01/01/98 01:10 PM

At 3:00 PM on January 1, 1998, the job runs again. This time the job does the following:

1. Reads the Last_Timestamp field from the status table (`01/01/98 01:10 PM`).
2. Selects rows from the source table whose timestamps are later than the value of Last_Timestamp. The SQL command to select these rows is:

   ```
   SELECT * FROM Source
   WHERE 'Update_Timestamp' > '01/01/98 01:10
   pm'
   ```

   This operation returns the second and third rows (key=2 and key=3).
3. Loads these new rows into the target table.
4. Updates the status table with the latest timestamp in the target table (`01/01/98 02:39 PM`) with the following SQL statement:

   ```
   UPDATE STATUS SET 'Last_Timestamp' = SELECT
   MAX('Update_Timestamp') FROM target_table
   ```

The target shows the new data:

Source table

Key	Data	Update_Timestamp
1	Alvarez	01/01/98 01:10 PM
2	Tanaka	01/01/98 02:12 PM
3	Lani	01/01/98 02:39 PM

Target table

Key	Data	Update_Timestamp
2	Tanaka	01/01/98 02:12 PM
3	Lani	01/01/98 02:39 PM
1	Alvarez	01/01/98 01:10 PM

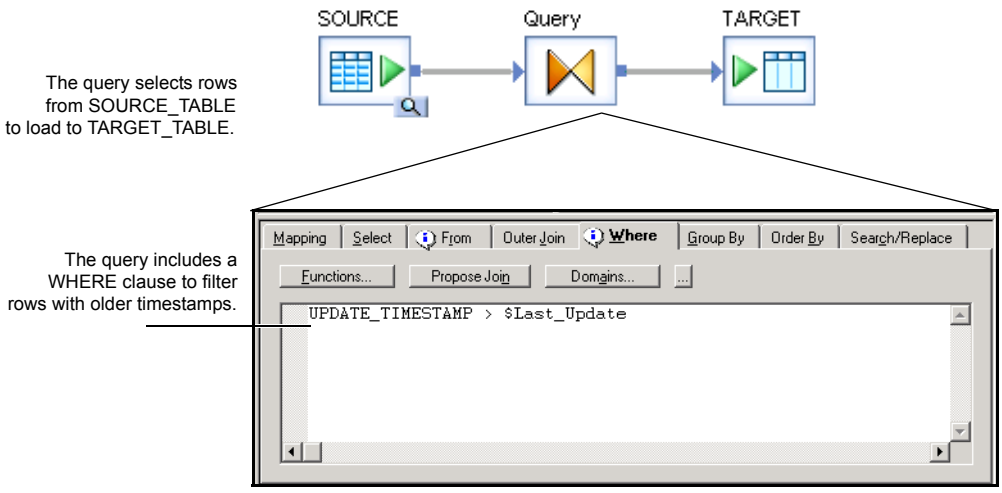
Status table

Last_Timestamp
01/01/98 02:39 PM

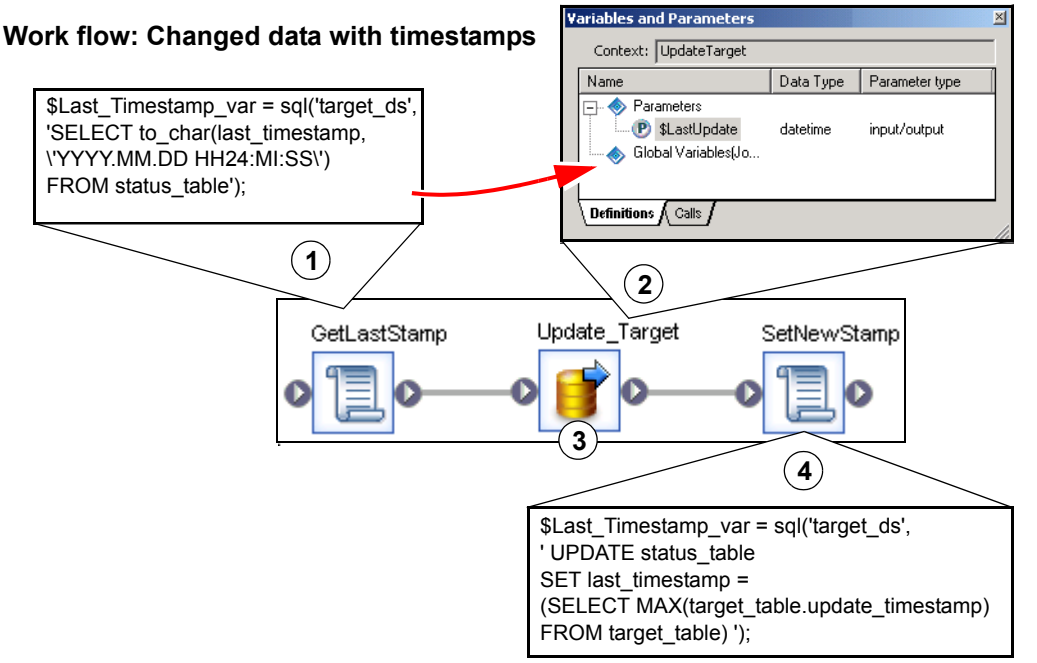
To specify these operations, a Data Integrator data flow requires the following objects (and assumes all the required metadata for the source and target tables has been imported):

- A data flow to extract the changed data from the source table and load it into the target table:

Data flow: Changed data with timestamps



- A work flow to perform the following:
  1. Read the status table
  2. Set the value of a variable to the last timestamp
  3. Call the data flow with the variable passed to it as a parameter
  4. Update the status table with the new timestamp



- A job to execute the work flow

Overlaps

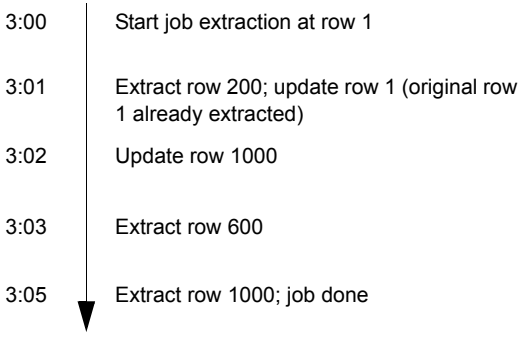
Unless source data is rigorously isolated during the extraction process (which typically is not practical), there is a window of time when changes can be lost between two extraction runs. This overlap period affects source-based changed-data capture because this kind of data capture relies on a static timestamp to determine changed data.

For example, suppose a table has 1000 rows (ordered 1 to 1000). The job starts with timestamp 3:00 and extracts each row. While the job is executing, it updates two rows (1 and 1000) with timestamps 3:01 and 3:02, respectively. The job extracts row 200 when someone updates row 1. When the job extracts row 300, it updates row 1000. When complete, the job extracts the latest timestamp (3:02) from row 1000 but misses the update to row 1.

Here is the data in the table:

Row Number	Column A
1	...
2	...
3	...
...	...
200	...
...	...
600	...
...	...
1000	...

Here is the timeline of events (assume the job extracts 200 rows per minute):



There are three techniques for handling this situation:

- Overlap avoidance
- Overlap reconciliation
- Presampling

The following sections describe these techniques and their implementations in Data Integrator. This section continues on the assumption that there is at least an update timestamp. For other situations, see "Types of timestamps" on page 522.

Overlap avoidance

In some cases, it is possible to set up a system where there is no possibility of an overlap. You can avoid overlaps if there is a processing interval where no updates are occurring on the target system.

For example, if you can guarantee that the data extraction from the source system does not last more than one hour, you can run a job at 1:00 AM every night that selects only the data updated the previous day until midnight. While this regular job does not give you up-to-the-minute updates, it guarantees that you never have an overlap and greatly simplifies timestamp management.

Overlap reconciliation

Overlap reconciliation requires a special extraction process that reapplies changes that could have occurred during the overlap period. This extraction can be executed separately from the regular extraction. For example, if the highest timestamp loaded from the previous job was 01/01/98 10:30 PM and the overlap period is one hour, overlap reconciliation reapplies the data updated between 9:30 PM and 10:30 PM on January 1, 1998.

The overlap period is usually equal to the maximum possible extraction time. If it can take up to *n* hours to extract the data from the source system, an overlap period of *n* (or *n* plus some small increment) hours is recommended. For example, if it takes at most two hours to run the job, an overlap period of at least two hours is recommended.

There is an advantage to creating a separate overlap data flow. A "regular" data flow can assume that all the changes are new and make assumptions to simplify logic and improve performance. For example, rows flagged as INSERT are often loaded into a fact table, but rows flagged as UPDATE rarely are. Thus, the regular data flow selects the new rows from the source, generates new keys for them, and uses the database loader to add the new facts to the target database. Because the overlap data flow is likely to apply the same rows again, it cannot blindly bulk load them or it creates duplicates. Therefore, the overlap data flow must check whether the rows exist in the target and insert only the ones that are missing. This lookup affects performance; therefore, perform it for as few rows as possible.

If the data volume is sufficiently low, you can load the entire new data set using this technique of checking before loading, avoiding the need to create two different data flows.

Presampling

Presampling eliminates the overlap by first identifying the most recent timestamp in the system, saving it, and then extracting rows up to that timestamp.

The technique is an extension of the simple timestamp processing technique described previously in "Processing timestamps" on page 511. The main difference is that the status table now contains a start and an end timestamp. The start timestamp is the latest timestamp extracted by the previous job; the end timestamp is the timestamp selected by the current job.

To return to the example: The last extraction job loaded data from the source table to the target table and updated the status table with the latest timestamp loaded:

Source table

Key	Data	Update_Timestamp
1	Alvarez	01/01/98 01:10 PM
2	Tanaka	01/01/98 02:12 PM
3	Lani	01/01/98 02:39 PM

Target table

Key	Data	Update_Timestamp
1	Alvarez	01/01/98 01:10 PM

Status table

Start_Timestamp	End_Timestamp
01/01/98 01:10 PM	NULL

Now it's 3:00 PM on January 1, 1998, and the next job runs; it does the following:

1. Selects the most recent timestamp from the source table and inserts it into the status table as the End Timestamp.

   The SQL command to select one row is:

   ```
   SELECT MAX(Update_Timestamp) FROM source
   table
   ```

   The status table becomes:

   Status table

Start_Timestamp	End_Timestamp
01/01/98 01:10 PM	01/01/98 02:39 PM

2. Selects rows from the source table whose timestamps are greater than the start timestamp but less than or equal to the end timestamp. The SQL command to select these rows is:

```
SELECT *
FROM source table
WHERE Update_Timestamp > '1/1/98 1:10pm'
AND Update_Timestamp <= '1/1/98 2:39pm'
```

   This operation returns the second and third rows (key=2 and key=3)

3. Loads these new rows into the target table.

4. Updates the status table by setting the start timestamp to the previous end timestamp and setting the end timestamp to NULL.

The table values end up as follows:

Source table

Key	Data	Update_Timestamp
1	Alvarez	01/01/98 01:10 PM
2	Tanaka	01/01/98 02:12 PM
3	Lani	01/01/98 02:39 PM

Target table

Key	Data	Update_Timestamp
1	Alvarez	01/01/98 01:10 PM
2	Tanaka	01/01/98 02:12 PM
3	Lani	01/01/98 02:39 PM

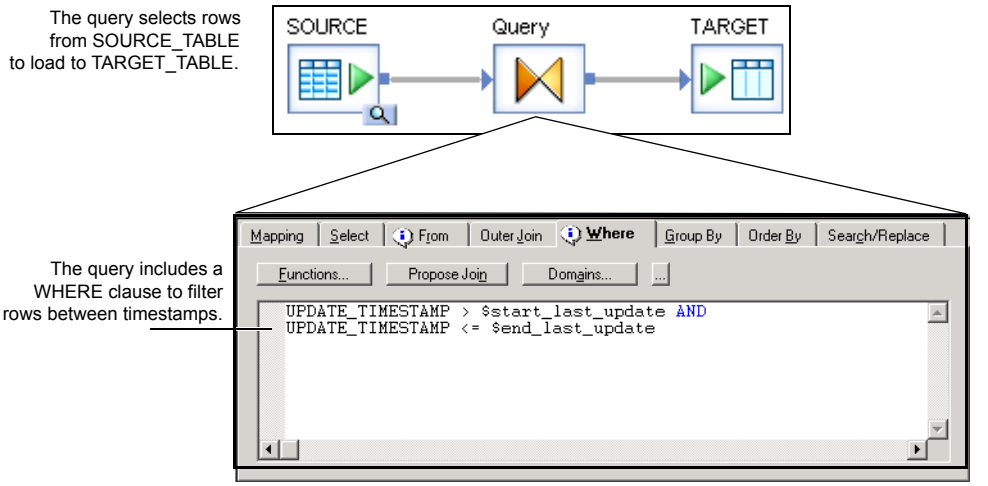
Status table

Start_Timestamp	End_Timestamp
01/01/98 02:39 PM	NULL

To enhance the previous example to consider the overlap time requires the following changes to the work flow:

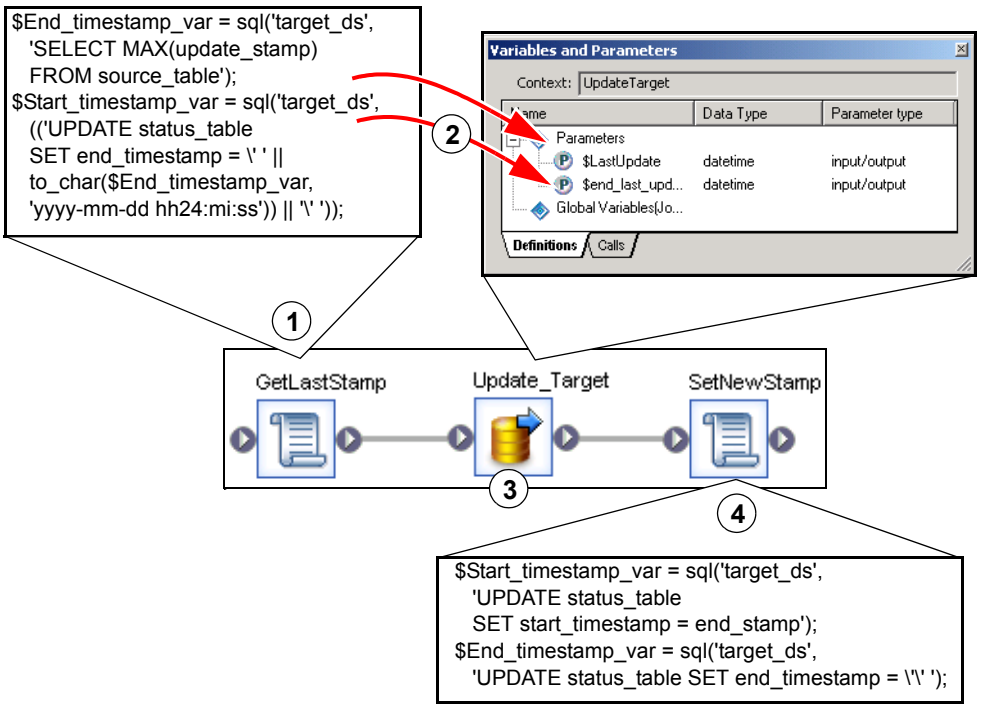
- A data flow to extract the changes since the last update and before the most recent timestamp.

Data flow: Changed data with overlap



- A work flow to perform the following:
  1. Read the source table to find the most recent timestamp.
  2. Set the value of two variables to the start of the overlap time and to the end of the overlap time, respectively.
  3. Call the data flow with the variables passed to it as parameters.
  4. Update the start timestamp with the value from end timestamp and set the end timestamp to NULL.

Work flow: Changed data with overlap



Types of timestamps

Some systems have timestamps that record only when rows are created. Others have timestamps that record only when rows are updated. (Typically, update-only systems set the update timestamp when the row is created or updated.) Finally, there are systems that keep separate timestamps that record when rows are created and when they are updated.

This section discusses these timestamps:

- Create-only timestamps
- Update-only timestamps
- Create and update timestamps

Create-only timestamps

If the source system provides only create timestamps, you have these options:

- If the table is small enough, you can process the entire table to identify the changes. The section "Using CDC for targets" on page 538, describes how to identify changes.
- If the table never gets updated, you can extract only the new rows.
- If the table is large and gets updated, you can combine the following two techniques:
  - Periodically (for example, daily) extract only the new rows.
  - Less frequently (for example, weekly) extract the updated rows by processing the entire table.

Update-only timestamps

Using only an update timestamp helps minimize the impact on the source systems, but it makes loading the target systems more difficult. If the system provides only an update timestamp and there is no way to tell new rows from updated rows, your job has to reconcile the new data set against the existing data using the techniques described in the section "Using CDC for targets" on page 538.

Create and update timestamps

Both timestamps allow you to easily separate new data from updates to the existing data. The job extracts all the changed rows and then filters unneeded rows using their timestamps.

Accomplish these extractions in Data Integrator by adding the WHERE clause from the following SQL commands into an appropriate query transform:

- Find new rows:

```
SELECT * FROM source_table
WHERE Create_Timestamp > $Last_Timestamp
```

- Find updated rows:

```
SELECT * FROM source_table
WHERE Create_Timestamp <= $Last_Timestamp
AND
Update_Timestamp > $Last_Timestamp)
```

From here, the new rows go through the key-generation process and are inserted into the target, and the updated rows go through the key-lookup process and are updated in the target.

For performance reasons, you might want to separate the extraction of new rows into a separate data flow to take advantage of bulk loading into the target. The updated rows cannot be loaded by bulk into the same target at the same time.

Timestamp-based CDC examples

This section discusses the following techniques for time-stamped based CDC:

- Preserving generated keys
- Preserving history

Preserving generated keys

For performance reasons, many data warehouse dimension tables use generated keys to join with the fact table. For example, customer ABC has a generated key 123 in the customer dimension table. All facts for customer ABC have 123 as the customer key. Even if the customer dimension is small, you cannot simply reload it every time a record changes: unless you assign the generated key of 123 to the customer ABC, the customer dimension table and the fact tables do not correlate.

You can preserve generated keys by:

- Using the lookup function
- Comparing tables

Using the lookup function

If history preservation is not an issue and the only goal is to generate the correct keys for the existing rows, the simplest technique is to look up the key for all rows using the `lookup` function in a query. If you do not find the key, generate a new one.

In the following example, the customer dimension table contains generated keys. When you run a job to update this table, the source customer rows must match the existing keys.

Source customer table

Company Name	Customer ID
ABC	001
DEF	002
GHI	003
JKL	004

Target dimension table

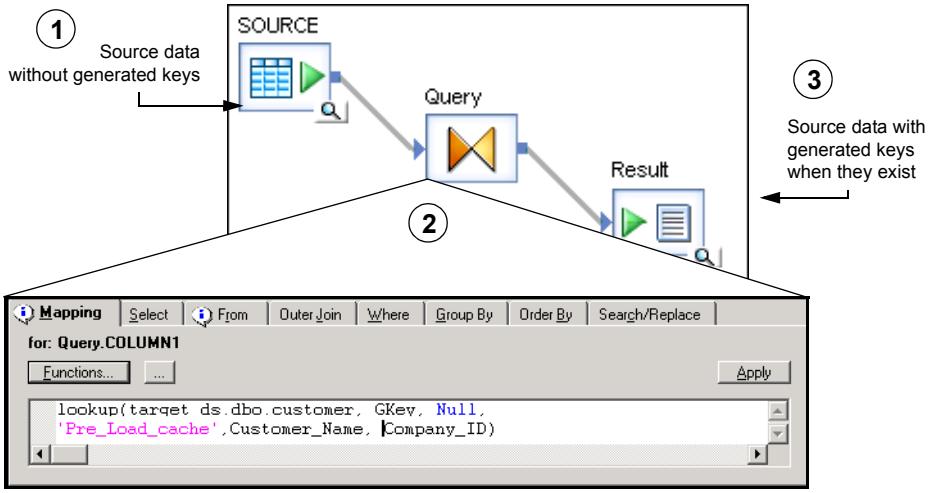
Gen_Key	Company Name	Customer ID
123	ABC	001
124	DEF	002
125	GHI	003

This example data flow does the following:

1. Extracts the source rows.
2. Retrieves the existing keys using a `lookup` function in the mapping of a new column in a query.

3. Loads the result into a file (to be able to test this stage of the data flow before adding the next steps).

Data flow: Replace generated keys



The `lookup` function compares the source rows with the target. The arguments for the function are as follows:

lookup function arguments	Description
target_ds.owner.customer	Fully qualified name of the target table containing the generated keys.
GKey	The column name in the target table containing the generated keys.
NULL	NULL value to insert in the key column if no existing key is found.
'PRE_LOAD_CACHE'	Caching option to optimize the lookup performance.
Customer_ID	The column in the target table containing the value to use to match rows.
Customer_ID	The column in the source table containing the values to use to match rows.

The resulting data set contains all the rows from the source with generated keys where available:

Result data set

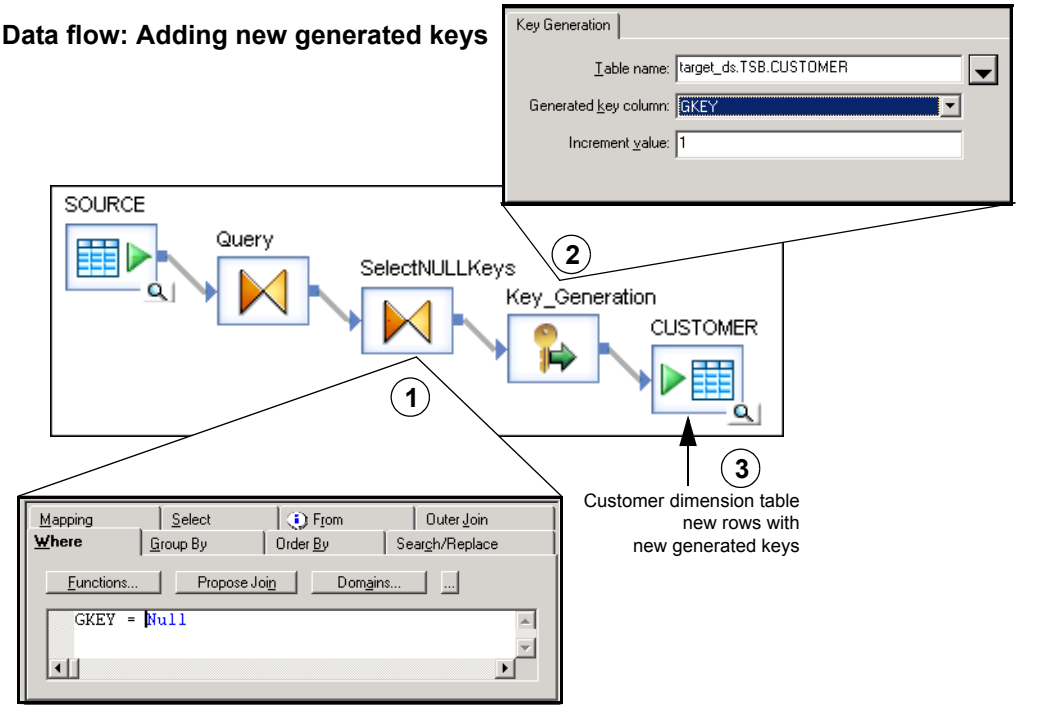
Gen_Key	Company Name	Customer ID
123	ABC	001
124	DEF	002
125	GHI	003
NULL	JKL	004

Adding a new generated key to the new records requires filtering out the new rows from the existing and updated rows. In the data flow, this requires the following steps:

1. A query to select the rows with NULL generated keys.
2. A Key_Generation transform to determine the appropriate key to add.
3. A target to load the new rows into the customer dimension table.

The data flow expands as follows.

Data flow: Adding new generated keys



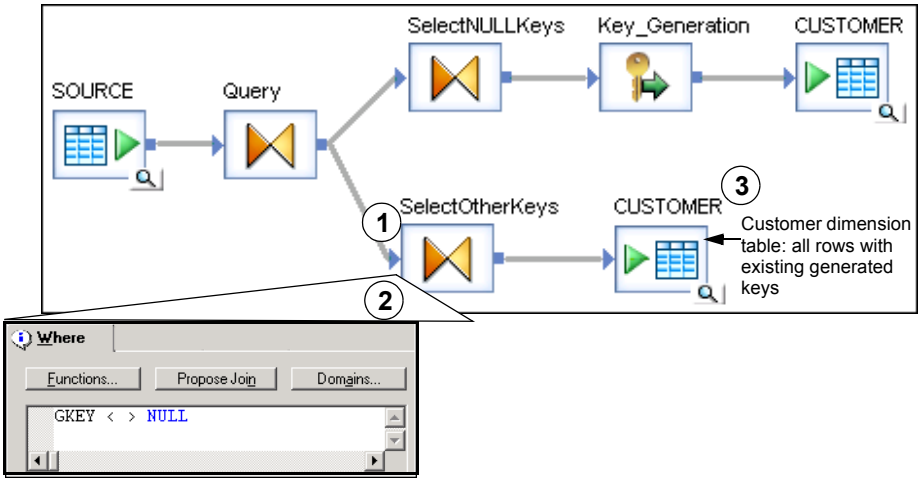
This data flow handles the new rows; however, the rows from the source whose keys were found in the target table might contain updated data. Because this example assumes that preserving history is not a requirement, Data Integrator loads all rows from the source into the target.

The data flow requires new steps to handle updated rows, as follows:

1. A new line leaving the query that looked up the existing keys.
2. A query to filter the rows with existing keys from the rows with no keys.
3. A target to load the rows into the customer dimension table.

The data flow expands as follows:

Data flow: Loading all rows into the target



Comparing tables

The drawback of the generated-keys method is that even if the row has not been changed, it generates an UPDATE and is loaded into the target. If the amount of data is large, a table-comparison transform provides a better alternative by allowing the data flow to load only changed rows.

The table-comparison transform examines all source rows and performs the following operations:

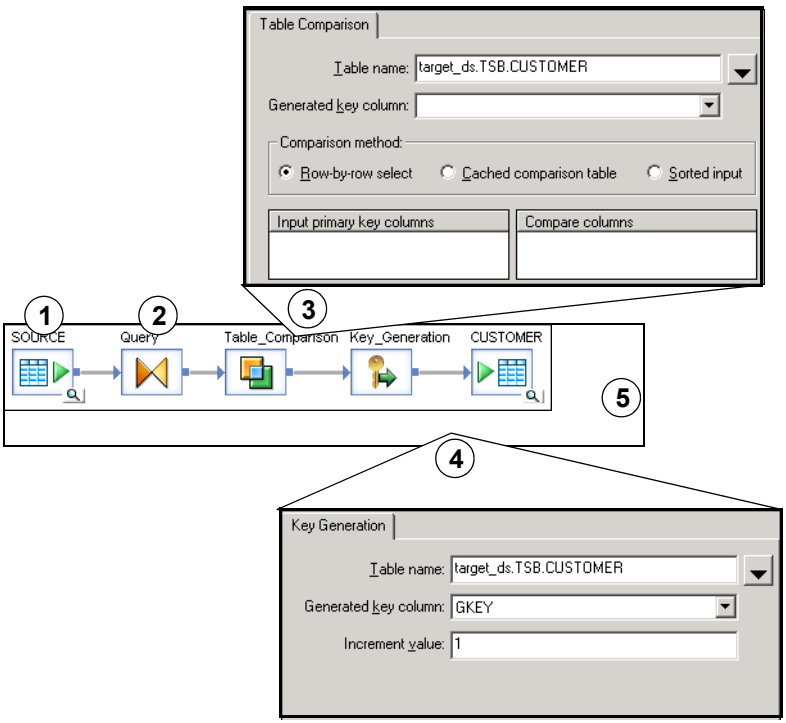
- Generates an INSERT for any new row not in the target table.
- Generates an UPDATE for any row in the target table that has changed.
- Ignores any row that is in the target table and has not changed.
- Fills in the generated key for the updated rows.

You can then run the result through the key-generation transform to assign a new key for every INSERT. This is the data set that Data Integrator loads into the target table.

The data flow that accomplishes this transformation includes the following steps:

1. A source to extract the rows from the source table(s).
2. A query to map columns from the source.
3. A table-comparison transform to generate INSERT and UPDATE rows and to fill in existing keys.
4. A key-generation transform to generate new keys.
5. A target to load the rows into the customer dimension table

Data flow: Load only updated or new rows



Preserving history

History preserving allows the data warehouse or data mart to maintain the history of data so you can analyze it over time. Most likely, you will perform history preservation on dimension tables.

For example, if a customer moves from one sales region to another, simply updating the customer record to reflect the new region would give you misleading results in an analysis by region over time because all purchases made by a customer before the move would incorrectly be attributed to the new region.

Data Integrator provides a special transform that preserves data history to prevent this kind of situation. The History_Preserving transform ignores everything but rows flagged as UPDATE. For these rows, it compares the values of specified columns and, if the values have changed, flags the row as INSERT. This produces a second row in the target instead of overwriting the first row.

To expand on how Data Integrator would handle the example of the customer who moves between regions:

- If `Region` is a column marked for comparison, the History_Preserving transform generates a new row for that customer.
- A Key_Generation transform gives the new row a new generated key and loads the row into the customer dimension table.
- The original row describing the customer remains in the customer dimension table with a unique generated key.

In the following example, one customer moved from the East region to the West region, and another customer's phone number changed.

Source Customer table

Customer	Region	Phone
Fred's Coffee	East	(212) 123-4567
Jane's Donuts	West	(650) 222-1212
Sandy's Candy	Central	(115) 231-1233

Target Customer table

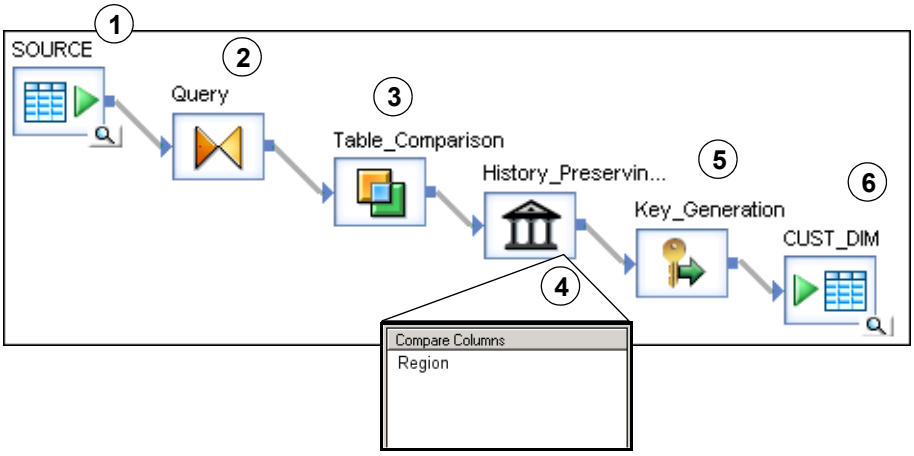
GKey	Customer	Region	Phone
1	Fred's Coffee	East	(212) 123-4567
2	Jane's Donuts	East	(201) 777-1717
3	Sandy's Candy	Central	(115) 454-8000

In this example, the data flow preserves the history for the `Region` column but does not preserve history for the `Phone` column. The data flow contains the following steps:

1. A source to extract the rows from the source table(s).
2. A query to map columns from the source.
3. A table-comparison transform to generate INSERTs and UPDATEs and to fill in existing keys.
4. A History_Preserving transform to convert certain `UPDATE` rows to INSERT rows.
5. A key-generation transform to generate new keys for the updated rows that are now flagged as INSERT.

6. A target to load the rows into the customer dimension table.

Data flow: Preserve history in the target



The resulting dimension table is as follows:

Target Customer table

GKey	Customer	Region	Phone	
1	Fred's Coffee	East	(212) 123-4567	
2	Jane's Donuts	East	(201) 777-1717	Updated rows
3	Sandy's Candy	Central	(115) 231-1233	
4	Jane's Donuts	West	(650) 222-1212	New row

Because the `Region` column was set as a **Compare** column in the History_Preserving transform, the change in the Jane's Donuts row created a new row in the customer dimension table. Because the `Phone` column was not used in the comparison, the change in the Sandy's Candy row did not create a new row but updated the existing one.

Now that there are two rows for Jane's Donuts, correlations between the dimension table and the fact table must use the highest key value.

Note that updates to non-history preserving columns update all versions of the row if the update is performed on the natural key (for example, `Customer`), and only update the latest version if the update is on the generated key (for example, `GKey`). You can control which key to use for updating by appropriately configuring the loading options in the target editor.

Valid_from date and valid_to date

To support temporal queries like "What was the customer's billing address on May 24, 1998," Data Integrator supports **Valid from** and **Valid to** date columns.

In history-preserving techniques, there are multiple records in the target table with the same source primary key values. A record from the source table is considered *valid* in the dimension table for all date values `t` such that the **Valid from** date is less than or equal to `t`, which is less than the **Valid to** date. (*Valid* in this sense means that the record's generated key value is used to load the fact table during this time interval.)

When you specify the **Valid from** and **Valid to** entries, the History_Preserving transform generates an UPDATE record before it generates an INSERT statement for history-preservation reasons (it converts an UPDATE into an INSERT). The UPDATE record will set the **Valid to** date column on the current record (the one with the same primary key as the INSERT) to the value in the **Valid from** date column in the INSERT record.

Update flag

To support slowly changing dimension techniques, Data Integrator enables you to set an update flag to mark the *current* record in a dimension table.

Value **Set value** in column **Column** identifies the current valid record in the target table for a given source table primary key.

When you specify **Column**, the History_Preserving transform generates an UPDATE record before it generates an INSERT statement.

This UPDATE record will set the **Column** value to **Reset value** in the target table record with the same source primary key as the INSERT statement.

In the INSERT statement the **Column** will be set to **Set value**.

When you specify entries in both the groups, the History_Preserving transform generates only one extra UPDATE statement for every INSERT statement it produces. This UPDATE statement updates the **Valid to** value.

Additional job design tips

When designing a job to implement changed-data capture (CDC), you must consider:

- Header and detail synchronization
- Capturing physical deletions

Header and detail synchronization

Typically, source systems keep track of header and detail information changes in an independent way. For example, if a line-item status changes, its "last modified date" column updates, but the same column at the order header level does not update. Conversely, a change to the default ship-to address in the order header might impact none of the existing line items.

In some instances, however, your source system might not consistently update those tracking columns, or you might not have access to such information (for example, when rows are physically deleted). In these cases, you might choose to extract all header and detail information whenever any changes occur at the header level or in any individual line item.

To extract all header and detail rows when any of these elements have changed, use logic similar to this SQL statement:

```
SELECT …
FROM HEADER, DETAIL
WHERE
   HEADER.ID = DETAIL.ID AND
   (HEADER.LAST_MODIFIED
      BETWEEN $G_SDATE AND $G_EDATE
      OR
      DETAIL.LAST_MODIFIED
      BETWEEN $G_SDATE AND $G_EDATE)
```

For some RDBMSs, this WHERE clause is not well optimized and might cause serious performance degradation. You might opt to relax that clause by removing one of the upper bounds, such as in:

```
…
WHERE HEADER.ID = DETAIL.ID AND
   (HEADER.LAST_MODIFIED
      BETWEEN $G_SDATE AND $G_EDATE
      OR
      DETAIL.LAST_MODIFIED >= $G_SDATE)
…
```

This might retrieve a few more rows than originally intended, but it might improve the final performance of your system while not altering the result of your target database.

Capturing physical deletions

When your source system allows rows to be physically deleted, your job should include logic to update your target database correspondingly. There are essentially five ways to do this:

- Scan a log of operations — If your system logs transactions in a readable format or if you can alter the system to generate such a log, then you can scan that log to identify the rows you need to delete.

- Perform a full refresh — Simply reload all of the data, therefore fully synchronizing the source system and the target database.
- Perform a partial refresh based on a data-driven time-window — For example, suppose that the source system only allows physical deletion of orders that have not been closed. If the first non-closed order in your source table occurred six months ago, then by refreshing the last six months of data you are guaranteed to have achieved synchronization.
- Perform a partial refresh based on a business-driven time-window — For example, suppose that the business that the job supports usually deletes orders shortly after creating them. In this case, refreshing the last month of orders is appropriate to maintain integrity.

1. Check every order that could possibly be deleted — You must verify whether any non-closed order has been deleted. To be efficient, this technique requires you to keep a record of the primary keys for every object that is a candidate for deletion.

When physical deletions of detail information in a header-detail relationship are possible (for example, removing line items from an existing order), then you must capture these physical deletions when synchronizing header and detail information.

Using CDC for targets

Source-based changed-data capture is almost always preferable to target-based capture for performance reasons. Some source systems, however, do not provide enough information to make use of the source-based changed-data capture techniques. Target-based changed-data capture allows you to use the technique when source-based change information is limited.

20 Monitoring jobs

This chapter contains the following sections:

Administrator

The Data Integrator Administrator is your primary monitoring resource for all jobs designed in the Data Integrator Designer. For detailed information, see the Data Integrator Administrator Guide.

SNMP support

Data Integrator includes an SNMP (simple network management protocol) agent that allows you to connect to third-party applications to monitor its jobs. You can use an SNMP-supported application to monitor Data Integrator job status and receive error events.

Topics in this section include:

- About the Data Integrator SNMP agent
- Job Server, SNMP agent, and NMS application architecture
- About SNMP Agent's Management Information Base (MIB)
- About an NMS application
- Configuring Data Integrator to support an NMS application
- Troubleshooting

About the Data Integrator SNMP agent

When you enable SNMP (simple network management protocol) for a Job Server, that Job Server sends information about the jobs it runs to the SNMP agent. The SNMP agent monitors and records information about the Job Server and the jobs it is running. You can configure NMS (network management software) applications to communicate with the SNMP agent. Thus, you can use your NMS application to monitor the status of Data Integrator jobs.

The SNMP agent is a license-controlled feature of the Data Integrator Job Server. When you have a Data Integrator SNMP license on a computer, you can enable SNMP for any number of Job Servers running on that computer. Like Job Servers, SNMP starts when the Data Integrator Service starts.

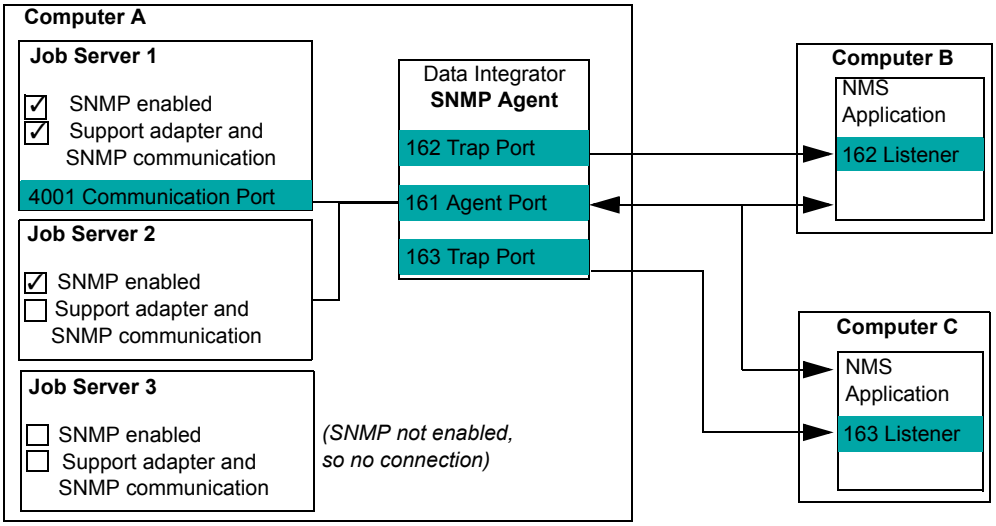
Job Server, SNMP agent, and NMS application architecture

You must configure one Job Server to communicate with the SNMP agent and to manage the communication for SNMP-enabled Job Servers.

You must also enable at least one Job Server for SNMP. This Job Server does not need to be the same one configured with the communication port. When you enable a Job Server for SNMP, it will send events to the SNMP agent via the Job Server with the communication port.

When you configure the SNMP agent, you specify one agent port and any number of *trap* receiver ports. The Data Integrator SNMP agent sends proactive messages (traps) to NMS applications. While you use an NMS application, traps notify you about potential problems.

The SNMP agent uses the agent port to communicate with NMS applications using UDP (user datagram protocol). The agent listens for requests from the NMS applications and responds to requests. The agent uses the trap receiver ports to send error events (traps or notifications) to NMS applications.



About SNMP Agent's Management Information Base (MIB)

The Data Integrator SNMP agent uses a management information base (MIB) to store information about SNMP-enabled Job Servers and the jobs they run. Metadata for the Data Integrator MIB is stored in two files which are located in the `LINK_DIR/bin/snmp/mibs` directory:

```
BOBJ-ROOT-MIB.txt

BOBJ-DI-MIB.txt
```

Consult these files for more detailed descriptions and up-to-date information about the structure of objects in the Data Integrator MIB.

The Data Integrator MIB contains five scalar variables and two tables. The scalar variables list the installed version of Data Integrator, the time the agent started, and the current system time. The tables contain information about the status of Job Servers and the jobs they run. Tables include:

Data Integrator MIB Job Server table

Column	Description
jsIndex	A unique index that identifies each row in the table
jsName	Name of Job Server
jsStatus	Status of the Job Server. Possible values are: • notEnabled • initializing • optimizing • ready • proceed • wait • stop • stopRunOnce • stopRecovered • stopError • notResponding • error • warning • trace

Data Integrator MIB Job table

Column	Description
jobIdDate	The part of a job's identifier that matches the date
jobIdTime	The part of a job's identifier that matches the time
jobIdN	The final integer in a job's identifier
jobRowN	A unique index that identifies an object in a job
jType	Associated object for this row. Possible values include: • job — A job • wf — A work flow • df — A data flow • error — An error message • trace — A trace message
jName	The name or identifier for this object, such as the job or work flow name or the error message identifier.
jStatus	The status of the object. Possible values include: • notEnabled • initializing • optimizing • ready • proceed • wait • stop • stopRunOnce • stopRecovered • stopError • notResponding • error • warning • trace

Data Integrator MIB Job table (Continued)

Column	Description
jRowsIn	Depends on the type of object: • Data flow — The number of input rows read • Work flow — Always zero • Job — Sum of values for all data flows in the job • Error, warning, trace — Always zero
jRowsOut	Depends on the type of object: • Data flow — The number of output rows written • Work flow — Always zero • Job — Sum of values for all data flows in the job • Error, warning, trace — Number of times that the error, warning, or trace has occurred during this job
jStatusTime	The time when the object's jStatus, jRowsIn, or jRowsOut last changed.
jExecTime	The number of milliseconds between the beginning of the object's execution and jStatusTime.
jInitTime	The number of milliseconds necessary to compile the object (job, work flow or data flow).
jMessage	For jobs, work flows, and data flows, either empty or an information message. For errors, warnings, or trace messages, jMessage contains the message text.

The Data Integrator SNMP agent receives data about jobs and Job Servers from SNMP-enabled Job Servers and maintains this data in the Data Integrator MIB for currently running jobs and recently completed jobs. To provide some historical context, each time the agent starts it loads data into the Job table for each Job Server. The data is from jobs that ran just before the Job Servers were shut down.

The MIB is stored in memory. The SNMP agent maintains the data for completed jobs for the specified lifetime. The agent summarizes and eliminates individual data flow and work flow records for completed jobs periodically to reduce the size of the MIB.

The data that remains includes:

- One Job table row with the statistics for the entire job
- For a successful job, zero additional rows
- For a failed job, additional error rows as needed.

If the MIB's size reaches the maximum table size, the agent eliminates 20 percent of the completed jobs, starting with the oldest jobs. During configuration, you set a job lifetime and a maximum table size.

About an NMS application

An NMS application can query the Data Integrator SNMP agent for the information stored in the Data Integrator MIB (iso.org.dod.internet.private.enterprises.businessObjects.dataIntegrator) or one of the standard SNMP MIBs:

- `iso.org.dod.internet.mgmt.mib-2.system`
- `iso.org.dod.internet.mtmt.mib-2.snmp`
- `iso.org.dod.internet.snmpv2.snmpModules`

The agent listens on the agent port for commands from an NMS application which communicates commands as PDUs (protocol data units). The agent responds to SNMP `GetRequest`, `GetNextRequest`, `GetBulkRequest`, and `SetRequest` commands that specify valid object identifiers (OIDs). Because there are no writable objects in the Data Integrator MIB, the agent gracefully rejects `SetRequest` commands for that MIB.

While you use an NMS application, traps notify you about potential problems. Specifically, the agent sends an `SNMPv2-Trap` PDU to the SNMP ports that you have configured. The agent sends traps when:

- Errors occur during batch or real-time jobs
- Job Servers fail

- Agent starts
- Agent has an internal error
- Agent has an orderly shut down
- Agent restarts and the previous agent was unable to send a trap caused by a job error (these traps include a historical annotation)

  NOTE: This can occur if the machine fails unexpectedly or is halted without an orderly shutdown.

- Agent denies a request due to an authentication failure (if configured to do so). See "Traps" on page 557.

NOTE: Status for real-time services does not appear until the real-time services have processed enough messages to reach their cycle counts. Similarly, Data Integrator does not send traps for real-time jobs until the jobs have reached their cycle count. After the jobs reach their cycle count, Data Integrator refreshes status or sends additional traps.

Configuring Data Integrator to support an NMS application

If you have an NMS application that monitors systems across your network using SNMP, you can use that application to monitor Data Integrator jobs. To do so:

- Select one Data Integrator Job Server on each computer to support SNMP communication. When you select a Job Server to support SNMP, you must also specify the communication port that connects the Job Server to the SNMP agent.
- Enable SNMP on each Job Server that you want to monitor. When you enable SNMP for a Job Server, you are telling Data Integrator to send events to the SNMP agent via the communication Job Server on the same machine.

  NOTE: Supporting SNMP communication and enabling SNMP are separate configurations.

- Configure the Data Integrator SNMP agent on the same computer for which you configured Job Servers.
- Configure your NMS application to query the Data Integrator MIB for job status using the agent port. Refer to the documentation for the NMS application. If the Data Integrator SNMP agent does not respond to the NMS application (for example, if the application gets a time-out), check the agent.

SNMP configuration in Windows

➢ To select a Job Server to support SNMP communication

1. Open the Data Integrator Server Manager (**Start** > **Programs** > **BusinessObjects Data Integrator 6.1** > **Server Manager**).
2. In the Data Integrator Server Manager, click **Edit Job Server Config**.

   The Job Server Configuration Editor opens. This window lists the Job Servers currently configured. The summary lists the Job Server name, the Job Server port, whether the Job Server supports adapters and SNMP communication, the communication port, and whether SNMP is enabled.

   Exactly one Job Server must be configured to support adapters and SNMP communication. If a Job Server is already configured to support adapters and SNMP communication, verify that the communication port is correct. Otherwise, continue with this procedure.
3. Select a Job Server and click **Edit**.

   If you want to add a new Job Server, click **Add**.
4. Select the **Support adapter and SNMP communication** check box.
5. In the **Communication port** box, enter the port you want to use for communication between the Data Integrator SNMP agent and Job Servers on this computer.

Enter a port number that is not used for other applications. The default value is 4001. Data Integrator uses the same port to communicate with adapters.

6. Verify that the repositories associated with this Job Server are correct.
7. You can enable SNMP for a Job Server from the Server Manager or from the SNMP agent. The SNMP agent allows you to enable or disable more than one Job Server at a time.
   - If you want to enable SNMP for the current Job Server:
     a. Select the **Enable SNMP** check box
     b. Click **OK**.
     c. In the Job Server Configuration Editor, select **OK**.
     d. In the Server Manager window, select **Restart**.
   - If you want to configure the SNMP agent (including enabling SNMP for Job Servers):
     a. Click **OK**.
     b. In the Job Server Configuration Editor, select **OK**.
     c. Skip to step 2 in the procedure, "To configure the SNMP agent" on page 551.

➢ To enable SNMP on a Job Server

1. You can enable SNMP for a Job Server from the Server Manager or from the SNMP agent. The SNMP agent allows you to enable or disable more than one Job Server at a time.
   - To use the SNMP agent, see "To configure the SNMP agent" on page 551.
   - To use the Server Manager, see step 2 below.
2. Open the Data Integrator Server Manager (**Start** > **Programs** > **BusinessObjects Data Integrator 6.1** > **Server Manager**).
3. In the Data Integrator Server Manager, click **Edit Job Server Config**.

The Job Server Configuration Editor opens. This window lists the Job Servers currently configured. The summary indicates whether the Job Server is SNMP-enabled.

You can enable SNMP on any Job Server. SNMP-enabled Job Servers send the SNMP agent messages about job status and job errors.

4. Select a Job Server and click **Edit**.

   If you want to add a new Job Server, click **Add**.

5. Select the **Enable SNMP** check box.
6. Click **OK**.
7. To enable SNMP on additional Job Servers, repeat steps 4 through 6.
8. In the Job Server Configuration Editor, select **OK**.
9. In the Server Manager window, select **Restart**.

   The Data Integrator Service restarts, which restarts the Job Servers using the new configuration information.

➢ To configure the SNMP agent

1. Open the Data Integrator Server Manager (**Start** > **Programs** > **BusinessObjects Data Integrator 6.1** > **Server Manager**).
2. In the Data Integrator Server Manager, click **Edit SNMP Config**.

   The SNMP Configuration Editor opens.

3. Select the **Enable SNMP on this machine** check box to enable the Data Integrator SNMP agent.

   After you enable the SNMP agent, you can modify current or default SNMP configuration parameters. Parameter categories include:

   - Job Servers for SNMP — Job Servers enabled for SNMP on this machine

- System Variables — Parameters that affect basic agent operation
- Access Control, v1/v2c — Parameters that affect how the agent grants NMS applications using the v1 or v2c version of SNMP access to the Data Integrator MIB and Data Integrator supported standard MIBs
- Access Control, v3 — Parameters that affect how the agent grants NMS applications using the v3 version of SNMP access to the Data Integrator MIB and Data Integrator supported standard MIBs
- Traps — Parameters that determine where the agent sends trap messages

4. Select a category and set configuration parameters for your SNMP agent. For details on each parameter category, see the next section, "SNMP configuration parameters".
5. Click **OK** after you enter the correct configuration parameters for the Data Integrator SNMP agent.
6. In the Data Integrator Server Manager window, click **Restart**.

   The Data Integrator Service restarts, which restarts the Data Integrator SNMP agent using the new configuration information.

SNMP configuration parameters

Job Servers for SNMP

Use this category to select any number of Job Servers and enable SNMP for them. When SNMP is enabled for a Job Server, the SNMP agent maintains and reports data for jobs that run on that Job Server.

The editor lists each configured Job Server in one of two columns: **Not enabled for SNMP** or **Enabled for SNMP**. To enable SNMP for a Job Server that is not enabled, select that Job Server and click **Enable**. To disable SNMP for a Job Server that is enabled, select that Job Server and click **Disable**. Similarly, to enable SNMP for all the configured Job Servers, click **Enable All**; to disable SNMP for all configured Job Servers, click **Disable All**.

You can also enable or disable SNMP for individual Job Servers using the Job Server Configuration Editor. See "To enable SNMP on a Job Server" on page 550.

System Variables

Use this category to set parameters that affect the SNMP agent's operation.

Parameter	Description
Minimum SNMP version	Select the earliest version of SNMP that NMS applications will use to query the agent: **v1**, **v2c**, or **v3**. Business Objects recommends that you not use v1. The security mechanism used by v1 is not robust, and trap messages sent by the agent are not compatible with v1. **NOTE**: Some NMS applications use v1 by default. If other devices or agents that the application monitors support v2c or v3, Business Objects recommends that you reconfigure the NMS application. If not, then you must set this value to **v1**.
Agent port	Enter the port at which the agent listens for commands (PDUs) from NMS applications and responds to those commands. The default port is 161, the standard SNMP input port.
System name	Enter the name of the computer.

Parameter	Description
System contact	Optional. Enter text that describes the person to contact regarding this system. This text is reported to the NMS application. Network monitors might contact this person to resolve identified problems.
System location	Optional. Enter text that identifies this computer such as physical location information. This text is reported to the NMS application.
JobTable cache lifetime (in min)	Enter the maximum number of minutes a job will remain in the Data Integrator MIB after the job completes. The agent summarizes jobs (that is, eliminates individual data flow and work flow records) after one-eighth of a job's lifetime. The agent eliminates jobs completely after reaching the lifetime limit. Default lifetime is 1440 (one day).
JobTable cache max size (in KB)	Enter the maximum number of kilobytes that the agent can use for the Data Integrator MIB. If the MIB reaches this size, the agent reduces the MIB by 20 percent by eliminating completed jobs, starting with the oldest jobs. The default is 819 (0.8 Megabytes) which will store approximately 1000 jobs.

Access Control, v1/v2c

Use this category to enter the information that allows NMS applications using SNMP version v1 or v2c to access the MIBs controlled by the Data Integrator SNMP agent.

NOTE: If an NMS application monitoring the Data Integrator SNMP agent uses SNMP version v1 or v2c, you must set the **Minimum SNMP version** to either **v1** or **v2c** under the **System Variables** category.

The editor lists community names and the type of access permitted.

To enable access for a new community:

a. Click **Add**.

b. Select **Read-only** to permit this community to read the Data Integrator MIB only.

With this setting, this community is not permitted to send `SetRequest` commands to any MIB or `GetRequest` commands to a standard SNMP MIB for trees that contain security information such as community strings, user password, or encryption pass phrases.

c. Select **Read-write** to permit this community to send `SetRequest` commands for all read-write variables in any MIB and `GetRequest` commands for variables in any MIB.

Remember that selecting this option gives the community the capability of reading and then modifying variables in the standard SNMP MIBs.

d. In **Community name**, enter the community name permitted to send requests to this agent.

Names are case-sensitive and must be unique. Typically, the administrator of the NMS application assigns the name. The NMS application includes this name in all requests to the agent.

e. Click **OK**.

To edit a community's name or access:

a. Select the community name and click **Edit**.

b. Change access type and community name as desired.

c. Click **OK**.

To delete access for a particular community:

a. Select the community name.

b. Click **Delete**.

Access Control, v3

Use this category to enter the information that allows NMS applications using SNMP version v3 to access the MIBs controlled by the Data Integrator SNMP agent.

The editor lists user names along with properties of each user.

To enable access for a new user:

a. Click **Add**.

b. Enter appropriate information for the user.

Parameter	Description
Read-only	Select **Read-only** to permit this user to read the Data Integrator MIB only. With this setting, this user is not permitted to send SetRequest commands to any MIB or GetRequest commands to a standard SNMP MIB for a tree that contains security information such as community strings, user passwords, or encryption passphrases.
Read-write	Select **Read-write** to permit this user to send SetRequest commands for all read-write variables in any MIB and Get request commands for variables in any MIB. Remember that selecting this option gives the user the capability of reading and then modifying variables in the standard SNMP MIB.
User name	Enter a name of a user to which the Data Integrator SNMP agent will respond. Names are case-sensitive. Each name must be unique. Typically, the administrator of the NMS application assigns the name. The NMS application includes this name in all requests to the agent.
Password	Enter the password for the user. The password is case-sensitive.
Confirm password	Re-enter the password to safeguard against typing errors.

c. Click **OK**.

To edit a user's name or access data:

a. Select the user name and click **Edit**.

b. Change access data, user name, and password as desired.

c. Click **OK**.

To delete access for a user:

a. Select the user name.

b. Click **Delete**.

Traps

Use this category to configure where to send traps.

Select the **Enable traps for authentication failures** check box if you want the agent to send traps when requests fail due to authentication errors, such as incorrect passwords or community names, in addition to traps about job errors.

The editor lists the receivers of the trap messages sent by the Data Integrator SNMP agent. A receiver is an NMS application identified by a machine and port.

To add a new trap receiver:

a. Click **Add**.

b. Enter identifying information about the trap receiver.

Parameters	Description
Machine name	Enter the name of the computer or the IP address of the computer where the agent sends trap messages. This is a computer where an NMS application is installed.
Port	Enter the port where the NMS application listens for trap messages. The default value is 162, the standard SNMP output port.
Community name	Enter the community name that the NMS application expects in trap messages.

c. Click **OK**.

To change information for a trap receiver:

d. Select the trap receiver and click **Edit**.

e. Update the identifying information about the trap receiver.

f. Click **OK**.

To delete a trap receiver:

a. Select the trap receiver.

b. Click **Delete**.

SNMP configuration on UNIX

This section lists the procedures to configure SNMP on UNIX. For more detailed descriptions about the options mentioned here, see "SNMP configuration in Windows" on page 549.

➢ To select a Job Server to support SNMP communication

1. Run the Server Manager. Enter:

```
$ cd $LINK_DIR/bin/
$ . ./al_env.sh
$ ./svrcfg
```

NOTE: The second command sets the environment variables before running the Server Manager.

2. Select option `2` to configure a Job Server.

```
*Data Integrator Server Manager Utility *

1 : Control Job Service
2 : Configure Job Server
3 : Configure Access Server
4 : Configure Web Server
5 : Configure SNMP Agent
x : Exit

Enter Option: 2
```

3. Enter option `e : Edit a JOB SERVER entry`.
4. Enter the serial number of the Job Server you want to work with when you see the following question:

   ```
   Enter serial number of Job Server to edit: 1
   ```

5. Enter a number that will be used as the SNMP communication port when you see the following question:

   ```
   Enter TCP Port Number for Job Server <S1>
   [19111]:
   ```

6. Enter 'y' when prompted with the following question:

   ```
   Do you want to manage adapters and SNMP
   communication for the Job Server 'Server1''Y|N'
   [Y]?:
   ```

When you return to the Current Job Server Information page, the Job Server set to manage adapters or SNMP is marked with an asterisk and noted below the list of Job Servers.

```
_____________________________________________________________

** Current Job Server Information **
_____________________________________________________________

   S#      Job Server Name  TCP      Enable     Repository
                            Port     SNMP       Connection
   --      ---------        -----    -------    ----------
   1*      Server1          19111    Y          repo1@orasvr1
   2       Server2          19112    N          repo2@orasvr1

*:JobServer <S1> supports adapter and SNMP communication
on port:19110
_____________________________________________________________

  c: Create a new JOB SERVER entry   a: Add a REPO to job server
  e: Edit a JOB SERVER entry         y: Resync a REPO
  d: Delete a JOB SERVER entry       r: Remove a REPO from job server
  u: UPDATE a REPO                   s: Set default REPO
  q: Quit

Enter Option: q
```

7. To exit the Server Manager, enter `q`, then enter `x`.

➢ To enable SNMP on a Job Server

1. Run the Server Manager. Enter:

```
$ cd $LINK_DIR/bin/
$ . ./al_env.sh
$ ./svrcfg
```

2. Select option `2` to configure a Job Server.

3. Enter option `e : Edit a JOB SERVER entry`.
4. Enter `y` when prompted with the following question:

```
Do you want to Enable SNMP for this
JobServer 'Y|N' [N]:
```

5. To exit the Server Manager, enter `q`, then enter `x`.

➢ To configure an agent

1. Run the Server Manager. Enter:

```
$ cd $LINK_DIR/bin/
$ . ./al_env.sh
$ ./svrcfg
```

2. Select option `5` Configure SNMP Agent

```
*Data Integrator Server Manager Utility *

1 : Control Job Service
2 : Configure Job Server
3 : Configure Access Server
4 : Configure Web Server
5 : Configure SNMP Agent
x : Exit

Enter Option: 5
```

One of these options appears based on the current configuration:

- SNMP is Disabled for this installation of Data Integrator

```
[ D = Keep Disabled / E = Enable ]? :
```

- SNMP is Enabled for this installation of Data Integrator

```
[ E = Keep Enabled / D = Disable ]? :
```

Once you enable SNMP for Data Integrator, the SNMP configuration menu appears.

The following is a sample SNMP configuration menu screen.

```
                    SYSTEM VARIABLES
                    ---------------
 Minimum SNMP Version: v2c                Default Port: 4961

System Name: hpsrvr3
System Contact: sysadmtr
System Location:
JobTable Cache Lifetime: 1440    JobTable Cache Max Size: 819
(in min)                         (in KB)


                    Access Control, v1/v2c
                    ----------------------

Read Community string : read_v2
Write Community string : write_v2


                    Access Control, v3
                    ----------------------

Read User : read_v3
Write User : write_v3
```

```
                    SNMP TRAPS
                    ----------

Authentication Traps : Enabled

Agent is configured to send trap to:
     aixserver3:20162 with Community String trap_co

     1 - Modify System Variables
     2 - Access Control, v1/v2c
     3 - Access Control, v3
     4 - Modify Traps
     X - Exit to previous Menu

Enter Option:
```

- To modify system variables, choose `1`. Values for each variable display. Press `ENTER` to keep the default values or enter new values for each variable.
- To modify SNMP v1 and V2c community names, choose `2`.

A submenu displays. At the prompt, enter a new value or press RETURN to keep the original value.

```
Access Control, v1/v2c
----------------------

Read Community string : read_v2
R - Add READ Community String
W - Add WRITE Community String
D - Delete Community String
S - Save Changes
X - Exit without save to Previous Menu

Enter Option:
```

- To modify SNMP v3 USER names, choose 3.

  A submenu displays. At the prompt, either enter a new value or press RETURN to keep the original value.

```
Access Control, v3
------------------

Read User : read_v3
R - Add READ User
W - Add WRITE User
D - Delete User
S - Save Changes
X - Exit without save to Previous Menu

Enter Option:
```

- To modify TRAP setup, choose 4.

A submenu displays. At the prompt, either enter a new value or press RETURN to keep the original value.

```
TRAP SETUP
----------

Authentication Traps : Enabled
Agent is configured to send trap to:
     aixserver3:20162 with Community String trap_co

A - Add TRAP receiver
D - Delete TRAP receiver
E - Enable sending traps on Authentication failures
F - Disable sending traps on Authentication failures
S - Save Changes
X - Exit without save to Previous Menu

Enter Option:
```

Use this menu to:

- Enable/disable authentication traps.
- Configure a trap receiver (host name and port number on which the trap receiver is listening, and community string the trap receiver will use to authenticate the traps).

  A "+" indicates newly added names.

  A "-" indicates deleted names.

  After you enter "S", all additions and deletions display and you are prompted to confirm them.

Troubleshooting

➢ To troubleshoot the Data Integrator SNMP agent

1. Check that you are using a valid v1/v2c community name or v3 user name.

   The SNMP agent does not reply to unauthorized requests. Unauthorized requests will fail due to a time-out. To determine whether the agent regards a request as unauthorized:

   a. Under the **Traps** category of the SNMP Configuration Editor:

      - Set a trap receiver
      - Select the **Enable traps for authentication failures** check box

   b. Restart the Data Integrator SNMP agent.

   c. Inspect the output of the trap receiver for authorization traps.

2. Increase the SNMP agent's time-out.

   Repeat until the agent responds or the time-out exceeds 20 seconds. If the agent does not respond and the time-out is more than 20 seconds, revert to the original time-out setting, and try the next step.

3. Verify that the agent and the NMS application are using compatible versions of SNMP.

   Under **System Variables** in the SNMP Configuration Editor, the **Minimum SNMP version** must not be greater than the version of the messages that the NMS application sends. If the versions are incompatible, change the NMS application setting or the configuration of the Data Integrator SNMP agent.

For example, if the NMS application sends messages using SNMP version v2c or v3, you can set the **Minimum SNMP version** to **v2c**.

NOTE: Some NMS applications use v1 by default. If other devices or agents that the NMS application monitors support v2c or v3, Business Objects recommends that you reconfigure the NMS application. If not, then you must configure the Data Integrator SNMP agent to accept version v1 commands.

4. Check errors in the SNMP error log:

   "*installation directory*"/bin/snmp/snmpd.log

   Use the Server Manager to resolve errors, if possible.

5. Contact Business Objects technical support. In your defect report, include:
   - The exact NMS command you are trying to run and the exact error message
   - Name and version of the NMS application you are running and the version of SNMP protocol that application uses.
   - Copies of the following four files from "*installation directory*"/bin/snmp directory:
     - snmp.conf
     - snmpd.conf
     - snmpd.p.conf
     - snmpd.log

6. Check that encryption pass phrases are accessible.

   Encryption passphrases are locked to a given SNMP engine ID based on the IP address of the computer running the SNMP agent. If you change the IP address of that computer, you must re-create all your SNMP users in the Data Integrator Server Manager.

A Glossary

This glossary defines terms used in this documentation set including common terms that have a special meaning with respect to Data Integrator. Within a definition, other terms that are defined in the glossary appear in **bold type**.

ABAP

Advanced Business Application Programming.
A fourth-generation programming language developed by SAP in which R/3 applications are written.

ABAP program

A program that executes database operations on an **SAP R/3** server. Data Integrator **R/3 data flows** generate ABAP programs.

Access Server

A process that starts, stops, and monitors real-time services (**real-time jobs**). The Access Server dispatches requests to real-time services, ensuring optimal load balancing and complete life cycle management. The Access Server also starts, stops, and monitors **adapters**.

adapter

An external Data Integrator **interface**. There are two types of adapters:

- Custom adapters — Adapters developed using the Data Integrator Adapter Development Kit
- Prepackaged adapters — Adapters prebuilt and purchased from Business Objects, such as the Data Integrator MQSeries Adapter

Administrator

A browser-based system administration application that supports Data Integrator. Use the Administrator to configure and manage:

- Services (**real-time jobs**)
- Service providers (instances of real-time services)
- Clients (Web application clients)
- Batch jobs
- Adapters
- Job logs

after-image

The values in an UPDATE row after the row changes. You might want to pair before- and after-images of UPDATE rows when designing log-based changed-data capture jobs which Data Integrator supports.

aggregate data

Data that results when a process combines elements. This data can be presented collectively or in summary form.

ALE

Application Link Enabling. An SAP R/3 programming-related interface designed to allow reliable communication across a distributed environment.

Analytic Rapid Mart

An **Data Integrator application** that produces a predefined **data warehouse** for a specific area of business analysis. Analytic Rapid Marts are typically based on dimensional modelling. With Analytic Rapid Marts, users can manage large volumes of data.

attribute

A **property** created for a type of **object**.

BAPI

Business Application Programming Interface. A standardized SAP R/3 programming interface that allows non-SAP R/3 applications to access specific business processes and data.

back-office application

A software system that an enterprise uses to run business operations, such as human resources, general ledger, and finance. Back-office applications store data in a relational **database** optimized for operational use.

There are several types of back-office applications:

- Enterprise resource planning (ERP) — For example, SAP R/3, BAAN, J.D. Edwards, PeopleSoft
- Supply chain management (SCM) — For example, i2
- Customer relationship management (CRM) — For example, Siebel, Clarify
- Legacy

Basis

The SAP infrastructure. Basis is the foundation for all SAP products.

batch job

A group of objects (data flows, work flows, conditionals, scripts, and so forth) that executes on a schedule. This type of job is recommended for operations on large amounts of data.

before-image

The values in an UPDATE row before the row changes. You might want to pair before- and after-images of UPDATE rows when designing log-based changed-data capture jobs which Data Integrator supports.

bulk loading

A software-based mechanism that moves large data files. This mechanism supports compression, blocking, and buffering to optimize transfer times.

business component

A set of tables Siebel applications use to create a logical object called a business object.

Business Objects Data Platform

A set of applications from Business Objects, Americas, that enterprises can use to manage all types of data—from data for supply chain partners to data for customers. The Business Objects Data Platform consists of **Data Integrator** and one or more **Rapid Marts**.

CDC datastore

A CDC datastore allows you to limit extracted data to changed data only. A CDC datastore connects a **changed-data capture** table on a source database to Data Integrator.

CDC checkpoint

A CDC checkpoint enables Data Integrator to restrict CDC subscription reads. After you enable a checkpoint, the next time the CDC job runs, it reads only the rows inserted into the CDC table since the last checkpoint.

CDC Subscription

A CDC subscription name allows different data flows to extract data from the same table without corrupting data extracted by other data flows. The datastore and table name can be the same; however, the table owner name must be different in each source CDC table.

changed-data capture

The process of retrieving changes made to a production data source. This process consolidates units of work, ensures data is synchronized with the original source, and reduces data volume in a warehouse environment.

Citrix MetaFrame XP

Citrix MetaFrame XP software provides an access infrastructure for enterprise applications. You can use this software to run Data Integrator on a server which publishes instances of the Designer and other Data Integrator interfaces to users on client computers.

client/server

A distributed technology approach where the processing is divided by function. The server performs shared functions (such as managing communications and providing database services), while the client performs individual user functions.

COM

Component Object Model. Provides a set of interfaces that allow clients and servers to communicate within the same computer (running on Windows 95 or later versions).

Commerce Rapid Mart

An **Data Integrator application** that produces a predefined **data warehouse** and that includes logic to support real-time requests for data from a **back-office application**. A Commerce Rapid Mart is usually based on different data model than an **Analytic Rapid Mart**. With a Commerce Rapid Mart, users can integrate a company's back-office system with an **e-commerce** application.

Communication Structure

In **SAP BW**, a data structure that defines a set of **InfoObject**s available from an **InfoSource** to put into **InfoCube**s.

conditional

A **single-use object**, available in **work flow**s**,** that allows you to branch the execution logic based on the results of an expression. The conditional takes the form of an if/then/else statement.

custom ABAP program

A custom ABAP program runs an **ABAP program** and generates a **data set**. With a custom ABAP program, you can run an existing ABAP program as part of a job. Use a custom ABAP program as a **source** in a **data flow** or an **R/3 data flow**.

data extraction

The process of moving data from a **database** or application source to a database target (either from a legacy database to a **data mart**, or from one data mart to another).

data flow

A **reusable object** containing **steps** to define the transformation of data from **source** to **target**. Data flows are called from inside a **work flow** or **job**. You can pass information into or out of data flows using **parameters**.

Data Integrator

A software system that allows users to build and execute applications with which they can create and maintain **data warehouses**.

Data Integrator consists of several components:

- **Access Server**
- **Data Integrator engine**
- **Data Integrator repository**
- **Data Integrator Service**
- **Administrator**
- **Designer**
- **Job Server**

Data Integrator application

A set of **jobs** that process batch data extractions and real-time requests using data from a variety of data sources.

Data Integrator engine

The core process that reads job information from the **Data Integrator repository** and sets up run-time processes that execute the job. The run-time processes extract, transform, and load relational and hierarchical data. The **Job Server** starts the Data Integrator engine to execute batch or real-time jobs.

Data Integrator interface

A program that Data Integrator uses to access data sources. Specific interfaces vary by installation. There are internal interfaces—those native to the installation—and external interfaces—those that you install separately. Internal interfaces allow Data Integrator to access applications like **SAP R/3** and **SAP BW**, **messages**, relational database systems, and **legacy systems**. An external interface is also known as an **adapter**. It allows Data Integrator to access applications using information exchange technologies such as Web Services and MQ Series.

Data Integrator repository

The database that contains information about an Data Integrator application. The repository contains information about defined **reusable objects**, the **metadata** for tables, and built-in **transforms** and **functions**. When you invoke Data Integrator, you log in to the repository containing the **objects** you want to use.

Data Integrator Service

The process that ensures that the **Access Server** and the **Job Server** are running. You can configure the Data Integrator Service to restart the Access Server and Job Server whenever the computer where they are located restarts.

data loading

The process of populating a **data warehouse**. Data loading is provided by **DBMS**-specific load processes, DBMS insert processes, and independent fast-load processes.

data mapping

The process of assigning a source data element to a target data element.

data mart

A highly-focused version of a **data warehouse**. Typically, created by a department or division of a company, data marts contain data for a specific subject area, such as finance or sales. Data Integrator can populate a data mart.

data transformation

The process of filtering, merging, decoding, and translating source data to create validated data for a **data warehouse**.

data set

Rows of data with a defined schema. A step in a **data flow**—such as reading data from a **source**, joining data in a **Query transform**, or transforming data though another **transform**—yields a data set. You can view individual data sets by placing a **target** table or file at that point in the **data flow**.

data transport

A step in an **R/3 data flow** that defines a **target** to store the **data set** extracted during the flow. You can locate the target file on the **SAP R/3** server or in a location accessible to both the SAP R/3 server and to Data Integrator across a network.

data type

The format used to store a value. Data types can imply a default format for displaying and entering the value. Data read from a **source** is converted to the appropriate Data Integrator data types; data loaded to a **target** is converted from its Data Integrator data type to the type appropriate for the target.

data warehouse

A relational **database** containing subject-oriented information that supports decision analysis. Typically, information in a data warehouse is integrated from multiple sources, including operational sources. Data warehouses provide a single, non-volatile point of reference for a particular time period. Data warehouses are composed of **data marts.** Data Integrator can populate a data warehouse.

database

A collection of **table**s managed by a **DBMS** such as Microsoft SQL Server or Oracle.

DataConnector

DataConnector operator instances are used to read data files generated by Data Integrator when performing bulk loading using the Teradata Warehouse Builder.

datastore

A logical channel connecting Data Integrator to a **database**. The datastore definition includes the name and location of the database as well as user authentication information. Data Integrator uses a datastore definition to qualify a table name wherever a table is indicated in a diagram or expression. You can access the datastore definition through the **object library**.

DBMS

Database management system. A software system that builds and maintains database tables.

debug mode

Allows you to debug a job using the **interactive debugger** features in the Designer.

declarative specification

A way of indicating the desired results of a **data transformation** without placing constraints on the method of reaching the results. When you create a **data flow** in Data Integrator, you specify what data you want to read from which **source**, what you want done to that data, and where you want it loaded. What you specify constitutes the declarative specification. Data Integrator then determines the most efficient way to reach the requirements you set, including pushing operations to the source **DBMS** for execution.

degree of parallelism

You can set a value at the data flow level that allows you to parallel-process a **transform** within a data flow.

For example, if from the Properties window of a data flow you set the **Degree of parallelism** to 5, then when the job executes, Data Integrator replicates each transform in the data flow five times. Each of these replicated transforms executes in parallel using a separate process thread. Using a value for degree of parallelism can improve job performance if a **Job Server**'s computer memory and number of CPUs supports a job's parallel-processing configuration settings.

dependency analysis

A category of metadata reports that produces reports based on search criteria you specify to find specific objects in the repository. For each object returned, dependency analysis reports provide object lineage as well as object impact information.

derived data

Data that results from a computational step applied to reference or event data. Derived data is the result of either relating two or more elements of a single transaction (such as aggregation), or relating one or more elements of a transaction to an external algorithm or rule.

Designer

A graphical user interface that allows you to design and test an **Data Integrator application**.

diagram

The icons and connections between the icons that make up the definition of a **job**, **work flow**, or **data flow**.

DMZ

Demilitarized zone. A perimeter network that is more accessible than your most protected network, but located inside a security layer protected from the internet.

drill down

A method of exploring detailed data that was used in creating a summary level of data. Drill down levels depend on the granularity of the data in the **data warehouse**.

DTD

Document type definition. A text file that describes the elements (tags) in an **XML** document and the relationship among them. When an XML document is used to describe a transaction, the DTD describes the data schema used in the transaction.

e-commerce

Buy and sell transactions conducted over the internet.

EDI

Electronic Data Interchange. Electronic exchange of structured data between businesses. This exchange is not dependent on hardware, software, or communication protocols.

element

A component found within XML Schemas and DTDs.

embedded data flow

A data flow created within one or more data flows.

enterprise data

Data that is defined for use across a corporate environment.

ERP system

Enterprise resource planning system. A **back-office application** from which Data Integrator can extract data.

exception

An error that occurs while executing a **job**. You can catch individual or groups of exceptions using a **try/catch block** inside a **work flow**. Catching an exception allows you to automatically execute a solution for the error.

Executive Information Systems (EIS)

Tools programmed to provide canned reports or briefing books to executives. They offer strong reporting and drill-down capabilities. These tools allow ad hoc querying against a multi-dimensional **database**, and most offer analytical applications along functional lines such as sales or financial analysis.

extract date

The date data was extracted.

extract frequency

The interval at which data is extracted, such as daily, weekly, monthly, or quarterly. The frequency that data extracts are needed in the **data warehouse** is determined by the shortest frequency requested through an order, or by the frequency required to maintain consistency of the other associated data types in the source data warehouse.

file

A **source** or **target** for data. To read or load data to a file within Data Integrator, define a **file format** that instructs Data Integrator about the structure of the data in the file.

file format

A description of how data is or should be organized in a **file** Data Integrator reads from or loads to. A file format can be specific to a single file or generic for many files.

front-office application

Software that companies use to support reporting, **e-commerce**, and other tasks. Types of front-office applications include:

- Supply chain collaboration
- Analytical
- Sell-side e-commerce
- Customer relationship management

function

A program that operates on values that are passed to it. Data Integrator functions are available through a function wizard in a **script**, **conditional**, or **Query transform**. Data Integrator also gives you access to functions provided by the **DBMS** you are using. In addition, you can define your own functions using the Data Integrator scripting language or by writing C-language programs.

IDoc

Intermediate Document. An SAP-specific format. Used for **EDI** and **ALE**.

IDoc type

Indicates the SAP format that is used to interpret the data of a business transaction. Consists of the following components:

- A control record

  Identical for each IDoc type.

- Several data records

  A single data record consists of a fixed key part and a variable data part. The data part is interpreted using segments, which differ depending on the IDoc type selected.

- Several status records

  Identical for each IDoc type. Describe the status states an IDoc has already passed through or the status an IDoc has attained.

import

The process of acquiring information for the **Data Integrator repository**. Import the following kinds of information into Data Integrator:

- The **metadata** for **source** and **target** databases
- Descriptions and code for user-defined and DBMS **functions** and **transforms**

InfoArea

In **SAP BW**, an arbitrary grouping of **InfoCube**s.

InfoCube

In **SAP BW**, a multidimensional star schema that is limited to one fact table and 16 dimensions.

InfoCubes store only numeric data at the fact level.

SAP BW reserves three of the dimensions for currencies, time, and units of measure. Dimensions have to be linked back to master data and hierarchy tables to gain access to the information stored there.

When a user query is initiated, BW creates a temporary **OLAP** cube that stores a constrained subset of the permanent OLAP cube. The temporary InfoCube persists until the user disconnects from BW or changes the query.

InfoObject

In **SAP BW**, an InfoObject is equivalent to a data element and its associated metadata (for example, description or business rules). Can be a **Key Figure** or a Characteristic.

InfoPackage

In **SAP BW**, a schedule for running BW updates.

Information resource (IR)

A producer or consumer of enterprise information. An information resource can be a flat file, a **back-office application**, an EAI system, or a Web application.

InfoSource

In **SAP BW**, a logical construct that defines a set of **InfoObject**s and their mappings to source data elements. InfoSources do not store data, but rather serve as conduits via **Transfer Structure**s.

Integration Rapid Mart

An **Data Integrator application** that provides point-to-point content mapping and data movement mechanisms between different **back-office applications**.

interactive debugger

A Designer feature that allows you to step through the data of a job one row at a time using filters and breakpoints on a line. Like executing a job, you can start the interactive debugger from the Debug menu when a job is active in the workspace. While in debug mode, all other Designer features are set to read-only.

interface

Data Integrator offers two types of interfaces:

An internal **Data Integrator interface**, allows you to create datastore connections to natively supported applications.

An external **Data Integrator interface** (or **adapter**), allows Data Integrator to communicate with information exchange technologies such as Web Services and MQ Series.

IR

See **Information resource (IR)**.

job

The unit of work that can be scheduled independently for execution by the **Administrator**. Jobs are special **work flows** that can be called by other work flows or jobs.

Job Server

A process that receives requests from the **Designer** and the **Administrator** to start and stop **jobs**. To start batch or real-time jobs, the Job Server triggers the **Data Integrator engine**. Engine processes run on the same computer as the Job Server process that triggers them.

join rank

A value given to or calculated for all data sets in a **data flow**. Data Integrator uses the join rank to determine which **source** to read first when assembling the **data set** in a join. Data Integrator uses the source with the lower join rank as the inner source of the join and uses the source with the higher join rank as the outer source of the join. If more than one of the sources is located in the same **DBMS**, the highest rank is used for all of the sources from that DBMS. When a source is processed through a **Query transform** or another **transform**, and then used in a join, the maximum join rank defined for the source is used as the join rank in the query.

Key Figure

In **SAP BW**, an **InfoObject** that represents a numeric fact.

legacy system

An information or transaction processing system used to store data such as bank balances, inventories, payroll, and manufacturing parts.

license-controlled feature

A Data Integrator feature that is enabled or disabled based on the product license. The product license controls which icons and settings are available in Data Integrator as an internal **Data Integrator interface**.

line

The line you use to connect objects in a workspace diagram. Lines represent a left-to-right flow path for data moving between two objects during job execution.

memory datastore

A datastore connection/container for memory tables.

memory table

Internal Data Integrator table used to store a data set in memory while a job runs. Use instead of staging tables to improve performance of a real-time job built with multiple data flows. Use a memory table to move a data set between data flows.

message

Represents hierarchical data, such as a header with line items, for document-oriented transactions, such as a purchase order.

metadata

In Data Integrator, information acquired and maintained to describe tables in **source** and **target** databases. This information includes the names of tables and their columns, and the data types of the columns. All the metadata for all user tables is imported at one time for a given **datastore**.

In general, metadata typically includes a description of data models, a description of the layouts used in database design, the definition of the system of record, the mapping of data from the system of record to other places in the environment, and specific database design definitions.

multiple data flow model

One of three data flow models you can use in real-time jobs. Use this model to develop real-time jobs in separate logical object segments (data flows) rather than in a single data flow. Allows for any number of data flows in a real-time processing loop under these conditions: The first data flow must have a single message source, the last data flow must have a single message target. Other data flows in the same real-time job cannot have a message source or message target. See also **put/acknowledge data flow model** and **single data flow model**.

nested data

Data in one **table** that is related to a single row of another table. A nested table appears in Data Integrator as a column in a parent table. Columns in the nested table can themselves contain tables.

object

Any item that you create in the **Designer**. Data Integrator distinguishes two classes of objects: **reusable objects** that are complete and can be reused in your projects (such as **data flows**) and **single-use objects** that only appear as components of other objects (such as a **try/catch block**). This distinction affects how you create and retrieve each type of object.

object definition

The options that describe the operation of an **object**. To view and modify an object definition, open the object so that its definition appears in the **workspace**.

object library

A tool in the **Designer** that gives you access to reusable objects.

ODBC

Open database connectivity. A standard for **database** access initiated by the SQL Access Group consortium and now owned by Microsoft.

ODS

Operational data store. An OLAP-designed relational **database** that an enterprise has designated as the operational database of record (for example, a finance department might use an ODS to close its books).

OLAP

Online analytical processing. A relational **database** design that is optimized for random access. OLAP systems are used in a query environment, such as for a **front-office application**.

OLTP

Online transaction processing. A relational database design optimized for operational use. OLTP systems are used in an operational environment, such as for a **back-office application**.

operation code

A flag associated with a row in a **data set** that indicates the status of the data in the row. The operation codes are INSERT, UPDATE, DELETE, and NORMAL.

operational statistics

A category of reports within the metadata reporting tool that provides you with job and data flow performance information over a given period of time or number of executions.

parameter

A value passed to a **work flow** or **data flow** when that flow is called.

partition

To divide table data into sets based on a criteria such as a range or list of values in each row. You can configure Data Integrator to read and write partitioned table data in parallel threads. Designing jobs with partitioned table data can improve job performance if a Job Server's computer memory and number of CPUs supports the job's parallel-processing configuration settings.

project

The collection of **jobs** available in the **Designer** at a given time. A project provides a way to organize the **objects** you create.

property

Detailed descriptive information about **objects**.

put/acknowledge data flow model

One of three data flow models you can use in real-time jobs. Similar to the multiple data flow model, this model does not require a data flow with a target message. Data Integrator creates a target message to automatically acknowledge a request message. Designed for use with IDoc sources in SAP R/3 systems. See also **multiple data flow model** and **single data flow model**.

Query transform

A **data transformation object** that you can use to map columns from a **source** to a **target** schema, add new columns to the target schema, determine the data to extract, and perform operations on the data. Like a SQL SELECT statement, a query creates a **data set** that satisfies the conditions you specify.

Rapid Mart

A prepackaged **Data Integrator application** that delivers data for a specific business purpose. Easily deployed and customized, BusinessObjects™ Rapid Marts include the extraction, transformation, and data loading logic to support analysis or **e-commerce**. There are three types of Rapid Mart:

- **Analytic Rapid Mart**
- **Commerce Rapid Mart**
- **Integration Rapid Mart**

R/3 data flow

A **data flow** that extracts data from an **SAP R/3** source table. Data Integrator translates steps you define in an R/3 data flow into **ABAP** and then passes the ABAP program back to your SAP R/3 system for execution. The resulting **table** or **file** resides on the SAP R/3 system to be used as a **source** in the parent data flow.

real-time job

A group of objects (data flows, work flows, conditionals, scripts, and so forth) that execute on-demand as a "request-response" system. You design real-time jobs in the **Designer**, then configure them as real-time services and associate them with an **Access Server** in the **Administrator**, where they are started, managed and monitored. When a real-time service receives a request from a caller, it processes the request and returns a reply. This type of job is recommended for operations on small amounts of data.

relational data

A **data set** in which data in each column contains a scalar value. Data Integrator can process relational data; it can also process **nested data**.

repository

See **Data Integrator repository**.

reusable object

An **object** that can be defined, stored, and reused independent of other objects. Create reusable objects from the **object library**.

RFC server

Remote Function Call server. The Data Integrator RFC server allows third-party programs, including SAP R/3, to schedule and initiate Data Integrator jobs and return the results to Data Integrator.

SAP BW

Business Information Warehouse, an SAP product.

SAP R/3

An **ERP system**.

script

A step in a **work flow** that allows you to calculate values to pass to other parts of the work flow. The script can call **function**s, execute if-then-else statements, and assign values to **variable**s. Write a script in the Data Integrator scripting language.

segment

Format with which the data records of **IDocs** are interpreted.

server group

A defined collection of Job Servers on different computers. A server group automatically measures resource availability on each Job Server in the group and distributes scheduled batch jobs to the Job Server with the lightest load at run-time. Use the Server Groups node in the Administrator's navigation tree to group Job Servers that are associated with the same repository into a server group.

service request

Any message sent from a Web client that requires processing by a **real-time job**.

session

Associated with the **Adapter Properties** tab in an adapter datastore, a session is created by a stream-oriented adapter to group adapter components and maintain state for interactivity between Data Integrator and the adapter.

single data flow model

One of three data flow models you can use in real-time jobs. Allows for one data flow in a real-time processing loop. This data flow must have a single message source and a single message target. See also **multiple data flow model** and **put/acknowledge data flow model**.

single-use object

A step in a **work flow** or **data flow** that cannot be saved independently of the flow. Create single-use objects from the tool palette.

smart editor

A flexible editing tool in Data Integrator used for creating scripts, expressions, and custom functions without having to type the names of existing elements like column, function, and variable names.

SNMP

System Network Management Protocol helps network administrators manage network routing hardware. The protocol can manage a variety of hardware and software devices.

source

An **object** from which Data Integrator reads data.

SQL

Structured Query Language. A query language for accessing relational, **ODBC**, DRDA, or non-relational **database** systems.

SQL query tool

An end-user tool that accepts **SQL** to be processed against one or more relational databases.

star schema

A **database** design you can use to format data in a **data mart**. This design is based on a single fact table to which any number of dimensional tables may be joined. This type of database design supports multi-dimensional database analysis.

step

An **object** that is part of the definition of a **work flow** or **data flow**. Each step is represented by an icon in the diagram of the flow and is connected to other steps to indicate the flow of data through the data flow or the order of execution in the work flow.

table

A **database** table that Data Integrator reads data from or loads data into. The path and mechanisms for reading and loading data and apportioning the data among rows and columns are defined in the **datastore** that the table is associated with. Writing a **data set** to a database table means sending a combination of rows with appropriate **operation code** to the database table.

target

An object in which Data Integrator loads extracted and transformed data in a **data flow**. Data Integrator only loads rows flagged as INSERT or UPDATE.

TCP/IP

Transmission Control Protocol/Internet Protocol. The basic communication protocol of the internet, and often intranets and extranets. A computer having direct access to the internet contains a copy of the TCP/IP program. TCP/IP makes it possible for computers to communicate with each other.

Tdpid

(Teradata Director Program ID) The server name Data Integrator uses when loading with the bulk loader option. Data Integrator uses tdpid as a Teradata Warehouse Builder operator attribute.

Transfer Structure

In **SAP BW**, a data structure that defines a set of **InfoObject**s to be moved into BW. To an ETL tool like Data Integrator, a Transfer Structure looks like a table.

transform

A step in a **data flow** that acts on a **data set**. Data Integrator transforms are available through the **object library**. You can also import user-defined transforms and transforms from your **DBMS**.

Transformation Rule (Transfer Rule)

In **SAP BW**, a data manipulation rule between a **Transfer Structure** and a **Communication Structure**. The results of these transformation are globally available to all **InfoCube**s.

try/catch block

A combination of a try object and one or more catch objects that define alternate execution paths in case an error occurs during the execution of a **job**. You can tune try/catch blocks to trap specific errors and to provide general alerts or messages if an error occurs.

Update Rule

In **SAP BW**, a data manipulation rule between a **Communication Structure** and an **InfoCube**. The result of the data manipulation is specific to one InfoCube.

variable

A symbolic placeholder for a value. Data Integrator lets you define local variables and global variables.

Local variables pertain to the **work flow** in which they are defined. You can pass the value into another **work flow** or **data flow** using a **parameter**.

Global variables pertain to the **job** in which they are defined. With global variables, there is no need to define parameters between **objects** in the job. Global variables can also be selected at execution time. This eliminates the need to open the Designer to set global variable values.

Web Services

A standard platform for integrating applications. Web Services allow different programs, constructed in different languages, on different platforms to communicate with each other. DATA INTEGRATOR supports three XML web service technologies: SOAP, WSDL, and **XML Schema**.

work flow

A **reusable object** containing **steps** to define the order of **job** execution. Work flows call **data flow**s, but cannot manipulate data themselves.

You call work flows from inside other work flows or **job**s. You pass information into or out of work flows using **parameter**s. You reuse work flows by dragging existing work flows from the **object library**.

workspace

The window inside the **Designer** in which you define, display, and modify **objects**. The workspace for a **data flow** contains an area to build a **diagram** representing the data flow definition. The workspace for a **transform** contains an editor for modifying the transform options.

WSDL

Web Services Definition Language. Web services are self-contained, modular business process applications based on open Internet standards.

XML

Extensible Markup Language. This markup language is like HTML (Hypertext Markup Language) in that it specifies a standard with which you can define your own markup languages with their own sets of tags. XML allows you to define various tags with various rules, such as tags that represent business rules, tags that represent data description, or tags that represent data relationships.

XML Schema

The XML format used by Data Integrator to support message processing that includes **Web Services**. XML Schemas describe the data structure of an **XML** file or message. Data flows can read and write data to messages or files based on a specified XML Schema format. You can use the same XML Schema to describe multiple XML sources or targets. XML Schema properties include: Name, Description, Imported from, Root element name, and Namespace.

Index

T

Business Objects, Americas
3030 Orchard Parkway
San Jose, CA 95134
USA

info@businessobjects.com
www.businessobjects.com

Data Integrator

Administrator Guide

Version 6.5

Document Number DI-65-0101-001

January 30, 2004

Contents

1 Welcome

The BusinessObjects™ Data Integrator Administrator is a web-based application written entirely in JAVA. You can install BusinessObjects™ Data Integrator Administrator on a separate computer from the other Data Integrator components. It runs on the Data Integrator Web Server, which is supported by the Data Integrator Web Server service. The Administrator uses a JDBC connection to repositories.

Use the Administrator to:

- Set up users and their roles
- Add connections to Access Servers and repositories
- Manage the retention of Job Server and Access Server logs
- Access job data published for Web Services
- Schedule and monitor batch jobs
- Configure and monitor:
  - Access Server status
  - Real-time services
  - Client interfaces including SAP R/3 client interfaces (to read IDocs) and message traffic moving in and out of an Access Server
  - Adapter instances (a prerequisite for creating adapter datastores)

About this document

This guide describes how to use the Data Integrator Administrator. This guide contains the following chapters:

- Chapter 2, "Administrator User Interface" — Describes the Administrator and how to navigate through its browser-based interface.
- Chapter 3, "Administrator Management" — Describes the Repositories, Users, Access Servers, Status Interval, Log Retention Period, and Web Services options of the Administrator.
- Chapter 4, "Using Server Groups"— Describes how you can configure job servers into logical groups for load balancing.
- Chapter 5, "Batch Jobs" — Describes how to schedule, execute, and monitor batch jobs. Also Data Integrator includes information about exporting execution commands and using the Data Integrator Job Launcher.
- Chapter 6, "Real-Time Jobs" — Describes how to configure and monitor real-time jobs. This chapter includes information about working with services, service providers and client interfaces. Also includes information about configuring an Access Server's parameters (using the Server Manager).
- Chapter 7, "Real-Time Performance" — Explains the configuration parameters and statistics that support real-time jobs. Explains how to improve the performance of your real-time applications.
- Chapter 8, "Adapters" — Describes how to add an adapter to the Data Integrator system, how to start an adapter instance, and how to monitor an adapter's operation instances.
- Chapter 9, "Support for Web Services" — Describes how Data Integrator supports Web Services through the Administrator.

- Chapter 10, "Troubleshooting" — Describes how to use the Administrator to find and help resolve problems.
- Appendix A, "Glossary" — Defines terms used in the Data Integrator documentation set.

More Data Integrator product documentation

Consult the *Data Integrator Getting Started Guide* for:

- An overview of Data Integrator products and architecture
- Data Integrator installation and configuration information
- A list of product documentation and a suggested reading path

After you install Data Integrator, you can view technical documentation from many locations. To view documentation in PDF format, you can:

- Select **Start** > **Programs** > **Data Integrator** ***version*** > **Data Integrator Documentation** and choose:
  - Release Notes
  - Release Summary
  - Technical Manuals
- Select one of the following from the Designer's **Help** menu:
  - Release Notes
  - Release Summary
  - Technical Manuals
- Select **Help** from the Data Integrator Administrator

You can also view and download PDF documentation by visiting Business Objects Customer Support online. To access this Web site, you must have a valid user name and password. To obtain your user name and password, go to http://www.techsupport.businessobjects.com and click **Register**.

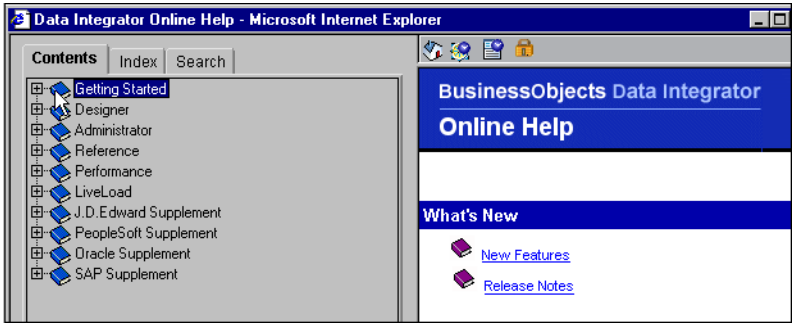
After installing Data Integrator Designer you can also view documentation in HTML format. **Online Help** opens automatically the first time you open the Designer.

You can also open **Help,** using one of the following methods:

- Choose **Contents** from the Designer's Help menu.
- Click objects in the object library or workspace and press **F1**.

  Online Help opens to the subject you selected.

Use **Online Help's** links and tool bar to navigate:



Online Help tool bar

Icon	Icon Name	Description
	Home	Returns to the Online Help home page.
	Show/Hide Navigation	Shows and hides the Table of Contents for the core suite of Data Integrator Technical Manuals. Includes a master **Index** and **Search** tabs.
	Show/Hide Descriptions	The Online Help window automatically updates its information when you click an object in the Designer window. In addition, you can click any object in the Designer window then click this button to show or hide an object's local and global variables.
	Start/Stop Help Updates	When the lock appears open, each time you click an object in the Designer window, the Online Help information automatically updates.
		When the lock icon appears closed, the Online Help window does not automatically update when you click objects in the Designer window; you must press F1.

2 Administrator User Interface

This chapter describes the Administrator and how to navigate through its browser-based, graphical user interface.

Specific sections include:

- Installation and configuration
- Logging in
- Navigation

Installation and configuration

See the *Data Integrator Getting Started Guide* for:

- General information about the components and architecture of Data Integrator
- Complete installation instructions for all Data Integrator components including connectivity testing for Data Integrator real-time functionality.

A summary of the connections used in Data Integrator is included here for your reference. You must complete the steps in the first four rows of the following table must before you can log in to the Administrator.

Data Integrator Connections

Component	Connection Tool	Connection Type	Purpose of this connection
Repository(s)	Repository Manager	Custom connection	• Connects Designer and repositories. • Provides location for storage of Data Integrator tables and job metadata. • Connection information is based on the database you use for a repository.
Job Server(s)	Server Manager	Default (3500) or custom port	Connects a Job Server to the Data Integrator Service and repository you specify. You can also set a Job Server to support adapters via a separate communication port (default 4001). Required to use the Adapter Instance node in the Administrator.
Access Server(s)	Server Manager	Default (4000) or custom port	Connects an Access Server to the Data Integrator Service and provides a port for Message Client libraries (allows applications to communicate with Data Integrator).
Administrator	Installer	Automatically assigned ports	Provides an HTTP port (28080) for connection between Administrator and all Access Servers. Use server-di.xml to change it. Includes an automatically assigned shutdown port (22828) which is not displayed. It is used by the Web Server service to start and stop the Data Integrator Web Server, which supports the Administrator.
Access Server(s)	Administrator	Computer name on which an Access Server is installed and port (you specified in Server Manager). For example: AT589:4000	Connects Access Server (s) to the Administrator
Repository(s)	Administrator	Settings based on each repository's database	Connects repositories to the Administrator. Job Servers (previously connected to each repository using the Server Manager) also link to the Administrator with this connection.

Logging in

The first time you log in to the Administrator, use the default user name and password. Business Objects recommends that you change the defaults thereafter.

➢ To log in to the Data Integrator Administrator

1. The initial step depends on your operating system:
   - On Windows, go to the **Start** menu, select **Programs** > **Business Objects** > **BusinessObjects Data Integrator *version*** > **Data Integrator Web Administrator**.

     The Administrator login page displays.

     If you encounter an error, check to see if the Data Integrator Web Server service is running. The Web Server and its service are installed with every Administrator. If the Web Server service is running, but you cannot log in, see "Reinstalling the Web Server service" on page 148.
   - On UNIX, open a browser, enter the following URL, then press Enter:

     http://hostname:28080/
2. Enter the default user name (`admin`) and password (`admin`) and click **Log in**.

   The home page opens.

   NOTE: Data Integrator Administrator sessions time out after 30 minutes of non-use. When this occurs, the log in window will re-appear. Re-enter your user name and password and click **Log in**.

Navigation

The layout of the Data Integrator Administrator consists of a window with a navigation tree on the left and pages on the right.

When the Administrator opens, it displays the Home page. The Home page shows the status of repositories, Access Servers, and adapters once they are connected to the Administrator. The red, green, and yellow icons indicate the overall status of each item based on the jobs, services, and other objects they support.

NOTE: Home page content does not display if you have not added a repository or Access Server connection to the Administrator. To see an adapter's status, enable a Job Server for adapters, then add the repository associated with that Job Server.

Navigation tree

The navigation tree is divided into four nodes: **Batch**, **Real-Time**, **Adapters**, **Management**.

Management node

The Management node refers to the configuration options for the Administrator component itself. Before you can use the Administrator, you must add connections to other Data Integrator components.

- From the **Management** tree, select **Repositories** and add a connection to the repositories that contain the jobs with which you want to work.
- From the Management tree, select **Access Servers** and add a connection to (for real-time jobs only) your Access Servers.

See Chapter 3, "Administrator Management" for more information.

Server Groups node

The Server Groups node allows you to group Job Servers that are associated with the same repository into a server group.

Use a server group if you want Data Integrator to automatically use the Job Server on a computer with the lightest load when a batch job is executed. This functionality improves load balancing (throughput) in production environments and also provides a hot backup method. When a job is launched, if a Job Server is down another Job Server in the same group executes the job.

For more information, see "Using Server Groups" on page 29.

Batch node

After you add a repository connection to the Administrator, you can expand the **Batch** node. Click a repository name to open the Batch Job Status page. Click the **All Repositories** option to see jobs in all repositories connected to this Administrator (only appears if more than one repository is connected).

The Status and Configuration tabs allow you to open pages to work with jobs in the selected repository.

The *Status* tab displays the status of executed jobs and job logs filtered by the Administrator's status interval. If more than 25 jobs exist, use the **Next** and **Previous** links to view them.

The *Configuration* tab displays job information sorted and filtered by project and allows you to configure existing objects such as executing a job or adding and exporting schedules.

Real-Time node

After you add a connection to an Access Server in the Administrator, you can expand the **Real-Time** node. Expand an Access Server name under the Real-Time node to view options.

Access Server node options	Description
Status	View status of real-time services and client interfaces supported by this Access Server. Control, restart, and set a service provider interval for this Access Server.
Real-time Services	View status for services and service providers, start and stop services, add or remove a service, configure Job Servers for a service.
Client Interfaces	View status for client interfaces, start and stop interfaces, add or remove an interface.
Logs - Current	View list of current Access Server logs, content of each log, clear logs, configure content of logs for display, enable or disable tracing for each Access Server.
Logs - History	View list of historical Access Server logs, view content of each log, delete logs.

Adapter Instances node

Use this node to configure a connection between Data Integrator and an external application by creating an adapter instance and dependent operations. This is a prerequisite requirement for creating a datastore for adapters in the Designer.

After you create a datastore, import data through the adapter and create jobs. Then use this node to view the status of Adapter instances. Options are listed by Job Server under the Adapter Instance node.

Pages

Each page has a unique name. Once you select a branch on the navigation tree to go to a page, use the page links and the tab row on the page to navigate further. The navigation tree does not redraw as you click page links and options. Only the first level of each branch displays in the navigation tree.

The tab row displays the page type (Status or Configuration). A dark blue (shaded) tab signifies the active page type. Click a light blue tab to go to that page type. Some pages do not include a tab row.

Below the tab row, use the grey navigation bar links as needed.

3 Administrator Management

Use the Management features to configure the Data Integrator Administrator.

This chapter discusses:

- Adding repositories
- Managing user roles
- Adding Access Servers
- Centralizing administration
- Setting the status interval
- Setting the log retention period
- Support for Web Services

Adding repositories

The Administrator allows you to manage batch and real-time jobs. To do so you must first add a repository connection to the Administrator so that you can view the jobs in that repository. After adding a repository connection:

- Jobs and logs (stored on a Job Server computer) appear in the Administrator.
- From the Batch Job Configuration page, you can execute batch jobs.
- Repositories become available to Metadata Reports users.

Working with the List of Repositories page

Use the List of Repositories page to connect an Administrator to a repository.

➢ To add a repository connection to the Administrator

1. Select **Management** > **Repositories**.
2. Click **Add**.

3. Enter the following information for the repository.

Option	Description
Repository Name	Logical name for a repository (used in the Administrator only).
Database type	The type of database storing your repository. There are several options: • DB2 • Informix • Microsoft SQL Server • Oracle • Sybase
Machine Name	Host name on which the database server is running.
Database Port	Port number of the database or data source.
Service Name/SID, Database name, Server name, or Data source	This field requires additional information based on the Database Type you select.
User name	The user or owner name for the database or data source.
Password	The user's account password for the database or data source.

4. (Optional) If you want to test the database information you have specified for the repository, before attempting to register it with the Administrator, you can click **Test**.
5. Click **Apply**.

   The Administrator validates repository connection information, and displays it on the List of Repositories page.

➢ To view the list of repositories connected to the Administrator

Select **Management** > **Repositories**.

The List of Repositories page lists the repositories that are connected to the Administrator. You can also remove a connection to a repository from this page.

NOTE: If you create a clean repository with the same name as a repository you have previously connected to the Administrator, you must reconnect the repository. To do so, go to the List of Repositories page, click the repository's name to open the Edit Repository page, then click **Apply**.

Adapter Considerations

To access adapter instances, a repository must be associated with a Job Server that is:

- Installed on the same computer as the adapter instance.
- Configured with the following adapter-related properties: **Support Adapter and SNMP communication** check box selected and the **Communication Port** number set. Configure these properties using the Server Manager utility.

If these conditions have not been met, you will not be able to use the **Adapter Instance** node of the Administrator.

Managing user roles

The Administrator allows you to add two types of user accounts:

Administrator	This role provides access to all of the Administrator's functionality
Monitor	This role provides access limited to options available from the Status tabs. For example, a Monitor can abort batch jobs but cannot execute or schedule them. A monitor can restart, abort or shut down an Access Server, service, adapter instance, or client interface but not add or remove them.

NOTE: Accounts you create for the Administrator also control access to the Metadata Reports tool (which you can access from the **Tools** menu in the Designer using the default user name and password). If you modify the `Admin` login ID or password, Metadata Reports tool users must use those new login IDs and passwords if they directly access Metadata Reports from its URL: `http://localhost:28080/reports/`. Repositories connected to the Administrator are also available to Metadata Reports users.

➢ To add users and their roles

1. Select **Management** > **Users**.
2. Click **Add** to open the Add Users page.
3. In the **Login name** box, enter a new user ID.

   Login names and passwords for the Administrator do not need to match those for your system or repository.
4. In the **Password** box, enter the new password.
5. In the **User Name** box, enter another identifier for the user such as the full name. If you have trouble recognizing a login name, you can use this value to label the account.

6. In the **Role** list, select a user role.

   Users with the Administrator role have rights to access all pages. Users with the Monitor role have rights to access all Status pages (to monitor, start, and stop processes) and the Status Interval and Log Retention Period pages.

7. In the **Status** list, select a status for this account.

   You may select active or suspended. If you want to delete the user, go to the User Management page.

8. Click **Apply**.

   View the new user in the **Users** table on the User Management page. You can also edit or delete user IDs using from the User Management page.

Adding Access Servers

The Administrator acts as a front end for Access Servers connected to it. Use the Administrator to:

- Configure real-time jobs as real-time services.
- Configure real-time services with service providers.
- Monitor Access Servers, real-time services, and service providers.

You first must connect an Access Server to the Administrator so that you can use the Administrator to create a real-time service from a real-time job. After a service starts, the Access Server brokers messages between external applications and Data Integrator.

When a message request comes in, the Access Server communicates with the Job Server to get the repository data needed to run a real-time service and process the message. A reply comes back through the Access Server to the message originator and the Access Server log records the event, which you can monitor from the Administrator.

Use the List of Access Servers page to connect an Administrator to a repository.

➢ To connect an Access Server to the Administrator

1. Select **Management** > **Access Servers**.
2. Click **Add**.
3. Enter the following information.

Option	Description
Machine name	Host name of the computer on which the Access Server is installed.
Communication Port	Port assigned to this Access Server in the Server Manager utility.

4. (Optional) To see if the Access Server is available and that it exists on the computer and port you have specified, before attempting to register it with the Administrator, click **Ping**.
5. Click **Apply**.

   The Administrator registers the Access Server, validates the Access Server connection information, and shows that information on the List of Access Servers page.

To view a list of Access Servers connected to the Administrator, select **Management > Access Servers**.

The List of Access Servers page lists the Access Servers that are connected to the Administrator. You can also remove a connection to an Access Server from this page.

Centralizing administration

You can connect any number of repositories and Access Servers to an Administrator, which allows you to administrate all jobs from a single, central location.

Alternatively, you can set up an Administrator to manage the jobs from an individual developer, the test repository, or different types of production jobs (batch and real-time). You can connect repositories to one Administrator, providing convenient access to a particular set of real-time jobs (for example, a set that serves a unique function such as development). However, Administrators cannot connect to each other.

➢ To group administration by job type

1. Configure Administrators that will process a particular type of job.

   For example, in your production environment you can configure one Administrator to process batch jobs and a different Administrator to process real-time jobs.

2. Connect each Administrator to the repositories that contain that type of job.

   You might want to name repositories so that you can easily see the types of jobs stored on them.

3. Connect Access Servers to any Administrators that process or manage real-time jobs.

Setting the status interval

Use the Status Interval page to specify the time period for which the Administrator displays the status (using the red, yellow, and green status icons) on the Batch Job Status page.

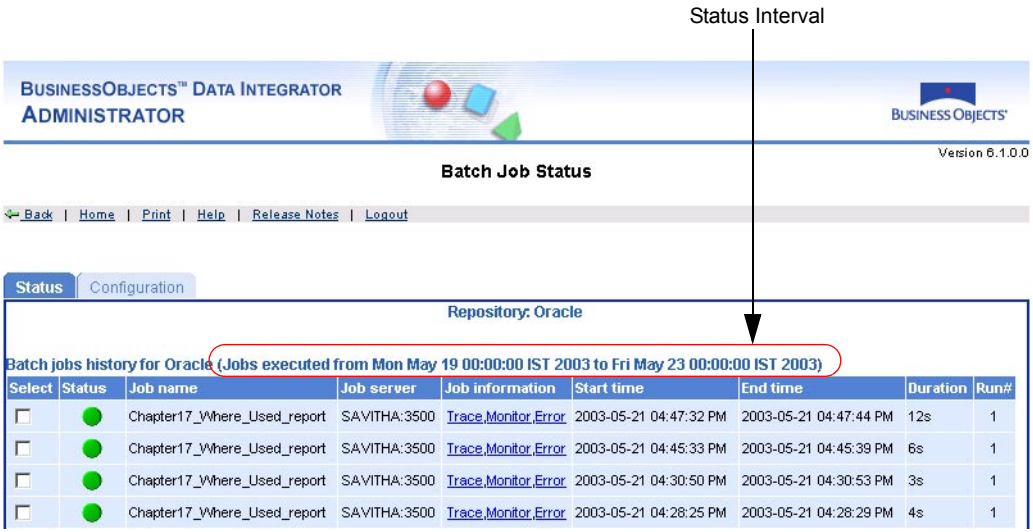
➢ To set the status interval

1. Select **Management** > **Status Interval.**
2. On the Status Interval page, specify the time period.

   You can filter the information on this page in three ways:

   - By the last execution
   - By number of days
   - By date
3. Click **Apply**.

   The Administrator updates the list of job executions and the status interval displays in the table title on the Batch Job Status page.



Setting the log retention period

The log retention period provides an automatic method of deleting log files.

➢ To delete log information

1. Select **Management** > **Log Retention Period**.
2. In the **Log Retention Period** box, enter the number of days you want to retain:
   - Historical batch job error, trace, and monitor logs
   - Current service provider trace and error logs
   - Current and historical Access Server logs

   The Administrator deletes all log files beyond this period. For example:
   - If you enter `1` then the Administrator displays the logs for today only. After 12:00 AM, these logs clear and the Administrator begins saving logs for tomorrow.
   - If you enter `-1` Data Integrator will not save logs.
   - If you enter `-1095` Data Integrator deletes logs older than approximately three years.

   NOTE: You can also delete Access Server logs manually using the Access Server Current Logs and Access Server History Logs pages.
3. Click **Apply**.

   Changes you make to the log retention period occur as a background clean-up process so they do not interrupt more important message processing. Therefore, you might not see logs deleted immediately when you select **Apply**. Changes can take up to an hour to take effect.

4. Choose a repository to view a list of executed batch job logs. When you select **Batch Jobs** > ***repository name***, the Administrator lists the most recent job first, providing a link to each job's log.

   For more information about monitoring jobs from the Administrator, see "Monitoring jobs" on page 53.

Support for Web Services

BusinessObjects Data Integrator can:

- Publish any job as a callable Web Service (call-in functionality)
- Call external Web Services from within its data flows using a built-in Web Services adapter (call-out functionality).

If you have an application that also supports Web Services, you can use that application to automatically start Data Integrator batch jobs and real-time services or to publish your application's functionality to be used by Data Integrator data flows.

Data Integrator publishes WSDL from the Administrator. To work with its call-in functionality, select **Management > Web Services**. See "Call-in functionality" on page 120 for more information.

After you install Data Integrator, you can immediately start working with its call-out functionality as the Web Services adapter is built-in. During the installation process, you are directed to select an adapter Job Server. Data Integrator automatically creates an interface for the Web Services adapter in the Administrator.

➢ To drop a function call to a Web Service into a Data Integrator job

1. Open the Designer.
2. Create a Web Services adapter datastore. See "Adapter datastores" on page 106 of the *Data Integrator Designer Guide* for more information about creating adapter datastores.
3. Import operation metadata from an external Web Service.
4. Drag the resulting function call from the object library into a data flow and configure it like any other function.

   See "Call-out functionality" on page 140 for more information.

4 Using Server Groups

You can group Job Servers on different computers into a logical Data Integrator component called a server group. A server group automatically measures resource availability on each Job Server in the group and distributes scheduled batch jobs to the Job Server with the lightest load at runtime.

This chapter describes how to work with server groups.

Specific sections include:

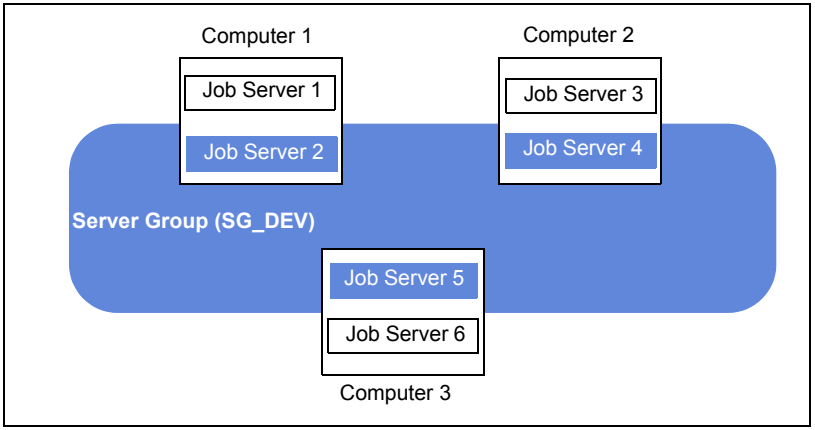
- Server group architecture
- Adding a server group
- Editing and removing a server group
- Monitoring Job Server status in a server group
- Executing jobs using server groups

Server group architecture

Use the Administrator to create and maintain server groups.

There are two rules for creating server groups:

- All the Job Servers in an individual server group must be associated with the same repository.
- Each computer (running NT, AIX, HPUX, or Solaris) can only contribute one Job Server to a server group.



The requirement that all Job Servers in a server group be associated with the same repository, simply allows you to more easily track which jobs are associated with a server group. Business Objects recommends that you use a naming convention for server groups that includes the name of the repository. For example, for a repository called DEV, a server group might be called `SG_DEV`.

On startup, all Job Servers check the repository to find out if they must start as part of a server group.

Compared to normal Job Servers, Job Servers in a server group each:

- Collect a list of other Job Servers in their server group

- Collect system load statistics every 60 seconds:
  - Number of CPUs (on startup only)
  - Average CPU load
  - Available virtual memory
- Service requests for system load statistics
- Accept server group execution requests

Load balance index

All Job Servers in a server group collect and consolidate system load statistics and convert them into a load balance index value for each Job Server. A Job Server's load balance index value allows Data Integrator to normalize statistics taken from different platforms. The Job Server with the lowest index value is selected to execute the current job. Data Integrator polls all Job Server computers every 60 seconds to refresh the load balance index.

Job execution

After you create a server group, you can select a server group to execute a job from the Designer's Execution Properties window or from the Administrator's Execute Batch Job, Schedule Batch Job, and Export Batch Job pages.

When you execute a job using a server group, the server group executes the job on the Job Server in the group that is running on the computer that has the lightest load. The Administrator will also resynchronize a Job Server with its repository if there are changes made to the server group configuration settings.

Job launcher

The Job Launcher, exported as part of a job's execution commands, includes a specific command line option for server groups. You can use this option to change the Job Servers in a server group. For more information, see "Data Integrator job launcher" on page 50.

Working with server groups and Designer options

Some Designer options assume paths are relative to a Job Server. If your Job Servers are on different machines from your Designer (typically the case in a production environment) you must ensure that connections and directory paths point to the Job Server host that will run the job. Such options include:

- Source and target directories for files
- Bulk load directories
- Source and target connection strings to databases
- Path to repositories

When using server groups consider the additional layer of complexity for connections. For example, if you have three Job Servers in a server group:

- Use the same directory structure across your three host computers for source and target file operations and use relative paths for file names.
- Use the same connection strings to your databases for all three Job Server hosts.

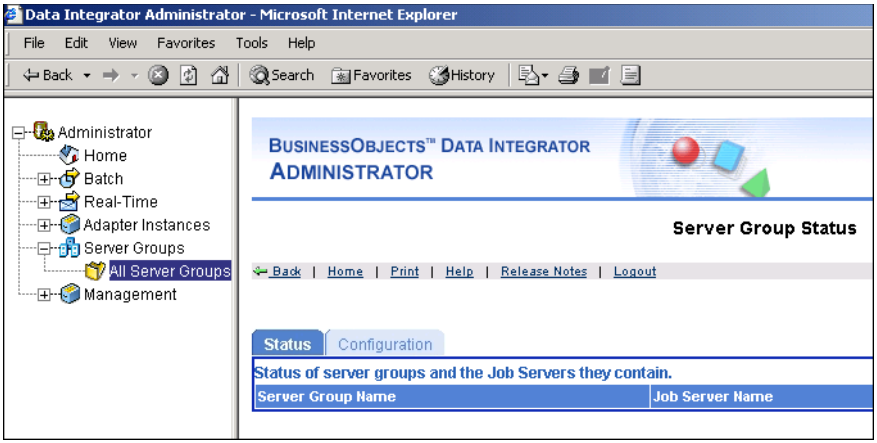
Thoroughly test Job Server centric job options when working with server groups.

Adding a server group

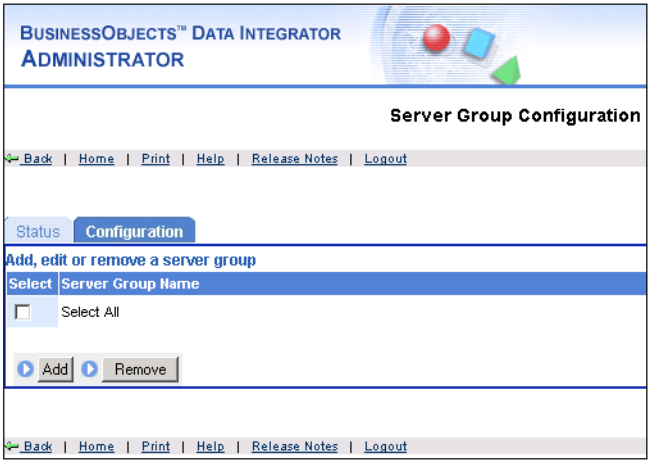
In the Administrator, use the Server Groups node to create and add a server group.

➢ To add a server group

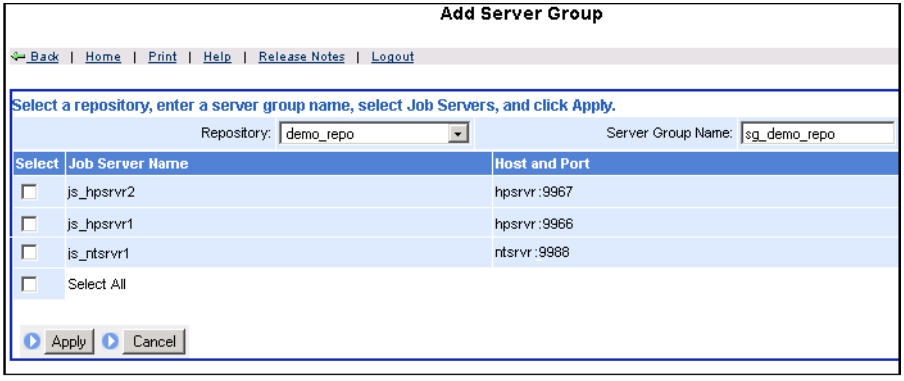
1. Select **Server Groups** > **All Server Groups.**



2. Click the **Configuration** tab.

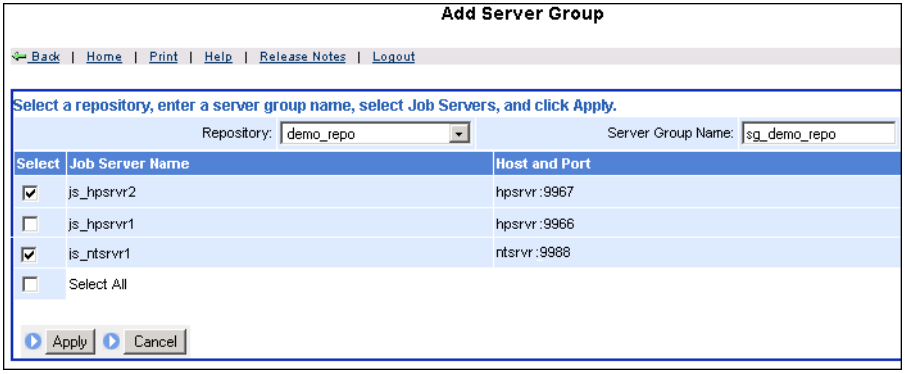


3. Click **Add**.



4. Follow the instructions on the Add Server Group page to create a server group.
   - When you select a repository, the Job Servers registered with the repository are displayed. You can create one server group per repository.
   - Notice that the Administrator provides a default server group name. It is the name of your repository with the prefix `SG_` (for server group). You can change the default name, however, labeling a server group with the repository name is recommended.
   - One Job Server on a computer can be added to a server group. Use the **Host and Port** column to verify that the Job Servers you select are each installed on a different host.

5. After you select the Job Servers for a server group, click **Apply**.



The display returns to the Server Group Configuration page.

For information about monitoring server groups, see "Monitoring Job Server status in a server group" on page 38.

Editing and removing a server group

You can select a new set of Job Servers for an existing server group or remove a server group.

Trace messages are written for a change in Job Server status when you create, edit, or remove server groups.

- When a Job Server is upgraded to membership in a server group, the trace message is:

  ```
  Collecting system load statistics,
  maintaining list of Job Server(s) for this
  server group, and accepting Job Server
  execution requests.
  ```

- When a Job Server is downgraded out of a server group, the trace message is:

  ```
  Deleting current system load statistics, and
  not collecting more. Not accepting job
  execution requests from a server group.
  ```

➢ To edit a server group

1. In the Server Group Status page, click the **Configuration** tab.
2. In the Server Group Configuration page, click the server group that you want to edit.
3. In the Edit Server Group page, select a new set of Job Servers.
4. Click **Apply**.

   Your edited server group is saved and the display returns to the Server Groups Configuration page.

➢ To remove a server group

1. In the Server Group Status page, click the **Configuration** tab.

2. In the Server Group Configuration page, select the check box for a the server group(s) that you want to remove.
3. Click **Remove**.

   The selected server group is removed as shown in the display.

NOTE: If you delete Job Servers from a repository, so as to delete all the Job Servers in a server group, the Administrator displays an invalid status for the server group.

Monitoring Job Server status in a server group

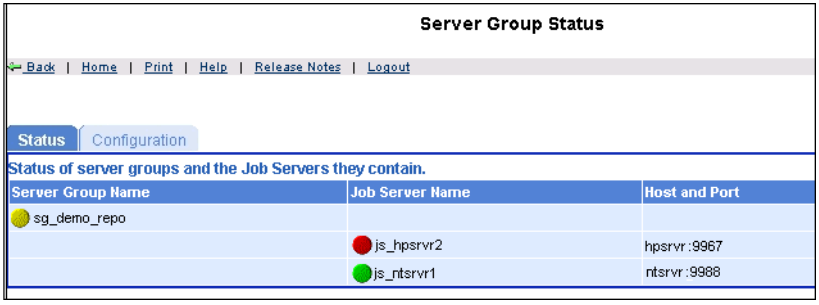
If Job Servers are in a server group, you can view their status in the Administrator.

- To monitor the status of these Job Servers, select **Server Groups** > **All Server Groups**.

  The Server Group Status page opens. All existing server groups are displayed with the Job Servers they contain.

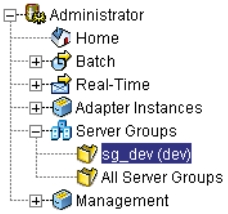
Indicator	Description
	A green indicator signifies that a Job Server is running.
	A yellow indicator signifies that a Job Server is running but not responding. Perhaps because it is too busy.
	A red indicator signifies that the Job Server is not running.

  If a server group contains Job Servers with a mix of green, yellow, or red indicators, then its indicator appears yellow:



  Otherwise, a server group indicator displays the same color indicator as its Job Servers.

- To view the status for a single server group, select its name.



Executing jobs using server groups

After you create a server group, you can select a server group to execute a job from the Designer's Execution Properties window or from the Administrator's Execute Batch Job and Schedule Batch Job pages.

For more information about executing, scheduling and monitoring batch jobs, see "Batch Jobs" on page 41.

5 Batch Jobs

This chapter describes how to execute, schedule, and monitor batch jobs from the Administrator.

Specific sections include:

- Executing batch jobs
- Scheduling jobs
- Monitoring jobs

Before you can manage batch jobs with the Administrator, add repository connections. See "Adding repositories" on page 16.

Executing batch jobs

You can execute batch jobs from the Administrator if their repositories are connected to the Administrator.

➢ To execute a job

1. Select **Batch** > ***repository***.

   The Administrator opens the Batch Job Status page, which lists all the jobs in the repository you just selected.

   To view jobs in all repositories from this page, select **Batch** > **All Repositories**. (The **All Repositories** option appears under the **Batch Job** node if more than one repository is connected to the Administrator.)

2. Click the **Configuration** tab.

3. To the right of the job you want to run, click **Execute**.

   The Administrator opens the Execute Batch Job page.

4. Under **Execution options**, set the parameters for the execution of this job.

   For more information about these parameters, see "Parameters" on page 17 of the *Data Integrator Reference Guide*.

5. Under **Trace options,** set the trace properties for this execution of the job.

   For more information about these properties, see "Trace properties" on page 20 of the *Data Integrator Reference Guide*.

6. Click **Execute** to run the job.

   The Administrator returns to the Batch Job Status page.

Scheduling jobs

There are two ways to schedule a job in the Data Integrator environment:

- Using the Administrator
- Using a third-party scheduler

Using the Administrator

When you schedule batch jobs using the Administrator, it creates an entry in the scheduling utility on the Job Server computer. Windows uses the task scheduler (AT.exe) and UNIX systems use the CRON utility. If you make changes to the schedule directly through these utilities, the Administrator will not reflect such changes.

Changes made to the Job Server, such as an upgrade, will not affect schedules created in Data Integrator as long as:

- The new version of Data Integrator is installed in the same directory as the original version (Data Integrator schedulers use a hard-coded path to the Job Server).
- The new installation uses the Job Server name and port from the previous installation. (This occurs automatically when you install over the existing DSConfig file.)

You can use the Administrator:

- To define a new schedule for a job
- To view, activate/deactivate a schedule
- To remove a job schedule

➢ To define a new schedule for a job

1. Select **Batch** > ***repository***.
2. Click the **Configuration** tab.

3. Find the job you want to schedule, and under **Action**, click **Add Schedule** for that job.

   The Administrator opens the Schedule Batch Job page.

4. Under **Schedule job**, enter the desired options.

Schedule Batch Job page options

Option	Description
Schedule name	Enter a unique name that describes this schedule.
Active	Check this box to activate this schedule. This option allows you to create several schedules for a job and then activate the one you currently want to run.
Recurring	Indicate whether the schedule repeats. There are two options: **No** — Select to execute this schedule once during the current week or month **Yes** — Select to execute this schedule repeatedly, every week or every month
Day of Week	Select **Day of Week** to schedule the job by the day of the week. You can select one or more days. To choose multiple days, press **Ctrl** when selecting additional days. If **Recurring** is set to **Yes**, then the Administrator schedules this job to repeat every week.
Days of Month	Select **Day of Month** to schedule the job by date. You can select one or more dates. To choose multiple dates, press **Ctrl** when selecting additional dates. If **Recurring** is set to **Yes**, then the Administrator schedules this job to repeat every month.
Start Time	Enter the time for the Administrator to start the job (hours, minutes, seconds, and either AM or PM). Select a time when all of the required resources are available. Typically, you want to schedule jobs to ensure they complete before the target database or data warehouse must meet an increased demand.
Enable Recovery	Select this check box to enable the Recovery mode when this job runs. See "Parameters" on page 17 of the *Data Integrator Reference Guide*.

Schedule Batch Job page options (Continued)

Option	Description
Recover from Last Failed Execution	Select this check box if an execution of this job has failed and you want to enable the Recovery mode. See "Parameters" on page 17 of the *Data Integrator Reference Guide*.
System profile	Select the system profile to use when executing this schedule. See "Parameters" on page 17 of the *Data Integrator Reference Guide*.
Job Server or server group	Select the Job Server or a server group to execute this schedule. See "Parameters" on page 17 of the *Data Integrator Reference Guide*.

5. Click **Schedule**.

When you export a repository using .atl files, jobs and their schedules (created in Data Integrator) automatically export as well.

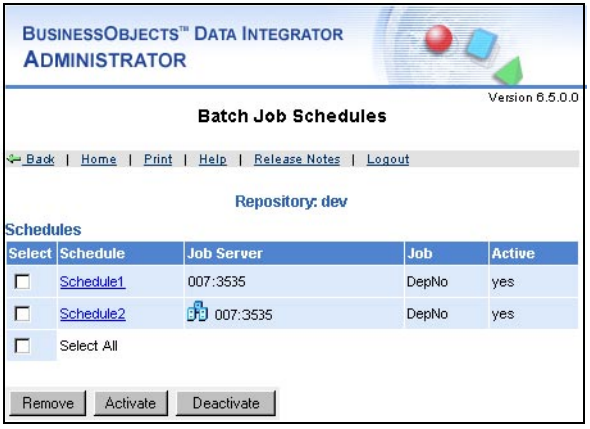
You can also import a repository .atl file including jobs and their associated schedules (previously created in Data Integrator) back into Data Integrator. See "Importing from a file" on page 347 of the *Data Integrator Designer Guide*.

Once imported, you must reactivate job schedules to use them.

➢ To view, activate/deactivate a schedule

1. If you have not already done so, connect the repository that contains the job to the Administrator.
2. In the Administrator, select **Batch** > ***repository*** to view the list of jobs.
3. Click the **Configuration** tab.

4. Click the **Schedules** link for the job with which you want to work.



The Batch Job Schedules page opens. It lists scheduled jobs and provides a link to each schedule so that you can view, edit, activate, or deactivate it. You can also simply select a schedule name and use the buttons on the Batch Job Schedules page.

The Job Server column listed next to each schedule indicates both where the schedule is stored and which Job Server will execute it. If there is a server group icon in the Job Server column, this indicates the schedule will be executed by the server group, and the schedule is stored on the Job Server shown. To see which server group is associated with the schedule, roll your cursor over the server group icon.

5. Select a schedule name.
6. Select the **Active** check box.
7. From the **Job Server or server group** list, select the Job Server (host and port) or the server group that will execute the job.
8. Make additional edits to the schedule as needed.

9. Click **Apply**.

   NOTE: You cannot edit an active job schedule. Deselect an active schedule and click **Apply**. Then navigate back to the schedule, edit it, and click **Apply** again to save changes.

➢ To remove a job schedule

1. Open the Batch Job Schedules page.
2. In the **Select** column, select a check box in front of a scheduled job.
3. Click **Remove**.

   The Administrator deletes information about this job schedule.

Using a third-party scheduler

When you schedule jobs using third-party software:

- The job initiates outside of Data Integrator.
- The job runs from an executable batch file (or shell script for UNIX), exported from Data Integrator.

  See "Data Integrator job launcher" on page 123 for more information about these files.

NOTE: When a third-party scheduler invokes a job, the corresponding Job Server must be running.

➢ To execute a job with a third-party scheduler

1. Export the job's execution command to an executable batch file (`.bat` file for Windows NT and Windows 2000 environments, or `.sh` file for UNIX environments).

   See "To export a job for scheduling" on page 48.

2. Ensure that the Data Integrator Service is running (for the job's Job Server) when the job begins to execute.

   The Data Integrator Service automatically starts the Job Server when the computer on which you installed the Job Server is rebooted.

   - You can also check that a Job Server is running at a particular time using the Designer. Log into the repository that contains your job and view the Designer's status bar to verify that the Job Server connected to this repository is running.
   - You can check that all Job Servers in a server group are running using the Administrator. In the navigation tree select **Server Groups** > **All Server Groups** to view the status of server groups and the Job Servers they contain.

3. Schedule the batch file from the third-party software.

   For information about parameters for the batch file, see "Data Integrator job launcher" on page 123.

➢ To export a job for scheduling

1. Select **Batch** > ***repository***.
2. Click the **Configuration** tab.

   The Administrator opens the Batch Job Configuration page, which lists all the jobs in the connected repositories.

3. Find the job that you want to export for scheduling, and click **Export Execution Command** for that job.

   The Administrator opens the Export Execution Command page.

4. Enter the parameters for the batch job command file you want the Administrator to create.

The following options appear:

Field	Description
File name	The name of the batch file or script containing the job. The third-party scheduler executes this file. The Administrator automatically appends the appropriate extension: • `.sh` for UNIX • `.bat` for Windows NT or Windows 2000
System profile	(Appears only if system profiles exist for a job) Select the system profile to use when executing this job. See "Parameters" on page 17 of the *Data Integrator Reference Guide*.
Job Server or server group	Select the Job Server or server group to execute this job. See "Parameters" on page 17 of the *Data Integrator Reference Guide*.
Enable Recovery	Select this check box to enable the automatic recovery feature. When enabled, Data Integrator saves the results from completed steps and allows you to resume failed jobs. See "Automatically recovering jobs" on page 463 of the *Data Integrator Designer Guide* for information about the recovery options.
Recover from last failed execution	Select this check box to resume a failed job. Data Integrator retrieves the results from any steps that were previously executed successfully and re-executes any other steps. This option is a run-time property. This option is not available when a job has not yet been executed or when recovery mode was disabled during the previous run.

5. Click **Export**.

   The Administrator creates a command file for the job and writes this file to the local `LINK_DIR\log` directory. `LINK_DIR` is the installation directory, for example `c:\DataIntegrator`.

   The Administrator returns to the Batch Job Configuration page.

Data Integrator job launcher

Data Integrator exports job execution command files as batch files on Windows or CRON files on UNIX. These files pass parameters and call `AL_RWJobLauncher`. Then, `AL_RWJobLauncher` executes the job, sends it to the appropriate Data Integrator Job Server, and waits for the job to complete.

WARNING: Under normal circumstances, you should not modify the exported file—do so only under the direction of Business Objects Customer Support.

The following shows a sample Windows NT batch file created when Data Integrator exports a job. `ROBOT` is the host name of the Job Server computer. All lines after `inet:ROBOT:3513` are `AL_Engine` arguments, not `AL_RWJobLancher` arguments.

```
d:\Data Integrator\bin\AL_RWJobLauncher.e

"inet:ROBOT:3513"

"-SrepositoryServer

-Uusername

-Ppassword

-G"b5751907_96c4_42be_a3b5_0aff44b8afc5"

-r100 -T14

-CTBatch -CmROBOT -CaROBOT

-CjROBOT -Cp3513"
```

Job launcher flag values and arguments

The following table lists job launcher flags and their values.

Flag	Value
`-w`	The job launcher starts the job(s) and then waits before passing back the job status. If -w is not specified, the launcher exits immediately after starting a job.
`-t`	The time, in milliseconds, that the Job Server waits before checking a job's status. This is a companion argument for `-w`.
`-s`	Status or return code. 0 indicates successful completion, non-zero indicates an error condition. Combine `-w`, `-t`, and `-s` to execute the job, wait for completion, and return the status.
`-C`	Name of the engine command file (path to a file which contains the Command line arguments to be sent to the engine).
`-v`	Prints `AL_RWJobLauncher` version number.

`-S`	Lists the server group and Job Servers it contains using the following syntax:
	`"SvrGroupName;JobSvr1Name:JobSvr1Host:JobSvr1Port;JobSvr2Name:JobSvr2Host:JobSvr2Port";`
	For example:
	`"SG_DEV;JS1:HPSVR1:3500;JS2:WINSVR4:3505";`

There are two arguments that do not use flags:

- `inet` address — The host name and port number of the Job Server. The string must be in quotes. For example:

  ```
  "inet:HPSVR1:3500"
  ```

  If you use a server group, `inet` addresses are automatically rewritten using the `-S` flag arguments. On execution, the first Job Server in the group checks with the others and the Job Server with the lightest load executes the job.

- `server log path` — The fully qualified path to the location of the log files. The server log path must be in quotes. The server log path argument does not appear on an exported batch job launch command file. It appears only when Data Integrator generates a file for an active job schedule and stores it in the following directory: *`LINK_DIR`*`/Log/`*`JobServerName`*`/`*`RepositoryName`*`/`*`JobInstanceName`*

  You cannot manually edit server log paths.

Job launcher error codes

The job launcher also provides error codes to help debug potential problems. The error messages are:

Error number	Error message
180002	Network failure.
180003	The service that will run the schedule has not started.
180004	`LINK_DIR` is not defined.
180005	The trace message file could not be created.
180006	The error message file could not be created.
180007	The GUID could not be found. The status cannot be returned.
180008	No command line arguments were found.
180009	Invalid command line syntax.
180010	Cannot open the command file.

Monitoring jobs

Using the Administrator, you can monitor job execution of any batch job in a connected repository. You can monitor jobs that you run from the Administrator or from the Designer.

This section discusses how you can use the Administrator to view a batch job's:

- Overall status
- Statistics

Overall status

The Batch Job Status page lists each batch job execution. Use this list to view the overall status of each execution and to access more detailed statistics and log files.

➢ To view overall status of executed jobs

1. Select **Batch** > ***repository***.

   The Batch Job Status page shows each instance of job execution on the selected repository. The list shows jobs that ran during the time period specified in the table title. To modify this status interval, see "Setting the status interval" on page 24.

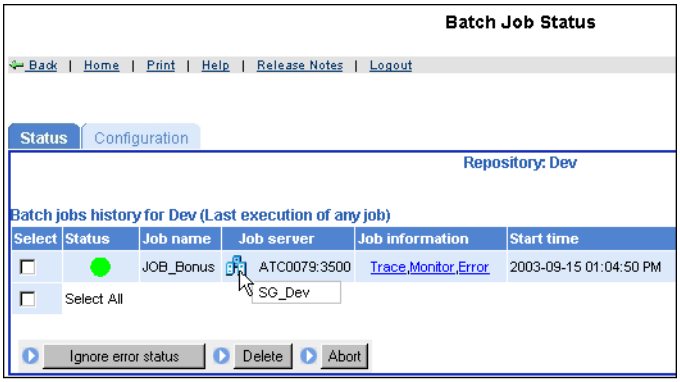
2. Find the overall status of a batch job execution by examining the indicator in the **Status** column.

Indicator	Description
	A green indicator signifies that the batch job ran without error.
	A red indicator signifies that the batch job experienced an error.

   Check the **End Time** column to see if or when the job completed.

3. If a batch job execution has a red status, examine the trace, monitor, and error logs for more information.
4. To view detailed information about a particular job execution, look at the data on the Batch Job Status page.

If the job includes a server group icon in the **Job Server** column, this indicates that the job was executed by a server group. You can roll your cursor over the server group icon to view the name of the server group. The Job Server listed is the Job Server in the server group that executed the job.



NOTE: All jobs can be executed by an explicitly selected Job Server or by a server group. If you choose to execute a job using a server group, you can use this page to see which Job Server actually executed the job. If you explicitly select a Job Server to execute a job, then even if it is also part of a server group, the server group icon does not appear for the job in the **Job Server** column on this page.

Statistics

For each job execution, the Administrator shows statistics. Statistics quantify the activities of the components of the job. Time spent in a given component of a job and the number of data rows that streamed through the component are examples of job statistics.

To help tune the performance of a job, review job statistics.

➢ To view job statistics

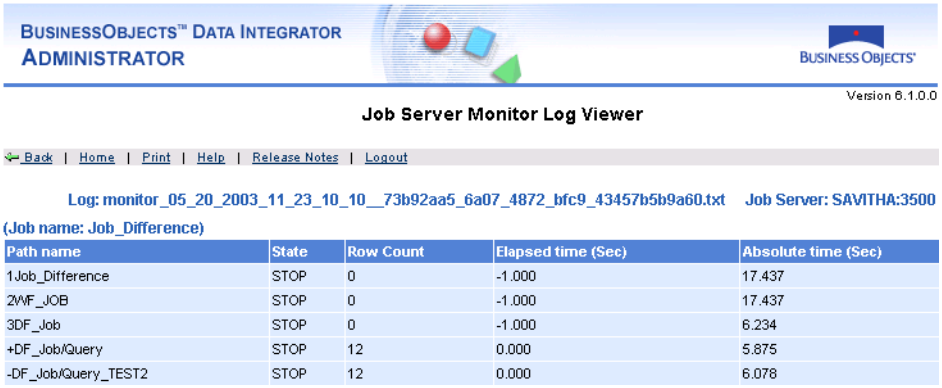
1. Select **Batch** > ***repository***.
2. On the Batch Job Status page, find the instance of the job execution in which you are interested.

   Identify an instance using the page sub-title (which provides the name of the repository on which Data Integrator stores the job) and the following column heading on this page:

   - Status — See "Overall status" on page 53.
   - Job Name— Name you gave the job in the Designer
   - Job Server— Server that ran this job
   - Start Time— Date and time this instance started
   - End Time— Date and time this instance stopped
   - Duration — Time (in seconds) the job took to complete
   - Run # — Times this instance ran before completing
3. Under **Job Information** for an instance, click **Monitor**.

   The Administrator opens the Job Server Monitor Log Viewer page. This page shows several statistics about this instance of job execution starting with the name of the monitor log file.

After the file name, each line in the log provides the following information:


BUSINESSOBJECTS™ DATA INTEGRATOR ADMINISTRATOR

BUSINESS OBJECTS

Version 6.1.0.0

Job Server Monitor Log Viewer

Back | Home | Print | Help | Release Notes | Logout

Log: monitor_05_20_2003_11_23_10_10__73b92aa5_6a07_4872_bfc9_43457b5b9a60.txt Job Server: SAVITHA:3500

(Job name: Job_Difference)

Path name	State	Row Count	Elapsed time (Sec)	Absolute time (Sec)
1Job_Difference	STOP	0	-1.000	17.437
2WF_JOB	STOP	0	-1.000	17.437
3DF_Job	STOP	0	-1.000	6.234
+DF_Job/Query	STOP	12	0.000	5.875
-DF_Job/Query_TEST2	STOP	12	0.000	6.078

- Path Name — Indicates which object (step in a data flow) is executing.
- State — Indicates the run time order of the processes in the execution of the transform object and the states of each process. These are not error status states. However, if a process state is **Proceed** and it never changes to **Stop**, this indicates the process ran with errors.
  - Initializing — Job is initializing
  - Optimizing — Job is optimizing
  - Proceed — Process is executing
  - Stop — Process ends without error
- Row Count — Indicates the number of rows processed through this object. This value updates based on the **Monitor sample rate (# of rows)** set as an execution option on the Execute Batch Job page.
- Elapsed Time — Indicates the time (in seconds) since this object received its first row of data.
- Absolute time — Indicates the time (in seconds) since the execution of this data flow (including the current object) began.

Ignore error status

The Batch Job Status page includes an option to **Ignore Error Status** (button at end of page). Use this option if you are working through jobs with warnings or errors on this page and you want to mark a row so that you know you are finished looking at its logs.

➢ To ignore error status

1. Select the job or jobs that you want to ignore.
2. Click the **Ignore Error Status** button.

   The page refreshes and the rows you selected now display a green status icon.

Deleting batch job history data

The Batch Job Status page includes an option to delete information about how a job ran. If you want to manually delete rows from this page, select the rows that you want to delete, then select **Delete**. You can also manage this information by setting the Administrator's log retention period.

Stopping a running job

The Batch Job Status page includes an option to abort batch jobs. If a batch job is running and you need to stop it, select the check box before the job name and click **Abort**.

Trace, monitor, and error logs

You can view and delete trace, monitor, and error logs for job instances from the Batch Job Status page. The corresponding Job Server must be up and running to view or delete these logs.

You can set trace log options on the Execute Batch Job page. For more information about batch trace and error logs, see "Batch job logs" on page 151. For information specific to the monitor log, see "Statistics" on page 54.

For detailed definitions about the type of information shown on each log, see "Log" on page 75 of the *Data Integrator Reference Guide*.

You can use the **Delete** button on the Batch Job Status page to delete a set of batch log history files from a Job Server computer and its corresponding repository.

➢ To delete trace, monitor, and error logs for a batch job

1. Select **Batch** > ***repository***.
2. Select the job or jobs for which you want to delete logs.

   Alternately, you can click **Select All**.
3. Click **Delete.**

6 Real-Time Jobs

This chapter describes how to support real-time jobs using the the Administrator.

Specific sections include:

- Supporting real-time jobs
- Configuring and monitoring real-time services
- Creating and monitoring client interfaces

Before configuring services, add real-time job repository and Access Server connections to the Administrator.

Supporting real-time jobs

The Access Server manages real-time communication between Data Integrator and external applications (such as ERP or web applications). The Access Server determines how to process incoming and outgoing messages based on the settings you choose for each real-time job in the Administrator.

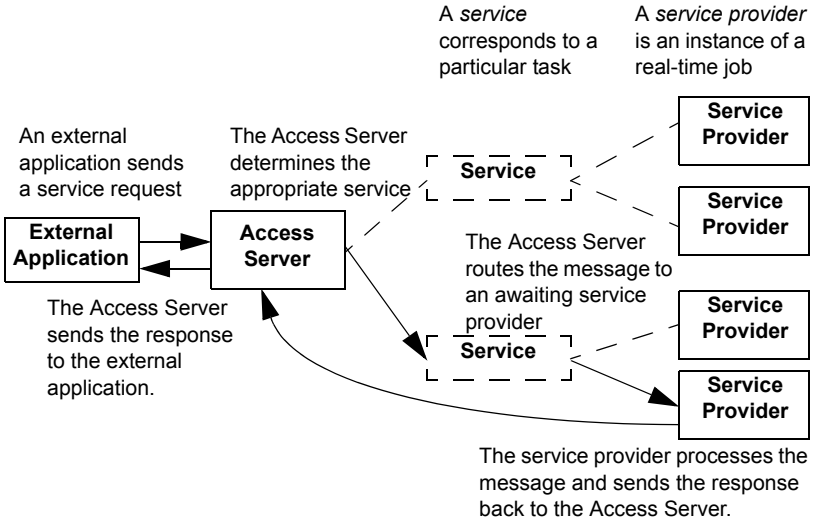
In particular you use the Administrator to define:

- Services — A service is a name that identifies a task. The Access Server receives requests for a service. You associate a service with a real-time job. The real-time job contains the real-time processing loop that can process requests for this service and generate a response.
- Service providers — A service provider is the computer process that performs a service; the service provider completes the tasks in a real-time job. A service provider is controlled by a Job Server. A Job Server can control several service providers—each service provider is a unique process or instance.

The Access Server uses services and service providers to process message requests.

- For example, suppose an external application sends a request to the Access Server.
- The Access Server determines the appropriate service for the request.
- Next, the Access Server finds the associated service providers and dispatches the request to the next available service provider.

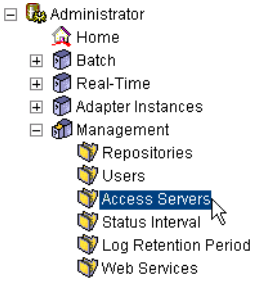
- Under the control of a Job Server, that service provider completes the processing for the request. A different Job Server might control each service provider.



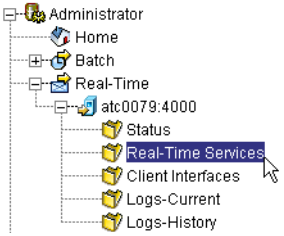
The Access Server manages the entire set of service providers, implementing configuration changes and telling the appropriate Job Servers to start and stop service providers. At a prescribed interval, the Access Server updates service providers, balancing loads and implementing configuration changes. To balance loads, the Access Server monitors requests for services to ensure that no service provider is over-used or under-used. Based on the number of requests for a service, the Access Server tells Job Servers to start or stop service providers.

To support real-time jobs, you must:

- Create any number of Access Servers using the Server Manager utility, then add a connection to each local or remote Access Server using the **Management** node in the Administrator.



- In the **Real-Time** node of the Administrator, create a service for each real-time job under each Access Server's node.



  See "Creating services and service providers" on page 63.

- Create one or more service providers for each service. See "Creating services and service providers" on page 63.
- Start the services. See "Starting and stopping services" on page 69.
- Monitor the services. See "Monitoring services" on page 74.

Configuring and monitoring real-time services

To enable an Access Server to support real-time jobs, you must configure and monitor real-time services and service providers for it.

- Configure services by specifying a real-time job and other operational parameters.
- Configure service providers by specifying a Job Server and indicating the maximum and minimum number of instances you want the Job Server to control. Each service provider is a unique process or instance controlled by a Job Server.

This section discusses:

Creating services and service providers

In the Administrator, you create a service that processes requests for each real-time job. You also create the service providers to perform that service. A service provider is the process that completes the tasks in a real-time job.

➢ To add a service

1. Select **Real-Time** > ***Access Server*** > **Real-Time Services.**
2. Click the **Configuration** tab.
3. Click **Add**.

4. In the **Service configuration** section, enter information that describes this service.

Real-time service configuration parameters

Parameter	Description
Service name	A unique name for this service.
Job name	Click **Browse Jobs** to view a list of all the real-time jobs available in the repositories you connected to the Administrator. Select a job name on this page to fill the service configuration form.
Job tracing enable	A flag that indicates whether the service will write trace messages. • Select **Enable** for the job to write trace messages.
Startup timeout (seconds)	The maximum time the Access Server waits for the service to register after startup. See "Service startup behavior" on page 89.
Queuing timeout (seconds)	The maximum time the Access Server waits for the service to process the request. See "High traffic behavior" on page 91.
Processing timeout (seconds)	The maximum time the Access Server waits for a response from the service. See "Response time controls" on page 92.
Processing retry count max	The number of times the Access Server attempts to restart a job that fails to respond. See "Response time controls" on page 92.

Real-time service configuration parameters (Continued)

Parameter	Description
Recycle request count max	The number of requests the Access Server sends to a given real-time service before automatically recycling the flow. See "Service startup behavior" on page 89.
System profile	Select the system profile to use when executing this service. See "Parameters" on page 17 of the *Data Integrator Reference Guide*.
Enable	A flag that indicates whether the Access Server attempts to automatically start this service when the Access Server restarts. • Select **Enable** if you want to automatically start this service when the Access Server restarts. This is the default setting. • If you clear **Enable,** when the Access Server restarts, it does not automatically start this service. If you manually attempt to start a disabled service, an error message appears in the Service's **Status** column.

5. Click **Apply**.

The Administrator updates the configuration parameters for this service. These configuration parameters apply to all providers of this service.

View status, apply an action, or select a service name to see statistics and service providers

Select	Name	Job name	Max Instances	Min Instances	Started Instances	Requests	Status
☐	Service1	Job_TestConnectivity	1	1	0	0	Service Initialized.

When you add a new service, the Administrator creates a default service provider controlled by the default Job Server.

Verify that the Job Server host name and port for the new service provider are correct by clicking the Service name.

Job Servers for Service

Select	Job Server	Max Instances	Min Instances	Started Instances	Status
☐	USERNAME:3500	1	1	0	Ready to be started

6. If you need to change the Job Server host name or port, or want to alter the number of service providers controlled by this Job Server:
   a. Click the Job Server name.

      The Administrator opens the Service Provider Configuration page.

   b. In the **Job Server** list, select a Job Server to control the service provider.

   c. In **Min instances** and **Max instances** enter a minimum and a maximum number of service providers you want this Job Server to control for this service.

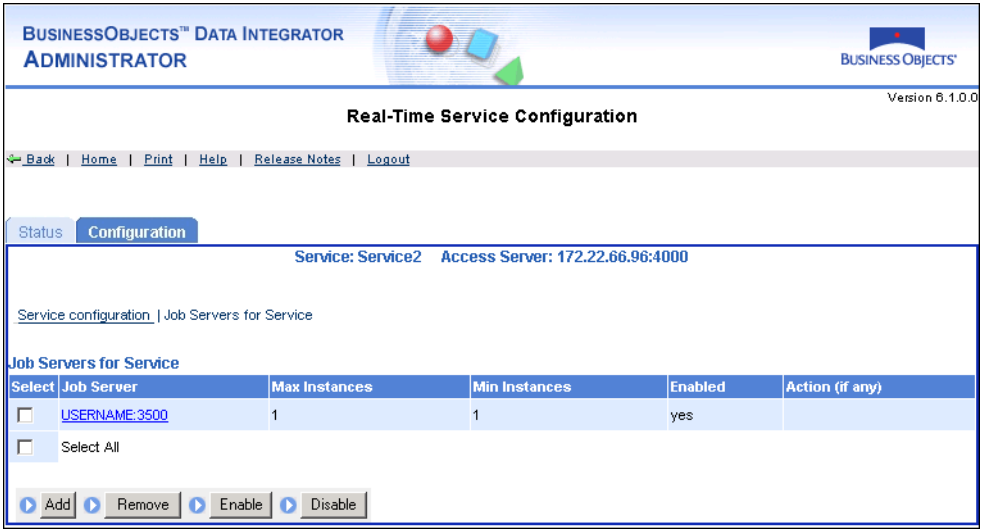
   d. Select **Enable** to have the Access Server start the service providers controlled by this Job Server. Deselect **Enable** to configure but not start the service providers controlled by this Job Server.

   e. Click **Apply**.

      The Administrator updates the information and returns to the **Job Servers for Service** section of the Real-Time Service Configuration page.

      If required, you can add other service providers now or use the instructions in the next section to add them later.

You can also add more Services now by toggling between the **Service Configuration** and **Job Servers for Service** sections of this page



7. When you are ready to have the Access Server process requests, start the service. See "To start a service" on page 70.

➢ To add a new service provider for a service

1. Select **Real-time** > ***Access Server*** > **Real-Time Services.**
2. Click the **Configuration** tab.
3. Click the name of the service for which you want to add a service provider.
4. Click **Job Servers for Service.**
5. Click **Add.**
6. In the **Job Server** list, select the Job Server that will control the service provider.

7. In **Min instances** and **Max instances**, enter a minimum and a maximum number of service provider instances you want this Job Server to control.
8. Select **Enable** to have the Access Server start service providers controlled by this Job Server. Deselect **Enable** to configure but not start the service providers controlled by this Job Server.
9. Click **Apply**.

   If the service has already started, the Access Server adds this service provider to the available list when it next updates the service providers (see "Updating service providers" on page 73).

   If the service has not yet started, the Access Server starts enabled service providers when the service starts.

➢ To set the service provider update interval

1. Select **Real-time** > ***Access Server*** > **Status**
2. Click the **Configuration** tab.
3. Enter the desired **Provider update interval**.

   This is the time interval, in seconds, between service provider updates. Valid values range from 10 to 120 seconds. The default is 30 seconds. When updating service providers, the Access Server balances loads and implements any configuration changes you have applied to a service provider. See "Updating service providers" on page 73.

   If the provider update interval is too small, performance can decrease because the Access Server must frequently check for events and collect statistics. Business Objects recommends that you set the **Provider update interval** to 30 seconds. On systems with heavy loads and production systems with fewer start and stop events, you can increase the interval.

➢ To change a Job Server that executes a service

1. Select **Real-time** > ***Access Server*** > **Real-Time Services.**
2. Click the **Configuration** tab.
3. Click the service name that you want to change.
4. Click the **Job Servers for Service** link.
5. In the **Job Server** list, select a Job Server to control the service providers.

   Job Servers are defined by host name and port number.
6. Update the maximum and minimum number of service providers, if necessary.
7. Select **Enable** to start service providers controlled by the changed Job Server when you apply this configuration. Deselect **Enable** to configure but not start service providers controlled by the Job Server when you apply this configuration.
8. Click **Apply**.

   The Administrator updates the service provider configuration.

   The Access Server automatically implements the changed configuration at the next service provider update. See "Updating service providers" on page 73.

Starting and stopping services

After creating required services and service providers, you must start them. After you start a service or service provider, Data Integrator ensures that it continues to run. You can also use the Administrator to stop a service (such as for maintenance). Similarly, use the Administrator to remove, enable, or disable services and service providers.

➢ To start a service

1. Select **Real-time** > ***Access Server*** > **Real-Time Services.**
2. Select the check box next to the service or services that you want to start.
3. Click **Start**.

   The Access Server starts the minimum number of service providers for this service.

➢ To abort or shut down a service

1. Select **Real-time** > ***Access Server*** > **Real-Time Services.**
2. Select the check box next to the service or services that you want to abort or shutdown.
   - **Abort** — Shuts down all service providers for this service without waiting for them to complete processing. The Access Server responds to current and new requests for this service with an error.
   - **Shutdown** — Shuts down all service providers for this service after they complete processing any current requests. The Access Server responds to new requests for this service with an error.
3. Click **Abort** or **Shutdown**.

   To restart a stopped service, see "To start a service" on page 70.

   To ensure that a stopped service does not restart automatically when the Access Server restarts, see "To disable a service" on page 71.

➢ To remove a service

1. Select **Real-time** > ***Access Server***> **Real-Time Services.**
2. Click the **Configuration** tab.

3. Select the check box next to the service or services that you want to remove.
4. Click **Remove**.

   The Administrator stops processing this service. The Access Server shuts down each of the service providers defined for this service and removes the service from the list.

➢ To disable a service

1. Select **Real-time** > ***Access Server*** > **Real-Time Services.**
2. Click the **Configuration** tab.
3. Click the service that you want to disable.
4. Deselect the **Enable** box.
5. Click **Apply**.

This change does not have an immediate effect on the service. Instead, the service is disabled only when the Access Server attempts to restart this service, such as after the Access Server restarts.

➢ To remove, enable, or disable a service provider

1. Select **Real-time** > ***Access Server*** > **Real-Time Services.**
2. Click the **Configuration** tab.
3. Select the check box next to a Job Server if you want to change its service providers.
4. Click one of the buttons below the list of Job Servers to perform the appropriate action:
   - **Remove** — Discontinue using the service providers controlled by the selected Job Servers to process requests for this service. The Access Server shuts down the service providers and removes the Job Server from the list.

- **Enable** — Start the service providers controlled by the selected Job Servers. Each Job Server starts the minimum number of service providers. The Access Server now includes the selected Job Servers in the set of available service providers. If a Job Server is already enabled, this choice has no effect.
- **Disable** — Shut down the service providers controlled by the selected Job Servers. The Access Server finishes processing any current requests and no longer includes the selected Job Servers in the set of service providers available to process requests for this service.

The Administrator completes this action during the next service provider update. See "Updating service providers" on page 73.

➢ To restart a service provider

1. Select **Real-time** > ***Access Server*** > **Real-Time Services**.
2. Select the check box next to a Job Server if you want to restart its service providers.

   NOTE: Only select **Restart** if the service providers controlled by this Job Server are currently enabled. To check, select the **Configuration** tab and view service provider status in the **Job Servers for Service** section.

3. Click **Restart**.

   The Administrator completes this action during the next service provider update. The Administrator shuts down any service providers controlled by the selected Job Servers and immediately restarts the minimum number of service providers. For example, you might restart a service provider after a computer running its Job Server reboots following a crash.

Updating service providers

At a specified provider update interval, the Access Server updates service providers. When updating service providers, the Access Server balances the work load—starting or stopping service providers as necessary—and implements other events that you initiated since the last update.

When balancing the work load, the Access Server checks the number of requests in a service queue and the minimum idle time for a service. If the number of requests in a service queue is greater than the number of service providers started, the Access Server tries to start a new service provider. Conversely, if the minimum idle time for a service is more than 10 minutes, the Access Server will shut down a service provider. However, the number of service providers cannot exceed the maximum number of instances configured nor can it be less than the minimum number of instances configured.

When implementing events that you initiated, the Access Server:

- Enables service providers
- Disables service providers
- Reconfigures service providers
- Restarts service providers
- Adds service providers
- Removes service providers

See "To set the service provider update interval" on page 68 for information about setting the provider update interval.

Monitoring services

Use the Administrator to monitor services. With the Administrator you can:

- View service status — From the Access Server Status page or Real-Time Service Status page, view whether a service is running or not. Based on this information, you can begin troubleshooting problems. See "To view the status of services" on page 74.
- View service provider status — From the Real-Time Service Status page, click a service name to view:
  - The statistics for a particular service. See "Service statistics" on page 94 for a detailed description.
  - Detailed statistics about each service provider. Using this information, you can monitor and evaluate system performance. See "Service provider statistics" on page 96 for a detailed description.
  - The status of all service providers in that service. See "To view the statistics for a service provider" on page 75.
- View logs — The "Access Server" node provides access to current and historical service provider trace and error logs. See "To view the logs for a service provider" on page 152.

➢ To view the status of services

1. Select **Real-Time** > ***Access Server*** > **Real-Time Services**.

   The Administrator opens the Real-time Service Status page. For each service, this page shows the overall status and statistics about the number of service providers started and the number that is possible. See "Service statistics" on page 94.

2. Verify that the services are working.

Indicator	Description
	A green indicator signifies that the service is operating properly.
	A yellow indicator signifies that some aspect of the service is not working, and that the Access Server is attempting to reestablish the service using error handling.
	A red indicator signifies that one or more aspects of the service is not working, and the Access Server cannot reestablish the service.

3. If a service shows a yellow or red status, click the service name to get more information.

For information about interpreting status information, see Chapter 10, "Troubleshooting".

➢ To view the statistics for a service provider

1. Select **Real-time** > ***Access Server*** > **Real-Time Services**.
2. Click the name of the service.

   This page shows the overall statistics for the service, the service providers for the service (listed by Job Server), and the status of each service provider. Start a service to see its service provider status information.
3. Under **Service Provider Status Information**, click the **Process ID** of a service provider to view its statistics.

   The Administrator opens the Service Provider Status page.

   Under **Service Provider Status Information**, the page shows the statistics for this service provider. See "Service provider statistics" on page 96 for a description of these statistics. For more information about service provider logs, see "To view the logs for a service provider" on page 152.

Creating and monitoring client interfaces

A client is an external application that communicates with Data Integrator through the Access Server.

There are two types of client interfaces in the Administrator:

- RFC clients
- Message broker clients

Configure RFC clients in the Administrator for real-time jobs that use SAP R/3 IDocs. To support these jobs, create a remote function call (RFC) client interface and attach IDoc configuration information to it.

Data Integrator creates message broker client interfaces when communication occurs between the Access Server and an external application that uses Data Integrator message client libraries. To monitor message statistics, view the message broker clients of each Access Server as needed. For information about configuring an external application, see "Using the Message Client library" on page 117 of the *Data Integrator Getting Started Guide*.

This section describes configuration and monitoring for each type of client. Specific topics are:

RFC clients

Configure IDoc *message sources* in the Administrator as well as in Data Integrator Designer. Other IDoc sources and targets need only be configured using the Data Integrator Designer. See the *Data Integrator Supplement for SAP* for detailed information about how Data Integrator handles IDocs in real-time jobs.

NOTE: Using the Administrator, create a *service* for your real-time job that contains an IDoc as a message source before you configure an RFC Client.

An RFC client uses the SAP R/3 RFC protocol to communicate with the Access Server. An RFC client requires connection information so that an Access Server can register to receive IDocs from an SAP R/3 application server. An RFC client can process one or more IDoc types. An RFC client specifies which service will process a particular IDoc type and whether or not the RFC client connection can process an IDoc type in parallel.

The process of creating an RFC client interface for IDocs has two parts:

- Adding an RFC client
- Adding IDoc configurations to an existing RFC client

Configure one RFC client per Access Server. This means that you can only process IDocs from one instance of SAP R/3. To process IDocs from more than one instance, configure more than one Access Server.

NOTE: SAP R/3 function modules are responsible for IDoc processing. In Data Integrator, the RFC client might fail if multiple IDocs are sent from SAP R/3 and you previously set SAP R/3's packet size to 1. Therefore:

- Do not enable the option of immediate IDoc dispatch in SAP R/3 unless the volume of produced IDocs is very light (no more than one IDoc per minute).

- For batch processing of IDocs, the packet size should never be smaller than 5 or larger than 1000. The following numbers are rough estimates for this parameter:

IDoc Processing Volume	IDocs per day	Packet size
Light	1 - 300	5
Medium	301 - 1000	20
Heavy	301 - 1000	80
Very Heavy	more than 5000	800

➢ To add an RFC client

1. Select **Real-time** > ***Access Server*** > **Client Interfaces**.
2. Click the **Configuration** tab.
3. Click **Add**.

   The Administrator opens the RFC Client Configuration page.
4. Enter the client information in the **Configuration** section.

All options, except the RFC program ID, SAP gateway host name, and SAP gateway service name must match SAP R/3 datastore settings. You might want to create a single SAP R/3 datastore for processing real-time jobs with IDoc message sources. See Chapter 3, "SAP Datastores," in the *Data Integrator Supplement for SAP*.

Field	Description
RFC program ID	This is the RFC Server registration ID and is used as the Program ID in the R/3 Destination Configuration.
R/3 user name	User name through which Data Integrator connects to this SAP R/3 application server. Use the same user name used to create the SAP R/3 datastore. You created this datastore to design the jobs that include this IDoc.
R/3 user password	Password for the user account through which Data Integrator connects to this SAP R/3 application server.
SAP application server name	The domain name of the computer where the SAP R/3 application server is running.
R/3 client number	The SAP R/3 application client number.
R/3 system number	The SAP R/3 application system number.
SAP gateway host name	The domain name of the computer where the SAP R/3 RFC gateway is located.
SAP gateway service name	The TCP/IP service name for the SAP R/3 application server gateway. Typically, this value is **SAPGW** and the system number.

5. Click **Apply**.

   The Administrator adds this client definition and returns to the Client Interface Status page.

Adding IDoc configurations to an RFC client

Once you have created an RFC client, you can list the IDoc types that you want to receive.

➢ To add an IDoc configuration to an RFC client

1. Select **Real-time** > ***Access Server*** > **Client Interfaces**.
2. Click the **Configuration** tab.
3. Click the name of an existing RFC client interface.

   The RFC Client Configuration page opens.
4. Click the **Supported IDocs** link.
5. Click **Add**.
6. Enter IDoc information:
   - In the **IDoc Type** box, enter the IDoc type that this SAP R/3 application server will send to the Access Server.
   - In the **Service Name** box, enter the name of the service that will process this IDoc.

     The service identifies the job that processes this IDoc. See "Configuring and monitoring real-time services" on page 63 for more information.
   - If you want the Access Server to read IDocs (of this type and from the specified SAP R/3 source) in parallel, check the **Parallel Processing** check box.

     Real-time services that contain an IDoc message source can be processed one at a time or in parallel. The **Parallel Processing** option allows you to increase the number of IDoc source messages processed per minute for the IDoc type specified. This option is disabled by default. The Parallel Processing option allows the Access Server to send an IDoc to a service queue (where it waits for a service provider) and continue with the next IDoc without waiting for reply. The maximum number of outstanding IDoc requests in the queue is the number of IDocs received or four, whichever is smaller.

     NOTE: Where a strict IDoc processing sequence is required, do not use the **Parallel Processing** option.

7. Click **Apply**.
8. (Optional) Click **Add** to include more IDocs. Repeat steps 6 and 7.
9. From the navigation tree, select Select **Real-time** > ***Access Server*** > **Client Interfaces**.
10. From the Client Interface Status page, select the check box next to the new RFC client and click **Start**.

    The Administrator starts the RFC client. A green indicator signifies that the client is running. Detailed status information is provided in the **Status** column.

➢ To close connections to an RFC client interface

1. Select **Real-time** > ***Access Server*** > **Client Interfaces**.
2. Select the check box next to the RFC client you want to disconnect.

   If you choose **Shutdown**, the Access Server allows the clients to finish processing any active requests before closing the connection. The Access Server responds with an error to any new requests that arrive during that interval.

   If you choose **Abort**, the Access Server closes the connection to the client without responding to requests currently being processed.
3. Click **Shutdown** or **Abort**.

Message Broker clients

A Message Broker client uses an XML message to communicate with a Data Integrator Access Server.

Message Broker clients include:

- External applications
- Adapters

- Service providers

Use the Administrator to monitor Message Broker clients.

Monitoring clients

From the Client Interface Status page, you can view the overall status of all client connections.

➢ To view the overall status of client connections

1. Select **Real-time** > ***Access Server*** > **Client Interfaces**.
2. Verify that the RFC client connections are working.

Indicator	Description
	A green indicator signifies that each client of this type has an open connection with Access Server.
	A yellow indicator signifies that at least one client of this type is disconnecting.
	A red indicator signifies that the Access Server could not reserve the specified port to listen for client requests.

If an RFC client interface has a red status:

a. View details in the Status column and click the name of the client to view statistics about the particular client connection with a problem.

b. If you want to restart, abort, or shut down a client interface, click 

in the navigation bar. The Administrator returns to the Client Interface Status page.

c. Click **Start**, **Abort**, or **Shutdown**.

See Chapter 10, "Troubleshooting" for more information.

➢ To monitor Message Broker clients

Select **Real-time** > ***Access Server*** > **Client Interfaces**.

Under **Message Broker Clients**, this page lists each message broker client that has registered with the Access Server along with statistics for that client.

NOTE: The first client in this list is the Administrator. You registered with the Access Server when you added connection information to the Administrator.

Message broker client interface information includes:

- Name — The name of the client.
- Time Connected — The total time that this client has been connected to the Access Server.
- Last Message Received — The length of time since the Access Server has received a message from this client.
- Last Message Sent — The length of time since the Access Server has sent a message to this client.
- Received Messages — The number of messages the Access Server has received from this client.
- Sent Messages — The number of messages that the Access Server has sent to this client.

7 Real-Time Performance

This chapter discusses the Access Server parameters, statistics for services and service providers, and how to tune the performance of services and service providers.

Specific sections include:

- Configuring Access Server output
- Service configuration parameters
- Service statistics
- Service provider statistics
- Using statistics and service parameters

Configuring Access Server output

You can configure the Access Server to control its operation and output such as sending specific event information to its trace log.

Data Integrator installation includes a server configuration utility called the Server Manager. The Server Manager allows you to view and change the following Access Server information:

Option	Description
Directory	The location of the configuration and log files for this instance of the Access Server. Do not change this value after the initial configuration.
Communication Port	The port on this computer the Access Server uses to communicate with the Administrator and through which you can add additional configuration information to an Access Server. Make sure that this port number is not used by another application on this computer.
Parameters	Command-line parameters used by the Data Integrator Service to start this Access Server. For development, consider including the following parameters: `-P -T16` where `-P` indicates that trace messages are recorded, and `-T16` indicates that the Access Server collects events for services and service providers. These parameters are described in "The following parameters are available to control the operation and output of an Access Server:" on page 88.
Enable Access Server	An option to control the automatic start of the Access Server when the Data Integrator Service starts.

➢ To configure an Access Server

1. Select **Start** > **Programs** > **Business Objects Data Integrator *version*** > **Server Manager**.
2. In the Data Integrator Server Manager, click **Edit Access Server Config**.

   The Access Server Configuration Editor opens.
3. Click **Add** to configure a new Access Server or select an existing Access Server, then click **Edit** to change the configuration for that Access Server.
4. Make the appropriate changes in the Access Server Properties window.
5. Click **OK** to return to the Access Server Configuration Editor.
6. Click **OK** to return to the Server Manager.
7. Click **Restart** to stop and start the Data Integrator Service with the new Access Server configuration.

The following parameters are available to control the operation and output of an Access Server:

Parameter	Description
`-A`	Specifies the communication port for an Access Server. The default value is `-A4000`.
`-C`	Disables display output.
`-H`	Prints the parameter list to the console.
`-P`	Enables trace messages to the console and log.
`-R`*`root_directory`*	Indicates the location of the Access Server directory.
`-T`*`value`*	Determines the type of tracing information displayed in the console and the Access Server log. Use any value or any combination of values: 1 system 2 real-time service flow 4 client 8 transaction 16 service 64 administration 128 request For example, to enable tracing for both system-level and service-level operations, include the value 17 after the `T` parameter.
`-V`	Displays the version number of the Access Server.
`-VC`	Displays communication protocol and version number.
`-X`	Validates the Access Server configuration without launching the Access Server.

The `-A` and `-R` parameters can also be set using the Server Manager.

The `-P` and `-T` parameters can be set using the Administrator. Select **Real-Time** > ***Access Server*** > **Logs-Current** then click the **Configuration** tab.

Service configuration parameters

Each service contains configuration parameters that control how the Access Server dispatches requests to the assigned real-time job. These parameters determine how the system handles errors that occur during operation. For information about setting these parameters, see "To add a service" on page 63.

Often, requirements during development differ from requirements during production. Therefore, the values of your configuration parameters differ during development and production. To ensure that the system works as expected, test the values before committing the Access Server configuration to production use.

Parameters control different categories of Access Server operation:

- Service startup behavior
- High traffic behavior
- Response time controls

Service startup behavior

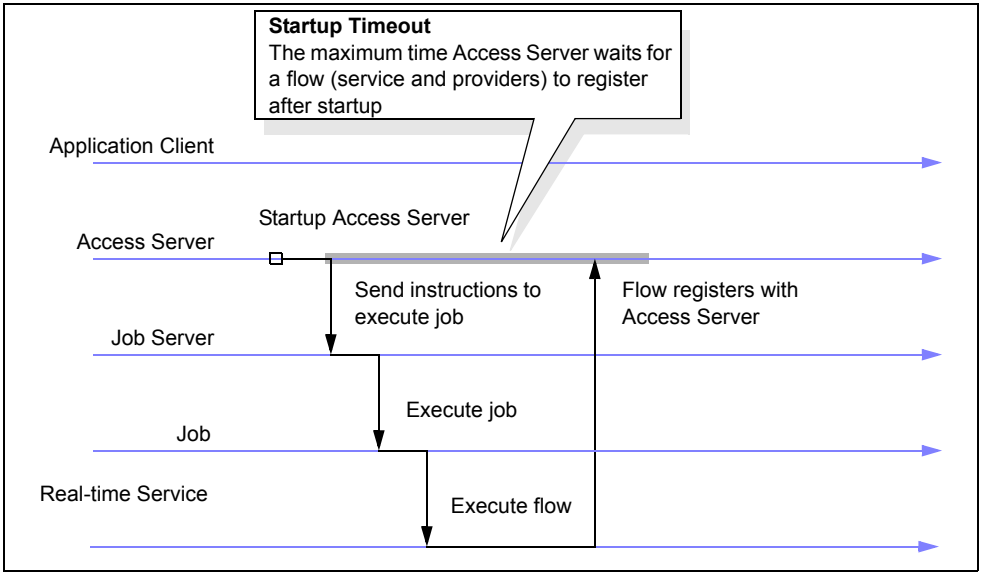
Use two parameters to configure how the Access Server starts service providers associated with a particular service:

- **Startup timeout** — The maximum time the Access Server waits for a flow (service and its providers) to register after startup.
- **Recycle request count max** — The number of requests the Access Server sends to a given flow before automatically recycling.

When the Access Server starts, it immediately starts the service providers for each service. If you want the Access Server to start more than one instance of a service to process a particular type of message, you must define more than one service provider for the service.

The Job Servers launch the jobs, which in turn initiate their corresponding real-time services. The first operation of each real-time service is to register with the Access Server.

If an error occurs and a real-time service fails to register, the Access Server instructs the Job Server to restart the job. The Access Server waits the length of time that you configure as the **Startup timeout** before instructing the Job Server to start the job again. The startup timeout is in seconds. The Access Server continues to instruct the Job Server to restart the job until the real-time service registers.



You can also control how many requests a particular service provider processes. After a provider processes the number of requests specified by **Recycle request count max**, the Access Server automatically recycles the service provider—that is, the Access Server automatically stops the current instance of the real-time service and starts a new instance of that service. Setting this parameter to a higher value increases the time that the service provider is available to accept requests for processing. Setting this parameter to a lower value refreshes any data cached in the real-time service more often.

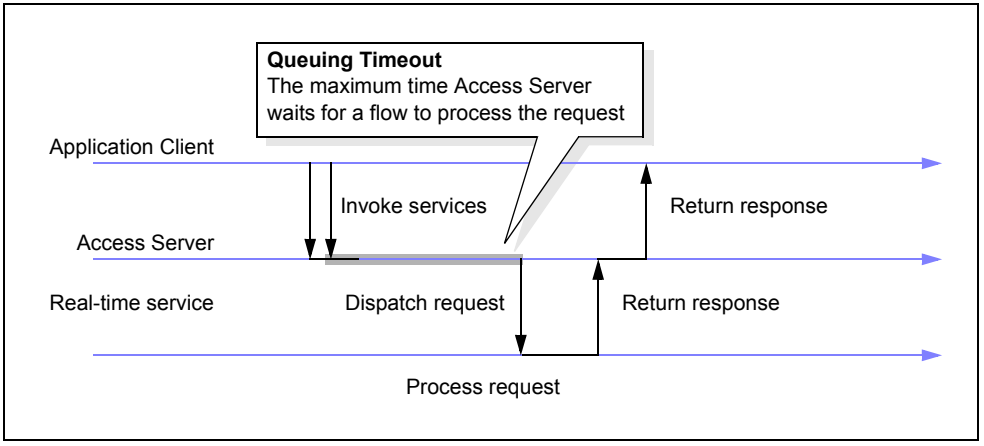
High traffic behavior

Use the **Queuing timeout** parameter to specify the maximum amount of time the client application must wait for a request to be processed.

If the number of requests the Access Server receives for a particular service exceeds the number of registered service providers that can process those requests, the Access Server queues the requests in the order they are received. When a service provider completes processing a request and responds to the Access Server, the Access Server dispatches the next request in the queue for that service to the open service provider.

If there are many requests and the queue causes requests to exceed the queuing timeout, the Access Server removes the oldest request from the queue and responds to the client with an error indicating that the request failed. You can use the queueing timeout to ensure that the client receives a timely response, even during high-traffic periods.

The queuing timeout is in seconds.



A service experiences *high traffic* when the available resources cannot process the received requests efficiently. High traffic occurs when the time messages wait to be processed exceeds the time required to process them. See "Using statistics and service parameters" on page 98.

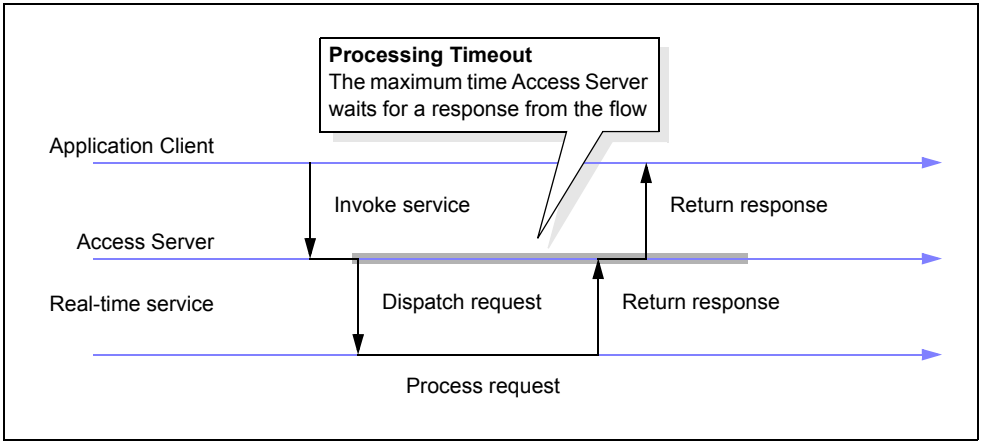
Response time controls

Use two parameters to configure how long the Access Server waits for responses from service providers for a particular service:

- **Processing timeout**
- **Processing retry count max**

After the Access Server sends a request to a service provider to process, the Access Server waits for the response. If the response does not arrive within the specified processing timeout, the Access Server sends the request to another waiting service provider. The **Processing timeout** is in seconds.

If the first attempt fails, the Access Server will attempt to process the request as many times as you specify in the **Processing retry count max** parameter.



If **Processing retry count max** is set to zero, the maximum response time is equal to the queuing timeout plus the processing timeout.

Service statistics

The Real-time Service Status page for a particular service shows overall statistics.

- Number of processed requests

  The number of requests for this service from any client that the Access Server received and responded to since the last time the Access Server started.

- Number of requests in the queue

  The number of messages the Access Server has received from a client for this service but has not sent to a service provider for processing.

  This value reflects the current state of the Access Server.

- Max queuing time (milliseconds)

  The maximum time any request for this service waited after the Access Server received the message and before the Access Server sent the request to a service provider for processing.

- Average queuing time (milliseconds)

  The average time that requests for this service waited after the Access Server received the request and before the Access Server sent the request to a service provider for processing.

- Queuing timeouts

  The number of requests to which the Access Server replied to the client with an error indicating that there was no service provider available to process the request.

- Max processing time (milliseconds)

  The maximum time required to process a request for this service. It is the difference between the time the Access Server sent the request to a service provider and the time that the Access Server responded to the client. The processing time does not include time the request spent in a queue waiting to be sent to a service provider.

- Average processing time (milliseconds)

  The average time required to process a request for this service. It is the difference between the time the Access Server sent the request to a service provider and the time that the Access Server responded to the client. The processing time does not include time the request spent in a queue waiting to be sent to a service provider.

- Processing timeouts

  The number of requests that the Access Server sent to a service provider and did not receive a response before exceeding the processing timeout. These requests are either successfully processed by another service provider, or if they are left unprocessed beyond the time indicated by the queuing timeout parameter, the Access Server returns an error to the client.

Service provider statistics

The Service Provider Status page shows the statistics for an instance of a real-time service.

When Data Integrator measures a statistic "from the start," the value does not restart when the Access Server restarts the service provider. The value restarts when the Access Server restarts.

When Data Integrator measures a statistic "for the current service provider," the value restarts when the Access Server restarts the service provider, either due to error or when the service provider reaches the maximum number of requests defined for the service.

- Max processing time (milliseconds)

  The longest time it took between when the Access Server sent a message to this service provider and when the service provider returned a response.

- Average processing time (milliseconds)

  The average time it took between when the Access Server sent a message to this service provider and when the service provider returned a response.

  If you are running more than one service provider for this service, compare this statistic with the same statistic from the other instances. If this instance is significantly different, look for processing constraints on the computer where this instance runs.

- Processed requests (for the current service provider)

  The number of requests that the Access Server sent to this service provider to which the service provider responded.

- Processed requests (since start)

  The number of requests that the Access Server sent to this service provider to which the service provider responded.

- Error replies received from the start

  The number of requests that the Access Server sent to this service provider to which the service provider responded with an error.

- Communication errors encountered from the start

  The number times that the communication link between the Access Server and this service provider failed.

- Timeout errors encountered from the start

  The number of times the Access Server sent requests to this service provider and did not receive a response within the time specified by the processing timeout.

- Service provider connections (restarts) from the start

  The number of times the Access Server restarted this service provider when it did not receive a response from the service provider.

- The last time of a successful flow launch

  The system time when the Access Server last started the real-time service associated with this service provider. If the Access Server never successfully started an instance of this service provider, the value is "N/A."

  This time is from the computer running the Access Server.

- Time since start attempt

  The amount of time since the Access Server last attempted to start this service provider. This value reflects successful and unsuccessful attempts.

- Time since last request start

  The amount of time since the Access Server last sent a request to this service provider. This value reflects successful and unsuccessful attempts.

Using statistics and service parameters

You can use the statistics for a service to tune the service parameters.

- Average and maximum processing time

  If the average or maximum processing time for a service provider is equal or close to the processing timeout value resulting in processing timeouts, consider increasing the processing timeout parameter for the service.

- Maximum queuing time

  In a tuned system, the maximum and average queuing times should be close together, the difference being an indication of the traffic distribution for this service. Values should not approach the value of the queuing timeout parameter listed for the service.

  If the maximum queuing time for a service provider is equal or close to the queuing timeout parameter and there are queuing timeouts listed, consider the following changes:

  - Increase the queuing timeout parameter for the service
  - Increase the number of service providers available, either controlled by the same Job Server host or by a different Job Server

  If you find that the average time in the queue is longer than the average processing time, the traffic for this service is too high for the resources provided. Consider running multiple service providers to process the same message type. You can add the same job many times in the service list, or you can add the same job controlled by a different Job Server on a separate computer to the service list.

  If you find that the average queuing time is growing, consider increasing the queuing timeout or adding processing resources.

- Processing timeouts

  If you see processing timeouts and service providers restarting successfully, consider increasing the number of processing retries allowed for the service.

8 Adapters

This chapter describes how to add an adapter to the Data Integrator system, how to start an adapter instance, and how to monitor an adapter's operation instances.

Specific sections include:

- Overview
- Adding and configuring adapter instances
- Starting and stopping adapter instances
- Monitoring adapter instances

Overview

A Data Integrator adapter is a Java program that allows Data Integrator to communicate with front-office and back-office applications. Depending on the adapter implementation, Data Integrator adapter capabilities include the ability to:

- Browse application metadata
- Import application metadata into the Data Integrator repository
- Move batch and real-time data between Data Integrator and information resource applications

Data Integrator adapters can handle the following types of metadata: tables, documents, functions, outbound messages, and message functions. Each of these can be used in real-time or batch jobs. Outbound messages and message functions are the only objects that include operations.

An adapter can process several predefined operations. An operation is a unit of work or set of tasks that the adapter completes. Possible operations include:

- Take messages from an application and send them to a real-time service for processing, possibly returning a response to the application
- Take messages from a real-time service and send them to an application for processing, possibly returning a response to the real-time service
- Take messages produced by a function call inside a real-time service, send the messages to an application, and return responses to the function

An adapter connects Data Integrator to a specific information resource application. You can create one or more instances of an adapter. Each adapter instance requires a configuration file. That configuration file defines the operations available.

All adapters communicate with Data Integrator through a designated Job Server. You must first install an adapter on the Job Server's computer before you can use the Administrator and Designer to integrate the adapter with Data Integrator. See your specific adapter's documentation for its installation instructions.

After installing the adapter, configure its instances and operations in the Administrator before creating adapter datastores in the Designer because you must select an adapter instance name as part of an adapter datastore configuration. It might help to think of the **Adapter Instances** node of the Administrator as part of your adapter datastore configuration.

➢ To create an adapter datastore connection in the Designer:

1. First, configure an adapter instance.
   a. Use the Server Manager utility to configure a Job Server that supports adapters and an associated repository — See "Adapter Considerations" on page 18 and "To configure Job Servers" on page 90 of the *Data Integrator Getting Started Guide*.
   b. Use the Administrator to:
      1. Add a connection to the Job Server's associated repository by selecting **Management** > **Repositories** > **Add**.
      2. Add at least one instance of the adapter to the Data Integrator system by selecting **Adapter Instances** > ***Job Server*** > **Configuration** > **Add.**
      3. If the adapter instance includes operations, add at least one operation for each adapter instance.
      4. Start the adapter instance (operations start automatically)
2. Second, use the Designer to create an adapter datastore and import metadata. Use the metadata accessed through the adapter to create batch and/or real-time jobs — See "Adapter datastores" on page 106 of the *Data Integrator Designer Guide*

When you are ready to administer jobs that include objects that use the adapter datastore, add the jobs' repository connection to the Administrator, open the **Batch** or **Real-Time** nodes, and use them as you would for any job.

You can also monitor adapter instances and operations in the **Adapter Instances** node of the Administrator.

Adding and configuring adapter instances

Use the Administrator to add adapter instance configuration information to the Data Integrator system and to edit an existing configuration.

How to add and configure an adapter instance

Until you add an adapter interface using the Administrator, you cannot run jobs using information from that adapter.

➢ To add an adapter instance

1. Select **Adapter Instances** > ***Job Server***.

   If there are no Job Servers listed in this navigation path, see "To create an adapter datastore connection in the Designer:" on page 103.

2. Click the **Configuration** tab.
3. Click **Add**.
4. Click an adapter from the list of those installed on the Job Server with which you are working.

   NOTE: The HTTP adapter and the Web Services adapter automatically install with every Job Server. Both adapters allow you to call external applications from Data Integrator, one using the HTTP or HTTPS protocol and the other using the SOAP protocol. Use the Web Services adapter to create outbound calls because it automatically configures and starts when a Job Server is enabled for use with adapters. However, if you want to use the HTTP adapter you can, but you must build it like any other Data Integrator adapter.

   Find more information on the Web Services adapter in Chapter 9, "Support for Web Services". For more information about the HTTP adapter see the *Data Integrator HTTP Adapter Guide*.

5. Enter the required information to create an adapter instance. See "Adapter instance configuration information" on page 106.
6. Click **Apply**.

   The Administrator adds the adapter instance to the list available to the Data Integrator system.

➢ To edit an adapter's configuration

1. Select **Adapter Instances** > ***Job Server***.
2. Click the **Configuration** tab.
3. Click the name of the adapter instance that you want to edit.

   The Administrator displays the current configuration information.
4. Edit the configuration information. See "Adapter instance configuration information" on page 106.
5. Click **Apply**.

   The Administrator updates the information.

Adapter instance configuration information

Complete the following fields in the Administrator to set up an adapter instance in the Data Integrator system:

NOTE: The **Adapter Instance Name** is the only option required if your adapter instance is for batch jobs.

- **Adapter Instance Name**

  A unique name that identifies this instance of the adapter.
- **Access Server Host**

To run an adapter instance in a real-time job, you must configure a service that will be called from a given Access Server. Enter the host ID of the computer running the Access Server for which you will configure a service for a real-time job that contains this adapter instance.

- **Access Server Port**

  The communication port of the Access Server host is used to both connect an Access Server to Data Integrator components and to broker messages with external applications. After you log in to the Administrator, select **Real-time** > ***Access Server*** > **Client Interfaces** to view an Access Server's message broker port information.

- **Adapter Retry Count**

  Number of times to retry an adapter instance if it fails or crashes. Enter `0` to indicate no retries. Enter a negative number to retry the instance indefinitely.

- **Adapter Retry Interval**

  Number of milliseconds to wait between adapter retry attempts.

- **Classpath**

  All adapter Java programs require specific jar files in the CLASSPATH to use when starting the `javaw.exe`. For example:

  - `LINK_DIR\lib\acta_adapter_sdk.jar`
  - `LINK_DIR\lib\acta_broker_client.jar`
  - `LINK_DIR\lib\acta_tool.jar`
  - `LINK_DIR\ext\lib\xerces.jar`

  Your adapter program might require different jar files.

  You can change the system CLASSPATH environmental variable, or you can use this option to enter a CLASSPATH parameter for the required jar files. If you use this option, enter, for example:

```
C:\Data Integrator\lib\acta_adapter_sdk.jar;
Data Integrator\lib\acta_broker_client.jar;
Data Integrator\lib\acta_tool.jar;
Data Integrator\ext\lib\xerces.jar
```

- **Autostart**

  Enable the adapter interface to start automatically when Data Integrator starts by setting this option to **True**.

- **Trace mode**

  A flag that controls the amount of trace messages the adapter writes. There are two settings:

  - **True**— Select to have the adapter interface write additional information messages to help debug problems
  - **False** — Select to have the adapter interface write only minimal information messages

  The adapter writes trace messages to the *adapter_instance_name*_trace.txt file in the LINK_DIR\adapters\logs directory.

- **Additional java launcher options**

  In addition to the classpath, you can use additional options when launching java processes (javaw.exe for Windows and java.exe for UNIX platforms). Here are some examples:

  - If you do not define a value in this box, the default options, for memory usage, are: `-Xms128m -Xmx256m.`
  - If you get an out-of-memory error from an adapter, then you can re-configure its instance by editing the additional java launcher options. For example: `-Xms512m -Xmx1024m`
  - If an adapter requires that you define a system property, do so by editing the additional java launcher options: `-Xms128m -Xmx256m foo="string"`

- **Adapter type name**

(Read-only) The name of the adapter used to create this instance.

- **Adapter version**

  (Read-only) The version of the adapter used to create this instance.

- **Adapter Class**

  (Read-only) A name that identifies the adapter class. The name depends on the type of adapter:

  - For prepackaged adapters, see the adapter documentation.
  - For custom adapters, this is the adapter's fully qualified Java class name:

    package_name.*class_name*

    where:

    package_name is the Java package name for the adapter as defined in the adapter's Java file.

    class_name is the Java class name for the adapter as defined in the adapter's Java file.

  For example, suppose the adapter's Java file contains these lines:

```
package com.acta.adapter.SiebelAdapter

public class SiebelAdapter implements Adapter
```

  Then, the adapter class name is:

```
com.acta.adapter.SiebelAdapter.SiebelAdapter
```

- **Root Directory**

  Examine the adapter's root directory name. Edit this name as needed.

➢ To add operation instances to an adapter instance

1. Select **Adapter Instances** > ***Job Server***.
2. Click the **Configuration** tab.
3. Click **Operations** under Dependent Objects.
4. Click **Add** to configure a new operation.

   Here you can also click the link of an existing operation instance to edit its configuration.
5. Select an operation type from the list.

   The options that appear on this page depend on the adapter's specific design.
6. Click **Apply**
7. Complete the operation instance configuration page.

   The options and descriptions that appear on this page depend on the adapter's specific design. Consult your adapter-specific documentation for details.
8. Click **Apply**.

Starting and stopping adapter instances

Use the Administrator to start and stop an adapter instance and its operations.

NOTE: After you configure an adapter instance, each time you stop and start the Access Server, you stop and start the adapter instance and its operations.

➢ To start an adapter instance

1. Select **Adapter Instances** > ***Job Server***.
2. Select the check box next to the adapter instance you want to start.
3. Click **Start**.

   The Administrator starts the adapter instance and all of its operations.

➢ To stop an adapter instance

1. Select **Adapter Instances** > ***Job Server***.
2. Select the check box next to the adapter instance you want to stop.
3. Click either **Shutdown** or **Abort**:
   - Select **Shutdown** to stop an adapter and all of its operations gracefully. The adapter will complete any pending operations before shutting down.
   - Select **Abort** if you want to stop all operations immediately. Select **Abort** only if incomplete operations are acceptable.

➢ To start or stop an adapter operation instance

1. Select **Adapter Instances** > ***Job Server***.

2. Select the check box next to the operation instance you want to start or stop.

   When you start an adapter instance, its operations will also start. However, you can also start and stop individual operation instances manually using this page.

3. Click either **Start** or **Shutdown**.

Monitoring adapter instances

Use the Administrator to monitor adapters and their operations.

➢ To monitor the adapter instances and operations

1. Select **Adapter Instances** > ***Job Server***.

   The Adapter Instance Status page lists each adapter instance and its operations.

2. Find the overall status of a particular adapter instance or operation by examining the indicators.

Indicator	Description
	A green indicator signifies that the adapter instance or operation has started and is currently running.
	A yellow indicator signifies that the adapter instance or operation is not currently running.
	A red indicator signifies that the adapter instance or operation has experienced an error.

   For each operation, this page lists four statistics:

   - Requests Processed

     The number of requests for this operation instance that were processed. Processing of these requests is complete.

   - Requests Pending

     The number of requests for this operation instance that are still pending. Processing of these requests is not complete.

   - Requests Failed

     The number of requests for this operation instance that have failed. The operation has stopped processing these requests.

   - Status

     For operations, displays error text.

You can also find more detailed adapter instance information in the **Status** column. Possible values include:

- Initialized
- Starting
- Started
- Shutting Down
- Shutdown
- Error text — Displays the last error message that occurred as the adapter instance shut down or indicates that the configuration has changed. To allow the adapter instance to use the changes, restart the adapter instance.

For more detailed information about the adapter instance, view the error and trace log files.

➢ To monitor adapter instance statistics

1. Select **Adapter Instances** > ***Job Server.***
2. Click the name of an adapter instance.

   The statistics for the instance appear. The options and descriptions that appear on this page depend on the adapter's specific design. Consult your adapter-specific documentation for details.

9 Support for Web Services

This chapter contains the following sections:

Overview

BusinessObjects Data Integrator can:

- Publish any job as a callable Web Service (call-in functionality)
- Call external Web Services from within its jobs using a built-in Web Services adapter (call-out functionality)

If you have an application that also supports Web Services, you can use that application to automatically launch Data Integrator batch jobs and use real-time services. You can also publish your application's functionality, which Data Integrator can use in its data flows.

Web Services are modular business applications based on open standards (WSDL, SOAP, and XML Schema primarily) that allow integration among different applications and environments through the Internet. Web Services allow parts of existing applications to be used by other applications.

For business intelligence (BI), you can use Web Services to:

- Access legacy systems
- Conduct computer-to-computer interaction over an internal or external Web
- Allow applications constructed in different languages on different platforms to communicate with each other in an enterprise environment

The Web Services Internet standard has emerged as a leading technology shift. In the same way that the HTTP standard turned the world's information systems into a single, searchable, content library, the Web Services standard can turn the world's applications into a single, function call library. Web Services allow called resources to interact with and modify data.

Web Service technologies

Data Integrator supports three XML Web Service technologies:

Web Services Technologies	Description
SOAP	Connection protocol
WSDL	Metadata for Web Services data (sub-set of XML)
XML Schema	Format used for the WSDL file

SOAP

In addition to allowing you to invoke real-time services using the Message Client Libraries (which support C++, COM, and JAVA connections), TCP/IP, and proprietary XML using HTTP, Data Integrator supports the Simple Object Access Protocol (SOAP). SOAP is the industry standard from the Word Wide Web Consortium (WC3.org) used to invoke network resources using XML over HTTP, HTTPS, and other standard protocols.

A SOAP gateway is built-in to the Data Integrator Administrator. Data Integrator supports SOAP version 1.1 over HTTP and HTTPS protocols.

WSDL

Web Services description language (WSDL) is a subset of XML. It is used as a transport mechanism for XML messages. Data Integrator publishes its jobs in WSDL so that your external applications can call them. Data Integrator also publishes all comments entered into the Designer's **Job Descriptions** box with each job in WSDL.

The WSDL file generated by Data Integrator includes tags (such as services, bindings, ports, and operations) that support the use of the SOAP protocol. Each tag has names associated with it that Data Integrator provides. For example:

- All Data Integrator jobs are published in a single, Web Service named `DataIntegrator_server`. In WSDL, a service is a set of business operations with connection endpoints.
- Binding names include `Connection_Operations`, `Batch_Jobs`, and `Real-time_Services`. WSDL uses bindings to associate operations with ports.
- Operation names have a one-to-one relationship with the names of Data Integrator batch jobs or real-time services.

Data Integrator supports WSDL version 1.1.

XML Schema

WSDL uses XML Schema to define input and output message formats. XML Schema has become an industry standard for formatting XML messages, replacing Document Type Definitions (or DTD formats).

- For call-in functionality, if a real-time service was defined with DTDs, Data Integrator translates the DTD format into the XML Schema format.
- For call-out functionality, the Web Services adapter imports metadata into Data Integrator using the XML Schema format only.

XML Schema formats are defined in imported XML Schema files and available to the WSDL through an import statement (`<import> </import>`). This statement is located above each set of message definitions for a real-time service in the WSDL file.

For general information about Data Integrator support for XML Schema, see "Using XML Schemas" on page 225 of the *Data Integrator Designer Guide*.

UDDI

UDDI is a method if publishing comments and other reference information about jobs to an external web site. Data Integrator does not publish information to a UDDI web site because most Data Integrator Web Services users work behind enterprise firewalls.

Call-in functionality

Data Integrator can publish its jobs as Web Services. Data Integrator allows you to invoke batch jobs and real-time services as well as set global variables and job parameters externally. The Administrator publishes Web Services. The Data Integrator Web Server hosts Web Services. When an external application calls into Data Integrator through Web Services, it acts as a SOAP client accessing a Web Services server.

SOAP clients call the published Web Services, pass in the appropriate parameters, and receive the results. Data Integrator routes SOAP calls to the appropriate Data Integrator Job Server and job for processing.

For example, Web Services might be used in the following scenarios:

- Dynamically update an internal Web site

  Suppose you have an internal web site that manages foreign exchange rate status worldwide for the Finance department. When foreign exchange rates change more than a certain percentage, a batch Data Integrator job updates exchange rates in your financial data mart. The rate change initiates a call to a Data Integrator-hosted Web Service that starts the appropriate batch ETL job.

- Solve a processing issue

  Suppose you have an existing Enterprise Application Integration (EAI) bus infrastructure and want to manage batch processes and EAI transactional processes from within the same infrastructure. The transactional processes are complex. Their staging is laid out in the order process. However, EAI work flows do not have the ability to run batch processes.

Data Integrator can publish Web Services that allow you to leverage EAI process management category tools (e.g. webMethods Business Process Manager) to control and stage batch processes alongside its transactional processes.

The work flows might call Data Integrator to:

- Perform an initial load of a data mart for real-time reporting
- Refresh the data cache depending on specified business criteria
- Perform complex transforms on hierarchical objects for mapping data between ERP systems

WSDL basics

WSDL is a subset of XML that you can use to describe network services as a collection of endpoints capable of exchanging messages.

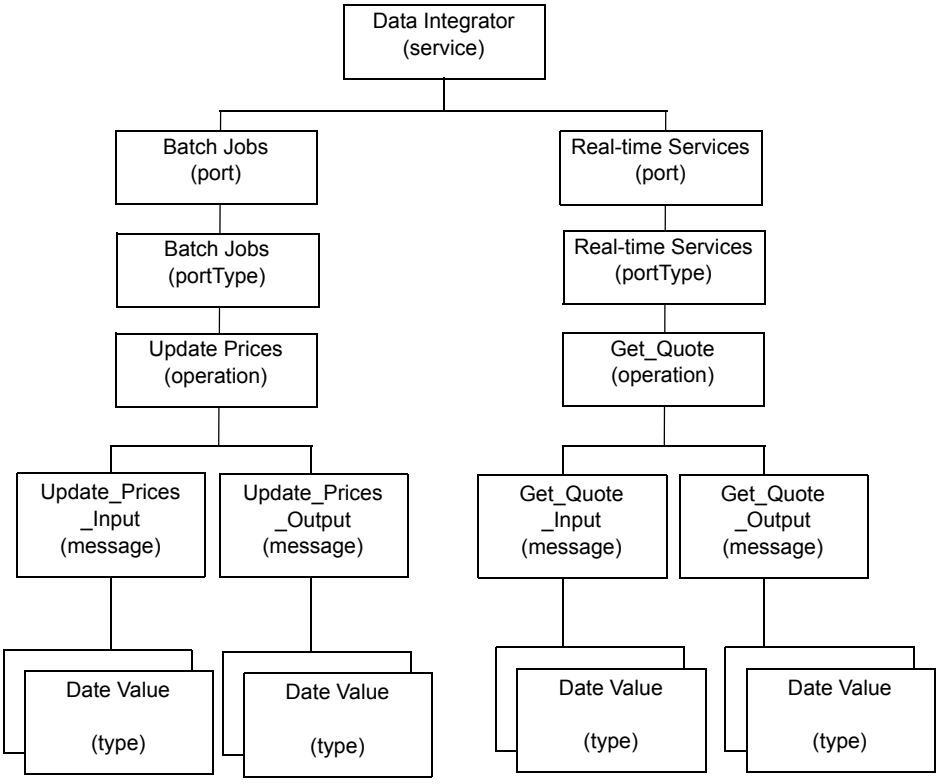
A WSDL file has the following elements:

Element Name	Description
definition	Root element
service	Used to group a set of related ports or endpoints to which a client application will connect. Data Integrator publishes a single service in the WSDL file it generates.
port	Defines a specific Web Service endpoint that a client can access. Each port has a unique name and a specific address used for binding. Data Integrator defines, ports for Web Services as: • `Connection_Operations` • `Real-time_Services` • `Batch_Jobs`
portType	Defines a set of operations that a Web Service publishes. A portType is bound to a particular port. The binding specifies the protocol and data formation for the operations defined by a portType. For batch jobs, see also SoapAction element.
operation	Defines a specific function call. Data Integrator publishes each batch job and real-time service as an operation. Data Integrator also publishes connection operations.
message	Defines the data to transmit. There is an input (request) message, which the Web Service receives from the client, and there is an output (response) message, which the Web Service sends back to the client.
type	Defines the data types used in messages sent to/from a Web Service.

The following is an example of the WSDL elements that Data Integrator uses to publish batch jobs and real time services. In this example, the WSDL publishes:

- A service named `DataIntegrator`
- Ports named `Batch_Jobs` and `Real-time_Services`
- A batch job operation named `Update_Prices` and a real-time service operation named `Get_Quote`.

- Each operation has an input and output message.



Data Integrator service and port definitions

Data Integrator creates a WSDL file with a single service definition. It is possible to create multiple service definitions in a WSDL, but many Web Service implementations do not support more than one service definition. Data Integrator only creates one service to avoid that limitation.

Within the service definition, Data Integrator defines ports for:

- `Connection_Operations`
- `Real-time_Services`

- `Batch_Jobs`

The following segment shows the syntax used in the generated WSDL file to define a service and ports. In this example, `URL_for_WebService represents` the URL of the Data Integrator Web Service.

```
<service name='DataIntegrator_Server'
   <port name='Connection_Operations'
    binding='tns:Connection_Operations'>
      <soap:address location='URL_for_WebService'/>
   </port>
   <port name='Batch_Jobs' binding='tns: Batch_Jobs'>
      <soap:address location='URL_for_WebService'/>
   </port>
   <port name='Real-time_Services'
    binding='tns:Real-time_Services'>
      <soap:address location='URL_for_WebService'/>
   </port>
</service>
```

The WSDL for the service and port definitions is enough for a SOAP client to know how to connect to a Web Service for an operation, but it does not describe operations. It is the equivalent of knowing the name of the computer to access and the folder in which to look for operations.

Security

You configure secure access to Web Services in the Administrator. If you enable secure access, a SOAP client must logon to Web Services with a valid Data Integrator Administrator login name and password (assign any level of privileges) before it can access the Web Services Data Integrator publishes.

If you do not enable secure access and you are using HTTP, SOAP clients have open access to:

- All batch jobs in repositories connected to the Administrator
- Configured real-time services

The long-term plan for Web Services is to use the Administrator to control which batch jobs and real-time services Data Integrator publishes as Web Services. Meanwhile, if you are using HTTP and SOAP, you can limit access to jobs by creating a repository that only includes the jobs and services what you want to publish.

Connection operations

Data Integrator generates WSDL that defines connection operations that belong to Web Services. There are three connection operations:

Ping	Verifies the connection to Data Integrator Web Services
Logon	Data Integrator uses this operation if you require secure access before establishing a session.
Logoff	Data Integrator uses this operation to terminate a session if you require secure access.

Ping operation

The Ping operation is an empty input message with a ping operation request. The output message is a text string that returns the current Data Integrator version, which indicates that a connection has been established.

Logon operation

The Logon operation is required when you enable Data Integrator to provide secure communication. To access Web Services, provide a Data Integrator Administrator login name and password (with any level of privileges). When Data Integrator validates them, the logon operation returns an Administrator session ID that you must include in all subsequent calls to Data Integrator Web Services.

Logout operation

The Logout operation is required when you enable Data Integrator to provide secure communication. When Web Services communication is complete, call the Logout operation to terminate the session.

Real-time service messages and operations

Data Integrator generates WSDL that defines how to invoke all real-time services for every Access Server known to the Administrator. Each real-time service name is represented as an operation name in the WSDL file.

Each real-time service operation has a set of messages that it uses to communicate with the real-time service. Real-time services use a defined XML message as input and a defined XML message as output. Real-time services obtain the format of these messages from the repository and provide the format in the WSDL.

Data Integrator supports both DTD and XML Schemas as message formats for real-time services. However, Web Services provides only XML Schemas in the WSDL. DTDs are converted to XML Schemas when necessary.

The repository stores XML Schemas that define the input and output messages as independent definitions. The WSDL file includes these definitions through an import statement. The SOAP client must obtain the imported message format as defined in the WSDL.

The messages that an XML Schema defines for each real-time service operation are:

- Input message

  When an external SOAP client invokes it, the input message passes information through Web Services to a Data Integrator real-time service. The name of the input message is the name of the operation that the Data Integrator publishes followed by the suffix `_Input`. The input message contains the following elements:

  - The message source defined by the real-time service
  - Secure session identifier (if you enabled security)

- Output message

  Data Integrator returns the output message when the real-time service completes. The output message contains the output generated by the real-time service. The name of the output message is the name of the operation followed by the suffix `_Output`. The output message contains the message target defined by the real-time service.

- Fault message

  Data Integrator returns a fault message if it cannot invoke the real-time service. It groups fault messages in the second section of the WSDL file, after the definition information. If there are no fault messages, you will still see the fault section heading in the WSDL file:

```
- <!--
    ********************************************************
    The fault message can be returned by any Data Integrator
    Web Service operation.  If an error occurs and the operation
    fails, this message type is returned instead of the
    output message configured for the operation
    ********************************************************
  -->
- <message name="Fault">
    <part name="body" element="localtypes:Fault" />
  </message>
- <!--
```

Fault messages are defined as Type elements in the WSDL file. Types are displayed near the end of the WSDL file after `soapAction` element. Fault messages consist of a code value and a reason, both configured as a string.

Fault message syntax is as follows:

```
<types>

   ...

   <xsd:element name='Fault'

   ...

   </xsd:element

   ...

</types>
```

Message formats

The following segment shows the syntax that Data Integrator generates in an WSDL file to define an operation's messages. In this example:

- `RTService` represents the name of the real-time service as defined in Data Integrator Administrator.
- `XMLSchemaName` represents the name of the XML Schema that was used to create an XML message source or XML message target in Data Integrator Designer.
- `RootElement` represents the root element of the XML Schema.

Data Integrator publishes WSDL as import statements that reference input and output XML Schema message formats. Each message calls its format using the real-time service name followed by the name of the XML Schema format name, followed by _Input (or _Output), followed by the name of the root element in the XML Schema (preceded by a colon)..

```
<!--The operation messages for real-time service 'RTService'-->

<import
namespace='http://businessobjects.com/service/RTService/input'
location='http://che:28080/admin/RTService_Input.xsd' />
<import
namespace='http://businessobjects.com/service/RTService/output'
location='http://che:28080/admin/RTService_Ouput.xsd' />

<message name='RTService_XMLSchemaName_Input'>
   <part name='sessionID' element='ltp:SessionID'/>
   <part name='inputBody'element='RTService_
    XMLSchemaName_Input:RootElement'/>
</message>
<message name='RTService_XMLSchemaName_Output'>
   <part name='outputBody'element='RTService_
    XMLSchemaName_Output:RootElement'/>
</message>
<message name='Fault'>
   <part name='body' type='ltp:Fault'/>
</message>
```

NOTE: Call-in support for real-time services requires that you use a valid URL for locating DTD (.dtd) and XML Schema (.xsd) files in an import statement. A local file name cannot be used.

The WSDL file displays the operations for real-time services within a `portType` tag. The following segment shows their syntax.

```
<!--The operations for Data Integrator real-time services -->

<portType name='RealTimeService'>
   <operation name='RTService'>
      <input message='tns:RTService_XMLSchemaName_Input'/>
      <output message='tns:RTService_XMLSchemaName_Output'/>
   </operation>
</portType>
```

Batch job messages and operations

Data Integrator publishes WSDL that defines how to start batch jobs. The WSDL file represents each batch job name as an operation.

In addition, the WSDL file defines an input and output message for each operation. An input message communicates the input needed by the job at startup (such as the global variables needed to start the job). An output message either confirms that the job started or presents the errors that prevent a job from starting.

WSDL defines the following messages for each operation:

- Input message

  The input message passes information needed by the batch job at startup. The name assigned to the input message is the name of the operation followed by the suffix `_Input`. The input message contains the following elements:

  - Global Variables
  - Secure session identifier (if you enable security)

- Output message

Data Integrator returns the output message when the batch job starts. The output message contains the job identification. The name of the output message is the name of the operation followed by the suffix `_Output`. The output message contains the following IDs:

- OS process ID of the started job
- Job Server Counter ID of the started job

- Fault message

  Data Integrator returns a fault message if the batch job fails to start. It returns a text description of the error that prevents the job from starting.

The following segment shows the syntax of the generated WSDL file used to define an operation and its associated messages for a batch job. In this example, `batchjob` represents the name of the batch job as defined in Data Integrator Designer.

```
<!--The operation messages for batch job 'batchjob'-->

<message name='batchjob_Input'>
   <part name='sessionID' element='ltp:SessionID'/>
   <part name='globalVarible'
    element='ltp:batchjob_GlobalVariable'/>
</message>
<message name='batchjob_Output'>
   <part name='pid' type='xsd:int'/>
   <part name='cid' type='xsd:int'/>
</message>
<message name='Fault'>
   <part name='body' type='ltp:Fault'/>
</message>
<operation name='batchjob'>
   <input message='tns:batchjob_Input'/>
   <output message='tns:batchjob_Output'/>
</operation>
```

The preceding WSDL segment defines the operation that provides access to a batch job along with the input/output messages associated with the operation. The messages include message parts, which provide the data values in the message.

In this example, the message part named `globalVariable` is a custom-defined data type. This is an example of how the WSDL file can include global variables.

Another important part of defining a batch job is the definition of the repository connection information. For batch jobs, the WSDL specifies repository connection information when it binds an operation to a port. Look for repository connection information in the service and ports section of the file. For more information, see "Finding code segments in the Data Integrator WSDL file" on page 136.

By defining connection information in the WSDL, it is much simpler for a SOAP client to start a batch job. The connection information only needs to be returned to the Web Service and not configured by the SOAP client.

SoapAction element

The definition of each batch job operation uses the `soapAction` element to define all information needed to actually launch the job:

- Batch job name
- Job Server host and port
- Repository connection information (user name, an encrypted password, database name, database type, and server name)

The WSDL file displays the `soapAction` element in the service and port section as shown in the following example:

```
- <binding name="Batch_Jobs" type="tns:BatchJob">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
  - <operation name="Job_Table_xml_source">
      <soap:operation soapAction="job=Job_Table_xml_source?jsHost=CHE?jsPort=3500?user=acta?
        pw=;1330246A9BB635292B421B854E83581DDE37A4748398D991D9D290BD7A33FCF4?db=cheryl?dbType=0?
        server=" style="document" />
    - <input>
        <soap:body use="literal" />
      </input>
    - <output>
        <soap:body use="literal" />
      </output>
    </operation>
  - <operation name="TestCall">
      <soap:operation soapAction="job=TestCall?jsHost=CHE?jsPort=3500?user=acta?
        pw=;1330246A9BB635292B421B854E83581D6AD5240A58C51987EEEA8A5CE2C5B88A?db=che?dbType=0?
        server=" style="document" />
    - <input>
        <soap:body use="literal" />
      </input>
    - <output>
        <soap:body use="literal" />
      </output>
    </operation>
```

The `soapAction` element should not require modification. However, if you want to add additional startup parameters for the repository (such as `enableRecovery`, `enableTestMode`, and `monitorSampleRate`), you can modify the `soapAction` parameters.

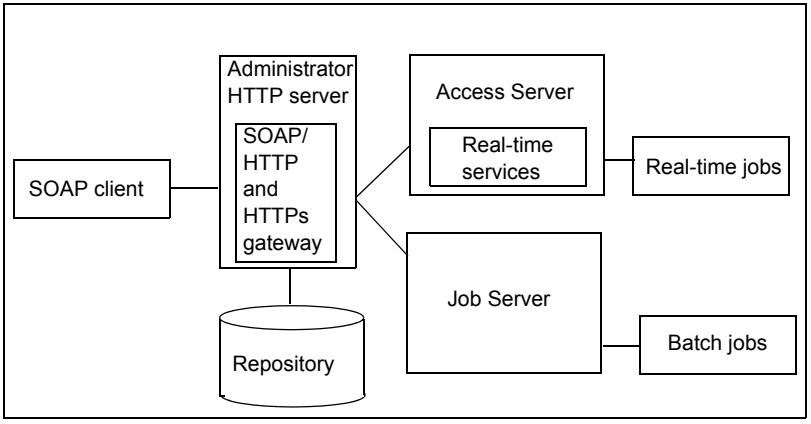
Generating a WSDL file

You can use the information in Data Integrator's WSDL file to build an application that can:

- Generate and get to the latest WSDL information programmatically using a reference URL
- Make function calls using the information in the WSDL to access Data Integrator batch jobs and real-time services

To view the WSDL file so that you can build your application you, can either use the Web Services page of the Data Integrator Administrator, or open a browser window and search for:
`http:\\<port>/admin/servlet/web services?wsdl`

Web Services interface to Data Integrator jobs



➢ To view Web Service configuration information using the Administrator

1. Open Data Integrator Administrator.
2. Make sure you have added connections to Access Servers and repositories to the Administrator.

   Data Integrator includes as Web Services:

   - Real-time services configured for all Access Servers known to the Administrator
   - Batch jobs in all repositories known to the Administrator
   - Connection Operations
     - Used to ping Web Services
     - Security operations (logon, logoff) that provide controlled access to Web Services.
3. Select **Management** > **Web Services**.

The Web Services page opens. The Web Services page contains a **View WSDL** section and a **Security** section. Security is enabled by default.

With security enabled, instead of making a single call to the Administrator to start a batch job or trigger a real-time service from an external application, you must write at least three calls:

- The first call logs in to the Administrator and gets a session ID.
- The second call accesses a job or service using the session ID as an input parameter. You must create a call for each job or service you want to access.
- The final call logs out of the session.

4. (Optional) Click **Enable Session Security** and **Apply** to disable security for the WSDL you are about to generate.
5. Click **View WSDL**.

   A new browser window opens with the WSDL displayed. Use the information in this file to:

   - Confirm that Data Integrator generates all jobs and services without error. The WSDL file displays error messages at the beginning of the file.
   - Create calls to Data Integrator

     Use the information in the browser window to configure your application to access Data Integrator batch jobs and real-time services.

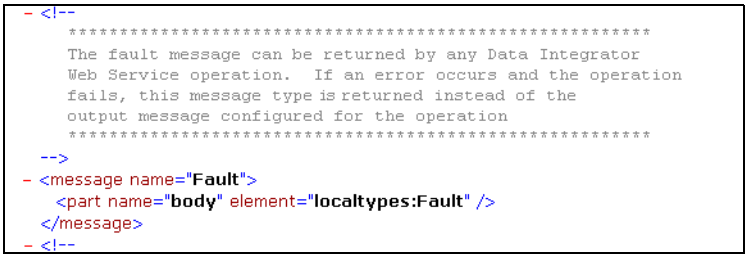
     When your application calls a job, Data Integrator performs the actions requested. To ensure that your application calls the latest version of the job, always generate WSDL in your request. Data Integrator generates WSDL when your application requests it.

Finding code segments in the Data Integrator WSDL file

The WSDL file that Data Integrator generates is organized into sections in the order shown in the following list:

1. Description
2. Fault message
3. `Connection_Operation` messages followed by operations
4. `Real-time_Service` operation messages followed by operations
5. `Batch_Job` operation messages followed by operations
6. Service, port, and binding (`soapAction`) information

Sections and sub-sections are labeled to help you find the information you need. An example of a WSDL file section label appears below:



You might want to use the following table as a quick reference to find the code you need in Data Integrator's WSDL file.

WSDL elements mapped to sections of Data Integrator's WSDL file

Element Name	Location in the generated WSDL file
definition	First section (beginning of file)
service	Sixth section, after: • Real-time services messages and operations • Batch job messages and operations.
port	Same location as the service
portType	PortTypes are listed in the operations sections. Bindings are listed in the service and ports section in the following order: • `Connection_Operations` • `Real-time_Services` • `Batch_Jobs`
operation	Data Integrator publishes connection operations in the third section of the WSDL file. If you register real-time services, their operations are in the fourth section. Batch job operations are in the section after real-time services.
message	Same as operations
types	Last section, after service and ports

Error reporting

Data Integrator uses Web Services to define. every operation with both an input and output message. In addition to the output message, Data Integrator returns a fault message if an error occurs.

In addition to the fault message, Data Integrator writes log and debug messages to the Administrator's log file. Fault messages include a short description of a failure. For detailed information about an error, use the Administrator's log file.

All Data Integrator Administrator components share the Administrator's log. Data Integrator prefixes these messages with the name of the component issuing the error message. For Web Services, the component name is the name of the java class issuing the error. All Web Service classes start with `com.acta.adapter.webservice`....

Data Integrator creates the Administrator's log file in: `<LINK_DIR>\log\webadmin.log.`

To control the level of detail in the `webadmin.log` file, you must edit the `log4j.properties` file. The properties file is located in:

`<LINK_DIR>\ext\webserver\webapps\acta_web_admin\WEB-INF`

To obtain a debug trace of events, change the log level from the default of `INFO` to `DEBUG` for example: `log4j.rootLogger=DEBUG, A`

Tips for using the generated WSDL file

The WSDL file:

- Appears in the View WSDL window or any browser window by searching for: `http:\\<port>/admin/servlet/webservices?wsdl`
- Displays all real-time services and jobs known to the Administrator. There is no mechanism to find a single job or to expose a sub-set of jobs except:
  - Connecting a repository to the Administrator that only contains the batch jobs you want exposed
  - Configuring only the real-time services you want exposed
- Always displays log on, log off, and session ID information. Deselecting the **Enable Session Security** option does not hide this content in the View WSDL window.

- Does not handle duplicate batch job or real time service names. If you use the same name in different repositories, there will be duplicate names in the WSDL.
- Does not display XML Schema formats for real-time services. The formats are defined in imported XML Schema files and available to the WSDL through an import statement (`<import> </import>`). This statement is located above each set of message definitions for a real-time service in the WSDL file.

  Data Integrator defines messages in the WSDL file by the real-time service name they are used in, followed by their XML Schema format name, followed by either `_Input` or `_Output` (depending on the type of message using the format) and the XML Schema root element name.

  For example:

  ```
  RTJob_OrderStatus_PurchaseOrder_Input:Order
  RTJob_OrderStatus_PurchaseOrder_Output:Order
  ```

  Bindings for each batch job contain a set of literal parameters defined as a `soapAction` element. Use these parameters to create a call to the Administrator to launch a batch job from your external application. If you are using security, use the log on and log off operations in the WSDL for your first and last calls.

Call-out functionality

You can add functionality to Data Integrator to invoke Web Services in external applications from Data Integrator data flows. This functionality requires configuring Data Integrator's built-in Web Services adapter. The Web Services adapter provides support for locating and importing metadata for Web Services as well as invoking Web Services.

The Web Services adapter works by sending a request and waiting until it receives a reply from an information resource (typically a database or an application).

For example, you might create a Web Service as a front-end to a legacy application. You could call the Web Service daily from a Data Integrator data flow to access inventory and update an inventory data mart.

The interaction between the Web Services adapter and an external Web Service has four parts:

- Downloading the WSDL file that describes the Web Service.
- Processing the WSDL file to extract the information needed to access the Web Service.
- Creating a SOAP request to access the Web Service.
- Processing the SOAP response to extract the response string from the SOAP envelope.

Adapter installation

The Web Services adapter is part of each Job Server installation. The installer automatically configures an adapter instance in the Administrator, which is the only adapter instance that is required for Web Services configuration. You do not need to configure adapter operations. The Web Services adapter is configured with

Autostart set to `FALSE` so that it does not consume resources if you do not use Web Services. However, you can invoke adapters set with **Autostart** disabled when needed. You do not need to edit the adapter instance that Data Integrator provides for the Web Services adapter.

The installer allows you to configure a Job Server to manage adapters by presenting a list of Job Servers to you during installation. To view any adapter instance in the Administrator, select **Adapter Instance** > ***Job Server***.

The following table shows the values Data Integrator automatically creates for the Web Services adapter instance.

Field name	Value automatically inserted
Adapter Instance Name	WebService
Access Sever host	(not required) Blank
Access Server port	(not required) Blank)

ClassPath	Jar files required in the classpath to use when starting the java process:
	`<LINK_DIR>/lib/acta_adapter_sdk.jar`
	`<LINK_DIR>/lib/acta_broker_client.jar`
	`<LINK_DIR>/lib/acta_tool.jar`
	`<LINK_DIR>/ext/lib/xerces.jar`
	`<LINK_DIR>/lib/acta_webservice_adapter.jar`

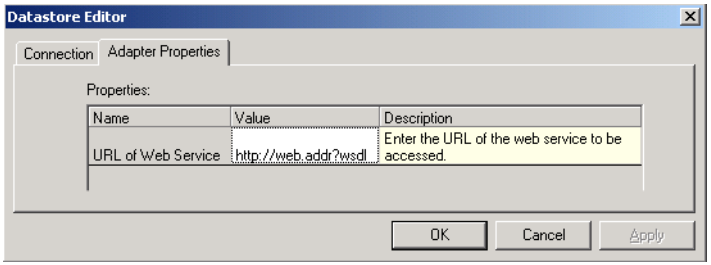
AutoStart	FALSE
Trace Mode	TRUE (The adapter writes trace messages to the `WebService.txt` file in the `LINK_DIR/adapters/logs` directory)
Application command line parameters	(not required) Blank
Adapter type name	(Read-only) The name of the adapter used to create this instance
Adapter version	(Read-only) The version of the adapter used to create this instance
Adapter class	(Read-only) The name that identifies the adapter entry point

Adapter configuration

To configure access to a specific Web Service, use the Designer. In the Designer's Datastore Editor window, select an adapter instance name. Data Integrator provides access to Web Services as stream-oriented function calls that it configures when you import metadata.

When you configure an adapter datastore, in addition to the normal adapter settings, described in "Defining an adapter datastore" on page 107 of the *Data Integrator Designer Guide*, specify the URL of the Web Service provider. It must be the same URL that accepts a Web Service connection and returns the WSDL.

The adapter will connect to the Service Provider at that URL to locate the definition of published services.



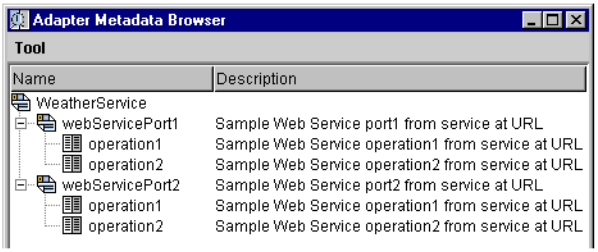
➢ To access a Web Service, using the Designer

1. Create an adapter datastore:
   a. Use the adapter instance that Data Integrator automatically creates during installation.
   b. Use the **Adapter Properties** tab of the Datastore Editor window to enter the URL of the Web Service.
   c. Click **OK**.
2. Import metadata:

a. From the object library, double-click a Web Service adapter datastore.

The Designer calls the adapter. The adapter calls the Web Service at the indicated URL and obtains a list services and ports published by the Web Service.

b. Expanded the ports to see the published operations available for import.



The list reflects the name and description of operations currently published by the configured Web Service.

c. Right-click an operation and select **Import**.

Data Integrator imports Web Service operations as function calls and lists them under the Web Services adapter datastore in the object library. Each function call includes a definition for both the input and output messages required for communication with a Web Service operation. The adapter extracts the details about the request and reply messages and generates XML Schema files that describe the messages.

3. From the Designer, drag the imported function calls into a data flow and use standard Data Integrator features to supply input to the function call and extract the response information obtained from the Web Service.

In call-out mode, Data Integrator calls a Web Service twice:

- During design time to import metadata for the functions and data types that a particular Web Service supports

- During runtime to call the Web Service and invoke this functionality

10 Troubleshooting

The Data Integrator Administrator provides status and error information. Use this information to discover problems with your implementation and to find the source of those problems. This chapter describes how to reinstall the Web Server service and how you can use the Administrator to find and help resolve job processing issues.

Specific sections include:

- Reinstalling the Web Server service
- Finding problems
- Error and trace logs
- Resolving connectivity problems
- Restarting the Access Server

Reinstalling the Web Server service

The error "`Error: 500 Location: /jsp/signin.jsp`" sometimes appears when you try to start the Administrator even though the Data Integrator Web Server service is running. This error is caused by an incomplete Web Server installation.

➢ To reinstall the Web Server service

1. Stop the Data Integrator Web Server service.
2. Delete the following directories in the Data Integrator installation location:
   - `/ext/webserver/webapps/acta_web_admin`
   - `/ext/webserver/webapps/acta_metadata_reports`
3. Restart the service.

   The service will re-expand the jar files and reinstall itself.

Finding problems

The Data Integrator Administrator uses colored indicators to show the status of the various system components. Generally, the indicators mean the following:

Indicator	Description
	A green indicator signifies that the object is running properly.
	A yellow indicator signifies that some aspect of this object is not working. Either the Access Server is in the process of its error-handling efforts to reestablish an operation, or the Access Server is waiting for a manual intervention. For example, when you first add a service to the Access Server configuration, the service shows a yellow indicator until you manually start the service or until you restart the Access Server.
	A red indicator signifies that one or more aspects of this object is not working, and the error handling efforts of Access Server were not able to reestablish the operation of the object.

When you see a yellow or red indicator, the system requires manual intervention. You must:

- Determine which object is not operating properly
- Determine the cause of the problem
- Fix the problem
- Restart the affected service providers if necessary

➢ To determine which object is not operating properly

1. In the Data Integrator Administrator, click **Home**.

   If there is an error anywhere in the system, you will see a red indicator next to a repository or Access Server name.

2. If you see a red indicator, click a repository or Access Server name.

   The page for that repository or Access Server appears.

3. Look for another red indicator on objects listed at this level.
4. Repeat steps 2 and 3 if you can identify lower-level objects that have a red indicator.

   When you have identified the lowest level object with the error, you are ready to determine the cause of the error.

➢ To determine the cause of the error

1. Examine the error log for the affected subsystem, such as a batch job, a service provider, an adapter interface, or for the Access Server itself.

   Use the timestamp on the error entries to determine which error accounts for the problem that you are experiencing.
2. Cross-reference the error to the trace information.

   When you identify the appropriate error message, you can use the timestamp to determine what other events occurred immediately before the error.

   For example, if an error occurred for a specific service provider, you can use the error timestamp in the service provider error log to look up Access Server activities that preceded the error.

Error and trace logs

The Administrator provides access to trace and error log files for each service provider, each batch job that ran, and for the Access Server. Use these detailed log files to evaluate and determine the cause of errors.

This section discusses:

- Batch job logs
- Service provider logs
- Access Server logs
- Adapter logs

Batch job logs

The Batch Jobs Status page provides access to trace, monitor, and error log files for each batch job that ran during a specified period. For information about setting that period, see "Setting the status interval" on page 24.

For information about monitor logs, deleting logs, and the **Ignore Error Status** button see, "Monitoring jobs" on page 53.

Batch job trace logs

The trace log file lists the executed steps and the time execution began. Use the trace log to determine where an execution failed, whether the execution steps occurred in the order you expect, and which parts of the execution were the most time-consuming.

➢ To view a trace log

1. Select **Batch Jobs** > ***repository.***
2. Under **Batch Job History**, find the instance of the job execution in which you are interested.

   Identify an instance by the job name, start time, etc.

3. Under **Job Information** for that instance, click **Trace**.

   The Administrator opens the Job Server Trace Log Viewer.

Batch job error logs

The error log file shows the name of the object being executed when an Data Integrator error occurred and the text of the resulting error message. If the job ran against SAP R/3 data, the error log might include some of the ABAP errors, also.

Use the error log to determine how an execution failed. If the execution completed without error, the error log is blank.

➢ To view an error log

1. Select **Batch Jobs** > ***repository***.
2. Under **Batch Job History**, find the instance of the job execution in which you are interested.

   Identify an instance by the job name, start time, etc.
3. Under **Job Information** for that instance, click **Error**.

   The Administrator opens the Job Server Error Log Viewer.

Service provider logs

The Service Provider Status page provides access to the error and trace log files for a service provider. These are the log files produced by the Job Server that controls the service provider.

➢ To view the logs for a service provider

1. Select **Real-Time** > ***Access Server*** > **Real-time Services**.
2. Click the name of a service.

   The Administrator opens the Real-time Service Status page. This page shows a list of service providers for the service and overall statistics for the service.

3. Click the name of the service provider process ID in which you are interested.

   The Administrator opens the Service Provider Status page.

4. Click a link to view the desired service provider log.

   To delete these logs, set the log retention period. See "Setting the log retention period" on page 25. To filter the list of real-time services on the Real-time Service Status page by date, set the status interval. See "Setting the status interval" on page 24.

Links to service provider logs

Link	Description
Trace Log	Opens the Trace Log page for the current service provider execution. This link appears only if the real-time service is registered with the Access Server.
Error Log	Opens the Error Log page for the current service provider execution. This page lists errors generated by Data Integrator, by the source or target DBMS, or the operating system for job execution. If the error log is empty, the job has not encountered errors in message processing. This link appears only if the real-time service is registered with the Access Server.

The computer running the Job Server stores text files containing the batch and service provider trace, error and monitor logs. If you installed Data Integrator in the default installation location, these files are located in the following directory:

```
LINK_DIR/Logs/JobServerName/RepoName
```

The name of the log file describes the contents of the file:

```
type_timestamp_sequence_jobname.txt
```

where

type	Trace, monitor, or error.
timestamp	System date and time from when the job created the log.
sequence	Number of this job related to all jobs run by this Job Server instance.
jobname	The name of the job instance.

Batch job trace and error logs are also available on the **Log** tab of the Designer project area. To see the logs for jobs run on a particular Job Server, log in to the repository associated with the Job Server when you start Data Integrator Designer.

Access Server logs

Trace and error logs for each Access Server are available under **Real-Time** > ***Access Server*** > **Logs-Current** and **Real-Time** > ***Access Server*** > **Logs-History**. In addition, these files are located in the Access Server configuration location, which you specify when you configure the Access Server.

NOTE: For remote troubleshooting, you can also connect to any Access Server through the Administrator. See "Adding Access Servers" on page 21.

➢ To view the current day's logs

1. Select **Real-time**> ***Access Server*** > **Logs-Current**.
2. This page lists the error log file followed by the trace log file.

   The date of the file is included in the name:

   - error_MM_DD_YYYY.log

- trace_MM_DD_YYYY.log

3. To view a file, click the name.

   The Administrator shows the last 100,000 bytes of the Access Server error log or trace log for the current date.

   The error log contains error information that the Access Server generates.

   The trace log contains a variety of system information. You can control the information the Access Server writes to the trace log. See "To configure the trace log file" on page 156 for more information.

➢ To view the previous day's logs

1. Select **Real-Time** > ***Access Server*** > **Logs-History**.
2. This page lists error log files followed by trace log files. The date of the file is included in the name:
   - error_MM_DD_YYYY.log
   - trace_MM_DD_YYYY.log
3. To view a file, click the name.

➢ To configure the trace log file

1. Select **Real-Time** > ***Access Server*** > **Logs-Current**.
2. Click the **Configuration** tab.
3. Under **Log Contents**, the Administrator lists several trace parameters that control the information the Access Server writes to the trace file.

Name	Description
Admin	Writes a message when an Access Server connection to the Administrator changes.
Flow	Writes a message when an Access Server exchanges information with a real-time service
Request	Writes a message when an Access Server receives requests
Security	Writes a message when an Access Server processes authentication information (IP addresses, user name, or password)
Service	Writes a message when an Access Server starts or stops a service
System	Writes a message when an Access Server initializes, activates, or terminates

4. Select the check box next to the name if you want the Access Server to write corresponding messages to the trace log file.
5. Under **Log Tracing**, select the **Enabled** check box.
6. Click **Apply**.

The Administrator will change the Access Server configuration. The Access Server will now write the selected trace messages to the trace log.

NOTE: Until you set the parameters on this page, the Access Server uses the startup parameters to determine trace options. For information about the startup parameters, see "Restarting the Access Server" on page 162. Each time you restart the Access Server, the startup parameters take precedence over parameters set on this page. You can control the content of this log by setting parameters when configuring the Access Server. See "The following parameters are available to control the operation and output of an Access Server:" on page 88.

➢ To delete Access Server logs

1. Select **Real-Time** > ***Access Server*** > **Logs-Current** or **Real-Time** > ***Access Server*** > **Logs-History**.
2. Select the check box next to any log file that you want to delete.

   Alternatively, to delete all the log files, select the **Select all** check box.
3. Click **Clear** or **Delete**.

   The Administrator clears the file size for current logs and deletes the selected history files from the display and from the Access Server log directory.

Adapter logs

For more detailed information about an adapter or an adapter's operations, see the adapter's error and trace log files.

➢ To view log files for an adapter instance

1. Select **Adapter Instance** > ***Job Server***.

2. Find the adapter instance for which you want to view logs and from the **Log Files** column, click the **Error Log** or **Trace Log** link.
3. The corresponding page opens.

   These log files are also found in the `LINK_DIR\adapters\log` directory. The error log file is named *`adapter_instance_name`*`_error.txt` and the trace log file is named *`adapter_instance_name`*`_trace.txt`.

Resolving connectivity problems

If you have determined that you have connectivity problems among your real-time system components, consider the following possible failures:

- Cannot log into the Administrator

  This occurs only if the local system account, that the Administrator's installer uses for the Web Server, does not have privileges to launch executables.

  Go into the Services panel and change the account used to start the Data Integrator Web Server service. Set properties to a user account, instead of a local system account, and add the privilege to log in as a service. Then stop and restart the Data Integrator Web Server service.

- Application client cannot connect to Access Server

  For example, an error appears in the logs generated by your application client or in the command prompt when executing the client test utility that looks like this:

  ```
  Error: unable to get host address
  ```

  If you specified an IP address and received this error, your network might not support static IP address resolution. Try using the computer name instead.

  Match the port number you specified in the Client Test utility (or in the Message Client library call) to the Access Server's port number.

  Make sure that the port you specified is not in use by other applications on the computer where an Access Server is installed.

- Access Server cannot connect to Job Server

  If this error occurs, you would see a red indicator for a service provider and an error log for the Access Server.

Match the host name and port number of the Job Server for for the service being called (configured in the Administrator) to the host name and port number that the Job Server is configured to use (as listed in the Server Manager).

Make sure that the Job Server is running by checking the Windows NT Task Manager for the Al_jobserver.ex and Al_jobservice.e processes or by opening the Designer, logging in to the repository that corresponds to the Job Server, and looking for the Job Server icon at the bottom of the window.

- Job Server cannot start real-time service

  If this error occurs, the status for the related service and its service provider would be red and you would be able to open the error log file from the Service Provider Status page.

  Make sure that the job is properly indicated for the service called in the Real-Time Service Status page.

  Make sure that the real-time jobs are available in the repository associated with the Job Server.

  Make sure the repository and the Access Server are connected to the Administrator and that the repository is available.

  If you change the password for your repository database, the job server will not be able to start real-time services. To fix this problem, re-register your repository in the Administrator and reconfigure the real-time services.

- Real-time service cannot register with Access Server

  If this error occurs, you would see:

  - A red indicator for the service provider
  - An error log in the Logs-Current page (the startup timeout will eventually be triggered)
  - An error log available from the Service Provider Status page.

  Make sure the Access Server host name correctly identifies the computer where the Access Server is running.

- Access Server cannot connect back to application client

  If this error occurs, you would see an error log under the Access Server's Logs-Current node.

  Make sure that the host name and port used by the message broker client to communicate with the Access Server is correct. For more information, see "Verifying connectivity" on page 105 of the *Data Integrator Getting Started Guide*.

Restarting the Access Server

To restart the Access Server, you can use either of two choices:

- Controlled Restart

  The Access Server responds to new and queued messages with a "shutdown" error. It waits for service providers to complete processing existing messages, then returns the responses to those clients. Next, the Access Server closes existing client connections (including adapters), stops, and restarts itself. Finally, the Access Server reads the current configuration settings and restarts services, service providers, and adapters.

  Restarting the Access Server this way requires as much time as it takes to processes requests in the system.

- Abort and Restart

  The Access Server responds to new and queued messages with a "shutdown" error. It shuts down existing service providers and responds to these messages with a "shutdown" error. Next, the Access Server closes existing client connections (including adapters), stops, and restarts itself. Finally, the Access Server reads the current configuration settings and restarts services, service providers, and adapters.

➢ To perform a controlled restart of the Access Server

1. Select **Real-Time** > **"Access Server"** > **Status**.
2. Under **Life Cycle Management**, click **Controlled Restart**.
3. Click **Real-Time Services** to verify that all services started properly.

The Access Server allows running services to complete and returns incoming and queued messages to the client with a message that the Access Server has shut down. When all services have stopped, the Access Server stops and restarts itself. The Access Server reads the new configuration settings and starts services as indicated.

If all service providers started properly, the Real-Time Service Status page shows a green indicator next to each service name. A red indicator signifies that a component of the service did not start.

➢ To perform an abort and restart of the Access Server

1. Select **Real-Time** > **"Access Server"** > **Status**.
2. Under **Life Cycle Management**, click **Abort and Restart**.
3. Click **Real-Time Services** to verify that all services started properly.

A Glossary

This glossary defines terms used in this documentation set including common terms that have a special meaning with respect to Data Integrator. Within a definition, other terms that are defined in the glossary appear in **bold type**.

ABAP

Advanced Business Application Programming.
A fourth-generation programming language developed by SAP in which R/3 applications are written.

ABAP program

A program that executes database operations on an **SAP R/3** server. Data Integrator **R/3 data flows** generate ABAP programs.

Access Server

A process that starts, stops, and monitors real-time services (**real-time jobs**). The Access Server dispatches requests to real-time services, ensuring optimal load balancing and complete life cycle management. The Access Server also starts, stops, and monitors **adapters**.

adapter

An external Data Integrator **interface**. There are two types of adapters:

- Custom adapters — Adapters developed using the Data Integrator Adapter Development Kit
- Prepackaged adapters — Adapters prebuilt and purchased from Business Objects, such as the Data Integrator MQSeries Adapter

Administrator

A browser-based system administration application that supports Data Integrator. Use the Administrator to configure and manage:

- Services (**real-time jobs**)
- Service providers (instances of real-time services)
- Clients (Web application clients)
- Batch jobs
- Adapters
- Job logs

after-image

The values in an UPDATE row after the row changes. You might want to pair before- and after-images of UPDATE rows when designing log-based changed-data capture jobs which Data Integrator supports.

aggregate data

Data that results when a process combines elements. This data can be presented collectively or in summary form.

ALE

Application Link Enabling. An SAP R/3 programming-related interface designed to allow reliable communication across a distributed environment.

Analytic Rapid Mart

An **Data Integrator application** that produces a predefined **data warehouse** for a specific area of business analysis. Analytic Rapid Marts are typically based on dimensional modelling. With Analytic Rapid Marts, users can manage large volumes of data.

attribute

A **property** created for a type of **object**.

BAPI

Business Application Programming Interface. A standardized SAP R/3 programming interface that allows non-SAP R/3 applications to access specific business processes and data.

back-office application

A software system that an enterprise uses to run business operations, such as human resources, general ledger, and finance. Back-office applications store data in a relational **database** optimized for operational use.

There are several types of back-office applications:

- Enterprise resource planning (ERP) — For example, SAP R/3, BAAN, J.D. Edwards, PeopleSoft
- Supply chain management (SCM) — For example, i2
- Customer relationship management (CRM) — For example, Siebel, Clarify
- Legacy

Basis

The SAP infrastructure. Basis is the foundation for all SAP products.

batch job

A group of objects (data flows, work flows, conditionals, scripts, and so forth) that executes on a schedule. This type of job is recommended for operations on large amounts of data.

before-image

The values in an UPDATE row before the row changes. You might want to pair before- and after-images of UPDATE rows when designing log-based changed-data capture jobs which Data Integrator supports.

bulk loading

A software-based mechanism that moves large data files. This mechanism supports compression, blocking, and buffering to optimize transfer times.

business component

A set of tables Siebel applications use to create a logical object called a business object.

Business Objects Data Platform

A set of applications from Business Objects, Americas, that enterprises can use to manage all types of data—from data for supply chain partners to data for customers. The Business Objects Data Platform consists of **Data Integrator** and one or more **Rapid Marts**.

CDC datastore

A CDC datastore allows you to limit extracted data to changed data only. A CDC datastore connects a **changed-data capture** table on a source database to Data Integrator.

CDC checkpoint

A CDC checkpoint enables Data Integrator to restrict CDC subscription reads. After you enable a checkpoint, the next time the CDC job runs, it reads only the rows inserted into the CDC table since the last checkpoint.

CDC Subscription

A CDC subscription name allows different data flows to extract data from the same table without corrupting data extracted by other data flows. The datastore and table name can be the same; however, the table owner name must be different in each source CDC table.

changed-data capture

The process of retrieving changes made to a production data source. This process consolidates units of work, ensures data is synchronized with the original source, and reduces data volume in a warehouse environment.

Citrix MetaFrame XP

Citrix MetaFrame XP software provides an access infrastructure for enterprise applications. You can use this software to run Data Integrator on a server which publishes instances of the Designer and other Data Integrator interfaces to users on client computers.

client/server

A distributed technology approach where the processing is divided by function. The server performs shared functions (such as managing communications and providing database services), while the client performs individual user functions.

COM

Component Object Model. Provides a set of interfaces that allow clients and servers to communicate within the same computer (running on Windows 95 or later versions).

Commerce Rapid Mart

An **Data Integrator application** that produces a predefined **data warehouse** and that includes logic to support real-time requests for data from a **back-office application**. A Commerce Rapid Mart is usually based on different data model than an **Analytic Rapid Mart**. With a Commerce Rapid Mart, users can integrate a company's back-office system with an **e-commerce** application.

Communication Structure

In **SAP BW**, a data structure that defines a set of **InfoObject**s available from an **InfoSource** to put into **InfoCube**s.

conditional

A **single-use object**, available in **work flow**s**,** that allows you to branch the execution logic based on the results of an expression. The conditional takes the form of an if/then/else statement.

custom ABAP program

A custom ABAP program runs an **ABAP program** and generates a **data set**. With a custom ABAP program, you can run an existing ABAP program as part of a job. Use a custom ABAP program as a **source** in a **data flow** or an **R/3 data flow**.

data extraction

The process of moving data from a **database** or application source to a database target (either from a legacy database to a **data mart**, or from one data mart to another).

data flow

A **reusable object** containing **steps** to define the transformation of data from **source** to **target**. Data flows are called from inside a **work flow** or **job**. You can pass information into or out of data flows using **parameters**.

Data Integrator

A software system that allows users to build and execute applications with which they can create and maintain **data warehouses**.

Data Integrator consists of several components:

- **Access Server**
- **Data Integrator engine**
- **Data Integrator repository**
- **Data Integrator Service**
- **Administrator**
- **Designer**
- **Job Server**

Data Integrator application

A set of **jobs** that process batch data extractions and real-time requests using data from a variety of data sources.

Data Integrator engine

The core process that reads job information from the **Data Integrator repository** and sets up run-time processes that execute the job. The run-time processes extract, transform, and load relational and hierarchical data. The **Job Server** starts the Data Integrator engine to execute batch or real-time jobs.

Data Integrator interface

A program that Data Integrator uses to access data sources. Specific interfaces vary by installation. There are internal interfaces—those native to the installation—and external interfaces—those that you install separately. Internal interfaces allow Data Integrator to access applications like **SAP R/3** and **SAP BW**, **messages**, relational database systems, and **legacy systems**. An external interface is also known as an **adapter**. It allows Data Integrator to access applications using information exchange technologies such as Web Services and MQ Series.

Data Integrator repository

The database that contains information about an Data Integrator application. The repository contains information about defined **reusable objects**, the **metadata** for tables, and built-in **transforms** and **functions**. When you invoke Data Integrator, you log in to the repository containing the **objects** you want to use.

Data Integrator Service

The process that ensures that the **Access Server** and the **Job Server** are running. You can configure the Data Integrator Service to restart the Access Server and Job Server whenever the computer where they are located restarts.

data loading

The process of populating a **data warehouse**. Data loading is provided by **DBMS**-specific load processes, DBMS insert processes, and independent fast-load processes.

data mapping

The process of assigning a source data element to a target data element.

data mart

A highly-focused version of a **data warehouse**. Typically, created by a department or division of a company, data marts contain data for a specific subject area, such as finance or sales. Data Integrator can populate a data mart.

data transformation

The process of filtering, merging, decoding, and translating source data to create validated data for a **data warehouse**.

data set

Rows of data with a defined schema. A step in a **data flow**—such as reading data from a **source**, joining data in a **Query transform**, or transforming data though another **transform**—yields a data set. You can view individual data sets by placing a **target** table or file at that point in the **data flow**.

data transport

A step in an **R/3 data flow** that defines a **target** to store the **data set** extracted during the flow. You can locate the target file on the **SAP R/3** server or in a location accessible to both the SAP R/3 server and to Data Integrator across a network.

data type

The format used to store a value. Data types can imply a default format for displaying and entering the value. Data read from a **source** is converted to the appropriate Data Integrator data types; data loaded to a **target** is converted from its Data Integrator data type to the type appropriate for the target.

data warehouse

A relational **database** containing subject-oriented information that supports decision analysis. Typically, information in a data warehouse is integrated from multiple sources, including operational sources. Data warehouses provide a single, non-volatile point of reference for a particular time period. Data warehouses are composed of **data marts.** Data Integrator can populate a data warehouse.

database

A collection of **table**s managed by a **DBMS** such as Microsoft SQL Server or Oracle.

DataConnector

DataConnector operator instances are used to read data files generated by Data Integrator when performing bulk loading using the Teradata Warehouse Builder.

datastore

A logical channel connecting Data Integrator to a **database**. The datastore definition includes the name and location of the database as well as user authentication information. Data Integrator uses a datastore definition to qualify a table name wherever a table is indicated in a diagram or expression. You can access the datastore definition through the **object library**.

DBMS

Database management system. A software system that builds and maintains database tables.

debug mode

Allows you to debug a job using the **interactive debugger** features in the Designer.

declarative specification

A way of indicating the desired results of a **data transformation** without placing constraints on the method of reaching the results. When you create a **data flow** in Data Integrator, you specify what data you want to read from which **source**, what you want done to that data, and where you want it loaded. What you specify constitutes the declarative specification. Data Integrator then determines the most efficient way to reach the requirements you set, including pushing operations to the source **DBMS** for execution.

degree of parallelism

You can set a value at the data flow level that allows you to parallel-process a **transform** within a data flow.

For example, if from the Properties window of a data flow you set the **Degree of parallelism** to 5, then when the job executes, Data Integrator replicates each transform in the data flow five times. Each of these replicated transforms executes in parallel using a separate process thread. Using a value for degree of parallelism can improve job performance if a **Job Server**'s computer memory and number of CPUs supports a job's parallel-processing configuration settings.

dependency analysis

A category of metadata reports that produces reports based on search criteria you specify to find specific objects in the repository. For each object returned, dependency analysis reports provide object lineage as well as object impact information.

derived data

Data that results from a computational step applied to reference or event data. Derived data is the result of either relating two or more elements of a single transaction (such as aggregation), or relating one or more elements of a transaction to an external algorithm or rule.

Designer

A graphical user interface that allows you to design and test an **Data Integrator application**.

diagram

The icons and connections between the icons that make up the definition of a **job**, **work flow**, or **data flow**.

DMZ

Demilitarized zone. A perimeter network that is more accessible than your most protected network, but located inside a security layer protected from the internet.

drill down

A method of exploring detailed data that was used in creating a summary level of data. Drill down levels depend on the granularity of the data in the **data warehouse**.

DTD

Document type definition. A text file that describes the elements (tags) in an **XML** document and the relationship among them. When an XML document is used to describe a transaction, the DTD describes the data schema used in the transaction.

e-commerce

Buy and sell transactions conducted over the internet.

EDI

Electronic Data Interchange. Electronic exchange of structured data between businesses. This exchange is not dependent on hardware, software, or communication protocols.

element

A component found within XML Schemas and DTDs.

embedded data flow

A data flow created within one or more data flows.

enterprise data

Data that is defined for use across a corporate environment.

ERP system

Enterprise resource planning system. A **back-office application** from which Data Integrator can extract data.

exception

An error that occurs while executing a **job**. You can catch individual or groups of exceptions using a **try/catch block** inside a **work flow**. Catching an exception allows you to automatically execute a solution for the error.

Executive Information Systems (EIS)

Tools programmed to provide canned reports or briefing books to executives. They offer strong reporting and drill-down capabilities. These tools allow ad hoc querying against a multi-dimensional **database**, and most offer analytical applications along functional lines such as sales or financial analysis.

extract date

The date data was extracted.

extract frequency

The interval at which data is extracted, such as daily, weekly, monthly, or quarterly. The frequency that data extracts are needed in the **data warehouse** is determined by the shortest frequency requested through an order, or by the frequency required to maintain consistency of the other associated data types in the source data warehouse.

file

A **source** or **target** for data. To read or load data to a file within Data Integrator, define a **file format** that instructs Data Integrator about the structure of the data in the file.

file format

A description of how data is or should be organized in a **file** Data Integrator reads from or loads to. A file format can be specific to a single file or generic for many files.

front-office application

Software that companies use to support reporting, **e-commerce**, and other tasks. Types of front-office applications include:

- Supply chain collaboration
- Analytical
- Sell-side e-commerce
- Customer relationship management

function

A program that operates on values that are passed to it. Data Integrator functions are available through a function wizard in a **script**, **conditional**, or **Query transform**. Data Integrator also gives you access to functions provided by the **DBMS** you are using. In addition, you can define your own functions using the Data Integrator scripting language or by writing C-language programs.

IDoc

Intermediate Document. An SAP-specific format. Used for **EDI** and **ALE**.

IDoc type

Indicates the SAP format that is used to interpret the data of a business transaction. Consists of the following components:

- A control record

  Identical for each IDoc type.

- Several data records

  A single data record consists of a fixed key part and a variable data part. The data part is interpreted using segments, which differ depending on the IDoc type selected.

- Several status records

  Identical for each IDoc type. Describe the status states an IDoc has already passed through or the status an IDoc has attained.

import

The process of acquiring information for the **Data Integrator repository**. Import the following kinds of information into Data Integrator:

- The **metadata** for **source** and **target** databases
- Descriptions and code for user-defined and DBMS **functions** and **transforms**

InfoArea

In **SAP BW**, an arbitrary grouping of **InfoCube**s.

InfoCube

In **SAP BW**, a multidimensional star schema that is limited to one fact table and 16 dimensions.

InfoCubes store only numeric data at the fact level.

SAP BW reserves three of the dimensions for currencies, time, and units of measure. Dimensions have to be linked back to master data and hierarchy tables to gain access to the information stored there.

When a user query is initiated, BW creates a temporary **OLAP** cube that stores a constrained subset of the permanent OLAP cube. The temporary InfoCube persists until the user disconnects from BW or changes the query.

InfoObject

In **SAP BW**, an InfoObject is equivalent to a data element and its associated metadata (for example, description or business rules). Can be a **Key Figure** or a Characteristic.

InfoPackage

In **SAP BW**, a schedule for running BW updates.

Information resource (IR)

A producer or consumer of enterprise information. An information resource can be a flat file, a **back-office application**, an EAI system, or a Web application.

InfoSource

In **SAP BW**, a logical construct that defines a set of **InfoObject**s and their mappings to source data elements. InfoSources do not store data, but rather serve as conduits via **Transfer Structure**s.

Integration Rapid Mart

An **Data Integrator application** that provides point-to-point content mapping and data movement mechanisms between different **back-office applications**.

interactive debugger

A Designer feature that allows you to step through the data of a job one row at a time using filters and breakpoints on a line. Like executing a job, you can start the interactive debugger from the Debug menu when a job is active in the workspace. While in debug mode, all other Designer features are set to read-only.

interface

Data Integrator offers two types of interfaces:

An internal **Data Integrator interface**, allows you to create datastore connections to natively supported applications.

An external **Data Integrator interface** (or **adapter**), allows Data Integrator to communicate with information exchange technologies such as Web Services and MQ Series.

IR

See **Information resource (IR)**.

job

The unit of work that can be scheduled independently for execution by the **Administrator**. Jobs are special **work flows** that can be called by other work flows or jobs.

Job Server

A process that receives requests from the **Designer** and the **Administrator** to start and stop **jobs**. To start batch or real-time jobs, the Job Server triggers the **Data Integrator engine**. Engine processes run on the same computer as the Job Server process that triggers them.

join rank

A value given to or calculated for all data sets in a **data flow**. Data Integrator uses the join rank to determine which **source** to read first when assembling the **data set** in a join. Data Integrator uses the source with the lower join rank as the inner source of the join and uses the source with the higher join rank as the outer source of the join. If more than one of the sources is located in the same **DBMS**, the highest rank is used for all of the sources from that DBMS. When a source is processed through a **Query transform** or another **transform**, and then used in a join, the maximum join rank defined for the source is used as the join rank in the query.

Key Figure

In **SAP BW**, an **InfoObject** that represents a numeric fact.

legacy system

An information or transaction processing system used to store data such as bank balances, inventories, payroll, and manufacturing parts.

license-controlled feature

A Data Integrator feature that is enabled or disabled based on the product license. The product license controls which icons and settings are available in Data Integrator as an internal **Data Integrator interface**.

line

The line you use to connect objects in a workspace diagram. Lines represent a left-to-right flow path for data moving between two objects during job execution.

memory datastore

A datastore connection/container for memory tables.

memory table

Internal Data Integrator table used to store a data set in memory while a job runs. Use instead of staging tables to improve performance of a real-time job built with multiple data flows. Use a memory table to move a data set between data flows.

message

Represents hierarchical data, such as a header with line items, for document-oriented transactions, such as a purchase order.

metadata

In Data Integrator, information acquired and maintained to describe tables in **source** and **target** databases. This information includes the names of tables and their columns, and the data types of the columns. All the metadata for all user tables is imported at one time for a given **datastore**.

In general, metadata typically includes a description of data models, a description of the layouts used in database design, the definition of the system of record, the mapping of data from the system of record to other places in the environment, and specific database design definitions.

multiple data flow model

One of three data flow models you can use in real-time jobs. Use this model to develop real-time jobs in separate logical object segments (data flows) rather than in a single data flow. Allows for any number of data flows in a real-time processing loop under these conditions: The first data flow must have a single message source, the last data flow must have a single message target. Other data flows in the same real-time job cannot have a message source or message target. See also **put/acknowledge data flow model** and **single data flow model**.

nested data

Data in one **table** that is related to a single row of another table. A nested table appears in Data Integrator as a column in a parent table. Columns in the nested table can themselves contain tables.

object

Any item that you create in the **Designer**. Data Integrator distinguishes two classes of objects: **reusable objects** that are complete and can be reused in your projects (such as **data flows**) and **single-use objects** that only appear as components of other objects (such as a **try/catch block**). This distinction affects how you create and retrieve each type of object.

object definition

The options that describe the operation of an **object**. To view and modify an object definition, open the object so that its definition appears in the **workspace**.

object library

A tool in the **Designer** that gives you access to reusable objects.

ODBC

Open database connectivity. A standard for **database** access initiated by the SQL Access Group consortium and now owned by Microsoft.

ODS

Operational data store. An OLAP-designed relational **database** that an enterprise has designated as the operational database of record (for example, a finance department might use an ODS to close its books).

OLAP

Online analytical processing. A relational **database** design that is optimized for random access. OLAP systems are used in a query environment, such as for a **front-office application**.

OLTP

Online transaction processing. A relational database design optimized for operational use. OLTP systems are used in an operational environment, such as for a **back-office application**.

operation code

A flag associated with a row in a **data set** that indicates the status of the data in the row. The operation codes are INSERT, UPDATE, DELETE, and NORMAL.

operational statistics

A category of reports within the metadata reporting tool that provides you with job and data flow performance information over a given period of time or number of executions.

parameter

A value passed to a **work flow** or **data flow** when that flow is called.

partition

To divide table data into sets based on a criteria such as a range or list of values in each row. You can configure Data Integrator to read and write partitioned table data in parallel threads. Designing jobs with partitioned table data can improve job performance if a Job Server's computer memory and number of CPUs supports the job's parallel-processing configuration settings.

project

The collection of **jobs** available in the **Designer** at a given time. A project provides a way to organize the **objects** you create.

property

Detailed descriptive information about **objects**.

put/acknowledge data flow model

One of three data flow models you can use in real-time jobs. Similar to the multiple data flow model, this model does not require a data flow with a target message. Data Integrator creates a target message to automatically acknowledge a request message. Designed for use with IDoc sources in SAP R/3 systems. See also **multiple data flow model** and **single data flow model**.

Query transform

A **data transformation object** that you can use to map columns from a **source** to a **target** schema, add new columns to the target schema, determine the data to extract, and perform operations on the data. Like a SQL SELECT statement, a query creates a **data set** that satisfies the conditions you specify.

Rapid Mart

A prepackaged **Data Integrator application** that delivers data for a specific business purpose. Easily deployed and customized, BusinessObjects™ Rapid Marts include the extraction, transformation, and data loading logic to support analysis or **e-commerce**. There are three types of Rapid Mart:

- **Analytic Rapid Mart**
- **Commerce Rapid Mart**
- **Integration Rapid Mart**

R/3 data flow

A **data flow** that extracts data from an **SAP R/3** source table. Data Integrator translates steps you define in an R/3 data flow into **ABAP** and then passes the ABAP program back to your SAP R/3 system for execution. The resulting **table** or **file** resides on the SAP R/3 system to be used as a **source** in the parent data flow.

real-time job

A group of objects (data flows, work flows, conditionals, scripts, and so forth) that execute on-demand as a "request-response" system. You design real-time jobs in the **Designer**, then configure them as real-time services and associate them with an **Access Server** in the **Administrator**, where they are started, managed and monitored. When a real-time service receives a request from a caller, it processes the request and returns a reply. This type of job is recommended for operations on small amounts of data.

relational data

A **data set** in which data in each column contains a scalar value. Data Integrator can process relational data; it can also process **nested data**.

repository

See **Data Integrator repository**.

reusable object

An **object** that can be defined, stored, and reused independent of other objects. Create reusable objects from the **object library**.

RFC server

Remote Function Call server. The Data Integrator RFC server allows third-party programs, including SAP R/3, to schedule and initiate Data Integrator jobs and return the results to Data Integrator.

SAP BW

Business Information Warehouse, an SAP product.

SAP R/3

An **ERP system**.

script

A step in a **work flow** that allows you to calculate values to pass to other parts of the work flow. The script can call **function**s, execute if-then-else statements, and assign values to **variable**s. Write a script in the Data Integrator scripting language.

segment

Format with which the data records of **IDocs** are interpreted.

server group

A defined collection of Job Servers on different computers. A server group automatically measures resource availability on each Job Server in the group and distributes scheduled batch jobs to the Job Server with the lightest load at run-time. Use the Server Groups node in the Administrator's navigation tree to group Job Servers that are associated with the same repository into a server group.

service request

Any message sent from a Web client that requires processing by a **real-time job**.

session

Associated with the **Adapter Properties** tab in an adapter datastore, a session is created by a stream-oriented adapter to group adapter components and maintain state for interactivity between Data Integrator and the adapter.

single data flow model

One of three data flow models you can use in real-time jobs. Allows for one data flow in a real-time processing loop. This data flow must have a single message source and a single message target. See also **multiple data flow model** and **put/acknowledge data flow model**.

single-use object

A step in a **work flow** or **data flow** that cannot be saved independently of the flow. Create single-use objects from the tool palette.

smart editor

A flexible editing tool in Data Integrator used for creating scripts, expressions, and custom functions without having to type the names of existing elements like column, function, and variable names.

SNMP

System Network Management Protocol helps network administrators manage network routing hardware. The protocol can manage a variety of hardware and software devices.

source

An **object** from which Data Integrator reads data.

SQL

Structured Query Language. A query language for accessing relational, **ODBC**, DRDA, or non-relational **database** systems.

SQL query tool

An end-user tool that accepts **SQL** to be processed against one or more relational databases.

star schema

A **database** design you can use to format data in a **data mart**. This design is based on a single fact table to which any number of dimensional tables may be joined. This type of database design supports multi-dimensional database analysis.

step

An **object** that is part of the definition of a **work flow** or **data flow**. Each step is represented by an icon in the diagram of the flow and is connected to other steps to indicate the flow of data through the data flow or the order of execution in the work flow.

table

A **database** table that Data Integrator reads data from or loads data into. The path and mechanisms for reading and loading data and apportioning the data among rows and columns are defined in the **datastore** that the table is associated with. Writing a **data set** to a database table means sending a combination of rows with appropriate **operation code** to the database table.

target

An object in which Data Integrator loads extracted and transformed data in a **data flow**. Data Integrator only loads rows flagged as INSERT or UPDATE.

TCP/IP

Transmission Control Protocol/Internet Protocol. The basic communication protocol of the internet, and often intranets and extranets. A computer having direct access to the internet contains a copy of the TCP/IP program. TCP/IP makes it possible for computers to communicate with each other.

Tdpid

(Teradata Director Program ID) The server name Data Integrator uses when loading with the bulk loader option. Data Integrator uses tdpid as a Teradata Warehouse Builder operator attribute.

Transfer Structure

In **SAP BW**, a data structure that defines a set of **InfoObjects** to be moved into BW. To an ETL tool like Data Integrator, a Transfer Structure looks like a table.

transform

A step in a **data flow** that acts on a **data set**. Data Integrator transforms are available through the **object library**. You can also import user-defined transforms and transforms from your **DBMS**.

Transformation Rule (Transfer Rule)

In **SAP BW**, a data manipulation rule between a **Transfer Structure** and a **Communication Structure**. The results of these transformation are globally available to all **InfoCubes**.

try/catch block

A combination of a try object and one or more catch objects that define alternate execution paths in case an error occurs during the execution of a **job**. You can tune try/catch blocks to trap specific errors and to provide general alerts or messages if an error occurs.

Update Rule

In **SAP BW**, a data manipulation rule between a **Communication Structure** and an **InfoCube**. The result of the data manipulation is specific to one InfoCube.

variable

A symbolic placeholder for a value. Data Integrator lets you define local variables and global variables.

Local variables pertain to the **work flow** in which they are defined. You can pass the value into another **work flow** or **data flow** using a **parameter**.

Global variables pertain to the **job** in which they are defined. With global variables, there is no need to define parameters between **objects** in the job. Global variables can also be selected at execution time. This eliminates the need to open the Designer to set global variable values.

Web Services

A standard platform for integrating applications. Web Services allow different programs, constructed in different languages, on different platforms to communicate with each other. DATA INTEGRATOR supports three XML web service technologies: SOAP, WSDL, and **XML Schema**.

work flow

A **reusable object** containing **steps** to define the order of **job** execution. Work flows call **data flow**s, but cannot manipulate data themselves.

You call work flows from inside other work flows or **job**s. You pass information into or out of work flows using **parameter**s. You reuse work flows by dragging existing work flows from the **object library**.

workspace

The window inside the **Designer** in which you define, display, and modify **objects**. The workspace for a **data flow** contains an area to build a **diagram** representing the data flow definition. The workspace for a **transform** contains an editor for modifying the transform options.

WSDL

Web Services Definition Language. Web services are self-contained, modular business process applications based on open Internet standards.

XML

Extensible Markup Language. This markup language is like HTML (Hypertext Markup Language) in that it specifies a standard with which you can define your own markup languages with their own sets of tags. XML allows you to define various tags with various rules, such as tags that represent business rules, tags that represent data description, or tags that represent data relationships.

XML Schema

The XML format used by Data Integrator to support message processing that includes **Web Services**. XML Schemas describe the data structure of an **XML** file or message. Data flows can read and write data to messages or files based on a specified XML Schema format. You can use the same XML Schema to describe multiple XML sources or targets. XML Schema properties include: Name, Description, Imported from, Root element name, and Namespace.

Index

Business Objects, Americas
3030 Orchard Parkway
San Jose, CA 95134
USA

info@businessobjects.com
www.businessobjects.com

Data Integrator™

Reference Guide

Version 6.5

BUSINESS OBJECTS®
Business Intelligence. If you have it, you know.™

Document Number DI-65-0106-001

January 30, 2004

Contents

1 Welcome

Welcome to the The *Data Integrator Reference Guide*. This guide provides detailed information about the objects, data types, transforms, and functions in the Data Integrator Designer.

This book contains the following chapters:

- Chapter 2, "Data Integrator Objects" — Describes options, properties, and attributes for objects, such as data flows and work flows.
- Chapter 3, "Smart Editor," in the *Data Integrator Reference Guide* — Describes the editor that can be used to create scripts, expressions, custom functions.
- Chapter 4, "Data Types" — Describes the data types used in Data Integrator, and how Data Integrator handles data type conversions.
- Chapter 5, "Transforms" — Describes the transforms included with Data Integrator and how to use these transforms.
- Chapter 6, "Functions and Procedures" — Describes the functions included with Data Integrator and how to use these functions.
- Chapter 7, "Data Integrator Scripting Language" — Describes the Data Integrator scripting language and how you can use this language to create scripts, expressions, and custom functions.

- Chapter 8, "Metadata in repository tables and views" — Describes the repository's reporting tables and views that you can use to analyze an Data Integrator application.
- Chapter 9, "Locales and Multi-Byte Functionality" — Describes how Data Integrator supports the setting of locales and multi-byte code pages for the Designer, Job Server, and Access Server.
- Chapter 10, "Reserved Words" — Lists words that have special meaning in Data Integrator. You cannot use these words in names that you create, such as names of data flows.
- Appendix A, "Glossary" — Defines terms used in the Data Integrator documentation set.

For source-specific information, such as information pertaining to a particular back-office application, consult the supplement for that application.

More Data Integrator product documentation

Consult the *Data Integrator Getting Started Guide* for:

- An overview of Data Integrator products and architecture
- Data Integrator installation and configuration information
- A list of product documentation and a suggested reading path

After you install Data Integrator, you can view technical documentation from many locations. To view documentation in PDF format, you can:

- Select **Start** > **Programs** > **Data Integrator** ***version*** > **Data Integrator Documentation** and choose:
  - Release Notes
  - Release Summary
  - Technical Manuals
- Select one of the following from the Designer's **Help** menu:
  - Release Notes
  - Release Summary
  - Technical Manuals
- Select **Help** from the Data Integrator Administrator

You can also view and download PDF documentation by visiting Business Objects Customer Support online. To access this Web site, you must have a valid user name and password. To obtain your user name and password, go to http://www.techsupport.businessobjects.com and click **Register**.

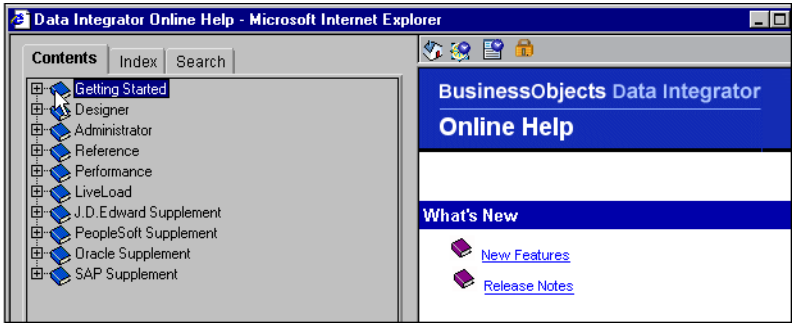
After installing Data Integrator Designer you can also view documentation in HTML format. **Online Help** opens automatically the first time you open the Designer.

You can also open **Help,** using one of the following methods:

- Choose **Contents** from the Designer's Help menu.
- Click objects in the object library or workspace and press **F1**.

  Online Help opens to the subject you selected.

Use **Online Help's** links and tool bar to navigate:



Online help tool bar

Icon	Icon Name	Description
	Home	Returns to the Online Help home page.
	Show/Hide Navigation	Shows and hides the Table of Contents for the core suite of Data Integrator Technical Manuals. Includes a master **Index** and **Search** tabs.
	Show/Hide Descriptions	The Online Help window automatically updates its information when you click an object in the Designer window. In addition, you can click any object in the Designer window then click this button to show or hide an object's local and global variables.
	Start/Stop Help Updates	When the lock appears open, each time you click an object in the Designer window, the Online Help information automatically updates.
		When the lock icon appears closed, the Online Help window does not automatically update when you click objects in the Designer window; you must press F1.

2 Data Integrator Objects

This chapter contains reference information about general Data Integrator objects, such as data flows, jobs, and work flows. This chapter contains two sections:

- Characteristics of objects
- Descriptions of objects

NOTE: For information about source-specific objects, consult the reference chapter of the *Data Integrator Supplement* document for that source.

Characteristics of objects

This section discusses common characteristics of all Data Integrator objects. Specifically, this section discusses:

- Object classes
- Object options, properties, and attributes

Object classes

An object's class determines how you create and retrieve the object. There are two classes of objects:

- Reusable objects
- Single-use objects

Reusable objects

After you define and save a reusable object, Data Integrator stores the definition in the repository. You can then reuse the definition as often as necessary by creating calls to the definition.

Most objects created in Data Integrator are available for reuse. You access reusable objects through the object library.

A reusable object has a single definition; all calls to the object refer to that definition. If you change the definition of the object in one place, and then save the object, the change is reflected to all other calls to the object.

A data flow, for example, is a reusable object. Multiple jobs, such as a weekly load job and a daily load job, can call the same data flow. If the data flow is changed, both jobs call the new version of the data flow.

When you drag and drop an object from the object library, you are creating a new reference (or *call*) to the existing object definition.

You can edit reusable objects at any time independent of the current open project. For example, if you open a new project, you can go to the object library, open a data flow, and edit it. The object will remain "dirty" (that is, your edited changes will not be saved) until you explicitly save it.

Functions are reusable objects that are not available in the object library. Data Integrator provides access to these objects through the function wizard wherever they can be used.

Some objects in the object library are not reusable in all instances:

- Datastores are in the object library because they are a method for categorizing and accessing external metadata.
- Built-in transforms are "reusable" in that every time you drop a transform, a new instance of the transform is created.

Saving reusable objects

"Saving" a reusable object in Data Integrator means storing the language that describes the object to the repository. The description of a reusable object includes these components:

- Properties of the object
- Options for the object
- Calls this object makes to other objects
- Definition of single-use objects called by this object

If an object contains a call to another reusable object, only the call to the second object is saved, not changes to that object's definition.

Data Integrator stores the description even if the object does not validate.

Data Integrator saves objects without prompting you:

- When you import an object into the repository.
- When you finish editing:
  - Datastores
  - Flat file formats
  - XML Schema or DTD formats

You can explicitly save the reusable object currently open in the workspace by choosing **Save** from the **Project** menu. If a single-use object is open in the workspace, the **Save** command is not available.

To save all objects in the repository that have changes, choose **Save All** from the **Project** menu.

Data Integrator also prompts you to save all objects that have changes when you execute a job and when you exit the Designer.

Single-use objects

Single-use objects appear only as components of other objects. They operate only in the context in which they were created.

Saving single-use objects

"Saving" a single-use object in Data Integrator means storing the language that describes the object to the repository. The description of a single-use object can only be saved as part of the reusable object that calls the single-use object.

Data Integrator stores the description even if the object does not validate.

Object options, properties, and attributes

Each object is associated with a set of options, properties, and attributes:

- *Options* control the operation of an object. For example, in a datastore, an option is the name of the database to which the datastore connects.
- *Properties* document an object. For example, properties include the name, description of an object, and the date on which it was created. Properties merely describe an object; they do not affect an object's operation.

  To view properties, right-click an object and select **Properties**.
- *Attributes* provide additional information about an object. Attribute values may also affect an object's behavior.

  To view attributes, double-click an object from an editor and click the **Attributes** tab.

Descriptions of objects

This section describes each Data Integrator object and tells you how to access that object.

The following table lists the names and descriptions of objects available in Data Integrator:

Objects available in Data Integrator

Object	Class	Description
Annotation	Single-use	Describes a flow, part of a flow, or a diagram in the workspace.
Catch	Single-use	Specifies the steps to execute if a given error occurs while a job is running.
Conditional	Single-use	Specifies the steps to execute based on the result of a condition.
Batch Job	Reusable	Defines activities that Data Integrator executes at a given time including error, monitor and trace messages. Jobs can be dropped only in the project tree. The object created is a direct reference to the object in the object library. Only one reference to a job can exist in the project tree at one time.
Data flow	Reusable	Specifies the requirements for extracting, transforming, and loading data from sources to targets.
Datastore	Single-use	Specifies the connection information Data Integrator needs to access a database or other data source. Cannot be dropped.
Document	Reusable	Available in certain adapter datastores, documents are data structures that can support complicated nested schemas.
DTD	Reusable	A description of an XML file or message. Indicates the format an XML document reads or writes. *See also:* XML Schema
File format	Reusable	Indicates how flat file data is arranged in a source or target file.
Function	Reusable	Returns a value.
Log	Single-use	Records information about a particular execution of a single job.
Message function	Reusable	Available in certain adapter datastores, message functions can accommodate XML messages when properly configured.
Outbound message	Reusable	Available in certain adapter datastores, outbound messages are XML-based, hierarchical communications that real-time jobs can publish to adapters.
Project	Single-use	Groups jobs for convenient access.

Objects available in Data Integrator (Continued)

Object	Class	Description
Query transform	Single-use	Retrieves a data set that satisfies conditions that you specify.
Real-time job	Reusable	Defines activities that Data Integrator executes on-demand. Real-time jobs are created in the Designer, then configured and run as services associated with an Access Server in the Administrator. Real-time jobs are designed according to data flow model rules and run as a request-response system.
Script	Single-use	Evaluates expressions, calls functions, and assigns values to variables.
Source	Single-use	An object from which Data Integrator reads data in a data flow.
Table	Reusable	Indicates an external DBMS table for which metadata has been imported into Data Integrator, or the target table into which data is or has been placed. A table is associated with its datastore; it does not exist independently of a datastore connection. A table retrieves or stores data based on the schema of the table definition from which it was created.
Target	Single-use	An object in which Data Integrator loads extracted and transformed data in a data flow.
Template table	Reusable	A new table you want added to a database. All datastores except SAP R/3 datastores have a default template that you can use to create any number of tables in the datastore. Data Integrator creates the schema for each instance of a template table at runtime. The created schema is based on the data loaded into the template table.
Transform	Reusable	Performs operations on data sets. Requires zero or more data sets; produces zero or one data set (which may be split).
Try	Single-use	Introduces a try/catch block.
While loop	Single-use	Repeats a sequence of steps as long as a condition is true.
Work flow	Reusable	Orders data flows and operations supporting data flows.
XML file	Single-use	A batch or real-time source or target. As a source, an XML file translates incoming XML-formatted data into data that DATA INTEGRATOR can process. As a target, an XML file translates the data produced by a data flow, including nested data, into an XML-formatted file.

Objects available in Data Integrator (Continued)

Object	Class	Description
XML message	Single-use	A real-time source or target. As sources, XML messages translate incoming XML-formatted requests into data that a real-time job can process. As targets, XML messages translate the result of the real-time job, including hierarchical data, into an XML-formatted response and sends the messages to the Access Server.
XML Schema	Reusable	A description of an XML file or message. Indicates the format an XML document reads or writes. *See also:* DTD
XML template	Single-use	A target that creates an XML file that matches a particular input schema. No DTD or XML Schema is required.

Annotation

CLASS

Single-use

ACCESS

Click the annotation icon in the tool palette, then click in the workspace.

DESCRIPTION

Annotations describe a flow, part of a flow, or a diagram in a workspace. An annotation is associated with the job., work flow, or data flow where it appears. When you import or export that job, work flow, or data flow, you import or export associated annotations.

For more information, see "Creating annotations" on page 56 of the *Data Integrator Designer Guide*.

NOTE: An annotation has no options or properties.

Batch Job

CLASS

Reusable

ACCESS

- In the object library, click the **Jobs** tab.
- In the project area, select a project and right-click **Batch Job**.

DESCRIPTION

NOTE: For information specific to SAP R/3, see *Data Integrator Supplement for SAP*.

A batch job is a set of objects that you can schedule and execute together. For Data Integrator to execute the steps of any object, the object must be part of a job.

A batch job can contain the following objects:

- Data flows
  - Sources
  - Transforms
  - Targets
- Work flows
- Scripts
- Conditionals
- Try/catch blocks
- While Loops

You can run batch jobs such that you can automatically recover from jobs that do not execute successfully. During automatic recovery, Data Integrator retrieves the results from steps that were successfully completed in the previous run and executes all other steps. Specifically, Data Integrator retrieves results from the following types of steps:

- Work flows
- Data flows
- Script statements
- Custom functions (stateless type only)
- SQL function
- EXEC function
- get_env function
- rand function
- sysdate function
- systime function

Batch jobs have the following built-in attributes:

Attribute	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object.
Description	Your description of the job.
Date created	The date when the object was created.

Batch and real-time jobs have properties that determine what information Data Integrator collects and logs when running the job. You can set the default properties that apply each time you run the job or you can set execution (run-time) properties that apply for a particular run. Execution properties override default properties.

To set default properties, select the job in the project area or the object library, right-click, and choose **Properties** to open the Properties window.

Execution properties are set as you run a job. To set execution properties, right-click the job in the project area and choose **Execute**. The Designer validates the job and opens the Execution Properties window.

You can set three types of execution properties:

- Parameters
- Trace properties
- Global variables

  For an introduction to using global variables as job properties and selecting them at runtime, see "Setting global variable values" on page 303 of the *Data Integrator Designer Guide*.

Parameters

Use parameter options to help capture and diagnose errors using log, View Data, or recovery options.

Data Integrator writes log information to one of three files (in the $LINK_DIR\log*Job Server name**repository name* directory):

- Monitor log file
- Trace log file
- Error log file

You can also select a system profile and a Job Server or server group from the **Parameters** tab of the Execution Properties window.

Select the **Parameters** tab to set the following options.

Parameters

Options	Description
Monitor sample rate (# of rows)	Enter the number of rows processed before Data Integrator writes information to the monitor log file and updates job events. Data Integrator writes information about the status of each source, target, or transform. For example, if you enter 1000, Data Integrator updates the logs after processing 1,000 rows. The default is 1000. When setting the value, you must evaluate performance improvements gained by making fewer calls to the operating system against your ability to find errors quickly. With a higher monitor sample rate, Data Integrator collects more data before calling the operating system to open the file: performance improves. However, with a higher monitor rate, more time passes before you are able to see any errors.
Print all trace messages	Select this check box to print all trace messages to the trace log file for the current Job Server. (For more information on log files, see "Log" on page 75) Selecting this option overrides the trace properties set on the **Trace** tab.

Parameters (Continued)

Options	Description
Capture data for viewing (# of rows)	In Capture data for viewing mode, Data Integrator executes a job only until it collects a specific number of rows for each transform object in a data flow. After execution, you can view the data flow in the workspace by clicking the magnifying glass button to open View Data for each transform in the data flow. View Data allows you to view and compare data at different steps in the data flow. When you enable Capture data for viewing mode, you automatically disable the automatic recovery feature. This property is only available as a run-time property. It is not available as a default property. Enter the number of data rows that you want scanned in each data flow transform. The default value is 100. For more information about View Data, see Chapter 14, "Design and Debug using View Data," in the *Data Integrator Designer Guide*.
Enable recovery	(Batch jobs only) Select this check box to enable the automatic recovery feature. When enabled, Data Integrator saves the results from completed steps and allows you to resume failed jobs. You cannot enable the automatic recovery feature when executing a job in data scan mode. See "Automatically recovering jobs" on page 463 of the *Data Integrator Designer Guide* for information about the recovery options. This property is only available as a run-time property. It is not available as a default property.
Recover from last failed execution	(Batch Job only) Select this check box to resume a failed job. Data Integrator retrieves the results from any steps that were previously executed successfully and re-executes any other steps. This option is a run-time property. This option is not available when a job has not yet been executed or when recovery mode was disabled during the previous run.

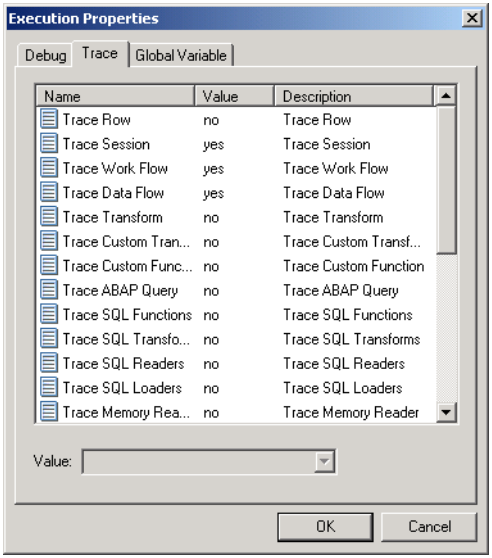
Parameters (Continued)

Options	Description
System profile	Select the system profile to use when executing this job. A system profile defines a set of datastore profiles, which define the datastore connections. For more information, see "Defining profiles" on page 111 of the *Data Integrator Designer Guide*. If a system profile is not specified, Data Integrator uses the default datastore profile for each datastore. This option is a run-time property. This option is only available if there are system profiles defined in the repository.
Job Server or server group	Select the Job Server or server group to execute this job. A Job Server is defined by a host name and port while a server group is defined by its name. The list contains Job Servers and server groups linked to the job's repository. For an introduction to server groups, see "Using Server Groups" on page 29 of the *Data Integrator Administrator Guide*. When selecting a Job Server or server group, remember that many objects in the Designer have options set relative to the Job Server's location. For example: • Directory and file names for source and target files • Bulk load directories

Trace properties

Use trace properties to select the information that Data Integrator monitors and writes to the trace log file during a job. Data Integrator writes trace messages to the trace log associated with the current Job Server and writes error messages to the error log associated with the current Job Server.

To set trace properties, click the **Trace** tab. To turn a trace on, select the trace, click **Yes** in the **Value** list, and click **OK**. To turn a trace off, select the trace, click **No** in the **Value** list, and click **OK**.



You can turn several traces on and off.

Trace properties

Trace	Description
Row	Writes a message when a transform imports or exports a row.
Session	Writes a message when the job description is read from the repository, when the job is optimized, and when the job runs.
Work Flow	Writes a message when the work flow description is read from the repository, when the work flow is optimized, when the work flow runs, and when the work flow ends.

Trace properties (Continued)

Trace	Description
Data Flow	Writes a message when the data flow starts, when the data flow successfully finishes, or when the data flow terminates due to error. This trace also reports when the bulk loader starts, any bulk loader warnings occur, and when the bulk loader successfully completes.
Transform	Writes a message when a transform starts, completes, or terminates.
Custom Transform	Writes a message when a custom transform starts and completes successfully.
Custom Function	Writes a message of all user invocations of the AE_LogMessage function from custom C code.
SQL Functions	Writes data retrieved before SQL functions: • Every row retrieved by the named query before the SQL is submitted in the key_generation function • Every row retrieved by the named query before the SQL is submitted in the lookup function (but only if PRE_LOAD_CACHE is not specified). • When mail is sent using the mail_to function.
SQL Transforms	Writes a message (using the Table_Comparison transform) about whether a row exists in the target table that corresponds to an input row from the source table. The trace message occurs before submitting the query against the target and for every row retrieved when the named query is submitted (but only if caching is not turned on).
SQL Readers	Writes the SQL query block that a script, query transform, or SQL function submits to the system. Also writes the SQL results.

Trace properties (Continued)

Trace	Description
SQL Loaders	Writes a message when the bulk loader: • Starts • Submits a warning message • Completes successfully • Completes unsuccessfully, if the **Clean up bulk loader directory after load** option is selected Additionally, for Microsoft SQL Server and Sybase, writes when the SQL Server bulk loader: • Completes a successful row submission • Encounters an error This instance reports all SQL that DATA INTEGRATOR submits to the target database, including: • When a `truncate stm` command executes if the **Delete data from table before loading** option is selected. • Any parameters included in PRE-LOAD SQL commands • Before a batch of SQL statements is submitted • When a template table is created (and also dropped, if the **Drop/Create** option is turned on) • When a `delete stm` command executes if auto correct is turned on (Informix environment only).
Optimized Dataflow	For Business Objects consulting and technical support use.
Tables	Writes a message when a table is created or dropped. The message indicates the datastore to which the created table belongs and the SQL statement used to create the table.
Scripts and Script Functions	Writes a message when Data Integrator runs a script or invokes a script function. Specifically, this trace links a message when: • The script is called. Scripts can be started any level from the job level down to the data flow level. Additional (and separate) notation is made when a script is called from within another script. • A function is called by the script. • The script successfully completes.

Trace properties (Continued)

Trace	Description
Access Server Communication	Writes messages exchanged between the Access Server and a service provider, including: • The registration message, which tells the Access Server that the service provider is ready • The request the Access Server sends to the service to execute • The response from the service to the Access Server • Any request from the Access Server to shut down
Trace Parallel Execution	Writes messages describing how data in a data flow is parallel processed.

Catch

CLASS

Single-use

ACCESS

With a work flow diagram in the workspace, click the catch icon in the tool palette.

DESCRIPTION

A catch is part of a serial sequence called a try/catch block. The try/catch block allows you to specify alternative work flows if errors occur while Data Integrator is executing a job. Try/catch blocks "catch" groups of errors, apply solutions that you provide, and continue execution.

For each catch in the try/catch block, specify the following:

- One exception or group of exceptions that the catch handles.

  To handle more than one exception or group of exceptions, add more catches to the try/catch block.

- The work flow to execute if the indicated exception occurs.

  Use an existing work flow or define a work flow in the catch editor.

If an exception is thrown during the execution of a try/catch block, and if no catch is looking for that exception, then the exception is handled by normal error logic.

Do not reference output variables from a try/catch block in any subsequent steps if you are using (for batch jobs only) the automatic recovery feature. Referencing such variables could alter the results during automatic recovery.

Also, try/catch blocks can be used within any real-time job component. However, try/catch blocks cannot straddle a real-time processing loop and the initialization or clean up component of a real-time job.

Catches have the following attribute:

Attribute	Description
Name	The name of the object. This name appears on the object in the diagram.

The following table describes the exception groups and errors passed as exceptions.

Catch exceptions

Exception	Number	Description
Catch All Exceptions	All	Catch All Exceptions
Parser Errors	1	Parser errors
Resolver Errors	2	Resolver errors
Execution Errors	5	Internal errors that occur during the execution of a data movement specification
Job initialization	50101	Initialization for a job failed
Job cleanup	50102	Job cleanup failed
Job unknown	50103	Unknown job error
Job failure	50104	Session failure
Work flow initialization	50201	Initialization for a work flow failed
Work flow cleanup	50202	Work flow cleanup failed
Work flow unknown	50203	Unknown work flow error
Work flow failure	50204	Work flow failure
Function initialization	50301	Initialization of the function failed
Function cleanup	50302	Function cleanup failed
Function unknown	50303	Unknown function error
Function failure	50304	Function failure
Step failure	50305	Step failure
System function failure	50306	System function execution failure (function returned an error status)

Catch exceptions (Continued)

Exception	Number	Description
System function initialization	50307	System function startup or launch failure
Data flow initialization	50401	Initialization for a data flow failed
Dataflow open	50402	Cannot open data flow
Dataflow close	50403	Data flow close failed
Dataflow cleanup	50404	Data flow cleanup failed
Data flow unknown	50405	Data flow unknown
Dataflow failure	50406	Data flow failure
Message bad	50407	Bad message error
Transform initialization	50501	Initialization for transform failed
Transform open	50502	Cannot open transform
Transform close	50503	Cannot close transform
Transform cleanup	50504	Cannot clean up transform
Transform unknown	50505	Unknown transform error
Transform failure	50506	Transform failure
OS unknown	50601	Unknown OS error
OS GetPipe	50602	OS GetPipe error
OS ReadPipe	50603	OS ReadPipe error
OS WritePipe	50604	OS WritePipe error
OS ClosePipe	50605	OS ClosePipe error
OS CreatePipe	50606	OS CreatePipe error
OS RedirectPipe	50607	OS RedirectPipe error
OS DuplicatePipe	50608	OS DuplicatePipe error
OS Stdin/stdout restore	50609	OS stdin/stdout restore error
OS CloseProcess	50611	OS CloseProcess error
OS CreateProcess	50612	OS CreateProcess error
OS ResumeProcess	50613	OS ResumeProcess error
OS SuspendProcess	50614	OS Suspend process error
OS TerminateProcess	50615	OS TerminateProcess error
Internal thread initialization	50701	Internal error: thread initialization
Internal thread join	50702	Internal error: thread join
Internal thread start	50703	Internal error: thread start

Catch exceptions (Continued)

Exception	Number	Description
Internal thread resume	50704	Internal error: thread resume
Internal thread suspend	50705	Internal error: thread suspend
Internal mutex acquire	50711	Internal error: acquire mutex
Internal mutex release	50712	Internal error: release mutex
Internal mutex recursive acquire	50713	Internal error: acquire recursive mutex
Internal mutex recursive release	50714	Internal error: mutex recursive release
Internal trap get error	50721	Internal error: trap get
Internal trap set	50722	Internal error: trap set
Internal trap make	50723	Internal error: trap make
Internal condition wait	50731	Errors encountered while processing files
Internal condition signal	50732	Cannot open file
Internal queue read	50741	Mismatch of the position of column
Internal queue write	50742	Premature end of row
Internal queue create	50743	Cannot move file pointer to the beginning of the file
Internal queue delete	50744	Null row is encountered during read
String conversion overflow	50800	String conversion overflow
Validation not done	50801	Validation not complete
Conversion error	50802	Conversion error
Invalid data values	51001	Invalid data values
Database Access Errors	7	Generic Database Access Errors
Unsupported expression	70101	Unsupported expression
Connection broken	70102	Connection broken
Column mismatch	70103	Column mismatch
Microsoft SQL Server	70201	Microsoft SQL Server error
Oracle server	70301	Oracle server error
ODBC	70401	ODBC error
Sybase server	70601	Sybase SQL error
Sybase server operation	70602	Sybase operation error
File Access Errors	8	Errors accessing files through file formats
Open	80101	Cannot open file
Read NULL row	80102	Null row is encountered during read

Catch exceptions (Continued)

Exception	Number	Description
Premature end of row	80103	Premature end of row
Position column mismatch	80104	Mismatch the position of column
LSEEK BEGIN	80105	Cannot move file pointer to the beginning of the file
Repository Access Errors	10	Errors accessing the DATA INTEGRATOR repository
Repository internal error	100101	Repository internal error
Repository ODBC error	100102	Repository ODBC error
Microsoft Connection Errors	12	Errors connecting to the Microsoft SQL Server
MSSQL Server Initialization	120201	Initialization for a Microsoft SQL Server failed
MSSQL Login Allocation	120202	Initialization for a Microsoft SQL Server failed
MSSQL Login Connection	120203	Connection to Microsoft SQL Server failed
MSSQL Database Context	120204	Failed to switch context to a Microsoft SQL Server database
ODBC Allocate environment	120301	ODBC Allocate environment error
Oracle connection	120302	Oracle connection failed
ODBC connection	120401	ODBC connection failed
Sybase server context allocation	120501	Sybase server failed to allocate global context
Sybase server initialization	120502	Sybase server failed to initialize CTLIB
Sybase user data configuration	120503	Sybase server failed to configure user data
Sybase connection allocation	120504	Sybase server failed to allocate connection structure
Sybase login connection	120505	Sybase connection failed
Predefined Transforms Errors	13	Predefined Transforms Errors
Column list is not bound	130101	Primary column information list is not bound
Key generation	130102	Transform Key_Generation error
Options not defined	130103	Transform options are not defined
ABAP Generation Errors	14	ABAP generation errors
ABAP syntax error	140101	ABAP syntax error

Catch exceptions (Continued)

Exception	Number	Description
R/3 Execution Errors	15	R/3 execution errors
R/3 RFC open failure	150101	R/3 RFC connection failure
R/3 file open failure	150201	R/3 file open failure
R/3 file read failure	150202	R/3 file read failure
R/3 file close failure	150203	R/3 file close failure
R/3 file open failure	150301	R/3 file open failure
R/3 file read failure	150302	R/3 file read failure
R/3 file close failure	150303	R/3 file close failure
R/3 connection open failure	150401	R/3 connection open failure
R/3 system exception	150402	R/3 system exception
R/3 connection broken	150403	R/3 connection broken
R/3 connection retry	150404	R/3 connection retry
R/3 connection tranid missing	150405	R/3 connection transaction ID missing
R/3 connection has been executed	150406	R/3 connection has been executed
R/3 connection memory low	150407	R/3 connection memory low
R/3 connection version mismatch	150408	R/3 connection version mismatch
R/3 connection call not supported	150409	R/3 connection call not supported
Email Errors	16	Email errors
System Exception Errors	17	System exception errors
Engine Abort Errors	20	Engine abort errors

Conditional

CLASS

Single-use

ACCESS

With a work flow diagram in the workspace, click the conditional icon in the tool palette.

DESCRIPTION

A conditional implements if/then/else logic in a work flow.

For each conditional, specify the following:

- **If**: A Boolean expression defining the condition to evaluate.

  The expression evaluates to TRUE or FALSE. You can use constants, functions, variables, parameters, and standard operators to construct the expression. For information about expressions, see Chapter 3, "Smart Editor".

  NOTE: Do not put a semicolon (;) at the end of your expression in the **If** box.

- **Then**: A work flow to execute if the condition is TRUE.
- **Else**: A work flow to execute if the condition is FALSE.

  This branch is optional.

The **Then** and **Else** branches of the conditional can be any steps valid in a work flow, including a call to an existing work flow.

Conditionals have the following attribute:

Attribute	Description
Name	The name of the object. This name appears on the object in the diagram.

Data flow

CLASS

Reusable

ACCESS

- In the object library, click the **Data Flows** tab.
- With a work flow diagram in the workspace, click the data flow icon in the tool palette.

DESCRIPTION

A data flow extracts, transforms, and loads data.

You can define parameters to pass values into the data flow. You can also define variables for use inside the data flow.

When Data Integrator executes data flows, it optimizes the extraction, transformation, and loading requirements into commands to the DBMS and commands executed internally. Where it can, DATA INTEGRATOR runs these operations in parallel.

The definition of a data flow can contain the following objects:

Sources	Files, tables, XMl files, XML messages (real-time jobs only), documents, or pre-defined template tables
Targets	Files, tables, XML files, XML messages (real-time jobs only), outbound messages, documents, XML template, or template tables
Transforms	Query is the most commonly used transform

You can view the SQL code Data Integrator generates for table sources in data flows and improve your data flow design accordingly. See "Viewing SQL" on page 34 of the *Data Integrator Performance Optimization Guide*.

To ensure that a job only executes a data flow one time, select the **Execute only once** check box on the data flow Properties window. When this check box is selected, Data Integrator only executes the first occurrence of the data flow; Data Integrator skips subsequent occurrences in the job. You might use this feature when developing complex jobs with multiple paths, such as jobs with try/catch blocks or conditionals, and you want to ensure that Data Integrator only executes a particular data flow one time.

You can run certain transforms and functions in parallel by entering a number in the **Degree of parallelism** box on a data flow's Properties window. When you drop a transform in to the data flow and a function into each transform, the number you enter in the **Degree of parallelism** box is the maximum number of instances that can be generated for each transform or function in the data flow. For more information, see "Degree of parallelism" on page 67 of the *Data Integrator Performance Optimization Guide*.

Data flows have several built-in properties.

Attribute	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object.
Description	Your description of the data flow.

If you delete a data flow from the object library, calls to the object are replaced with an icon indicating that the calls are no longer valid in the workspace.

Datastore

CLASS

Reusable

ACCESS

In the object library, click the **Datastores** tab.

DESCRIPTION

A datastore provides a connection to a data source, such as a database. Through the datastore connection, Data Integrator is able to import descriptions of the data source, such as its metadata. When you specify tables as sources or targets in a data flow, Data Integrator uses the datastore to determine how to read data from or load data to those tables. In addition, some transforms and functions require a datastore name to qualify the tables they access.

If you delete a datastore from the object library, you must remove references to the datastore from the following locations:

- Source or target tables using this datastore in your diagrams
- The lookup and key_generation functions and Key_Generation, History_Preserving, Table_Comparison, and SQL transform references

Datastores have the following properties:

Property	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object. You cannot change the name of a datastore after creation.
Description	Text that you enter to describe and document the datastore.
Date_created	The date that you created the datastore. You cannot change this value.

The information you must enter for a datastore depends on the type of application to which you are connecting. This document describes the options for two types of applications:

- Custom application datastores
- Adapters

After you create a datastore, you can import metadata about the objects, such as tables and functions, in that datastore. See Chapter 5, "Datastores," in the *Data Integrator Designer Guide*.

NOTE: For information about connecting to other applications, see the Data Integrator supplement for that application:

- For J.D. Edwards, see "Datastores" on page 6 of the *Data Integrator Supplement for J.D. Edwards*.
- For Oracle Applications, see "Datastores" on page 5 of the *Data Integrator Supplement for Oracle Applications*.
- For PeopleSoft, see "Datastore" on page 34 of the *Data Integrator Supplement for PeopleSoft*.
- For SAP R/3, see "Datastore" on page 242 of the *Data Integrator Supplement for SAP*.
- For Siebel Applications, see "Datastores" on page 6 of the *Data Integrator Supplement for Siebel*.

CUSTOM APPLICATION DATASTORES

You can define datastores so that Data Integrator can read from and write to the following types of databases:

- DB2
- DETAIL DB2 (use for DB2 databases on mainframe systems)
- DETAIL NRDB (use for non-relational databases on mainframe systems that use three part names like: aa.bb.cc)
- Informix

- Microsoft SQL Server
- ODBC
- Oracle
- Sybase
- Teradata

NOTE: Data Integrator also allows you to create a custom datastore using *Memory* as the Database type. For more information, see "Memory datastores" on page 98 of the *Data Integrator Designer Guide*.

Each database requires its own connection information in the datastore definition. In addition to the unique options described below, any built-in or custom datastore can be set to use a language and code page of a locale. Select the **Set Locale** button for any of the database types described below to set a language and code page. For more information about Data Integrator support for locales, see "Locales and Multi-Byte Functionality" on page 549.

Datastore connection information for custom applications

Database type	Option	Description
DB2	**Connection** tab	
	Multiple profiles	See "Multiple profiles" on page 45.
	DB2 data source	Enter an existing DB2 data source through which Data Integrator accesses sources and targets in this datastore. If you are going to use the Auto correct load feature for DB2 targets, be sure that your data source allows your user name to create/ replace stored procedures.
	DB2 version	Select the version of your DB2 client. This is the version of DB2 that this datastore accesses.
	Rows per commit	Enter the default number of rows you want Data Integrator to process before a commit to the database using this datastore. This value can be overwritten in individual target table options.
	Bulk loader directory	Enter the location where command and data files are written for bulk loading. For Solaris systems, the path name must be less than 80 characters.
	Overflow file directory	Enter the location of overflow files written by target tables in this datastore.
	User name	Enter the name of the account through which Data Integrator accesses the database.
	Password	Enter the user's password.
	Owner name	See "Owner name" on page 46.
	LiveLoad	See "LiveLoad" on page 47.

Datastore connection information for custom applications (Continued)

Database type	Option	Description
	DB2 Properties tab	
	Bulk loader user name	The user name Data Integrator uses when loading data with the bulk loader option. For bulk loading, you might specify a different user name. For example, specify a user who has import and load permissions.
	Bulk loader password	The password Data Integrator uses when loading with the bulk loader option.
	FTP host name	If this field is left blank or contains the name of the computer where the Data Integrator Job Server resides, Data Integrator assumes that DB2 and Data Integrator share the same computer and that FTP is unnecessary. When FTP is unnecessary, all other FTP-related fields can remain blank. See "Using the DB2 bulk load utility" on page 46 of the *Data Integrator Performance Optimization Guide* for a discussion about when FTP is necessary.
	FTP user name	Must be defined to use FTP.
	FTP password	Must be defined to use FTP.
	Server working directory	The working directory for the load utility on the computer that runs the DB2 server. You must complete this field whenever the DB2 server and the Data Integrator Job Server run on separate machines.
DETAIL DB2	**Connection** tab	*See also:* "Mainframe Interfaces (DETAIL)" on page 80 of the *Data Integrator Designer Guide*.
	Node name	Enter the name of the node defined in the dbmover.cfg file during DETAIL installation. This node name identifies the location of the database and the DETAIL Listener.
	DB name	Enter the name of the database to connect to.
	User name	Enter the name of the account through which Data Integrator accesses the database.
	Password	Enter the user's password.
DETAIL NRDB	**Connection** tab	*See also:* "Mainframe Interfaces (DETAIL)" on page 80 of the *Data Integrator Designer Guide*.
	Node name	Enter the name of the node defined in the dbmover.cfg file during DETAIL installation. This node name identifies the location of the database and the DETAIL Listener.
	User name	Enter the name of the account through which Data Integrator accesses the database.

Datastore connection information for custom applications (Continued)

Database type	Option	Description
	Password	Enter the user's password.
Informix	**Connection** tab	
	Multiple profiles	See "Multiple profiles" on page 45.
	Informix data source	Enter an existing Informix data source through which Data Integrator accesses sources and targets defined in this datastore.
	Informix version	Select the version of your Informix client. This is the version of Informix that this datastore accesses.
	Bulk loader directory	Enter the location where command and data files are written for bulk loading.
	Overflow file directory	Enter the location of overflow files written by target tables in this datastore.
	User name	Enter the name of the account through which Data Integrator accesses the database.
	Password	Enter the user's password.
	Owner name	See "Owner name" on page 46.
	LiveLoad	See "LiveLoad" on page 47.
Microsoft SQL Server		
	Connection tab	
	Multiple profiles	See "Multiple profiles" on page 45.
	Database server name	Enter the name of machine where the SQL Server instance is located.
	Database name	Enter the name of the database to which the datastore connects.
	SQL server version	Select the version of your SQL Server client. This is the version of SQL Server that this datastore accesses.
	Rows per commit	Enter the default number of rows you want Data Integrator to process before a commit to the database using this datastore. This value can be overwritten in individual target table options.
	Overflow file directory	Enter the location of overflow files written by target tables in this datastore.
	User name	Enter the name of the account through which Data Integrator accesses the database.
	Password	Enter the user's password.

Datastore connection information for custom applications (Continued)

Database type	Option	Description
	Owner name	See "Owner name" on page 46.
	LiveLoad	See "LiveLoad" on page 47.
ODBC	**Connection** tab	
	Multiple profiles	See "Multiple profiles" on page 45.
	ODBC data source	Enter the name used to create an ODBC data source in the ODBC Administration utility. This option specifies an existing data source through which Data Integrator accesses sources and targets in this datastore.
	Rows per commit	Enter the maximum number of rows loaded to a target table before saving the data. This value is the default commit size for target tables in this datastore. You can overwrite this value for individual target tables.
	Overflow file directory	Enter the directory where you want the data source to write data files when data cannot be loaded to target tables.
	User name	Enter the name of the account through which Data Integrator accesses the database.
	Password	Enter the user's password.
	Owner name	See "Owner name" on page 46.
	Additional connection information	Enter information for any additional parameters that the data source supports (parameters that the data source's ODBC driver and database support). Use the format: `<parameter1=value1; parameter2=value2>` You can leave this option blank.
	LiveLoad	See "LiveLoad" on page 47.

Datastore connection information for custom applications (Continued)

Database type	Option	Description
	ODBC Properties tab	
	Date format	Enter a date format supported by the data source (a date format that the data source's ODBC driver and database supports).
	Time format	Enter a time format supported by the data source (a time format that the data source's ODBC driver and database supports).
	Date-time format	Enter a date-time format supported by the data source (a date-time format that the data source's ODBC driver and database supports).
	Decimal separator	Enter the character that the data source uses to separate the decimal portion of a number.
	Driver does not support array fetch	If you encounter errors when reading from an ODBC custom datastore, especially if the error message involves the ODBC call: SQLFetchScroll, it is safe to assume that your ODBC driver does not support array fetch. In this case, select this option to turn off Data Integrator's array fetch size feature. All source editors using this datastore, will not display the array fetch size performance enhancement option. Reads are performed without the array fetch feature.

Datastore connection information for custom applications (Continued)

Database type	Option	Description
Oracle	**Connection** tab	
	Multiple profiles	See "Multiple profiles" on page 45.
	Database connection name	Enter an existing Oracle connection through which Data Integrator accesses sources and targets defined in this datastore.
	Oracle version	Select the version of your Oracle client. This is the version of Oracle that this datastore accesses.
	Rows per commit	Enter the default number of rows you want Data Integrator to process before a commit to the database using this datastore. This value can be overwritten in individual target table options.
	Bulk loader directory	Enter the location where command and data files are written for bulk loading.
	Overflow file directory	Enter the location of overflow files written by target tables in this datastore.
	User name	Enter the name of the account through which Data Integrator accesses the database.
	Password	Enter the user's password.
	Owner name	See "Owner name" on page 46.
	LiveLoad	See "LiveLoad" on page 47.

Datastore connection information for custom applications (Continued)

Database type	Option	Description
Sybase	**Connection** tab	
	Multiple profiles	See "Multiple profiles" on page 45.
	Database server name	Enter the name of the computer where the Sybase instance is located.
	Database name	Enter the name of the database to which the datastore connects.
	Sybase version	Select the version of your Sybase client. This is the version of Sybase that this datastore accesses.
	Rows per commit	Enter the default number of rows you want Data Integrator to process before a commit to the database using this datastore. This value can be overwritten for a particular target table in a data flow.
	Overflow file directory	Enter the location where Data Integrator writes overflow files when loading target tables in this datastore.
	User name	Enter the name of the account through which Data Integrator accesses the database.
	Password	Enter the password for the user name.
	Owner name	See "Owner name" on page 46.
	LiveLoad	See "LiveLoad" on page 47.

Datastore connection information for custom applications (Continued)

Database type	Option	Description
Teradata	**Connection** tab	
	Multiple profiles	See "Multiple profiles" on page 45.
	Database server name	Enter the name of the computer where the Teradata instance is located.
	Database name	Enter the name of the database to which the datastore connects.
	Teradata server version	Select the version of your Teradata client. This is the version of Teradata that this datastore accesses.
	Rows per commit	Enter the default number of rows you want Data Integrator to process before a commit to the database using this datastore. This value can be overwritten for a particular target table in a data flow.
	Bulk loader directory	Enter the location where command and data files are written for bulk loading.
	Overflow file directory	Enter the location where Data Integrator writes overflow files when loading target tables in this datastore.
	User name	Enter the name of the account through which Data Integrator accesses the database.
	Password	Enter the password for the user name.
	Owner name	See "Owner name" on page 46.
	LiveLoad	See "LiveLoad" on page 47.
	Teradata Properties tab	
	Log directory	Enter the location where log files are written.
	Tdpid	The password Data Integrator uses when loading with the bulk loader option.

Datastore connection information for custom applications (Continued)

Database type	Option	Description
multiple	Multiple profiles	Select this check box to specify multiple profiles and change configuration information at runtime. After creating a datastore with multiple profiles, you cannot disable this option. If you enable multiple profiles, you must specify the **Owner name**. Creating datastores with multiple profiles is optional. If you do not select this check box, Data Integrator treats this as a datastore with a single profile. You can change an existing single profile datastore to a datastore with multiple profiles. However, remember that all tables in a multiple-profile datastore must have the same owner. Existing datastores might contain tables with different owners. If you change an existing single-profile datastore to a multiple-profile datastore, you must verify that all tables in the datastore have the same owner. If they do not, Data Integrator will generate runtime errors.

Datastore connection information for custom applications (Continued)

Database type	Option	Description
	Owner name	Enter the default table owner name for all tables in this datastore. You can specify a value even if you do not enable multiple profiles: • If you do not enable multiple profiles and do not specify the **Owner name**, you can specify a table owner name from the Import By Name window when importing metadata. You can import tables with different owners from the same datastore. Template tables use datastore user name as default table owner name. • If you do not enable multiple profiles but do specify an **Owner name**, then the owner name becomes the default table owner name. Template tables use the default table owner name and the name also appears in the Import By Name window as the default. If you choose to edit the owner name in the Import by Name window, then the table imported must belong to the specified owner. • If you enable multiple profiles, the **Owner name** is required. All the database objects in a datastore with multiple profiles must have the same owner name. In the repository, the owner name of all database objects is set to "DSOWNER." When running a job, Data Integrator resets the owner to the value of the selected datastore profile. Specifically, during runtime, Data Integrator changes the owner in appropriate objects, functions, and transforms such as: • Source table • Target table • Template table • Key_generation function • Lookup function • Lookup_seq function • Table_Comparison transform • Key_Generation transform

Datastore connection information for custom applications (Continued)

Database type	Option	Description
	LiveLoad	Select this check box to activate LiveLoad for this datastore. When you activate LiveLoad, you must enter connection information for both the master and the mirror databases. (Select **Master** and enter connection information for the master database, and select **Mirror** and enter connection information for the mirror database.) See "Datastore object" on page 18 of the *Data Integrator LiveLoad User's Guide* for more information. If you activate LiveLoad, you cannot create a datastore with multiple profiles.

ADAPTERS

An adapter datastore provides a connection to a particular adapter.

Connection tab

- **Name** — Enter a unique name that identifies the adapter datastore. You might relate the name of the datastore to the name of the adapter instance.
- **Job Server** — Select the Job Server that runs the adapter.
- **Adapter Instance Name** — Select the adapter instance. The list shows adapter instances on the Job Server computer that are registered with the selected Job Server. These names match those specified in the adapter's XML configuration file.

Adapter Properties tab

Enter all adapter property information required to complete the datastore connection. The adapter documentation lists all information required for datastore connection. Obtain this documentation from your adapter vendor. In some cases, your adapter's data source administrator can provide connection information.

NOTE: If you do not select a **Job Server** and an **Adapter instance name**, you cannot access the **Adapter Properties** tab.

Database Properties tab

Enter all database property information required to complete the datastore connection. (This can include such information as database type, connection name, database version(s), user name, and password.)

NOTE: This tab is available only if the adapter indicates that it can get the data from the source directly using SQL.

An adapter datastore cannot be set to use a locale. Settings entered using the **Set Locale** button on the Datastore Editor are ignored. Locale settings for sources and targets must be handled in an Adapter SDK. Data Integrator uses the UTF-8 code page to process adapter data. For more information about Data Integrator support for locales, see "Locales and Multi-Byte Functionality" on page 549

Document

CLASS

Reusable

ACCESS

In the object library, click the **Datastores** tab.

DESCRIPTION

Available in some adapter datastores, documents describe a data schema. Documents can support complicated nested schemas. You can use documents as sources or targets.

See your adapter's documentation for more specific information about the options available for documents.

DTD

CLASS

Reusable

ACCESS

In the object library, click the **Formats** tab, then open the DTD category.

DESCRIPTION

A DTD (document type definition) describes the data schema of an XML message or file.

NOTE: XML Schemas can be used for the same purpose. See "XML Schema" on page 152.

Data flows can read and write data to messages or files based on a specified DTD format.You can use the same DTD to describe multiple XML sources or targets.

To use DTDs, import metadata into Data Integrator. You can import a DTD directly, or you can import an XML document that contains or references a DTD. During import, Data Integrator converts the structure defined in the DTD into the Data Integrator nested-relational data model (NRDM). See "Rules for importing DTDs" on page 57.

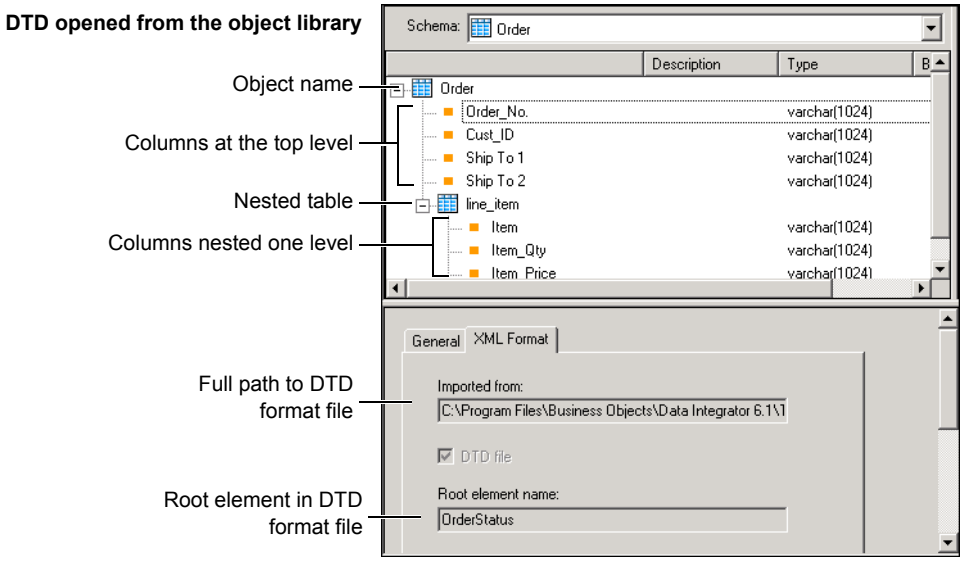
EDITOR

Open the DTD editor by double-clicking a DTD name in the object library.

Document Type Definition

```
<?xml encoding="UTF-8"?>
<!ELEMENT Order (OrderNo, CustID, ShipTo1, ShipTo2, LineItems+)>
<!ELEMENT OrderNo (#PCDATA)>
<!ELEMENT CustID (#PCDATA)>
<!ELEMENT ShipTo1 (#PCDATA)>
<!ELEMENT ShipTo2 (#PCDATA)>
<!ELEMENT LineItems (Item, ItemQty, ItemPrice)>
<!ELEMENT Item (#PCDATA)>
<!ELEMENT ItemQty (#PCDATA)>
<!ELEMENT ItemPrice (#PCDATA)>
```

DTD opened from the object library



For import procedures, see "Using Document Type Definitions (DTDs)" on page 229 of the *Data Integrator Designer Guide*.

PROPERTIES

DTDs have the following properties.

Property	Description
Name	The name of the format. This name appears in the object library under the Formats tab and is used for sources and targets (XML files or messages) that reference this format in data flows.
Description	Text that you enter to describe and document the DTD.
Imported from	The full path to the format. For example, `C:\data\test.dtd`
DTD file	(Read-only) If the check box is selected, the DTD format was originally imported from a DTD file. Otherwise, it was imported from an XML file with an associated DTD.
Root element name	The name of the primary node of the XML that the DTD is defining. Data Integrator only imports elements of the format that belong to this node or any sub nodes.

ATTRIBUTES

Data Integrator supports the following column attributes for DTDs.

Column attributes supported for DTDs

Attribute	Description
Enumeration	Contains a list of all possible values separated by vertical bars. For example: "Red | White | Blue Green | Magenta". A string display is cut off at 256 characters.
Fixed Value	The only value the column can have.
Native Type	String. The original data type of the of the element or attribute in the DTD.
Required	Indicates whether this column always has to be mapped (YES/NO). If a column is optional (required =no), then validation will allow mapping expressions to be missing for these columns and at runtime the engine will substitute NULLs for the missing values.
XML Type	Allows you to track whether the column was an element or attributes in the original DTD.

Nested table attributes supported for DTDs

Attribute	Description
Any One Column	If choice (for example, "white | black | almond"), then Data Integrator sets the value of Any One Column to YES. If sequence (for example, "first, last, street, city, state") then Data Integrator sets the value to NO. If both are present in the DTD, the value is set to NO.
Minimum Occurrence	If ()* then minimum occurrence is set to zero. If ()+, then minimum occurrence is set to 1. Indicates minimum number of rows that can be in the table.

SUPPORTED DTD COMPONENTS

Data Integrator reads the following DTD components. To process the data read in an XML file or message, Data Integrator translates the DTD into its internal nested-relational database model.

Each component in the DTD is defined by its *content model*. Data Integrator supports the declarations in XML content models as follows:

Content model support

DTD declaration	Support in Data Integrator
DOCTYPE	
SYSTEM	Supported.
PUBLIC	No support.
Declarations	
ELEMENT	Supported. The XML Type attribute of the corresponding column is set to Element.
ATTRIBUTE	Supported. The XML Type attribute of the corresponding column is set to Attribute.
ENTITY	Supported. All entity references that can be expanded are expanded. Any that cannot be expanded cause an error at the time that you import the DTD.
NOTATION	No support. Elements defined with NOTATION cause an error at the time that you import the DTD.
Content model	
ANY	No support. Elements defined with ANY cause an error at the time that you import the DTD.
EMPTY	Supported.
#PCDATA	Supported. Data Integrator converts to varchar(1024).
MIXED	Supported.
Attribute declarations	
CDATA	Supported. Data Integrator converts to varchar(1024).
ID	Supported. Data Integrator converts to varchar(1024). When producing XML output, Data Integrator cannot ensure that ID values are unique throughout the schema.
IDREF	Supported. Data Integrator converts to varchar(1024).
IDREFS	Supported. Data Integrator converts to varchar(1024).
NMTOKEN	Supported. Data Integrator converts to varchar(1024).
NMTOKENS	Supported. Data Integrator treats multiple tokens as a single token with more than one space-separated values.

Content model support (Continued)

DTD declaration	Support in Data Integrator
Enumerated value	Supported. Data Integrator saves the enumerated values in the Enumeration attribute of the column. When producing XML output, Data Integrator checks to ensure that the value generated by the real-time job for the corresponding column is from the list; if no value is generated, Data Integrator uses the provided default value. If you validate XML messages against the DTD in a real-time job and the message includes a value that is not allowed based on the DTD, the XML source produces an error.
Attribute declaration defaults	
#REQUIRED	Supported. Data Integrator saves this as the Required attribute with a value of `YES` and as data type varchar(1024). When producing XML output, Data Integrator always provides a value. If there is no value supplied, the output value is NULL (`''`).
#IMPLIED	Supported. Data Integrator saves this as the Required attribute with a value of `NO` and as the data type varchar(1024). When producing XML output, Data Integrator provides whatever value is generated in the data flow for the corresponding column, including a NULL value (`''`).
#FIXED (default value)	Supported. Data Integrator saves this as the Fixed Value attribute and the data type varchar(1024). When producing XML output, Data Integrator checks to ensure that the value generated by the real-time job for the corresponding column is from the list; if no value is generated, Data Integrator uses the provided default value.
Default values	Supported. Data Integrator converts to data type varchar(1024). When producing XML output, Data Integrator uses the default value if the value generated in the real-time job for the corresponding column is NULL.

To produce a data model that can include all possible configurations of an element, Data Integrator can simplify some of the content model operations:

Operator	Description	Support in Data Integrator
No operator	One and only one	One and only one.
Comma (,)	Sequence	Supported. Data Integrator uses the ordering given in the DTD as the column ordering in the internal data set. Also the Any One Column attribute is set to a value of `NO`.
Vertical bar (|)	Choice (either/or)	Supported. Data Integrator uses the ordering given in the DTD as the column ordering in the internal data set. Also the Any One Column attribute is set to a value of `YES`. The internal data set must include both options.
Plus (+)	One or more	Supported. Saved as nested table attribute Minimum Occurrence with a value of "1". The internal data set must include options for one or more elements.
Asterisk (*)	Zero or more	Supported. Saved as nested table attribute Minimum Occurrence with a value of "0". Data Integrator translates an item or grouping including zero or more items into a nested table.
Question mark (?)	Optional	Supported. The internal data set includes the Required attribute set to a value of NO for the corresponding column or nested table.
Parentheses ()	Group	Dropped. The internal data set does not maintain groupings unless the group is operated on by the * operator. If the group can allow more than one item, Data Integrator makes a new nested table into which it places the elements in the group.

After these simplifications, Data Integrator needs only work with two DTD operators: sequence (strict ordering) and the combined operators of the group operator with the zero or more item operator. For the purpose of representing the data internally in Data Integrator, all DTDs can now be written using only , or ()*.

Rules for importing DTDs

Data Integrator applies the following rules to convert a DTD to a Data Integrator internal schema:

1. Any element that contains an PCDATA only and no attributes becomes a column.
2. Any element with attributes or other elements (or in mixed format) becomes a table.
3. An attribute becomes a column in the table corresponding to the element it supports.
4. Any occurrence of choice (,)or sequence (|)operators uses the ordering given in the DTD as the column ordering in the internal data set.
5. Any occurrence of a multiple entities, such as ()* or $()^+$, becomes a table with an internally generated name (an implicit table).

   The internally generated name is the name of the parent followed by an underscore, then the string “nt” followed by a sequence number. The sequence number starts at 1 and increments by 1.

After applying these rules, Data Integrator uses two additional rules, except where doing so would allow more than one row for a root element:

1. If an implicit table contains one and only one nested table, then the implicit table can be eliminated and the nested table can be attached directly to the parent of the implicit table.

For example, the SalesOrder element might be defined as follows in a DTD:

```
<!ELEMENT SalesOrder (Header, LineItems*)>
```

When converted into Data Integrator, the LineItems element with the zero or more operator would become an implicit table under the SalesOrder table. The LineItems element itself would be a nested table under the implicit table.

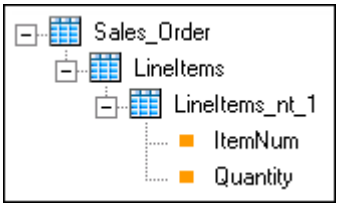
Because the implicit table contains one and only one nested table, the format would remove the implicit table.

2. If a nested table contains one and only one implicit table, then the implicit table can be eliminated and its columns placed directly under the nested table.

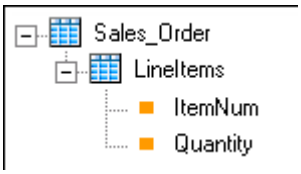
   For example, the nested table LineItems might be defined as follows in a DTD:

   ```
   <!ELEMENT LineItems (ItemNum, Quantity)*>
   ```

When converted into Data Integrator, the grouping with the zero or more operator would become an implicit table under the LineItems table. The ItemNum and Quantity elements would become columns under the implicit table.



Because the LineItems nested table contained one and only one implicit table, the format would remove the implicit table.



DESIGN CONSIDERATIONS

The following areas provide opportunities for you to improve performance and tune the nested-relational data model results for a given DTD:

- Recursion

  If the DTD contains an element that uses an ancestor element in its definition, Data Integrator expands the definition of the ancestor for a fixed number of levels. For example, given the following definition of element "A":

  ```
  A: B, C

  B: E, F

  F: A, H
  ```

Data Integrator produces a table for the element "F" that includes an expansion of "A." In this second expansion of "A," "F" appears again, and so on until the fixed number of levels. In the final expansion of "A," the element "F" appears with only the element "H" in its definition.

- Repeated column names

  Data Integrator does not allow more than one column with the same name at the same level in a table. If the internal schema that Data Integrator produces from a DTD contains duplicate names, Data Integrator adds a suffix to each instance of the duplicated name to ensure unique column names. The suffix is an underscore followed by a sequence number, starting at 1 and incrementing by 1.

  A DTD can produce duplicate names when the DTD contains a repeated element at one level or an element with a scalar value with an attribute of the same name.

- Ambiguous DTDs

  You can create a DTD such that Data Integrator does not have enough information to make a unique decision when generating the internal data set. Data Integrator reacts to an ambiguous DTD by throwing an error for the XML message source at run time. An example of an ambiguous definition is as follows:

DTD	`A: ((B, (C)*) | (B, (D)*))+`
Schema in Data Integrator	`A: (B, (C)*, B, (D)*)*`

XML input	`A: <B>text</B>`
	`<D>1</D>`
	`<D>2</D>`

Data Integrator will use the B element data to populate the first B column, then use the D element data to populate the D element. If this data is then translated back into XML, it would be invalid relative to the DTD.

METADATA

If you delete a DTD from the object library, XML file and message sources or targets that are based on this format are invalid. Data Integrator marks the source or target objects with an icon that indicates the calls are no longer valid.

To restore the invalid objects, you must delete the source or target and replace it with a source or target based on an existing DTD.

ERROR CHECKING

Data Integrator allows you to control whether it checks each incoming XML file or message for validity. If you choose to check each XML file or message, Data Integrator uses the DTD imported and stored in the repository rather than a DTD specified by a given XML file or message. If a file or message is invalid relative to the DTD, the real-time job produces an error and shuts down.

During development, you might validate all files and messages to test for error conditions with representative messages. During production, you might choose to accept rare invalid files or messages and risk ambiguous or incorrect data.

All files or messages that Data Integrator produces for an XML file or message target are validated against the imported DTD.

You can enable or disable validation for an XML file or message source or target in that object's editor.

File format

CLASS

Reusable

ACCESS

In the object library, click the **Formats** tab.

DESCRIPTION

A flat file format describes the structure of an ASCII file. You store templates for file formats in the object library. You use the templates to define the file format of a particular source or target file in a data flow.

A flat file format consists of multiple properties. You set the properties in the file format editor. Available properties vary by the mode of the file format editor:

- New mode — Use to create a new file format template. To open in new mode, go to the **Formats** tab in the object library, right-click **Flat Files**, and select **New**. Data Integrator opens the File Format editor window.
- Edit mode — Use to edit an existing file format template. To open in edit mode, go the **Formats** tab in the object library, select an existing flat file format, double-click, or right-click and select **Edit**. Data Integrator opens the File Format window, containing the editor
- Source mode — Use to edit the file format of a particular source file. To open in source mode, click the name of the source file in the workspace. The file format editor appears below the file's schema.

- Target mode — Use to edit the file format of a particular target file. To open in target mode, click the name of the target file in the workspace. The file format editor appears below the file's schema.

The **Property-Value** work area in the file format editor lists file format properties that are not field-specific. The following table lists all **Property-Value** work area options. These options are filtered by the mode you are using.

File format properties

Property	Possible values	Description	Mode
General			
Type	Delimited, Fixed width, R/3	The format of the data in the text file. Available properties change based on the selected file format type. For information about R/3 file formats, see "File format" on page 246 of the *Data Integrator Supplement for SAP*.	New, Edit
Name	Any alphanumeric character and underscores (_)	A descriptive name for the file format. This name appears in the object library.	New
Join rank	Integer greater than or equal to 0	Specifies join rank number. For more information, see "Source" on page 92.	Source
Cache	Yes, No	Indicates whether DATA INTEGRATOR should load the data into memory. For more information, see "Source" on page 92.	Source

File format properties (Continued)

Property	Possible values	Description	Mode
Adaptable schema	Yes, No	Indicates whether the schema of a delimited file format is adaptable or fixed. • **Yes** indicates that the schema is adaptable. The actual file can contain fewer or more columns than indicated by the file format. If a row contains fewer columns than expected, Data Integrator loads null values into the columns missing data. If a row contains more columns than expected, Data Integrator ignores the additional data. • **No** indicates that the schema is fixed. DATA INTEGRATOR requires the number of columns in each row to match the number of columns specified in the file format. The default is **No**. If you select **Yes**, you must ensure that the selected column and row delimiters do not appear inside the actual data.	Source
Make port	Yes, No	Indicates whether the file is an embedded data flow port. Choose **Yes** to make a source or target file an embedded data flow port. Default is **No**. For more information, see Chapter 11, "Embedded Data Flows," in the *Data Integrator Designer Guide*.	Source, Target
Rows to read	Integer or blank	Indicates the maximum number of rows that Data Integrator reads. The default is blank. Data Integrator reads all rows. If the value is zero or negative, Data Integrator reads no rows.	Source
Custom transfer program	Yes, No	Enables Data Integrator to use a third-party file transfer program. Displays additional Custom transfer program options in the file format editor below the Input/Output properties.	All

File format properties (Continued)

Property	Possible values	Description	Mode
Parallel process threads	Integer greater than 0	Specifies the number of threads that read or load data when this file format is used as a source or target in a data flow. This option allows you to parallel process flat file reads and loads. For example, if you have four CPUs on your Job Server computer, enter the number 4 in the Parallel process threads box to maximize performance. For more information, see "File multi-threading" on page 76 of the *Data Integrator Performance Optimization Guide*.	All
Data File(s)			
Location	Local, Job Server	During design, indicates whether files are located on the local machine that runs the Designer or on the machine that runs the Job Server. If you select Job Server, you must enter the absolute path to files. Remember that UNIX systems are case-sensitive. During execution, all files must be located on the Job Server machine that executes the job. If you use different files to design your job, change the file specified (through the **Root directory** and **File** properties) before execution.	All
Root directory	Path name for the file or blank	The directory where the file is located. If you enter a directory name, then enter only the file name for the **File** property. If you leave the root directory blank, then enter a file name that includes the full path name in the **File** property.	New, Edit

File format properties (Continued)

Property	Possible values	Description	Mode
File (s)	File name, file name including full path name, or blank	In new and edit modes, specifies an existing file on which you base the file format description. Data from this file appears in the Column Attributes area. In these modes, you can leave this property blank. In source and target modes, specify the location of the actual file for this source or target. In these modes, you cannot leave this property blank. For added flexibility, you can enter: • A variable that is set to a particular file with full path name. Use variables to specify file names that you cannot otherwise enter, such as file names that contain multi-byte characters. • A list of files, separated by commas, or a file name containing a wild card. In this case, Data Integrator reads all these files as a single source. See "Reading multiple files at one time" on page 135 of the *Data Integrator Designer Guide*. During design, you can specify a file located on the computer where the Designer runs or on the computer where the Job Server runs. Indicate the file location in the **Location** property. During execution, you must specify a file located on the Job Server computer that will execute the job.	All
Delete file	Yes, No	Specifies whether Data Integrator should delete the file before loading. • **Yes** indicates that Data Integrator should delete the file before loading. • **No** indicates that Data Integrator should append to the existing file.	Target

File format properties (Continued)

Property	Possible values	Description	Mode
Delimiters			
Column	Tab, Semicolon, Comma, Space, or any character sequence	For delimited file formats, this is the character sequence that indicates the end of one column and the beginning of the next.	New, Edit
Row	`{New line}` or any character sequence	A character sequence that indicates where one row of data ends and the next begins.	New, Edit
Text	Single quotation marks ('), double quotation marks ("), or `{none}`	Denotes the start and end of a character string. All characters (including those specified as column or row delimiters) between the first and second occurrence of this character is a single text string.	New, Edit
Default Format			
Escape char	Any character sequence, or `{none}`	A special character sequence that causes DATA INTEGRATOR to ignore the normal column delimiter. Characters following the escape character sequence are never used as column delimiters. For example, suppose you specify a forward slash as the escape character and a comma as the column delimiter. Then, you must have a forward slash to have a comma appear inside a field.	New, Edit
NULL indicator	`{none}` or any other character sequence	Special character sequence that Data Integrator interprets as NULL data.	New, Edit

NOTE: Data in columns cannot include the column or row delimiter, unless you also specify a text delimiter. For example, if you specify a comma as your column delimiter, none of the data in the file can contain commas. However, if you specify a comma as the column delimiter and a single quote as the text delimiter, commas are allowed in strings in the data.

You can use any ASCII characters (including non-printing characters) for column and row delimiters.

You can specify an ASCII character by entering a forward slash (/) followed by the decimal representation of the character. For example, to use Y umlaut (Ÿ) as a delimiter, enter /255 in the delimiter property box.

File format properties (Continued)

Property	Possible values	Description	Mode
Ignore row marker(s)	`{none}` or any other character sequence	Character sequence, which when found at the beginning of rows, cause Data Integrator to ignore the row when reading the file or automatically creating metadata. To enter multiple character sequences, separate each with a semi-colon. To include a semi-colon or backslash as a marking character, precede with a backslash.	New, Edit, Source
Blank padding	leading, trailing	For fixed width file formats, the method for adding extra blanks to fields. • Leading — Blanks added before data • Trailing — Blanks added after data	New, Edit
Date	`yyyy.mm.dd` or other combinations	The date format for reading or writing date values to and from the file.	New, Edit
Time	`hh24:mi:ss` or other combinations	The time format for reading or writing time values to and from the file.	New, Edit
Date-Time	`yyyy.mm.dd hh24:mi:ss` or other combinations	The date-time format for reading or writing datetime values to and from the file.	New, Edit
Validate decimal data	`Yes, No`	For file targets, by default, Data Integrator converts data in delimited files to the decimal data type (even if it is in string form due to lazy decimal conversion) to make sure that the decimal format is valid. To improve performance you can manually deselect this conversion and validation operation. In the Target File Editor, set **Validate decimal data** to NO.	Target
Input/Output			
Bad rows limit	Integer greater than 0	A bad row contains data that does not match the format specified for the file being processed. When the bad row limit is reached, the job stops processing. To ignore bad rows, continue processing, and log warnings, specify as `none` or `0`.	New, Edit, Source

File format properties (Continued)

Property	Possible values	Description	Mode
Style	Headers or BOF/EOF	The format of the start and end of the file. Available properties in the Input/Output property group change based on this selection.	New, Edit
Skipped rows	Integer	For file formats using Headers style, the number of rows skipped when reading the file. Specify a non-zero value when file includes comments or other non-data information.	New, Edit
Skip row header	Yes or No	For file formats using Headers style, indicates whether the first row of data in the file contains the column names and should be skipped when reading the file. DATA INTEGRATOR uses this property in addition to the **Skipped rows** property. When you select **Yes**, Data Integrator does not read data from the first row, and uses data in the first row to determine the file's column names.	New, Edit
Write row header	Yes or No	For file formats using Headers style, indicates whether to write column names in first row of output file.	New, Edit
BOF Marker	Any character sequence, including a blank space, an empty string, or `{none}`	For file formats using BOF/EOF style, the string that marks the start of data in the file.	New, Edit
EOF Marker	Any character sequence, including a blank space, an empty string or `{none}`	For file formats using BOF/EOF style, the string that marks the end of data in the file.	New, Edit
Custom Transfer			
Program executable	File name	(required) The name of the custom transfer program or its initialization script. For example: "`MyProgram.exe or MyProgram.cmd`"	New, Edit

File format properties (Continued)

Property	Possible values	Description	Mode
User name	Any character sequence, including a blank space, an empty string or `{none}`	(optional) Log in ID for the server to which the custom transfer program connects. You may want to allow a custom program user to enter their user name when they enter their password in Data Integrator.	New, Edit
Password	Any character sequence, including a blank space, an empty string or `{none}`	(optional) Password for the server to which the custom transfer program connects. Passwords entered into this option are encrypted by Data Integrator.	New, Edit
Arguments	Any character sequence, including a blank space, an empty string or `{none)`	(optional) You can create arguments in your custom transfer program and then specifically flag them from within Data Integrator using this box. For example, you might have security or compression mechanisms in your program. You can also link Data Integrator connection data to your transfer program's flags.	New, Edit
Locale		For more information about locales for sources, targets and Data Integrator's internal processing, see "Locales and Multi-Byte Functionality" on page 549.	
Language	For the list of supported languages, see "Languages" on page 568.	Specifies the human language (for example, Korean, Japanese, or English) in which data is stored or processed. Select from the displayed list.	New, Edit

File format properties (Continued)

Property	Possible values	Description	Mode
Territory	For the list of supported territories, see "Territories" on page 569.	Represents the geographical location (usually the country) where the language is used. The paring of a language and a territory determines factors such as date format, time format, decimal separator, etc. For example, English is used differently in the United States and the United Kingdom.	New, Edit
Code Page	For the list of supported code pages, see "Code pages" on page 570.	Specifies the sequence of bits that defines a character. For example, the Japanese code page contains ASCII, Greek, Cyrillic, and Japanese characters, thereby supporting the English, Greek, Russian, and Japanese languages.	New, Edit

The Column Attributes work area in the file format editor contains properties about the fields in the file format.

Property	Possible values	Description
Field name	Any sequence of letters or numbers, not including blank spaces	A name that identifies data in this column. If your file format uses the Headers style and you select **Yes** for the **Write row header** property, Data Integrator writes the field names in the target file.
Data type	VarChar, Date, DateTime, Time, Decimal, Double, Int, Long, Numeric, Real	The data type of values in this column. The Long data type is not available in fixed-width formats. For more information about data types, see "Descriptions of data types" on page 186.
Field size	Positive integer	If the data type is VarChar, specifies the number of characters in the field.
Precision	Positive integer	If the data type is Decimal or Numeric, specifies the total number of digits in the field.
Scale	Positive integer	If the data type is Decimal or Numeric, specifies the number of digits to the right of the decimal point.
Format	{none}	For all data types other than VarChar, specifies the format for this particular field. You can use this property to overwrite the default format. For example, if one date field is different than others, you can specify the different format here.

If you delete a file format template from the object library, you must also delete all file sources and targets that are based on that file format template.

Function

CLASS

Reusable

ACCESS

- For existing functions, click the **Functions** button in object editors.
- For imported functions, in the object library, click the **Datastores** tab, expand a datastore, and expand the **Functions** node.
- For custom functions, click the **Custom Functions** tab in the object library or select **Tools** > **Custom Functions**.

DESCRIPTION

Use functions to process values. There are three types of functions:

- Built-in functions
- DBMS and application functions or stored procedures imported into Data Integrator
- Custom functions you create

Functions have the following attributes:

Attribute	Description
Name	The name of the function. This name appears in the function wizard and smart editor. It is also used when the function appears in a script or expression.
Description	Descriptive text entered when the function is created or imported into Data Integrator.
Function type	Each imported or custom function has a type. For imported and custom functions, right-click the function from the object library and select **Properties** to view the type. Descriptions and syntax for built-in functions is listed in the function wizard and the smart editor.
Enable Parallel Execution	Check box on the **Properties** window. Enables Data Integrator to run stored procedures and custom functions in parallel. This option must be selected in addition to entering a positive number for the parent data flow's degree of parallelism. For more information see, "Degree of parallelism" on page 67 of the *Data Integrator Performance Optimization Guide*.

Functions are described in detail in Chapter 6, "Functions and Procedures".

Log

CLASS

Single-use

ACCESS

- To see the logs for jobs run on a particular Job Server, log in to the repository associated with the Job Server when you open the Designer. In the project area of the Designer, click the **Log** tab, and expand the job tree.
- To see the logs for jobs run on a particular Job Server, in the Administrator, select **Batch Jobs** > "**Repository**" (selecting the repository associated with the Job Server). Then, on the **Batch Jobs Status** page, click a log from the **Job Information** column for a job execution.

DESCRIPTION

A log records information about a particular execution of a single job.

- The **Log** tab in the Designer displays all logs for each execution. When you are finished with the logs for a given job or project, delete them from the **Log** tab. Right-click the log and select **Delete Log**.
- The **Job Information** column, of the Batch Job Status page in the Administrator also displays all logs for each execution.

There are three types of logs:

- Trace logs
- Statistics logs
- Error logs

Trace logs

The trace log shows progress of the execution through each component of the job. It lists the thread ID, the component of the job being executed, a description of the component, and the time each stage began.

For unsuccessful jobs, use the trace log to see which components of a partially executed job completed or where an error occurred.

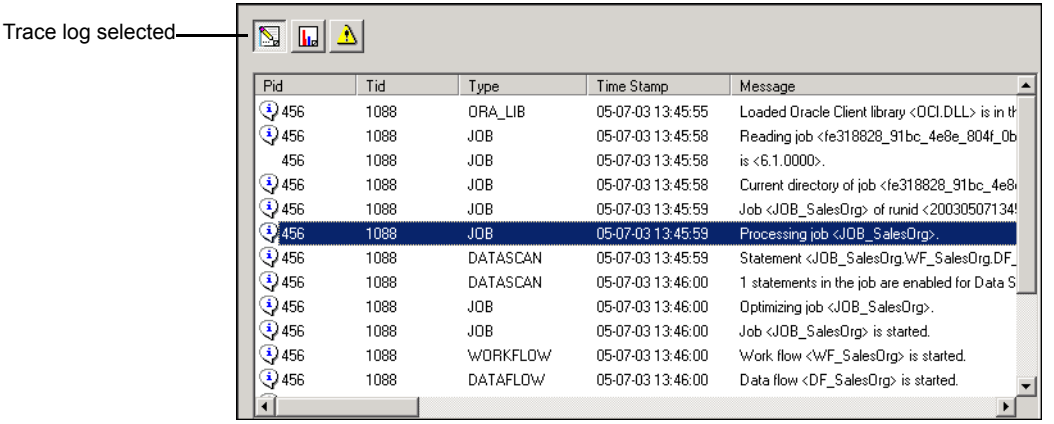
Trace logs have the following information:

Entry	Description
Pid	Indicates the process thread identification number of the thread executing.
Tid	Indicates the thread identification number of the thread executing.
Type	Indicates the object being executed, such as a data flow or a transform. The generic job events are labeled TRACE. Possible types are listed and described in the following table.
TimeStamp	Indicates the date and time when the thread generated the message.
Message	Gives a description of the event that occurred as the thread was executing.

There are several types of traces.

Error number prefix	Description
ABAP	Traces the ABAP query execution.
ADMIN	Prints administrative information like “server not responding” or “power failure.”
BLKLOAD	Traces bulk loading.
DATAFLOW	Traces the data flow execution.
EMAIL	Traces e-mail messages.
FTP	Traces FTP transport.
JOB	Traces the job execution.
OPTIMIZE	Records optimized details.
REPO	Traces objects in the repository.
ROW	Traces the row as it passes from one transform to another. It prints the row that is input to the transform and the output row it generates.
SQLFUNC	Traces function execution.
SQLLOAD	Traces loader execution, including the SQL sent to the target database.
SQLREAD	Traces reader execution, including the SQL sent to the source database.
SQLTRAN	Traces SQL transforms such as Table_Comparison and Key_Generation. The trace includes the SQL query sent to the underlying database and SQL results returned.
TRAN	Traces the transform execution.
USERFUNC	Traces user functions.

The following trace log (accessed in this case from the Output window of the Designer immediately after job execution), shows the status of the SalesOrgJob.



Statistics logs

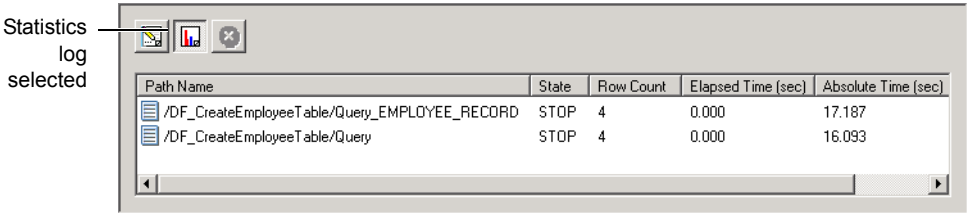
The statistics log quantifies the activities of the components of the job. It lists the time spent in a given component of a job and the number of data rows which streamed through the component.

Use the statistics log to help tune the performance of a job.

Statistics logs have the following information:

Entry	Description
Path Name	Indicates which object is executing.
State	Indicates the current status of the execution of the object. If you view the log while the job is running, this value changes as the status changes. The possible values are START, PROCEED, and STOP. In a successfully run job, all of these values are STOP to indicate that they finished successfully.
Row Count	Indicates the number of rows processed through this object. This value is updated based on the **Monitor sample rate (# of rows)** set as a debug property.
Elapsed Time	Indicates the time (in seconds) since this object received its first row of data.
Absolute Time	Indicates the time (in seconds) since the time execution of this data flow (including the current transform) began.

The following statistics log shows statistics for the DF_CreateEmployeeTable data flow.



Error logs

The **error log** lists errors generated by Data Integrator, by the source or target DBMS, or the operating system. If the error log is empty (that is, the button is dimmed), the job completed successfully.

Error logs have the following information:

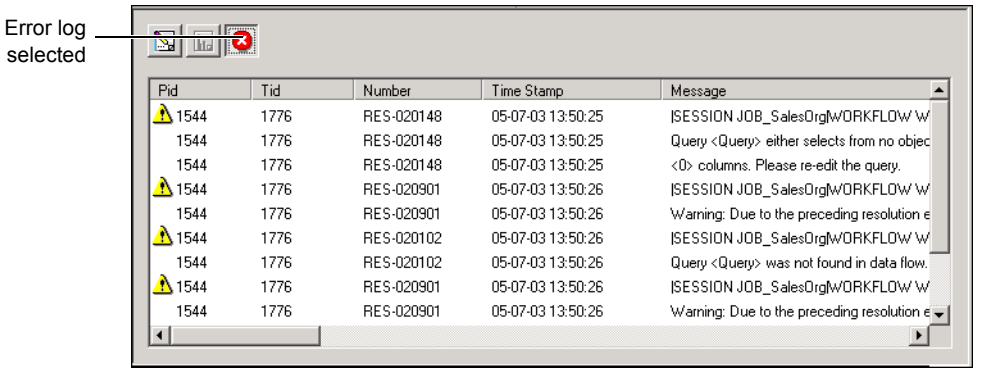
Entry	Description
Pid	The process thread identification number of the thread executing.
Tid	The thread identification number of the thread executing.
Number	An error number prefix (explained in the following table) and a number.
TimeStamp	The date and time when the thread generated the message.
Message	A description of the error that occurred as the thread was executing.

The error number prefixes are as follows:

Error number prefix	Description
ADM	Administration errors.
BAP	BAPI errors.
BIW	SAP BW errors.
CON	Connection errors. The connection indicated could not be initialized or failed during execution.
DBS	Database management system errors.

Error number prefix	Description
EML	Email errors.
FIL	Filespec errors.
OPT	Optimization errors.
PAR	Parser errors.
R3C	SAP R/3 connectivity errors.
R3S	SAP R/3 syntax errors.
REP	Repository errors.
RES	Resolver errors.
RUN	Runtime errors.
SCH	Job launcher errors.
SRV	Job Server errors.
SYS	System exceptions.
USR	User function errors.
VAL	Validator errors.
XRN	Transform errors.

The following error log shows error messages associated with the SalesOrgJob job.



Message function

CLASS

Reusable

ACCESS

In the object library, click the **Datastores** tab.

DESCRIPTION

Available in certain adapter datastores, message functions can accommodate XML messages when properly configured.

See your adapter's documentation for more specific information about the options available for a message function.

Outbound message

CLASS

Reusable

ACCESS

In the object library, click the **Datastores** tab.

DESCRIPTION

Available in some adapter datastores, outbound messages are XML-based, hierarchical communications that Data Integrator can publish to adapters. Outbound messages only wait for an acknowledgement from an external system; they do not wait for a reply. You can use outbound messages as targets only. You cannot use outbound messages as sources.

See your adapter's documentation for more specific information about the options available for outbound messages.

Project

CLASS

Single-use

ACCESS

- Choose **Project** > **New**.
- In the object library, click the **Projects** tab.

DESCRIPTION

A project allows you to group jobs. It is the highest level of organization offered by Data Integrator. Opening a project makes one group of jobs easily accessible in the user interface.

Projects have the following attribute:

Attribute	Description
Name	The name of the object. This name appears on the object in the project area.

Query transform

CLASS

Single-use

ACCESS

With a data flow diagram in the work space, click the Query transform icon in the tool palette, then click in the work space.

DESCRIPTION

A Query transform, like a SQL SELECT statement, retrieves a data set that satisfies the conditions you specify. With a Query transform, you can:

- Map columns from input to output schema
- Add new columns, nested schemas, and functions to the output schema
- Choose the data to extract
- Perform operations on the data
- Join data from multiple sources

For a complete description of the Query transform, see "Query" on page 287.

A Query transform can operate on nested data. Using a Query transform, you can nest data or unnest nested data. See Chapter 9, "Nested Data," in the *Data Integrator Designer Guide* for more information about using the Query transform with nested data.

Query transforms have one attribute:

Attribute	Description
Name	The name of the object. This name appears on the object in the diagram.

Real-time job

CLASS

Reusable

ACCESS

- In the object library, click the **Jobs** tab.
- In the project area, right-click a project and select **Real-time Job**.

DESCRIPTION

NOTE: For information specific to SAP R/3, see *Data Integrator Supplement for SAP*.

A real-time job is a set of objects that you can execute together to process messages.

A real-time job is made up of three logical components:

- Initialization (optional)
- Real-time processing loop
- Clean-up (optional)

Each component can include the same objects as a batch job.

A real-time job is created in the Designer and then configured in the Administrator as a real-time service associated with an Access Server.

Start real-time services in the Administrator. If you have included any objects in the initialization component of a a real-time job, they run when the service starts. When a real-time service starts, a real-time processing loop registers itself and its message type with the Access Server and waits for the Access Server to send

requests. The real-time processing loop continues to run until it encounters an error or you shut it down using the Data Integrator Administrator. The objects you placed inside the clean-up component of a real-time job run only when a service is shut down.

The message type that a given real-time job processes is determined (when it is designed) by the real-time source you include in the real-time processing loop; the format of the response is determined by the real-time target you include.

A real-time job has the same built-in attributes as a batch job:

Attribute	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object.
Description	Your description of the job.
Date created	The date when the object was created.

Like batch jobs, real-time jobs use the debug and trace properties to determine what information Data Integrator collects and logs when running the job. However, real-time jobs do not support the Enable Recovery debug options. For more information about Debug and Trace properties, see "Log" on page 75.

Content of real-time job

A real-time processing loop can contain the following objects:

- A single data flow which can contain:
  - A single real-time source — XML message (required)
  - Sources — Files, XML files, and tables, including R/3 tables
  - A single real-time target — XML message (required)
  - Targets — Files, XML files, tables, and template tables
  - Transforms, including queries

- Multiple data flows which can contain:
  - A single real-time source in the first data flow — XML message (required)
  - Sources — Files, XML files, and tables, including R/3 tables
  - A single real-time target in the last data flow — XML message (required)
  - Targets — Files, XML files, tables, and template tables
  - Transforms, including queries
  - Memory tables — used to as staging tables to move data sets to the next data flow in the job.
- Multiple work flows, scripts, conditionals, while loops, etc.

Real-time jobs can also be built using IDocs. See the SAP R/3 Guide for more information.

Arranging Metadata

If you delete an object used in a real-time job from the object library, calls to the object are replaced with an icon indicating that the calls are no longer valid.

You can insert any number of work flows and data flows into a real-time job as long as the data flow models for a real-time processing loop are followed.

- A single data flow in a loop — Must have both a message source and target
- Multiple data flows in a loop — Must have a message source in the first data flow and a message target in the last data flow

At the job level, work flows and data flows cannot be designed to run in parallel. Inside a job level work flow, they can.

The messages in a data flow are significant to Data Integrator. The following table indicates how data flows can be used.

If a data flow has:	It can be used as the:
One XML message source	First data flow in a real-time processing loop.
One XML message target	Last data flow in a real-time processing loop.
One XML message source and one XML message target.	Only data flow in a real-time processing loop.
No message source or target	Not the first, last, or only data flow in a real-time processing loop. Data flow in a batch job.

Data flows that do not contain messages can be used in batch jobs or in real-time processing loops (between its first and last data flows).

NOTE: The sources and targets in a real-time job cannot reference a LiveLoad-enabled datastore.

MESSAGE PROCESSING

Unlike batch jobs, real-time jobs are designed to process multiple messages rather than just files or tables of data.

For transforms that require all of the message's data at one time, such as queries that include aggregation functions, data is cashed temporarily. The transform performs the specified operation and then clears caches in preparation for the next message.

To test a real-time job using the Designer, the recommended procedure is to test one message and create a target test file to receive the data. A real-time job will clear data after processing each message if system defaults are used. Therefore, deselect:

- **Delete data from table before loading** for a table target
- **Delete file** for a flat file target
- **Delete and recreate file** for an XML target

Loading targets as a single transaction

In a real-time job, you can load more than one table from a single datastore in a single transaction. When transactional loading is turned on for table targets, Data Integrator sends `INSERT` statements for any of the tables included in the transaction to the database to process. You can also control which tables are loaded first by specifying the transaction order for the tables.

If the data flow includes a real-time target, it is always loaded in parallel with other targets to ensure load time is as short as possible.

Starting and stopping real-time Services

For development and testing, you can manually execute a real-time job from the Designer. The Designer runs a real-time job in test mode.

When testing a real-time service or when running in production, the Access Server triggers the Job Server to process a request using the logic you built inside a real-time processing loop. The Access Server can also trigger the Job Server to shut down real-time processing loops. In a production environment, you control the operation of real-time services using the Data Integrator Administrator.

Script

CLASS

Single-use

ACCESS

With a work flow or job diagram in the workspace, click the script icon in the tool palette.

DESCRIPTION

A script is a single-use object that assigns values to local, global or environment variables in a job or work flow. Define the script using the Data Integrator scripting language. See Chapter 7, "Data Integrator Scripting Language".

Scripts have the following attribute:

Attribute	Description
Name	The name of the object. This name appears on the object in the diagram.

Source

Document

File

Table

Template table

XML file

XML message

CLASS

Single-use

ACCESS

- To insert a document, table, or template table as a source, open the object library, go to the **Datastores** tab, select the object, drag it into the workspace, and select **Make Source**.
- To insert a flat file as a source, open the object library, go to the **Formats** tab, select the file format template for the file, drag it into the workspace, and select **Make Source**. Use the file format editor to specify the file's location.
- To insert an XML message or file as a source, open the object library, go to the **Formats** tab, select the an XML Schema or DTD format template, drag it into the workspace, and select **Make XML message source** or **Make XML file source**. Use the source editor to specify the test file name for the message or the source XML name for the file.
- To view options of a particular source, click the name of the source in the workspace or in the project area. This opens the appropriate editor, such as the table editor, or the XML file, XML message, or flat file format editors.

DESCRIPTION

A source is an object from which Data Integrator reads data.

In a batch job, a source can be a document, a file, a table, a previously defined template table, an XML file, or a source-specific data flow (see your source-specific supplement for more options).

In a real-time job, a source can be a table, a previously defined template table, a flat file, an XML message, or an XML file. Each real-time job must have exactly one real-time data source.

You can make a source an embedded data flow. For more information, see Chapter 11, "Embedded Data Flows," in the *Data Integrator Designer Guide*.

This section describes the options available for different sources:

- Table source
- Flat file source
- XML file source
- XML message source

Options available for sources from adapter datastores depend on the adapter implementation. Thus, options vary by data source and adapter version. See your adapter documentation for more information.

Table source

You can tune performance using table options:

Option	Description
Make Port	Makes the source table an embedded data flow port. For more information, see Chapter 11, "Embedded Data Flows," in the *Data Integrator Designer Guide*.
Enable partitioning	Enables Data Integrator to use the partition information in this table. If this option is selected, Data Integrator reads table data using the number of partitions in the table as the maximum number of parallel instances. For more information, see Chapter 5, "Using Parallel Execution," in the *Data Integrator Performance Optimization Guide*.
Join rank	Indicates the rank of the source relative to other tables and files in the data flow when creating a join. Data Integrator joins sources with higher join ranks before joining sources with lower join ranks. For more information, see "Join ordering" on page 24 of the *Data Integrator Performance Optimization Guide*. **NOTE:** Must be a non-negative integer. When set to its default value (zero), Data Integrator determines join order.
Cache	Indicates whether Data Integrator should read the required data from the source and load it into memory. Because an inner source of a join must be read for each row of an outer source, you might want to cache a source when it is used as an inner source in a join and when the data retrieved will fit into available memory. There are two options: • **Yes**: The source is always cached unless it is the outer-most source in a join. • **No**: The source is never cached. For tables, the default option is **Yes** in a batch job and **No** in a real-time job. For files, the default option is **No**.
Array fetch size	Indicates the number of rows retrieved in a single request to a source database. The default value is 1000. Higher numbers reduce requests, lowering network traffic, and possibly improve performance. This option is available for source tables from DB2, Detail DB2, Detail NRDB, Informix, ODBC, and Oracle datastores. When retrieving a column with an Oracle LONG data type, Data Integrator automatically sets **Array Fetch Size** to 1. If a column has an Oracle LONG data type, Data Integrator can only retrieve one row at a time.

For Oracle source tables you can use an overflow file for error handling. Select **Use overflow file** and enter a name for the file. Errors that occur while reading data are logged into the overflow file and the job execution proceeds while ignoring the rows that cause the error. To set the location of the overflow file directory use the table's Datastore Editor.

In addition to the options shown above, if the table uses a CDC datastore, click the **CDC Options** tab and complete the following to set Oracle and mainframe changed-data capture options:

Option	Description
CDC subscription name	The name used to mark sets of changed data read from the continuously growing CDC table.
Enable check-point	Enables Data Integrator to restrict CDC reads using check-points. Once a check-point is placed, the next time the CDC job runs, it reads only the rows inserted into the CDC table since the last check-point. For more information, see "Using check-points" on page 497.
Get before-image for each update row	Some databases allow two images to be associated with an UPDATE row: a before-image and an after-image. If your source can use before-images and you want to read them during change-data capture jobs, enable this option. By default, only after-images are retrieved. For more information, see "Using before-images" on page 497.

Flat file source

A flat file source has the **Join rank** and **Cache** options in addition to the file format options. For these two options, Data Integrator uses the same interpretation for both files and tables (see "Table source" on page 94). See "File format" on page 62 for a detailed description of flat file format options.

XML file source

An XML file source has the same **Join rank** and **Make port** options as tables (see "Table source" on page 94).

An XML file source has the following options in addition to its read-only XML Schema or DTD format information:

Option	Description
XML file	The location relative to the Job Server of an XML-formatted file to use as the source.
Enable validation	A check box to turn on the comparison of the incoming data to the stored XML Schema or DTD format. When this option is enabled, the data flow throws an exception if the incoming source is not valid.

XML message source

An XML message source has the same **Make port** option as tables (see "Table source" on page 94).

The XML message source has these options in addition to its read-only XML Schema or DTD format information:

Option	Description
XML test file	The location relative to the Job Server of an XML-formatted file to use as the message source when you execute the job in test mode.
Enable validation	A check box to turn on the comparison of the incoming message to the stored XML Schema or DTD format. When this option is selected, the real-time job throws an exception if the incoming message is not valid. When you are developing a real-time job, this validation helps you to make sure sample data is both valid and well-formed. If you select this option in production, make sure to include appropriate error handling either in the Data Integrator job or the client application to process an error caused if a data flow in the real-time job receives data that does not validate against the imported format.
implied: Join rank	The XML message source is always the outer table in a join. You cannot change its join rank. This option is implied and does not appear in the editor.

Table

CLASS

Reusable

ACCESS

In the object library, click the **Datastores** tab. Expand a datastore to find the tables node. Expand this node to view the list of imported tables. Right-click and select **Properties** to view and edit table properties.

DESCRIPTION

You can use a table as a source or target in a data flow.

The **Indexes** tab on the Properties window for a table shows information about the table's indices. Under **Index**, the window lists the primary index followed by any secondary index. Select an index and the window lists the columns in that index under **Column**.

The **Partition** tab on the Properties window displays how table metadata is partitioned. Partitions can be imported with a table or you can create metadata for them within Data Integrator. For more information, see Chapter 5, "Using Parallel Execution," in the *Data Integrator Performance Optimization Guide*.

The **Attributes** tab on the Properties window displays built-in table attributes:

Table Attribute	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object.
Description	A configurable description field.
Business_Name	Supports metadata exchanged between Data Integrator and BusinessObjects through the Universal Metadata Bridge (UMB) 1.1. A configurable field for the logical Business_Name used in BusinessObjects. Data Integrator can extract, transform and load data while transferring this information intact.
Business_Description	Supports metadata exchanged between Data Integrator and BusinessObjects through the Universal Metadata Bridge (UMB) 1.1. A configurable, business-level description of the table. Data Integrator transfers this information separately and adds it to any BusinessObjects Class description.
Table_Usage	A configurable label field. Use it to mark a table as `fact` or `dimension` for example.
Total_Number_Of_ Rows_Processed	The number of rows loaded into the table in the last successful load.
Date_last_loaded	The time the table was last successfully loaded by Data Integrator.
Number_Of_Rows_ Rejected	The number of rows rejected in the last successful load.
Number_Of_Inserts	The number of rows inserted in the last successful load.
Number_Of_Updates	The number of rows updated in the last successful load.
Date_Created	The date that the object was created.
Estimated_Row_ Count	A configurable estimate of the table size used in calculating the order in which tables are read to perform join operations. Used for SAP R/3 tables only.

Table Attribute	Description
Number_Of_Deletes	The number of rows deleted in the last successful load.
Elapsed_Time_For_Load	The time it took to load this table in the last successful load.
Table_Type	The type of datastore object for tables and hierarchies. Most often the value `TABLE` is displayed. However, Data Integrator might display the following values for SAP R/3 sources: BW master data transfer, BW transaction data transfer, BW hierarchy data transfer, SAP hierarchy.
SAP_Table_Class_Name	Imported with SAP R/3 table metadata.
Loader_Is_Template_Table	If `YES`, indicates that the table is an internal, template table created in Data Integrator. Before running production jobs, execute the job to load the target table then right-click the template table in the object library or in a data flow and select **Import Table.** Data Integrator creates the table in your database and imports it. For more information, see "Template table" on page 136.
SavedAfterCheckOut	If `YES`, indicates you saved the table after it was checked out of the central repository. Data Integrator uses this information to determine whether to save the table in the central repository when it is checked in.
PartitionModified	If `YES`, indicates that you modified the partitions in this table using Data Integrator after you imported the table's metadata.

COLUMN ATTRIBUTES FOR TABLES

Data Integrator also supports column attributes for tables.

➢ To view column attributes for a table

1. In the **Datastores** tab of the object library, double-click a table.

   The Table Metadata window opens.

2. Right-click a column name and select **Properties.**

   The Column Attributes window opens.

3. Click the **Attributes** tab.

Column attributes for tables

Column Attribute	Description
Business Name	A configurable logical name used by BusinessObjects. For more information see, "Attributes that support metadata exchange" on page 413 of the *Data Integrator Designer Guide*.
Business_ Description	A configurable business-level description of the column. For more information see, "Attributes that support metadata exchange" on page 413 of the *Data Integrator Designer Guide*.
Column_Usage	Supports metadata exchanged between Data Integrator and BusinessObjects through the Universal Metadata Bridge (UMB) 1.1. Enter `Dimension`, `Measure`, or `Detail` as a value for a corresponding object in BusinessObjects. • If set to `Dimension`, the corresponding object created in BusinessObjects is of qualification type **Dimension**. • If set to `Measure`, the corresponding object is of qualification type **Measure**. • If set to `Detail`, the corresponding object is of qualification type **Detail** and requires you to set a value for **Associated_Dimension**.
Associated_ Dimension	Set this value only if **Column_Usage** is set to `Detail`. The value must be in the format: `table.column`. The Detail column is created under the Dimension column you specify.
Acta_autojoin	Generated by Data Integrator. Not configurable.
Associated_ domain	Use for databases that use domains such as PeopleSoft.
Physical_Name	Use for applications that allow logical names for a column such as Oracle Applications.

The attributes listed above are available for all tables.

- For descriptions of column attributes that support DTDs see "Column attributes supported for DTDs" on page 53.
- For descriptions of column attributes that support XML Schemas see "Column attributes supported for XML Schemas" on page 155.

Target

Document

File

Outbound message

Table

Template table

XML file

XML message

XML template

CLASS

Single-use

ACCESS

- To display target options, click the name of the target in the workspace or in the project area. This opens the object editor.
- To display target properties, right-click a target and choose **Properties**.

DESCRIPTION

A target is an object in which Data Integrator loads extracted and transformed data in a data flow.

In a data flow, a target can be a document, a flat file, an outbound message, a table or template table, an XML file, XML message, or an XML template.

You can make a target an embedded data flow port: set the **Make port** option to **Yes** for flat files; select the **Make port** check box for other targets. For more information, see Chapter 11, "Embedded Data Flows," in the *Data Integrator Designer Guide*.

This section describes:

- Target files
- Target tables
- Target XML files, messages, and templates

Documents and outbound messages are only available from adapter datastores. Options available for these targets depend on the adapter implementation. Thus, options vary by data source and adapter version. See your adapter documentation for more information

Target files

You can use any flat file format as a target in a data flow. To add a target file, select a file format in the object library, drag the file format into the data flow workspace, and select **Make Target**.

If the schema defined in the file format does not match the schema that is input to the target, Data Integrator provides validation errors to identify the mismatch.

Use the file format editor in target mode to edit the file format of a target file. (See "File format" on page 62 for more information and a complete list of properties.) You cannot edit all properties of a particular target file. You can change some properties of the file format.

You can also change the name of the target file object using the object's properties. Right-click the object and choose **Properties**.

Target tables

You can add a table to a data flow diagram as a target if Data Integrator can write to the application or database containing the table. To add a target table, select the table in the object library, drag the table into the workspace, and select **Make Target**.

If the schema defined in the table does not match the schema that is input to the target, Data Integrator provides validation errors to identify the mismatch.

When loading DB2, ODBC, or Oracle tables, Data Integrator parameterizes the SQL. Parameterized SQL statements result in quicker load times. To parameterize SQL, Data Integrator must be able to generate, parse, and compile the statement. For example, Data Integrator is unable to parameterize SQL when the auto-correct option, transactional loading, or triggers are used.

You configure a target table by setting options in the target editor. Available options depend on the database in which the table is defined. This section describes:

- Target table options available in all databases
- DB2 target table options
- Informix target table options
- Microsoft SQL Server target table options
- Oracle target table options
- Sybase target table options
- Teradata target table options

Target table options available in all databases

Tab	Option	Description
Target		
	Make port	Select this check box to make the target table an embedded data flow port. For more information, see Chapter 11, "Embedded Data Flows," in the *Data Integrator Designer Guide*.
Options		
	Rows per commit	Specifies the transaction size in number of rows. If set to 1000, Data Integrator sends a commit to the underlying database every 1000 rows. Load triggers are never split across transaction boundaries, so if the load trigger crosses transaction boundaries, then the size of the transaction is automatically extended to accommodate the entire trigger requirement. For example, suppose you set a Rows per commit to 3 and specified an insert trigger, where an incoming insert statement is converted into 5 statements. Rows per commit would automatically be extended to 5 to accommodate each insert trigger statement in a single transaction. ODBC datastores are always set to 1 regardless of what is entered in this field. This option is not available for targets in real time jobs.

Target table options available in all databases *(Continued)*

Tab	Option	Description
	Column comparison	Specifies how the input columns are mapped to output columns. There are two options: • **compare_by_position** — Data Integrator disregards the column names and maps source columns to target columns by position • **compare_by_name** — Data Integrator maps source columns to target columns by name Validation errors occur if the data types of the columns do not match.
	Delete data from table before loading	For batch jobs, sends a TRUNCATE statement to clear the contents of the table before loading. Defaults to not selected. For real-time jobs, clears data after processing each message. For real-time jobs you may want to deselect this option during development and testing. For more information, see "Message processing" on page 89.
	Enable Partitioning	(Displayed only if the target table data is partitioned) Loads data using the number of partitions in the table as the maximum number of parallel instances. You can only select one of the following loader options: • Number of Loaders • Enable Partitioning • Transactional Loading NOTE: If you select both **Enable Partitioning** and **Include in transaction**, the **Include in transaction** setting overrides the **Enable Partitioning** option. For example, if your job is designed to load to a partitioned table, but you select **Include in transaction** and enter a value for **Transaction order**, when the job is executed, Data Integrator will include the table in a transaction load and does not parallel load to the partitioned table.
	LiveLoad Cache	(Displayed only for tables from datastores that are LiveLoad activated.) This option enables the creation of reconciliation files for this target table. See "Target table options" on page 22 of the *Data Integrator LiveLoad User's Guide*.

Target table options available in all databases *(Continued)*

Tab	Option	Description
	Number of loaders	Loading with one loader is known as "single loader loading." Loading when the number of loaders is greater than one is known as "parallel loading." The default number of loaders is 1. The maximum number of loaders is 5. When parallel loading, each loader receives the number of rows indicated in the Rows per commit option, in turn, and applies the rows in parallel with the other loaders. For example, if you choose a Rows per commit of 1000 and set the number of loaders to 3, the first 1000 rows are sent to the first loader. The second 1000 rows are sent to the second loader, the third 1000 rows to the third loader, and the next 1000 rows back to the first loader.
	Use overflow file	This option is used for recovery purposes. If a row cannot be loaded it is written to a file. When this option is selected, options are enabled for the file name and file format. The overflow format can include the data rejected and the operation being performed (write_data) or the SQL command used to produce the rejected operation (write_sql).
	Ignore columns with value	Enter a value that might appear in a source column and that you do not want updated in the target table. When this value appears in the source column, the corresponding target column is not updated during auto correct loading. You can enter spaces.
	Ignore columns with null	Select this check box if you do not want NULL source columns updated in the target table during auto correct loading. This option is only available when you select the **Auto correct load** check box.
	Use input keys	If the target table contains no primary key, this option enables Data Integrator to use the primary keys from the input.

Target table options available in all databases *(Continued)*

Tab	Option	Description
	Auto correct load	Select this check box to use auto correct loading. Auto correct loading ensures that the same row is not duplicated in a target table. This is particularly useful for data recovery operations. **NOTE**: This option is not available for targets in real time jobs or target tables that contain LONG column(s). When this check box is selected, Data Integrator: **1.** Reads a row from the source. **2.** Checks if a row exists in the target table with the same values in the primary key. If the **Use input keys** check box is selected, Data Integrator uses the primary key of the source table. Otherwise, Data Integrator uses the primary key of the target table; if the target table has no primary key, Data Integrator considers the primary key to be all the columns in the target. • If a matching row does not exist, inserts the new row regardless of other options. • If a matching row exists, updates the row depending on the values of **Ignore columns with value** and **Ignore columns with null**: • When the column data from the source matches the value in **Ignore columns with value**, the corresponding column in the target table is not updated. The value may be spaces. Otherwise, the corresponding column in the target is updated with the source data. • When the **Ignore columns with null** check box is selected and the column data from the source is NULL, then the corresponding column in the target table is not updated. Otherwise, the corresponding target column is updated as NULL since the source column is NULL. Data Integrator can optimize data flows such that an Oracle source database completes the auto correct load operation. When all other operations in the data flow can be pushed down to the source database, the auto-correct loading operation is also pushed down. The generated SQL implements the **Ignore columns with value** and **Ignore columns with null** options.

Target table options available in all databases *(Continued)*

Tab	Option	Description
	Include in transaction	Indicates that this target is included in the transaction processed by a batch or real-time job. This option allows you to commit data to multiple tables as part of the same transaction. If loading fails for any one of the tables, no data is committed to any of the tables. Transactional loading can require rows to be buffered to ensure the correct load order. If the data being buffered is larger than the virtual memory available, Data Integrator reports a memory error. The tables must be from the same datastore. If you choose to enable transactional loading, other options are not available: • Rows per commit • Use overflow file, and overflow file specification • Number of loaders • Enable partitioning • Delete data from table before loading • Bulk loader options • Pre load commands • Post load commands Data Integrator also does not parameterize SQL or push operations to the database if transactional loading is enabled.
	Transaction order	Transaction order indicates where this table falls in the loading order of the tables being loaded. By default, there is no ordering. All loaders have a transaction order of zero. If you specify orders among the tables, the loading operations are applied according to the order. Tables with the same transaction order are loaded together. Tables with a transaction order of zero are loaded at the discretion of the data flow process.

Target table options available in all databases *(Continued)*

Tab	Option	Description

Load triggers		Specifies SQL commands performed by the database on an INSERT, UPDATE, or DELETE operation.
		You can specify a *load trigger* (a template SQL statement) that has placeholders for column and variable values. Data Integrator sets the placeholders at execution time based on the fields in the transform's input schema. For each row, the template is filled out and applied against the target.
		The special operations you specify in a load trigger can occur before, after, or instead of normal operations.
		Use load triggers in situations such as archiving updates to a warehouse or incremental updates of aggregation value.
		DATA INTEGRATOR does not parse load triggers. Thus, when you specify a load trigger, Data Integrator does not parameterize SQL statements. As a result, load times might be higher when you use load triggers.
		Data Integrator does not validate load triggers.
		NOTE: If you use an override, you cannot specify auto correct load.
		For example, instead of applying an insert of a new sales order rows, you use a load trigger that applies inserts and updates of aggregated values of sales_per_customer and sales_per_region.
		The templates give you a row with customer_id, order_amount, region_id, and so forth.
		The INSERT and UPDATE statements are:
		`INSERT into order_fact`
		`values ([customer_id], [order_amount]);`
		`UPDATE region_fact`
		`SET order_amount =`
		`order_amount + [order_amount]`
		`WHERE region_id = [region_id];`
		Enter your load triggers manually or drag column names from the input schema. Column names need to be enclosed in curly braces or square brackets. For example, `{SalesOffice}` or `[SalesOffice]`.
		With curly braces, Data Integrator encloses the value in quotation marks, if needed. With square brackets, it will not. To avoid unintended results, use curly braces for varchar or char column names.

Target table options available in all databases *(Continued)*

Tab	Option	Description

		If you insert column names into the SQL statement by dragging the column names, Data Integrator inserts square brackets for you. If you need curly braces, you must change the square brackets to curly braces.
		`[#insert], [#update],` and `[#delete]` represent the default operations.
		To delimit a SQL statement, use `[#new]`. For example:
		`[#insert] [#new]`
		`insert into foo values ([col1], {col2}, ...)`
		For UPDATE operations you must specify both the "before" and the "after" image values. You can specify both images for INSERT and DELETE operations, also, but it is not required.
		To specify "before" images, add the suffix `.before` to the column name. To specify "after" images, add the suffix `.after` to the column name.
		The default suffix for UPDATE and INSERT operations is `.after`. The default suffix for DELETE operations is `.before`.
		You can include variables in the SQL statements, but not expressions.
		You can map a batch of SQL statements. Each SQL statement is separated by a new separator (`[#new]`).
		The following statement is an example for mapping insert SQL:
		`INSERT into log_table values ({col1}, {col2})`
		`[#new]`
		`[#insert] [#new]`
		`delete from alt_junk where . . .`

Target table options available in all databases *(Continued)*

Tab	Option	Description
Pre Load Commands Post Load Commands		Specify SQL commands that Data Integrator executes before starting a load or after finishing a load. When a data flow is called, Data Integrator opens all the objects (queries, transforms, sources, and targets) in the data flow. Next, Data Integrator runs the target's preload script. Therefore, Data Integrator executes any preload SQL commands before processing any transform. **NOTE:** Because Data Integrator executes the SQL commands as a unit of transaction, you should not include any transaction commands in preload or postload SQL statements. Both the **Pre Load Commands** tab and the **Post Load Commands** tab contain a **SQL Commands** box and a **Value** box. The **SQL Commands** box contains command lines. When you first open the tab, an empty line appears. To edit a line, select the line in the **SQL Commands** box. The text for the SQL command appears in the **Value** box. Edit the text in that box. To add a new line, determine the desired position for the new line, select the existing line immediately before or after the desired position, right-click, and choose **Insert Before** to insert a new line before the selected line, or choose **Insert After** to insert a new line after the selected line. Finally, type the SQL command in the **Value** box. To delete a line, select the line in the **SQL Commands** box, right click, and choose **Delete**.

Target table options available in all databases *(Continued)*

Tab	Option	Description
		You can include variables and parameters in preload or postload SQL statements. Put the variables and parameters in either brackets, braces, or quotes. Data Integrator translates each statement differently, writing a statement that depends on the variable or parameter type.

Entered statement	Variable value	Written statement
[$X]	5	5
[$X]	John Smith	John Smith
{$X}	5	5
{$X}	John Smith	'John Smith'
'$X'	5	'$X'
'$X'	John Smith	'John Smith'

You cannot use Pre Load and Post Load SQL commands in a real-time job.

DB2 target table options

Tab	Option	Description
Target		This tab lists the table name and other read-only values from the associated datastore. Also contains the **Make port** option to specify the table as an embedded data flow port (see "Make port" on page 105).
Options		
	Rows per commit	See "Rows per commit" on page 105.
	Column comparison	See "Column comparison" on page 106.
	Delete data from table before loading	See "Delete data from table before loading" on page 106.
	Drop and re-create table	See "Delete data from table before loading" on page 106
	Number of Loaders	See "Number of loaders" on page 107.
	Use overflow file	See "Use overflow file" on page 107.
	Overflow file name	The full path is determined by the datastore overflow directory setting. See "Use overflow file" on page 107.
	Overflow file format	See "Use overflow file" on page 107.
	Ignore columns with value	See "Ignore columns with value" on page 107
	Ignore columns with NULL	See "Ignore columns with null" on page 107
	Use input keys	See "Use input keys" on page 107.
	Auto correct load	See "Auto correct load" on page 108.
	Include in transaction	See "Include in transaction" on page 109.
	Transaction order	See "Include in transaction" on page 109.
Bulk loader options		
	Bulk load	Indicate the bulk load method. There are three options: • **no** — Data Integrator will not bulk load data. • **import** — Data Integrator will use the DB2 import utility to bulk load data.The import utility uses a SQL INSERT statement to write data from an input file into a table or view. • **load** — Data Integrator will use the DB2 load utility. The load utility improves performance by writing data directly into the data file.

DB2 target table options *(Continued)*

Tab	Option	Description
	Generate files only	Select this check box to have Data Integrator generate a data and command file. Rather than loading data into the target shown in the data flow, Data Integrator generates a command file and a data file that you can later load using DB2 bulk loading. This option is useful when the DB2 server is located on a system running a different operating system than the Data Integrator Job Server. DATA INTEGRATOR writes the data and command files in the bulk loader directory specified in the datastore definition. If you have not specified a bulk loader directory, Data Integrator writes the files in the `\$LINK_DIR\log\`*`Job Server name\repository name`*`\bulkloader` directory. To load the data, you must manually copy the files to the remote system and start the bulk load execution. When you select this check box, only the **Text delimiter** and **Column delimiter** options are available.
	Clean up bulk loader directory after load	Select this check box to delete all bulk load-oriented files after the load is complete. When this option is selected, Data Integrator deletes the following files after each bulk load unless an error has occurred: • Control file • Log file • Bad file

DB2 target table options *(Continued)*

Tab	Option	Description
	Mode	Specify the mode for loading data in the target table. Available modes depend on the bulk load method. Available modes for input method (**Bulk load** is set to **input**): • **insert** — Adds new records to table. Use when loading data into an empty table or when appending data to an existing table that contains data you want to maintain. • **insert-update** — If a record with matching primary keys exists in the table, updates that record; otherwise adds new record to table. This method requires that the target table has primary keys. • **replace** — Deletes all existing records in table, then adds new records. • **truncate** — Deletes all existing records in table, then adds new records. Available modes for load method (**Bulk load** is set to **load**): • **insert** — Appends the new records into the target table. • **replace** — Deletes the existing records, then inserts the loaded data.
	Rows per commit	Enter the number of rows that will be loaded before a commit takes place. If no value is entered, the load utility uses the default value at run time. This option is available only for the input method (**Bulk load** is set to **input**).
	Save count	Enter the minimum number of rows loaded before the load utility establishes a consistency point. This is the point when DB2 saves the data. DB2 converts this value to a page count, and rounds up to intervals of the extent size. If you enter zero, the load utility establishes no consistency points. This option is available only for the load method (**Bulk load** is set to **load**).
	Text delimiter	Enter a single character string delimiter. The default value is a double quote ("). The specified character is used in place of double quotation marks to enclose a character string. The specified character can be any printable or non-printable ASCII character, escaped with a double slash "\\".
	Column delimiter	Enter a single-character column delimiter. The default value is a comma (,).

DB2 target table options *(Continued)*

Tab	Option	Description
	Warning row count	Enter the number of warnings allowed for each load operation. This option is available only for the load method (**Bulk load** is set to **load**).
	Exception table name	Enter the table into which the DB2 server loads rows that violate a table constraint. Rows that violate constraints are deleted from the target table and inserted into the exception table. This option is available only for the load method (**Bulk load** is set to **load**).
	Recoverable	Select this check box to support data recovery through the DB2 roll-forward recovery feature. When this option is not selected, you cannot recover from failure using DB2 roll-forward. When this option is selected, DB2 writes a backup copy of the loaded data. You can use DB2 roll-forward recovery after failure. You must specify the directory for writing the backup file (**Copy target directory**). Only select this option if your DB2 target database is roll-forward enabled. This option is available only for the load method (**Bulk load** is set to **load**).
	Copy target directory	Enter the directory where copy files are stored when both the database forward log recovery and the **Recoverable** option are enabled. Data Integrator supports local media only This option is available only for the load method (**Bulk load** is set to **load**).
	Data file on client machine	Select this check box to have the load utility process the local file directly rather than using FTP to send the data file to the DB2 server. To use this option: • You must use DB2 version 7.x or later. • The target DB2 cannot be a DB2 enterprise (extended edition environment). • The target table and database must not be partitioned. This option is only applicable if Data Integrator and DB2 are on different servers. This option is available only for the load method (**Bulk load** is set to **load**).
Load Triggers		See "Load triggers" on page 110.
Pre Load Commands		See "Pre Load Commands" on page 112.

DB2 target table options *(Continued)*

Tab	Option	Description
Post Load Commands		See "Post Load Commands" on page 112.

Informix target table options

Tab	Option	Description
Target		This tab lists the table name and other read-only values from the associated datastore. Also contains the **Make port** option to specify the table as an embedded data flow port (see "Make port" on page 105).
Options		
	Rows per commit	See "Rows per commit" on page 105.
	Column comparison	See "Column comparison" on page 106.
	Delete data from table before loading	See "Delete data from table before loading" on page 106.
	Drop and re-create table	Drops the existing table and creates a new one with the same name before loading.
	Number Of Loaders	See "Number of loaders" on page 107.
	Use overflow file	See "Use overflow file" on page 107.
	Overflow file name	The full path is determined by the datastore overflow directory setting. See "Use overflow file" on page 107.
	Overflow file format	See "Use overflow file" on page 107.
	Ignore columns with value	See "Ignore columns with value" on page 107
	Ignore columns with NULL	See "Ignore columns with null" on page 107
	Use input keys	See "Use input keys" on page 107.
	Auto correct load	See "Auto correct load" on page 108.
	Include in transaction	See "Include in transaction" on page 109.
	Transaction order	See "Include in transaction" on page 109.

Informix target table options *(Continued)*

Tab	Option	Description
Bulk Loader Options		
	Bulk load	Select this check box to use Informix bulk loading options to write the data.
	Generate files only	Select this check box to have Data Integrator generate a data and control file. Rather than loading data into the target shown in the data flow, Data Integrator generates a control file and a data file that you can later load using Informix bulk loading. This option is often useful when the Informix server is located on a system running a different operating system than the Data Integrator Job Server. Data Integrator writes the data and control files in the bulk loader directory specified in the datastore definition. If you have not specified a bulk loader directory, Data Integrator writes the files in the `\$LINK_DIR\log\`*`Job Server name`*`\`*`repository name`*`\bulkloader` directory. You need to copy the files to the remote system manually. Data Integrator names these files *`tablename`*`.ctl` and *`tablename`*`.dat`, where *`tablename`* is the name of the target table.
	Lock table	Select this check box to lock the table for the duration of the load.
	Clean up bulk loader directory after load	Select this check box to delete all bulk load-oriented files after the load is complete, unless an error occurs. When you select this option, Data Integrator deletes these files after a successful bulk load: • Control file • Log file • Bad file
	Mode	Select the mode for loading data in the target table: • **append** — Adds new records to table. • **replace** — Deletes all existing records in table then adds new records.
	Bulk loader server name	Enter the name of the Informix database server.
	Bulk loader database name	Enter the name of the target information warehouse database.
	Rows per commit	Enter the number of rows that must be loaded before a commit takes place.

Informix target table options *(Continued)*

Tab	Option	Description
	Field delimiter	Enter the character that separates columns. Make sure the character you designate is not used in any of the data columns.
	Maximum rejects	Enter the maximum number of acceptable warnings. Bulk load stops after this many warnings. Set this parameter when you expect no warnings, but want to verify that the correct file and table are used. If you enter 0, or do not specify a value, the load continues regardless of the number of warnings issued. The default value is 10.
	Begin/end column character	Enter the character that designates the beginning or ending of the column.
Load Triggers		See "Load triggers" on page 110.
Pre Load Commands		See "Pre Load Commands" on page 112.
Post Load Commands		See "Post Load Commands" on page 112.

Microsoft SQL Server target table options

Tab	Option	Description
Target		This tab lists the table name and other read-only values from the associated datastore. Also contains the **Make port** option to specify the table as an embedded data flow port (see "Make port" on page 105).
Options		
	Rows per commit	See "Rows per commit" on page 105.
	Column comparison	See "Column comparison" on page 106.
	Delete data from table before loading	See "Delete data from table before loading" on page 106.
	Number Of Loaders	See "Number of loaders" on page 107.
	Use overflow file	See "Use overflow file" on page 107.
	Overflow file name	The full path is determined by the datastore overflow directory setting. See "Use overflow file" on page 107.
	Overflow file format	See "Use overflow file" on page 107.
	Ignore columns with value	See "Ignore columns with value" on page 107
	Ignore columns with NULL	See "Ignore columns with null" on page 107

Microsoft SQL Server target table options *(Continued)*

Tab	Option	Description
	Use input keys	See "Use input keys" on page 107.
	Auto correct load	See "Auto correct load" on page 108.
	Include in transaction	See "Include in transaction" on page 109.
	Transaction order	See "Include in transaction" on page 109.
Bulk loader options		
	Bulk load	Select this check box to use Microsoft SQL Server bulk loading options to write the data.
	All rows	Select this option to load all rows at the same time.
	Rows per commit	Select this option to load a specified number of rows per commit.
	Mode	Specify the mode for loading data in the target table: • **append** — Adds new records to table. • **truncate** — Deletes all existing records in table then adds new records.
	Maximum rejects	A rejected row contains data that does not match the expected format or information specified for the table being loaded. When the value for **Maximum rejects** is exceeded, the bulk loader stops. To ignore rejected rows, continue processing, and log warnings, specify as `none` or `0`.
Load Triggers		See "Load triggers" on page 110.
Pre Load Commands		See "Pre Load Commands" on page 112.
Post Load Commands		See "Post Load Commands" on page 112.

Oracle target table options

Tab	Option	Description
Target		This tab lists the table name and other read-only values from the associated datastore. Also contains the **Make port** option to specify the table as an embedded data flow port (see "Make port" on page 105).
Options		
	Rows per commit	Defaults to Oracle's default commit size. See "Rows per commit" on page 105.
	Column comparison	Choose from comparing by name or comparing by position. See "Column comparison" on page 106.
	Delete data from table before loading	See "Delete data from table before loading" on page 106.
	Drop and re-create table	Drops the existing table and creates a new one with the same name before loading. This option is displayed only for template tables. Template tables are used in design or test environments.
	Number of loaders	See "Number of loaders" on page 107.
	Use overflow file	See "Use overflow file" on page 107.
	Overflow file name	The full path is determined by the datastore overflow directory setting. See "Use overflow file" on page 107.
	Overflow file format	See "Use overflow file" on page 107.
	Ignore columns with value	See "Ignore columns with value" on page 107.
	Ignore columns with NULL	See "Ignore columns with null" on page 107.
	Use input keys	See "Use input keys" on page 107.
	Auto correct load	See "Auto correct load" on page 108.
	Enable partitioning	See "Enable Partitioning" on page 106
	Include in transaction	See "Include in transaction" on page 109.
	Transaction order	See "Include in transaction" on page 109.
	Commit at end of INSERT ... SELECT	When selected (default), Data Integrator commits a single transaction log statement per load. When deselected, you can limit transaction size to the value specified in **Rows per commit** to create a smaller transaction size than the single commit per job default behavior.

Oracle target table options *(Continued)*

Tab	Option	Description
	Commit at end of INSERT ... SELECT (*continued*)	If your transaction log size is too small for a single transaction of this type, set a commit size when the following is true: • The job has a source and target that use the same datastore • The job has an Oracle target table • Data Integrator is optimizing and executing the job by pushing down the read operation to the Oracle target table host. This type of operation requires that Data Integrator generate an INSERT ... SELECT SQL statement. Data Integrator commits a transaction for an INSERT ... SELECT by default at the end of the job.) • The job failed with an Oracle transaction log full error. While troubleshooting: **1.** View the SQL that Data Integrator generates to see if an `INSERT...SELECT` statement is in use. (In the Designer, open a data flow and select **Validate > Display Optimized SQL**.) **2.** If so, deselect the **Commit at end of INSERT...SELECT** check box and enter a value for **Rows per commit**. If you use this option, expect to see a decrease in performance.
Bulk Loader Options		
	Bulk load	Select a bulk loading method. Use database tools to load data in bulk instead of using SQL statements. • **API** — Allows you to use an Oracle direct-path load API to load table data directly to database files. The target database must be Oracle 8.1 or later. • **File** — Allows you to use a staging file and the Oracle SQL*Loader to load table data. To use this feature, the version of the Oracle SQL*Loader (specified on this tab) and the database (specified in the datastore for the target) must match. If you also want to use the direct-path load method, manually select it from the File Options section. Otherwise, Data Integrator will perform a convention load. Conventional loads generally are slower than direct-path loads because data is loaded to tables rather than directly to database files associated with tables • **None** — Allows you to use normal load functionality. See the Options tab.

Oracle target table options *(Continued)*

Tab	Option	Description
	Mode	Specify the mode for loading data in the target table: • **append** — Adds new records containing the loaded data. • **insert** — Adds new records containing the loaded data. Requires that the table be empty before loading. SQL loader terminates with an error if the table is not empty. • **replace** — Deletes all existing records in the table, then adds new records containing the loaded data. • **truncate** — Deletes all existing records in the table, then adds new records containing the loaded data.

Oracle target table options *(Continued)*

Tab	Option	Description
	Rows per commit	Specifies the transaction size in number of rows or bulk loading. If Rows per commit is set to 1000, Data Integrator sends a commit to the underlying database every 1000 rows. If you do not enter a value, Data Integrator uses the default: 1000 Bulk loading is not available for targets in a real-time jobs.
	Maximum rejects	Enter the maximum number of error records allowed before Data Integrator terminates the job. If you do not enter a value, Data Integrator uses the default: 10.
	Recoverable	Select this check box to log direct-path information about the loaded data in the Oracle redo log. **NOTE:** All conventional loads are automatically specified as recoverable.
	SQL *Loader version	(File method only) The version used to load data into the table. The version of the Oracle SQL*Loader and the database (specified in the datastore for the target) must match.
	Text delimiter	(File method only) Enter the character used to delimit `char` or `varchar` columns. The default character is a double quotation mark ("). Make sure the character you enter is not used in any of the data columns.
	Field delimiter	(File method only) Enter the character used to separate columns. The default character is a comma (,). Make sure the character you enter is not used in any of the data columns. You can specify a non-printable character by entering the ASCII equivalent, such as: /*ASCII_number*
	Maximum bind array	(File method only) Enter the maximum bind array. The bind array needs to be large enough to contain a single row. For good performance, make this large enough to hold 100 rows. If you don't enter a value, Data Integrator uses the default Oracle Bulk Loader value.

Oracle target table options *(Continued)*

Tab	Option	Description
	Use the control file:	(File method only) Select this check box to load data from a specific bulk loading control file and data file. Rather than loading data from the source shown in the data flow, DATA INTEGRATOR directs Oracle to load data from the data file associated with the named control file. Enter the name of the control file. Do not include the `.ctl` extension in the file name. If you do not specify a complete path, Data Integrator searches for the file in: • The path you have specified as the bulk loader directory in the datastore definition • `\$LINK_DIR\log\`*`Job Server name`*`\`*`repository name`*`\bulkloader` Both a control file and an associated data file must be in the same directory. If you select this option, you can also select these options or specify these values: • Direct path • Cleanup after bulk load • Rows per commit • Maximum rejects • Maximum bind array
	Generate files only	(File method only) Select this check box to have Data Integrator generate a data and control file. Rather than loading data into the target shown in the data flow, Data Integrator generates a control file and a data file that you can later load using Oracle bulk loading. This option is often useful when the target database is located on a system running a different operating system than the Data Integrator Job Server. Data Integrator writes the data and control files in the bulk loader directory specified in the datastore definition. You need to copy the files to the remote system manually. Data Integrator names these files *`tablename`*`.ctl` and *`tablename`*`.dat`, where *`tablename`* is the name of the target table.

Oracle target table options *(Continued)*

Tab	Option	Description
	Direct path	(File method only) Select this check box to specify a direct-path load. To use direct-path load, the version of SQL*Loader available to the Job Server executing the job must be the same as the target database version. For example, you cannot perform a SQL*Loader Version 7.1.2 direct path load to load into a Oracle Version 7.1.3 database. For more information, see the Oracle server documentation.
	Clean up bulk loader directory after load	(File method only) Select this check box to have DATA INTEGRATOR delete all bulk loader-related files (control file, data file, log file) after the load is complete. If an error occurs during the bulk load, Data Integrator creates a .bad file and does not delete any files. Errors occur when: • Log file was not created • Log file contains "ORA-" or "SQL*Loader-"
	Trailing nullcols	(File method only) Select this check box to indicate that columns not represented in the data being loaded should be treated as null columns. Use when a data record is not complete but the existing data needs to be loaded. If this option is not selected, the system generates an error.
Load Triggers		See "Load triggers" on page 110.
Pre Load Commands		See "Pre Load Commands" on page 112.
Post Load Commands		See "Pre Load Commands" on page 112.

Sybase target table options

Tab	Option	Description
Target		This tab lists the table name and other read-only values from the associated datastore. Also contains the **Make port** option to specify the table as an embedded data flow port (see "Make port" on page 105).
Options		
	Rows per commit	See "Rows per commit" on page 105.
	Column comparison	See "Column comparison" on page 106.
	Delete data from table before loading	See "Delete data from table before loading" on page 106.
	Number Of Loaders	See "Number of loaders" on page 107.
	Use overflow file	See "Use overflow file" on page 107.
	Overflow file name	Enter a file name. The full path is determined by the datastore's overflow directory setting. See "Use overflow file" on page 107.
	Overflow file format	See "Use overflow file" on page 107.
	Ignore columns with value	See "Ignore columns with value" on page 107
	Ignore columns with NULL	See "Ignore columns with null" on page 107
	Use input keys	See "Use input keys" on page 107.
	Auto correct load	See "Auto correct load" on page 108.
Bulk loader options		
	Bulk load	Select this check box to use Sybase bulk loading options to write the data.
	All rows	Select this option to load all rows at the same time.
	Rows per commit	Select this option to load a specified number of rows per commit.
	Mode	Specify the mode for loading data in the target table: • **append** — Adds new records to table. • **truncate** — Deletes all existing records in table then adds new records.
Load Triggers		See "Load triggers" on page 110.
Pre Load Commands		See "Pre Load Commands" on page 112.
Post Load Commands		See "Pre Load Commands" on page 112.

Teradata target table options

Tab	Option	Description
Target		This tab lists the table name and other read-only values from the associated datastore. Also contains the **Make port** option to specify the table as an embedded data flow port (see "Make port" on page 105).
	Table name	Read-only.
	Database owner	Read-only.
	Datastore name	Read-only.
	Database type	Read-only.
	Teradata datasource	Read-only. The name used to create an ODBC data source in the ODBC Administration utility. This option specifies an existing data source through which Data Integrator accesses sources and targets in this datastore.
Options		
	Column comparison	Choose from comparing by name or comparing by position. See "Column comparison" on page 106.
	Delete data from table before loading	See "Delete data from table before loading" on page 106.
	Number Of Loaders	See "Number of loaders" on page 107.
	Use overflow file	See "Use overflow file" on page 107.
	Overflow file name	The full path is determined by the datastore overflow directory setting. See "Use overflow file" on page 107.
	Overflow file format	See "Use overflow file" on page 107.
	Ignore columns with value	See "Ignore columns with value" on page 107
	Ignore columns with NULL	See "Ignore columns with null" on page 107
	Use input keys	See "Use input keys" on page 107.
	Auto correct load	See "Auto correct load" on page 108.
	Include in transaction	See "Include in transaction" on page 109.
	Transaction order	See "Include in transaction" on page 109.
Bulk Loader Options		

Teradata target table options (Continued)

Tab	Option	Description
	Bulk loader	Select a method from this list to load data from a specific bulk loading control file and data file. Select from: • Warehouse Builder • Load Utilities (non-Teradata Warehouse Builder methods) • None (use ODBC to load to Teradata) Note that your bulk loader options are different depending on the method you choose.
	Operator or Utility	Operator values for Warehouse Builder include: • Load • SQL Inserter • Stream • Update Utilities you can use under Load Utilities include: • Multiload • FastLoad • TPump
	Attribute values or Command line	A different set of attributes displays depending on the Warehouse Builder method and operator you choose. Bolded attribute names indicate that the value cannot be left blank and requires a value. You can accept default values or select attribute values and modify them. For more information on specific attributes, consult your Teradata Warehouse Builder documentation. For a Load Utility, you can enter the command line for the specified utility (example: For FastLoad, you could enter <C:\Teradata\FLScripts\myScript.ctl).
	Number of instances	Available only with Warehouse Builder method. Specify the number of target instances for your selected operator. This information is included in the Warehouse Builder script that Data Integrator generates.
	Number of DataConnector instances	Available only with Warehouse Builder method. Specify the number of DataConnector operator instances. A DataConnector operator is used to read data files generated by Data Integrator. This information is included in the Warehouse Builder script that Data Integrator generates.

Teradata target table options (Continued)

Tab	Option	Description
	Log directory	Available only with Warehouse Builder method. Specify the tbuild log directory. As a default, Data Integrator automatically populates this field with information provided when the Teradata datastore was created.
	Ignore duplicate rows	Available only with Warehouse Builder **Stream** or **Update** operators. Corresponds to "Ignore Duplicate Insert Rows" Warehouse Builder statement. When selected, duplicate rows are not placed in the error table. See Teradata Warehouse Builder documentation for more information.
	Generate files only	When selected, Data Integrator generates a data file and a script file, and ignores the **Number of loaders** option (in the Options tab). Rather than loading data into the target shown in the data flow, Data Integrator generates a control file and a data file that you can later load using Teradata bulk loading. This option is often useful when the target database is located on a system running a different operating system than the Data Integrator Job Server. Data Integrator writes the data and control files in the bulk loader directory specified in the datastore definition. You must copy the files to the remote system manually. Data Integrator naming conventions for these files is: <*OwnerName_TableName_TIDPID_n*>.ctl and <*OwnerName_TableName_TIDPID_n*>.dat where: • *OwnerName* is the table owner • *TableName* is the target table • *n* is a positive integer, optionally used to guarantee that Data Integrator does not overwrite a pre-existing file • *TID* is thread ID • *PID* is process ID
	Clean up bulk loader directory after load	Select this check box and Data Integrator deletes all bulk loader-related files (script, data files, temporary file) after the load is complete. If an error occurs during bulk load, Data Integrator does not delete script and data files. Errors usually occur when: • There is a syntax error in the script. • Error tables are not empty. Error tables contain rows that cannot be inserted into the target table due to data conversion or constraint violation.

Teradata target table options (Continued)

Tab	Option	Description
	Mode	Specify the mode for loading data in the target table: • **append** — Adds new records to the table • **replace** — Deletes all existing records in the table, then inserts the loaded data as new records
	Field delimiter	Specify a single-character field delimiter. Default value is /127 (non-printable character).
	Debug all tasks	Available only with Warehouse Builder method. Corresponds to the tbuild "-d" option. For more information consult your Teradata Warehouse Builder documentation.
	Trace all tasks	Available only with Warehouse Builder method. Corresponds to the tbuild "-t" option. For more information consult your Teradata Warehouse Builder documentation.
	Latency interval (sec)	Available only with Warehouse Builder method. Corresponds to the tbuild -l option. For more information consult your Teradata Warehouse Builder documentation.
	Checkpoint interval (sec)	Available only with Warehouse Builder method. Corresponds to tbuild -z option and specifies a time interval, in seconds, between checkpoints. Data Integrator default for this parameter is10 seconds. For more information consult your Teradata Warehouse Builder documentation.
Load Triggers		See "Load triggers" on page 110.
Pre Load Commands		See "Pre Load Commands" on page 112.
Post Load Commands		See "Pre Load Commands" on page 112.

Target XML files, messages, and templates

An XML Schema or DTD format can be added to a job as a target. Choose **Make XML File Target** or **Make XML Message Target** from the context menu that opens when you drop either format into the workspace.

You can also create an XML file target without creating a format by using an XML template. See "XML template" on page 170.

XML target file and message options

Option	Description
Make port	Select this check box to make the target file an embedded data flow port. For more information, see Chapter 11, "Embedded Data Flows," in the *Data Integrator Designer Guide*.
XML file	(File targets only) The location relative to the Job Server of a file to use as the target. If the file does not exist, Data Integrator creates it. If the file exists, Data Integrator clears the content of the file before writing the output to it.
XML test file	(Message targets only) The location relative to the Job Server of a file to use as the message target when you run the job in test mode. If the file does not exist, Data Integrator creates it. If the file exists, Data Integrator clears the content of the file before writing the output to it.
Delete and recreate file	Allows you to override the default behavior in Data Integrator which is to append new data sets to the file. If selected, Data Integrator deletes the old file and creates a new one containing only the current data set.
Print comment	Allows you to include or exclude a comment in the target file data that identifies the data as having been processed by BusinessObjects Data Integrator.
Replace NULL or blank	Allows you to specify a value that will replace NULL or blank values in element data. Select the check box, then enter a value in the **With** field.

XML target file and message options (Continued)

Option	Description
Enable validation	Enable a comparison of the outgoing data to the stored XML Schema or DTD from which this XML file was created. When this option is enabled, the data flow throws an exception if the outgoing data is not valid. When you are developing real-time jobs, this validation helps you ensure sample data is both valid and well-formed. If you turn on this option in production, make sure to include appropriate error handling either in the Data Integrator job or the client application to process an error caused if the real-time job receives data that does not validate against the imported XML Schema or DTD.
Include DTD	(For targets created from DTD formats only) The content of an XML target does not normally include the DTD format (which Data Integrator uses internally). If you want to add the DTD format to the target file or message, select this check box.
XML encoding	Select an XML encoding for the XML target file. If you do not select a value, the encoding in the XML header field is used. If that field is empty, UTF-8 is used. XML file targets can be saved with a different encoding/code page than Data Integrator's system locale. XML message source and target encodings default to UTF-8 and cannot be changed. For more information, see "Locales and Multi-Byte Functionality" on page 549.
XML header	You can use a unique header for each file target. To use this option, you must first enter the header information you want to use for the target. Thereafter, you can edit it from this field. For example, if your header includes more information than the XML Schema version and the encoding, you may want to view and edit this information in the Designer. If you only need to change the XML encoding for this file target, use the XML encoding option instead of editing the header.
DTD file in DOCTYPE	(For targets created from DTD formats only) The content of an XML target does not normally include the DTD format (which Data Integrator uses internally). If you want to add a DOCTYPE element to the target file or message, that specifies a path to a DTD format enter the path here or use the Browse button to select one.

XML target file and message options (Continued)

Option	Description
Format Name	(Read only) The name of the DTD or XML Schema format used in the Designer.
Root element name	(Read only) The name of the root element used in the DTD or XML Schema.
Namespace	(Read only) The name space used in the XML Schema.

The validation for an XML target allows columns and nested tables marked as optional in the output schema to not be present in the input schema. At run-time the XML target will handle missing columns appropriately.

Template table

CLASS

Reusable

ACCESS

- To insert as a target, open a data flow diagram in the work space, click the template table icon in the tool palette, and click anywhere in the data flow.
- To insert as a source, open the object library, click the **Datastores** tab, select the desired template table, and drag into the data flow.
- To view options, click the name of the template table in the workspace or in the project area. This opens the object editor.

DESCRIPTION

Template tables are new tables you want to add to a database. You can use a template table one time as a target and multiple times as a source. You cannot use a template table in an R/3 data flow.

A template table provides a quick way to add a new target table to a data flow. When you use a template table, you do not have to specify the table's schema or import the table's metadata. Instead, during job execution, Data Integrator has the DBMS create the table with the schema defined by the data flow. After you create a template table as a target in one data flow, you can use the same template table as a source in any other data flow.

Use template tables in the design and testing phases of your projects. You can modify the schema of the template table in the data flow where the table is used as a target. Any changes are automatically applied to any other instances of the template table. During the validation process, Data Integrator warns you of any errors, such as errors that result from changing the schema.

Before you can use a template table as a source in a data flow design, the data flow where the template table was created as a target has to be valid and you have to save the data flow.

Before executing any job where a template table is used as a source, you must execute the job where the template table is used as a target at least one time. If the template table is used as a target and a source in the same job, then the data flow where it is used as a target must be executed first.

When running a job where a template table is used as a target, use care if the table already exists. If the **Drop Create Table** option is selected in the template table editor (this is the default option), Data Integrator drops the existing table and creates a new one. If the **Drop Create Table** option is not selected, Data Integrator attempts to load data in the existing table. In this case, Data Integrator generates run-time errors if the existing table schema does not match the schema generated in the data flow.

When used as a target, the options available from the target editor for template tables are the same as those available for target tables (see "Target" on page 103) with some exceptions.

Tab	Option	Description
Target		
	Table name	The name of the table. Can contain alpha, numeric, and underscores; cannot include blank spaces.
Options		
	Column comparison	Not applicable for template tables.
	Drop and re-create table	Instructs Data Integrator to drop an existing table with the same name before creating the table specified by the template table. When using template tables in real-time jobs, deselect the this and the **Delete data from table before loading** option. These options are selected by default when you create an template tablet. For more information see "Message processing" on page 89.
Bulk Loader Options		
	Bulk load	Not available for template tables.
Load Triggers		
	On operation	Not available for template tables.

Before running production jobs, execute the job to load the target table if you have not already do so, then right-click the template table in the object library or in a data flow and select **Import Table**. Data Integrator creates the table in your database and imports it. All information about the table is marked as part of the database and you can make no further changes to the schema. You can now use the new table in expressions, functions, transform options, or for bulk loading. Other features, such as exporting an object, are available for template tables.

Transform

CLASS

Reusable

ACCESS

In the object library, click the **Transforms** tab.

DESCRIPTION

Transforms define your data transformation requirements. Transforms use the operation codes associated with each row of data read from a source. The descriptions of individual transforms indicate which operation codes the transforms ignore or use.

Transforms have the following built-in attributes:

Attribute	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object.
Description	Your description of the transform. Description is not available for query transforms.

If you delete a user-defined transform from the object library, calls to the object are replaced with an icon indicating that the calls are no longer valid, and it is deleted from the project area.

For a list of available transforms and detailed information about using each transform see "Descriptions of transforms" on page 225.

Try

CLASS

Single-use

ACCESS

With a work flow diagram in the workspace, click the try icon in the tool palette.

DESCRIPTION

A try is part of a serial sequence called a try/catch block. Use a single try with each try/catch block; there can be more than one catch with a single try. The try/catch block allows you to specify alternative work flows if errors occur while Data Integrator is executing a job. Try/catch blocks "catch" classes of errors, apply solutions that you provide, and continue execution.

Do not reference output variables from a try/catch block in any subsequent steps if you plan on using the automatic recovery feature. Referencing such variables could alter the results during automatic recovery.

Tries have the following attribute:

Attribute	Description
Name	The name of the object. This name appears on the object in the diagram.

For more information, see "Catch" on page 25.

While loop

CLASS

Reusable

ACCESS

With a work flow diagram in the workspace, click the while loop icon in the tool palette.

DESCRIPTION

A while loop repeats a sequence of steps as long as a condition is true.

For each while loop, specify:

- Condition — In the **While** box enter a Boolean expression that the job evaluates.

  The expression must evaluate to TRUE or FALSE. You can use constants, functions, variables, parameters, and standard operators to construct the expression. For information about expressions, see Chapter 3, "Smart Editor".

- Set of steps — In the while loop workspace, enter the steps you want completed when the condition is true.

  You can add any objects valid in a work flow, including scripts, work flows, and data flows. Connect these objects to represent the order that you want the steps completed.

  NOTE: Though you can include the parent work flow in the while loop, recursive calls can create an infinite loop.

For information about design considerations and using while loops with data scan, see "While loops" on page 193 of the *Data Integrator Designer Guide*.

Work flow

CLASS

Reusable

ACCESS

- In the object library, click the **Work Flows** tab.
- With a job or work flow diagram in the workspace, click the work flow icon in the tool palette.

DESCRIPTION

A work flow contains data flows and the operations that support data flows. The work flow defines the execution order of the data flows and supporting operations. A job is also a work flow.

You can define parameters to pass values into the work flow. You can also define variables for use inside the work flow.

The definition of a work flow can contain the following objects:

- Other work flows
- Data flows
- Scripts
- Try/catch blocks
- Conditionals
- While loops

In some cases, steps in a work flow depend on each other and should always be executed together. You can designate such a work flow (batch jobs only) as a "recovery unit." When designated as a recovery unit, the entire work flow must complete successfully during execution. If any step in such a work flow does not complete successfully, Data Integrator

re-executes all steps in the work flow during automatic recovery, except for R/3 data flows, Data Integrator re-executes data flows that executed successfully earlier. Data Integrator may or may not re-execute R/3 data flows. See *Data Integrator Supplement for SAP* for more information.

To designate a work flow as a recovery unit, select the **Recover as a unit** check box on the work flow Properties window.

On the workspace diagram, a symbol indicates when a work flow is a recovery unit.

To ensure that a job only executes a work flow one time, select the **Execute only once** check box on the work flow Properties window. When this check box is selected, Data Integrator only executes the first occurrence of the work flow; Data Integrator skips subsequent occurrences in the job. You might use this feature when developing complex jobs with multiple paths, such as jobs with try/catch blocks or conditionals, and you want to ensure that Data Integrator only executes a particular work flow one time.

Work flows have several built-in attributes.

Attribute	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object.
Description	Your description of the work flow.
Date_created	The date that the object was created.

If you delete a work flow from the object library, calls to the object are replaced with an icon indicating that the calls are no longer valid.

XML file

XML file source

XML file target

CLASS

Single-use

ACCESS

In the object library, click the **Formats** tab.

DESCRIPTION

An XML file object allows you to indicate a source or target in a batch or real-time job. When used as a source, an XML file object translates incoming XML-formatted data into an internal Data Integrator data set. When used as a target, an XML file object translates the data produced by a data flow, including nested data, into an XML-formatted file.

The data read into or written out of an XML file must have a single row at the top-level table. When writing out an empty nested table, Data Integrator includes a single row of the nested table, with null values in each column of the table.

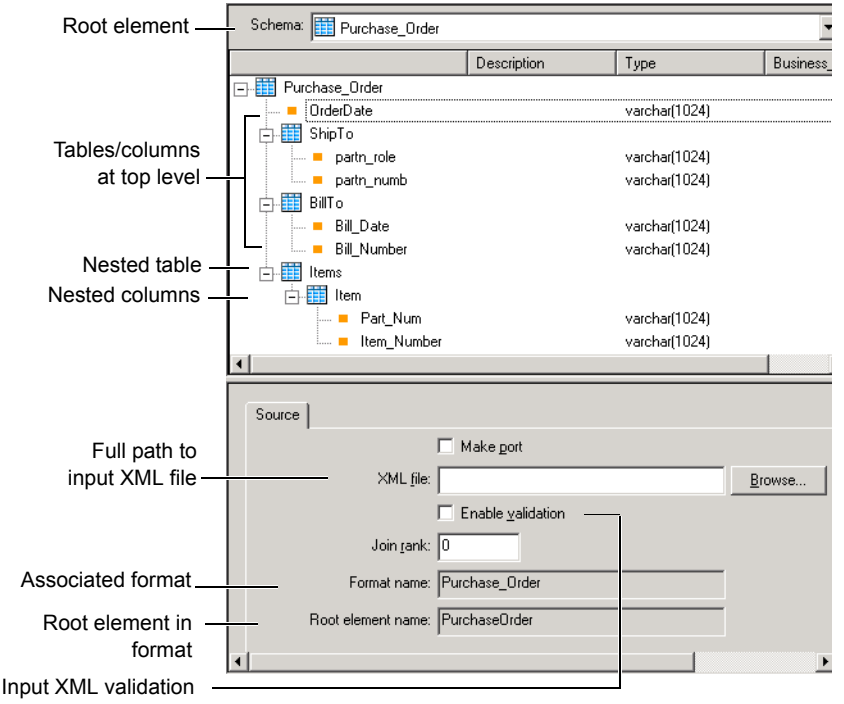
To produce the metadata that describes the data that an XML file object handles, Data Integrator reads an XML Schema or DTD. The metadata is stored in the repository as either an XML Schema or DTD object.

SOURCE OR TARGET

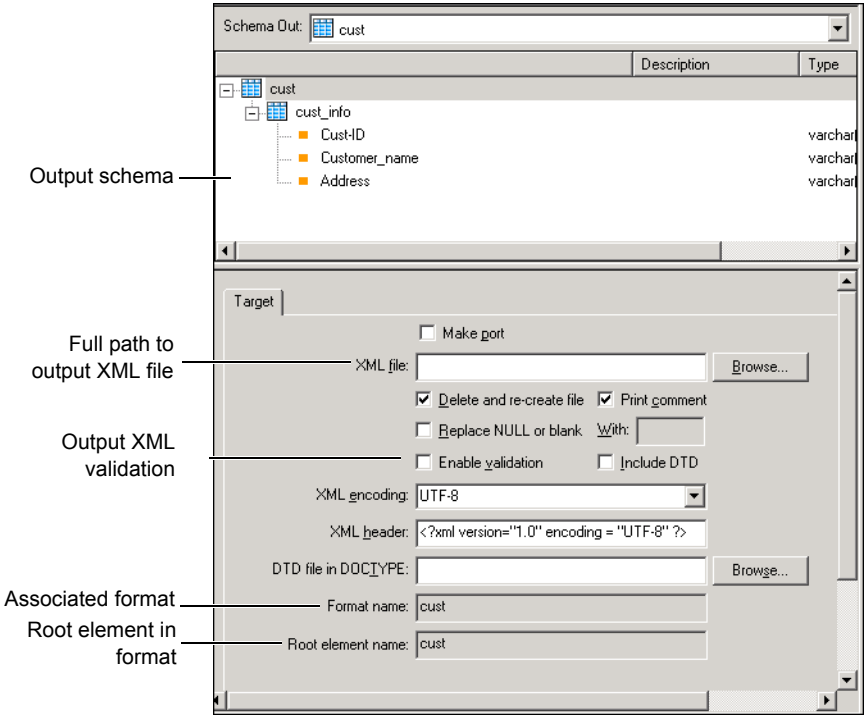
You can insert an XML file into a data flow by dragging either an XML Schema or DTD format from the **Formats** tab of the object library into the workspace of a data flow. When you drop the format in the workspace, Data Integrator prompts you to specify that the resulting object is a source or target file or a source or target message.

SOURCE AND TARGET FILE EDITORS

When used as a source in a data flow, an XML file has the following editor. See "XML file source" on page 96 for a detailed description of source file editor options.



When used as a target in a data flow, an XML file has the following editor. See "Target XML files, messages, and templates" on page 133 for a detailed description of target file editor options.



PARSING INPUT AND PRODUCING OUTPUT

Mapping rules govern how Data Integrator translates an XML Schema or DTD into its internal schema definition and produces XML formatted data from an internal data set. These rules are described in "Formatting XML documents" on page 224 of the *Data Integrator Designer Guide*.

PROPERTIES

XML file properties are the same as those of its format. See "DTD" on page 50 and "XML Schema" on page 152 for information about properties and column attributes produced by a particular format.

XML message

XML source

XML target

CLASS

Single-use

ACCESS

In the object library, click the **Formats** tab.

DESCRIPTION

An XML message object allows you to indicate a real-time source or target in a job.

When used as a source, an XML message object translates incoming XML-formatted messages into an internal Data Integrator data set. When used as a target, an XML message object translates the data produced by a job, including nested data, into an XML-formatted message and sends the message to the Access Server.

When a real-time job contains an XML message source, it must also contain an XML message target.

The data read into or written out of an XML message must have a single row at the top-level table. When writing out an empty nested table, Data Integrator includes a single row of the nested table, with null values in each column of the table.

To produce the metadata that describes the data that an XML message handles, Data Integrator reads the format for the XML message. The metadata is stored in the repository as an XML Schema or DTD. See "DTD" on page 50 and "XML Schema" on page 152.

Source or target

You can insert an XML message into a real-time job by dragging a XML Schema or DTD format from the **Formats** tab of the object library into the workspace of a data flow. When you drop the format in the workspace, Data Integrator prompts you to specify that the resulting XML message as a source or target.

Source and Target editors

See "XML message source" on page 97 for a detailed description of XML message source editor options.

See "Target XML files, messages, and templates" on page 133 for a detailed description of XML message target editor options.

XML test files

During the design phase of your application, you can execute a real-time job in "test mode." In test mode, the real-time job reads messages from an XML test file specified in the source editor, and writes XML-formatted messages to an XML test file specified in the target editor.

Parsing input and producing output

Mapping rules govern how Data Integrator translates an XML Schema or DTD into its internal schema definition and produces XML from an internal data set. These rules are described in "DTD" on page 50 and "XML Schema" on page 152.

See the Nested Data chapter of the Designer Guide for an introduction to the nested relational data model which Data Integrator uses to generate an internal hierarchical schema.

PROPERTIES

XML file properties are the same as those of its format. See "DTD" on page 50 and "XML Schema" on page 152 for information about properties and column attributes produced by a particular format.

XML Schema

CLASS

Reusable

ACCESS

In the object library, click the **Formats** tab, then double-click the XML Schema category.

DESCRIPTION

Data Integrator supports W3C XML Schemas Specification 1.0. This XML Schema version is documented on the following web site: http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/.

XML Schemas describe the data structure of an XML file or message. Data flows can read and write data to messages or files based on a specified XML Schema format. You can use the same XML Schema to describe multiple XML sources or targets.

To use XML Schemas, import XML Schema metadata into Data Integrator. During import, Data Integrator converts the structure defined in the XML Schema into the Data Integrator internal schema based on the nested relationship data model. See "Rules for importing XML Schemas" on page 159.

EDITOR

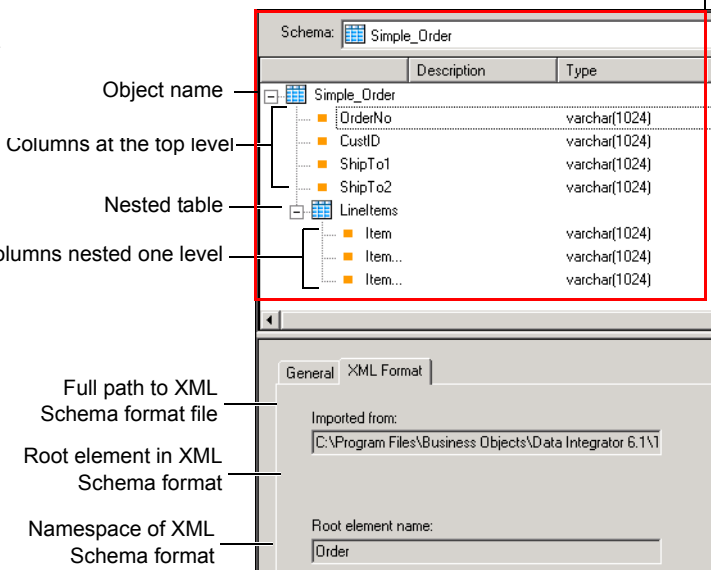
Open the XML Schema editor by double-clicking an XML Schema name in the object library.

XML Schema

```
<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Order">
    <xs:complexType>
      <xs:sequence>
          <xs:element name="OrderNo"  type="xs:string" />
          <xs:element name="CustID"   type="xs:string" />
          <xs:element name="ShipTo1"  type="xs:string" />
          <xs:element name="ShipTo2"  type="xs:string" />
          <xs:element maxOccurs="unbounded" name="LineItems">
             <xs:complexType>
                <xs:sequence>
                   <xs:element name="Item"      type="xs:string" />
                   <xs:element name="ItemQty"   type="xs:string" />
                   <xs:element name="ItemPrice" type="xs:string" />
                </xs:sequence>
             </xs:complexType>
          </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The URL for the namespace must be as shown to match the XML Schema standard supported by Data Integrator. Otherwise, an error is thrown.

Schema opened from the object library



Object name

Columns at the top level

Nested table

Columns nested one level

Full path to XML Schema format file

Root element in XML Schema format

Namespace of XML Schema format

For import procedures see "Formatting XML documents" on page 224 of the *Data Integrator Designer Guide*.

PROPERTIES

XML Schemas have the following properties.

Property	Description
Name	The name of the format. This name appears in the object library under the Formats tab and is used for sources and targets (XML files or messages) that reference this format in data flows.
Description	Text that you enter to describe and document the XML Schema.
Imported from	The full path to the format. For example, `D:\data\test.xsd`.
Root element name	The name of the primary node you want to import. Data Integrator only imports elements of the format that belong to this node or any sub nodes.
Namespace	(Optional) The Namespace URL of the root element.

ATTRIBUTES

Data Integrator supports the following column attributes for XML Schemas:

Column attributes supported for XML Schemas

Attribute	Description
Required	Indicates whether this column always has to be mapped (YES/NO)
Native Type	Original data type of the column. Saved as a string during import.
Default Value	Default value for this column.
Fixed Value	The only value the column can have.
Max Value Inclusive Of Min Value Inclusive Of Max Value Exclusive Of Min Value Exclusive Of	Mapped from the following data type constraining facets: MaxInclusive, MinInclusive, MaxExclusive, and MinExclusive.
Pattern	For a string: the pattern to which its value should match.
Enumeration	Contains a list of all possible values separated by vertical bars. For example: "Red | White | Blue Green | Magenta". A string is cut off at 256 characters.
XML Type	Allows you to track whether the column was an element or attribute in the original XML Schema.
Length	Number of characters in this column.
Max Length	Maximum number of characters allowed in this column.
Min Length	Minimum number of characters allowed in this column.
Namespace	Column namespace.

Nested table attributes supported for XML Schemas

Attribute	Description
Any One Column	If choice (for example, "white | black | almond"), then Data Integrator sets the value of Any One Column to YES. If sequence (for example, "first, last, street, city, state") then Data Integrator sets the value to NO. If both are present in the XML Schema, the value is set to NO.
Maximum Occurrence	Indicates maximum number of rows in the table. If maximum occurrence equals zero, Data Integrator indicates that your XML Schema structure is not valid.
Minimum Occurrence	Indicates minimum number of rows that can be in the table.

Supported XML Schema components

Data Integrator supports all valid XML Schemas (see "Description" on page 152) and their components except those listed under "Not supported in Data Integrator" on page 163.

Features like abstract types, blocking, etc. are supported (imported and accepted without error) but not discussed here as they do not have a direct impact on the ability of Data Integrator to support XML Schemas.

Data Integrator imports XML Schema data types as well as element and attribute names and their structure. Once imported, double-click an XML Schema format from the object library to view table and column names and structure. From the XML Format editor, right-click a column name and select **Properties** to edit properties, attributes, and data types.

XML Schema elements in Data Integrator

The following XML Schema *elements* are mapped to Data Integrator attributes when they are imported as metadata.

XML Schema elements mapped to Data Integrator attributes

XML Schema Element	Data Integrator Attribute (nested table or column)
All	All. Elements should occur but they can occur in any order. See Choice.
Choice	Any One Column. If the complex type for an element has been specified as choice then a Data Integrator attribute called Any One Column is created and set to YES. If the complex type has been defined with sequence or several nesting levels containing a mix of choice and sequence then the Any One Column table attribute is created and set to NO. *Sequence*, *choice*, and *all* are handled in Data Integrator as follows: • Sequence becomes "Any One Column = NO". Attributes A, B, and C become columns A, B, and C. • Choice becomes "Any One Column = YES": Attributes A, B, and C become columns A or B or C. • All becomes "All": B, C, and A or any combination of the three.
Default	Default Value
Enumeration	Enumeration. The value for this attribute is cut off after 256 characters. As a result, all the enumerated values may not be visible.
Fixed	Fixed Value
Length MinLength MaxLength	Length, Min Length, and Max Length.
MaxInclusive MinInclusive MinExclusive MaxExclusive	Max Value Inclusive Of, Min Value Inclusive Of, Min Value Exclusive Of, Max Value Exclusive Of.
MaxOccurs	Maximum Occurrence (only applies to tables).
MinOccurs	Minimum Occurrence (only applies to tables).

XML Schema elements mapped to Data Integrator attributes

XML Schema Element	Data Integrator Attribute (nested table or column)
Name	Column name
Pattern	Pattern
TotalDigits and FractionDigits	None. Digits are handled as decimal data types. See "Not supported in Data Integrator" on page 163.
Type	Saved as the Native Type attribute (string). The Type element is also translated into a Data Integrator data type (usually varchar). See "Not supported in Data Integrator" on page 163 for more information.
Sequence	See Choice.

XML Schema attributes in Data Integrator

The following XML Schema *attributes* are mapped to Data Integrator column attributes when they are imported as metadata.

XML Schema Attribute	Data Integrator Column Attributes
Default	Default Value
Fixed	Fixed Value The only value the column can have.
Name	Column name
Type	Saved as the Native Type attribute (string). The Type element is also translated into a Data Integrator data type (usually varchar). See "Not supported in Data Integrator" on page 163 for more information.
Use	An XML Schema *Use* attribute with a value of OPTIONAL becomes the Data Integrator *Required* attribute with a value of NO. An XML Schema *Use* attribute with a value of REQUIRED becomes the Data Integrator *Required* attribute with a value of YES. An XML Schema *Use* attribute with a value of PROHIBITED is not supported.

Included XML Schemas

An XML Schema can be extended by including pointers to other XML Schema files. This is done by using *import, include* and *redefine*. These elements are defined at the schema level.

The difference between *include* and *import* is that for *include* the name spaces must be identical in both XML Schemas. *Redefine* is similar to *include* except the caller can redefine one or more components in the related XML Schema.

When you import an XML Schema, Data Integrator follows the links to included files to define additional metadata. The included schema information is saved in the repository so that at run time there is no need to access these files again. Inclusions can be files or URLs.

Groups

XML Schemas allow you to group elements and then refer to the group. A similar concept is available for attributes (called an attribute group). In Data Integrator any reference to a group will be replaced by the contents of that group.

Rules for importing XML Schemas

Data Integrator applies the following rules to convert an XML Schema to a Data Integrator internal schema:

1. Any element that contains an element only and no attributes becomes a column.
2. Any element with attributes or other elements becomes a table.
3. An attribute becomes a column in the table corresponding to the element it supports.
4. Any occurrence of *choice, sequence* or *all* uses the ordering given in the XML Schema as the column ordering in the internal data set.

5. Any occurrence of *maxOccurs,* from greater than 1 to "unbounded", becomes a table with an internally generated name (an implicit table).

   The internally generated name is the name of the parent followed by an underscore, then the string "nt" followed by a sequence number. The sequence number starts at 1 and increments by 1.

After applying these rules, Data Integrator uses two additional rules, except where doing so would allow more than one row for a root element:

1. If an implicit table contains one and only one nested table, then the implicit table can be eliminated and the nested table can be attached directly to the parent of the implicit table.

   For example, the SalesOrder element might be defined as follows in an XML Schema:

```
<xs:complexType>
    <xs:sequence>
      <xs:element ref="Header"/>
      <xs:element ref="LineItems"
minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

When converted into Data Integrator, the LineItems element with `MaxOccurs ="unbounded"` would become an implicit table under the SalesOrder table. The LineItems element itself would be a nested table under the implicit table.

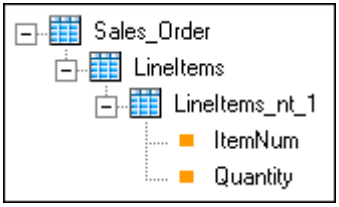
Because the implicit table contains one and only one nested table, the format would remove the implicit table.

2. If a nested table contains one and only one implicit table, then the implicit table can be eliminated and its columns placed directly under the nested table.

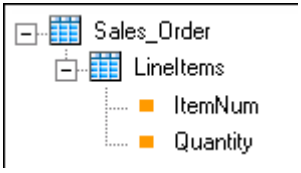
For example, the nested table LineItems might be defined as follows in an XML Schema:

```
<xs:element name="LineItems"
minOccurs="0" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ItemNum"/>
      <xs:element ref="Quantity"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

When converted into Data Integrator, the grouping with `MaxOccurs ="unbounded"` would become an implicit table under the LineItems table. The ItemNum and Quantity elements would become columns under the implicit table.



Because the LineItems nested table contained one and only one implicit table, the format would remove the implicit table.



NOT SUPPORTED IN DATA INTEGRATOR

The following XML Schema elements and attributes are not supported in Data Integrator. They are ignored and not imported.

Component	Description
Annotation	Both documentation and appinfo annotation components are ignored.
Any	If an XML file or message contains elements or attributes in the Any category, (`element xsd:any or attribute xsd:any`) Data Integrator throws a run time error. In an imported XML Schema format the Any component is ignored.
MaxOccurs set to "0"	
Mixed	Column with the same name as the parent table exists in addition to its children.
Non-native attributes	Non-native attributes are attributes that come from a name space other than the one your XML Schema uses. The W3C XML Schema standard enables users to add non-native attributes to attributes and elements. However, Data Integrator ignores all such attributes.
Substitution groups	Schema are imported if they contain substitution groups. However, at run time, substitutions are not accepted. An error is thrown.
Use set to PROHIBITED	
XDR files	Data Integrator does not support XML Data Reduced (XDR) files. XDR files were used as a format in some products before XML Schema became the standard. There are third-party tools on the market which can automatically convert XDR to XML Schema.

DATA TYPE MAPPINGS

Data Integrator imports data types for XML Schema elements and attributes.

There are two types of built-in data types, the Primitive data types and the Derived data types (derived from primitive). Each data type has the following values defined: space, lexical space, and constraining facet.

If the constraining facet *length* is missing when metadata is imported into Data Integrator, the default varchar(1024) is applied. Similarly, for a decimal the default values 28 and 2 are applied for *precision* and *scale*. All other facets like *minInclusive*, *maxInclusive*, *minLength* are imported as column attributes. Enumeration values are also imported as column attributes.

Primitive types

The table below lists Primitive XML Schema types, examples, and the corresponding data type in Data Integrator. The constraining facets used are shown in bold.

Primitive types mapped to Data Integrator data types

XML Schema type	Example	Data Integrator data type
AnyURI	http://www.example.com/	Varchar(len) : len = **length** (in chars)
Base64Binary	GpM7	Varchar(len) : len = **length** (in octets)
Boolean	{true, false, 0, 1}	Varchar(5)
Date	CCYY-MM-DD	Datetime
DateTime	Format = CCYY-MM-DD HH:MM:SS	Datetime
Decimal	7.999	Decimal(p, s) : p = **totalDigits**, a maximum of 28 and s = **fractionDigits**, default =28,2
Double	64 bit floating point	Double (In Data Integrator there is no difference between real and double)
Duration	`P1Y2M3DT10H30M`	Varchar(64)
Float	32 bit floating point, 12.78e-2	Real
gDay		Varchar(12)
gMonth		Varchar(12)
gMonthDay		Varchar(12)
gYear		Varchar(12)
gYearMonth	Gregorian CCYY-MM	Varchar(12)
HexBinary	0FB7	Varchar(len) : len = **length** (in octets)
Notation		N/A
Qname	po:USAddress	Varchar(len) : len = **length** (in chars)
String	“Hello World”	Varchar(len) : len = **length** (in characters)
Time	HH:MM:SS	Datetime

Derived types

The table below lists Derived XML Schema types, examples, and the corresponding data type in Data Integrator. The constraining facets used are shown in bold.

Derived types mapped to Data Integrator data types

XML Schema type	Example	Data Integrator data type
NormalizedString	[No tab/CR/LF in string]	Varchar(len) : len = **length** (in characters)
Token		Varchar(len) : len = **length** (in characters)
Language	En-GB, en-US, fr	Varchar(len): len = **length** (in characters)
NMTOKEN	US, Brésil	Varchar(len): len = **length** (in characters)
NMTOKENS	Brésil Canada Mexique	Varchar(len): len = **length** (in characters)
Name	ShipTo	Varchar(len): len = **length** (in characters)
NCName	USAddress	Varchar(len): len = **length** (in characters)
ID		Varchar(len): len = **length** (in characters)
IDREF		Varchar(len): len = **length** (in characters)
IDREFS		Varchar(len): len = **length** (in characters)
ENTITY		Varchar(len): len = **length** (in characters)
ENTITIES		Varchar(len): len = **length** (in characters)
Integer		Int
NonPositiveInteger		Int
NegativeInteger		Int
Long		Decimal 28,0
Int		Int
Short		Int
Byte		Int
NonNegativeInteger		Int
UnsignedLong		Long
UnsignedShort		Int
UnsignedByte		Int
PositiveInteger		Int
AnyType (ur-type)	unspecified type	Varchar(255)

User-defined types

User-defined types are XML Schema attributes with a non-XML Schema name space. The XML Schema W3C standard uses a SimpleType element for a user-defined type.

When Data Integrator finds a user-defined type it finds the base type and uses it to assign a Data Integrator data type for the element. For example: If element X has type TelephoneNumber its type in Data Integrator is varchar(8).

Some simple types are based on other simple types. In such cases Data Integrator traces back to the base type.

List types

XML Schemas have list types. When a list is encountered, Data Integrator makes its corresponding data type a varchar(1024). All the elements of the list are placed in the value of that column as a string (exactly as it is represented in the XML).

Union types

A union type enables an attribute or element value to be one or more instances of one type drawn from the union of multiple primitive type and list types. When a union is encountered, Data Integrator makes its corresponding data type a varchar (1024).

METADATA

If you delete an XML Schema from the object library, XML sources or targets that are based on this format are invalid. Data Integrator marks the source or target objects with an icon that indicates the calls are no longer valid.

To restore the invalid objects, you must delete the source or target and replace it with a source or target based on an existing XML Schema.

ERROR CHECKING

Data Integrator allows you to control whether it checks each incoming XML file or message for validity against the imported XML Schema. Select **Enable Validation** for an XML source or target in its editor. If you choose to check each XML file or message, Data Integrator uses the XML Schema imported and stored in the repository rather than a XML Schema specified by a given XML file or message. If a file or message is invalid relative to the XML Schema, the job produces an error and shuts down.

A typical example of when Data Integrator does throw validation errors is when either a required element is missing or a new and unexpected element is present in the input. This is true of both the source and target. Consider the following examples:

- A element is defined in the XML Schema with enumeration values of “Black”, “White”, “StainlessSteal”, and “Almond.” If the mapping of that element from the XML document yields “Red” that would be incorrect XML. But the XML file target generates the XML regardless. If validation is enabled, then this error is detected.

- If an element's whitespace attribute is set to `collapse`, Data Integrator does not change the data in sources or targets to respect this setting. The whitespace attribute is not supported in Data Integrator.

During development, you might validate all messages to test for error conditions with representative messages. During production, you might choose to accept rare invalid messages and risk ambiguous or incorrect data.

Data Integrator supports XML Schema legal naming such as allowing multiple elements and attributes to have the same name. However, name conflicts should be identified and tested before you import an XML Schema into Data Integrator. Data Integrator cannot detect naming conflicts and may not report accurate errors which could later lead to runtime errors.

XML template

CLASS

Single-use

ACCESS

To insert as a target:

- Select the XML Template icon in the tool palette, then click the data flow diagram in the workspace.

To view options:

- Click the name of the XML template in the workspace or in the project area. This opens the object editor. See "Target XML files, messages, and templates" on page 133 for a description of target editor options

DESCRIPTION

Use an XML template to create an XML file that matches a particular input schema. The XML template does not require and does not produce a corresponding XML Schema or DTD format. Likewise when it generates XML, it does not create column attributes if they are present in its input schema.

Thus, you can use the XML template to produce an XML file without predefining an XML format. You can use an XML template as a target in a batch or real-time job. In an XML template, all data types are converted to varchar.

After adding an XML template to a data flow, specify the name and location of the file. In the XML template target file editor, specify the file in the **XML file** box.

NOTE: When using XML templates in real-time jobs, deselect the **Delete and recreate file** option in the target editor. This option is selected by default when you create an XML target. For more information see "Message processing" on page 89.

3 Smart Editor

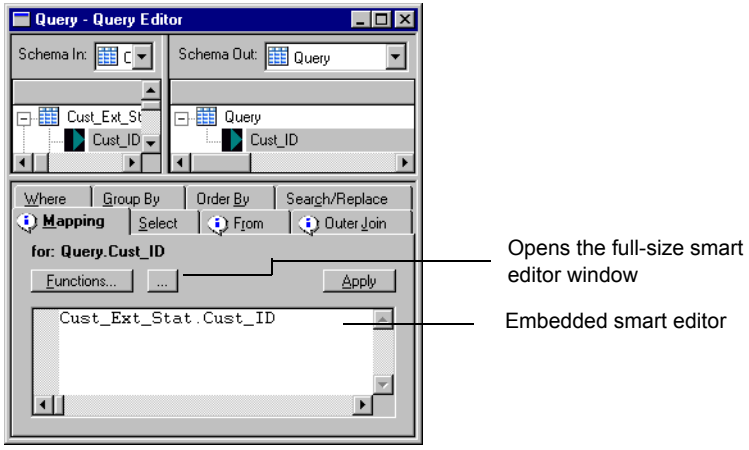
This chapter provides details about options available in the Data Integrator Designer smart editor. Use the smart editor to create scripts, expressions, and custom functions without having to type the names of existing elements like column, function, and variable names.

- For information about formatting scripts and expressions, see Chapter 7, "Data Integrator Scripting Language".
- For information about creating custom functions, see "Custom functions" on page 333.

This chapter contains the following sections:

Accessing the smart editor

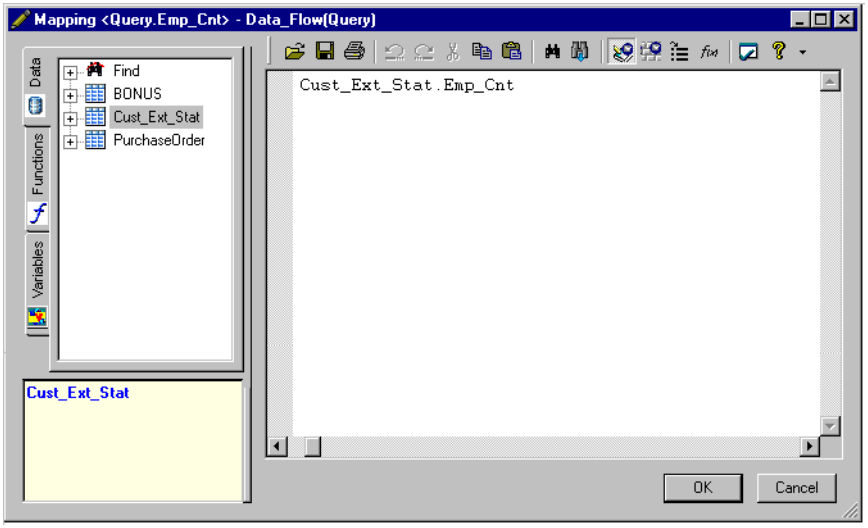
Access and use the embedded smart editor as a pane within any object editor in the Designer, or open the smart editor as a separate, full-size window:



For example, using the **Mapping** tab in a query:

1. Drag a column from an input schema into an output schema to enable the smart editor.
2. Enter text and select options using the smart editor's right-click menu, or click the ellipsis button to open the full-size smart editor window.

When you open the smart editor window, the context of the object from which you opened it is displayed in the title bar. For example, the following example shows a mapping clause for column `Emp_Cnt` in a query called `query`. The query is in a data flow called `Data_Flow`.



You can open the smart editor from the following locations:

- Query Editor **Mapping** tab
- Query Editor **Where** tab
- Script Editor
- Conditional Editor
- While Loop Editor
- Custom Function Editor
- Function wizard, "Define Input Parameter(s)" page
- SQL - Transform Editor
- Case Editor

Smart editor options

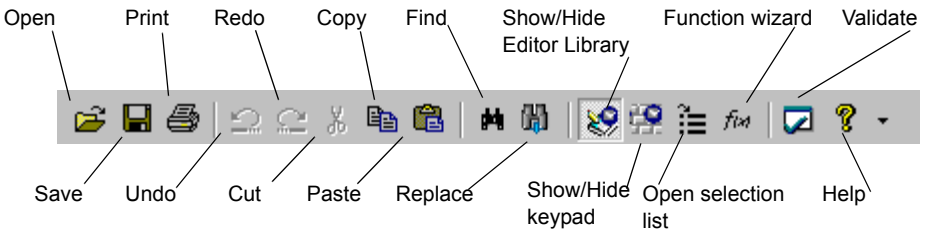
The smart editor window contains a toolbar and two panes:

- Editor Library pane
- Editor pane

Smart editor toolbar

In addition to standard toolbar icons (Open, Save, Print, Undo, Redo, Cut, Copy, Paste, Find, Replace, and Help), the smart editor toolbar also includes special icons to speed up your editing experience. Special smart editor toolbar icons include:

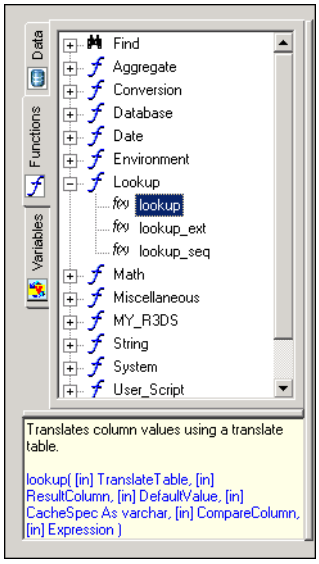
Icon	Description
Show/Hide Editor Library	Toggle to hide or show the Editor Library pane.
Show/Hide keypad	Toggle to hide or show an editing keypad.
Open selection list	Click to show a list of scripting options. Scroll and select the option you want. Double-click or press Enter/Return to add the option to your script.
Function wizard	Click to open the Function wizard window.
Validate	Click to check your script for errors.



Editor Library pane

Use the **Show/Hide Editor Library** icon in the toolbar to show or hide the smart editor library. The library:

- Displays functions, variables, and data using Tabs.
- Allows you to search each tab using the Find option.



Tabs

The **Functions** tab displays existing functions in Data Integrator: built-in, custom, and imported.

The **Variables** tab displays variables, parameters, data type formats, and right-click menu options that can be used in the current context. For example, if you open the smart editor to create a custom function, the **Variables** tab will include options on its right-click menu that you can use to insert, delete, and define properties for new parameters and local variables.

The **Data** tab displays the schemas of data flow sources including nested schemas for the current context. For example, if you open the smart editor from an object in a data flow, such as a WHERE clause of a query, schemas are displayed for connected sources. If you open the smart editor from a script object, the **Data** tab is not displayed.

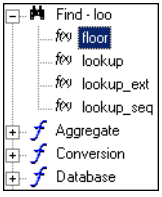
Find option

You can search or browse through each tab to find the content you want to include in your script or expression. For example:

➢ To search for a function

1. Select the position in the editor where you want to place the function.
2. In the **Functions** tab of the editor library, select the **Find** node.
3. Enter a string such as `loo`.
4. Press **Enter** or **Tab**.

   All functions that contain the string are returned under the node.



5. To place the function into the editor, do one of the following:
   - Double-click
   - Drag-and-drop
   - Right-click the function and select **Enter**
   - Select the function and press **Enter** or **Tab**

➢ To browse for a function

1. Expand the nodes to find the function you need:
   - Built-in functions are grouped by type
   - Custom functions are listed under the **User_Script** node

- Imported functions and stored procedures are listed under the name of the datastore used to import them.

2. Click a function and read its description and syntax in the yellow area below the tabs.
3. When you have the function you need, place it into the editor.

Editor pane

The editor pane provides:

- Syntax coloring
- Selection list and tool tips (optional)
- Right-click menu and toolbar
- Validation

Syntax coloring

When you type in the editor pane, the text changes color indicating the type of script language element it represents:

- Quoted strings are shown in pink
- Keywords in blue
- Comments in green
- Functions, operators, and variables are shown in black

Selection list and tool tips

You can use the smart editor with or without showing the editor library. The same list of context-based items available for use when the library is shown is also available in the editor when the selection list is enabled. The selection list shows these items in alphabetical order instead of grouping them into the categories shown in the library. In addition, the selection list displays keywords available for the context in which the editor is opened.

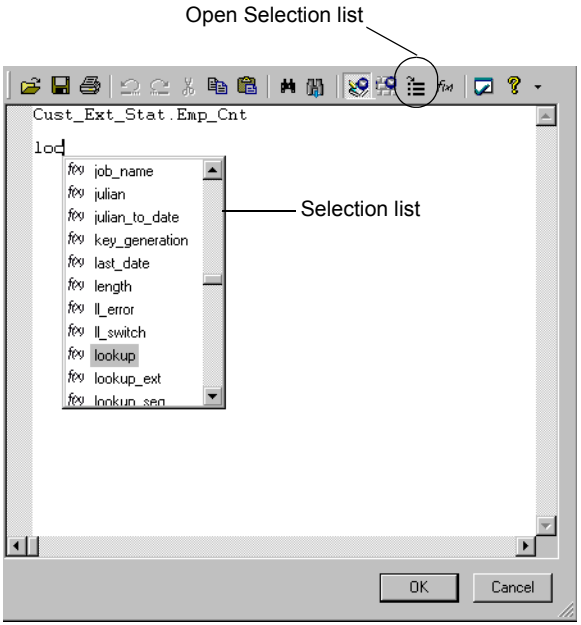
When the selection list is enabled, you can open it from the tool bar. The selection list also opens automatically when it recognizes a string pattern as you type into the editor.

➢ To use the selection list and a tool tip

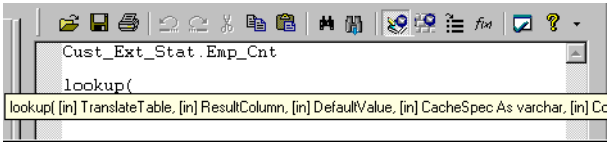
1. Right-click the editor, view the menu, and make sure that the **Enable Selection List** and **Enable Tool Tip** items are selected.
2. In the editor, enter at least three characters, or the dollar sign variable symbol ($).

   The selection list opens over the editor and highlights the first item (in this alphabetized list) that matches the characters you entered.

   Alternatively, you can click the **Open Selection list** icon in the tool bar.



3. Double-click an item in the selection list to insert the item in the editor and view an associated tool tip.



   The tool tip displays the same description, definition, or syntax that you would see if the item were selected from the editor library.

4. Complete the script or expression using these tools.

   For example, if you are completing a look-up function, the tool tip will remain on the screen so that you can follow the syntax of the function. If you enter an input value of the wrong data type, the tool tip closes, indicating an error.

Right-click menu and toolbar

The right-click menu and the tool bar share many commands.

Menu	Toolbar	Key Command
	Open	
	Save As	
	Print	Ctrl + P
Undo	Undo	Ctrl + Z
Redo	Redo	Ctrl + Y
Cut	Cut	Ctrl + X
Copy	Copy	Ctrl + C
Paste	Paste	Ctrl + V
Select All		Ctrl + A
Find	Find	Ctrl + F
Replace	Replace	Ctrl + H
Validate	Validate	
Enable ToolTips		
Enable Selection List		
	Show/Hide Editor library	
	Show/Hide keypad	
	Open selection list	Alt+Down
	Function wizard	
	Help	

Note that:

- **Enable ToolTips** and **Enable Selection List** can only be selected using the right-click menu.
- The library, key pad, selection list, and function wizard can be opened from the tool bar.
- Keyboard shortcuts are available for most commands.

Validation

The smart editor has an embedded error display that includes a script error highlight feature. If validation can occur for the current context, the **Validate** option will be available from the tool bar and right-click menu.

➢ To validate

1. Select the Validate icon in the tool bar or right-click and select **Validate**.

   If errors occur, they are listed in a separate pane below the editor.

2. Double-click each error.
3. The editor redraws to show you where the error occurred in your text.

NOTE: If the **Validate** option is not displayed, the expression must be validated in the context of the whole object. Close the full-size smart editor window. The expression is shown on the embedded smart editor. From the Designer menu select **Debug** > **Validate**.

For more information about validation options, see "Debugging and Validation" on page 520.

4 Data Types

Data types are internal storage formats used to store values. A data type implies a default format for displaying and entering values. *Expressions* are a combination of constants, operators, functions, and variables that evaluate to a value of a given data type.

This chapter discusses how Data Integrator processes, converts, and evaluates data types.

This chapter contains the following sections:

- Descriptions of data types
- Data type processing

Descriptions of data types

Data types are internal storage formats used to store values. Data types also imply certain default formats for displaying and entering values. In Data Integrator:

- Data read from sources is converted to the appropriate Data Integrator data types.
- Data loaded to targets is converted from their Data Integrator data types to types appropriate for the target.

Data Integrator recognizes the following data types:

- date
- datetime
- decimal
- double
- int
- interval
- long
- numeric
- real
- time
- timestamp
- varchar

All of these data types allow NULL values. The following sections describe their use in Data Integrator.

date

The `date` data type defines calendar dates.

Data Integrator automatically converts date values to and from the formats used by an external DBMS. Conversion operations from strings to dates, or from dates to strings or numbers require you to specify the format of the `date` value. To specify a date format, generate a string from the following codes and other literal strings or punctuation.

Date format code	Description	Example
DD	2-digit day of the month value (1-31)	The 2nd day of the month: `02`
MM	2-digit month number (1-12)	The month of March: `03`
MONTH	Full name of the month Data Integrator always generates month names in lower case.	The first month of the year: `january`
MON	Abbreviated 3-character name of month Data Integrator always generates month names in lower case.	The first month of the year: `jan`
YY	2-digit year A `YY` less than 15 is interpreted as being 20*yy*; for example, 10 would be interpreted as being the year 2010. A `YY` greater than or equal to 15 is interpreted as being 19*yy*; for example, 35 would be interpreted as being the year 1935. To change the value that Data Integrator uses to interpret 2-digit year dates, change the **Century change year** value in the Data options. (Select **Tools** > **Options** to open the Options window, and then select **General** under the **Data** category). The value must be a positive integer between 0 and 99.	The year 1998: `98`
YYYY	4-digit year	The year 1999: `1999`

When you enter a date, use the default date format `YYYY.MM.DD`.

You can add and subtract date, datetime, interval, and time values; see "Date arithmetic" on page 207.

The following examples show the use of date formats with Data Integrator functions. The value of the variable `MyDate` is the first day of 1996.

Example	Output
`to_char($MyDate, 'YYYY.MM.DD')`	`1996.01.01`
`to_char($MyDate, 'MONTH DD, YYYY')`	`january 01, 1996`
`to_char($MyDate, 'DD/MM/YY')`	`01/01/96`
`to_date('01/01/96','DD/MM/YY')`	`1996.01.01` stored as a date
`to_date('01/01/19','DD/MM/YY')`	`2019.01.01` stored as a date

datetime

The `datetime` data type defines calendar dates and times.

Data Integrator manages date operations in the format used by your DBMS. Conversion operations to or from `datetime` values require you to specify the format of the `datetime`. This data type behaves like a concatenation of two data items: The rules for the datetime type are the `date` rules for the date part, and the `time` rules for the time part.

If a `date` is converted to a `datetime` value, the default time added to the value is 00:00:00. If a `time` is converted to a `datetime` value, the default date added to the value is 1900.01.01. You can also add and subtract date, datetime, interval, and time values; see "Date arithmetic" on page 207.

When converting datetime values to strings, you can choose the sections of the value not to convert by excluding them from the format description. For example, to convert a datetime value to a string containing only the time, specify the function parameters as follows:

```
to_char($MyDateTime, 'hh24:mi:ss.ff')
```

For Oracle, if you load datetime data from Data Integrator into a char field in an Oracle table, Oracle puts the data in its default datetime format—which includes only date values—and loses the time from the value.

This data type allows NULL values.

decimal

The `decimal` data type defines exact decimal numbers.

When specifying a decimal data type in Data Integrator, you indicate the following characteristics of the type:

precision	The total number of digits in the value.
scale	The number of digits to the right of the decimal point.

In Data Integrator, the following relations must hold for *precision* and *scale*:

1 <= *precision* <= 28

0 <= *scale* <= *precision*

The decimal value can have a plus or minus sign indicating a positive or negative value. The sign can appear before or after the value with any number of blanks between the value and the sign. Unsigned values are considered to be nonnegative. The sign does not count as part of *precision*.

Leading zeros are permitted in the integer digit, and trailing zeros are permitted in the fraction part.

Input that is more precise than the data type of the column or variable in which it is stored is rounded. Input out of range (absolute value is too large) causes a runtime error.

This data type allows NULL values.

The decimal data type and the numeric data type are identical in Data Integrator.

double

The `double` data type defines an 8-byte floating point value, with radix, exponent range, and precision of the platform on which Data Integrator is running.

For more information, consult the appropriate reference material for Windows NT.

This data type allows NULL values.

int

The `int` (integer) data type defines a 4-byte signed binary integer.

The int value can have a plus or minus sign indicating a positive or negative value. The sign can appear before or after the value with any number of blanks between the value and the sign. Unsigned values are considered to be nonnegative.

This data type allows NULL values.

interval

The interval data type defines differences between dates or times. The value is in days unless you specify another unit, such as in a conversion function.

Data Integrator provides conversion functions to make interval values accessible: `interval_to_char` and `num_to_interval`.

You can add and subtract date, datetime, interval, and time values; see "Date arithmetic" on page 207.

This data type allows NULL values.

long

The LONG datatype stores character strings longer than 4000 bytes. It can also be used to store character strings with less than 4000 bytes. However, the restrictions for a LONG data type always apply.

The Data Integrator LONG data type supports mapping from:

- Oracle LONG data type

  The Oracle LONG data type stores variable-length character data containing up to two gigabytes of information.

- SQL_LONGVARCHAR data type

  SQL data types store data in a data source. Each data source defines its own SQL data type. To identify SQL data types, use an ODBC datastore connection. It defines type identifiers and describes the general characteristics of the SQL data types that might be mapped to each type identifier.

  The SQL_LONGVARCHAR data type stores variable-length character strings. The maximum length is data-source dependent.

  You can use a SQL_LONGVARCHAR data type column to map to columns using the following database data types if Data Integrator access them using a ODBC datastore:

Database name	Data type (s)
Access	MEMO
DB2	CLOB, LONG VARCHAR
Informix	CLOB
Microsoft SQL Server	TEXT
Oracle	CLOB
Sybase	TEXT

IMPORT

Data Integrator can successfully import metadata for an Oracle or ODBC table containing a long column. In earlier versions of the product (before 4.4), the metadata for the table bypassed the long columns.

MAP

When defining a Query transform in Data Integrator, you can map a column that contains a LONG data type to a:

- LONG type column in an Oracle table
- SQL_LONGVARCHAR type column in an ODBC table
- Long type column in an Data Integrator delimited file format

CACHE

You can store Oracle, ODBC tables, and a delimited file format containing a long column in memory when using Data Integrator.

TRANSFORM

You can perform multiple transforms on Oracle or ODBC tables that contain a LONG data type in a column, however there are several restrictions. Transforms are successful when the long column is not used in an expression evaluation.

Long columns cannot be compared. Subsequently, long columns will fail as primary key columns or compare columns in Table_Comparison or History_Preserving transforms.

Long columns, when used as non-key and non-compare columns, cause the transform to forego modifications to long column values in the source table. In other words, the Table_Comparison or History_Preserving transforms will not generate updates or insert information to the target table.

Long columns that appear in the SQL_TEXT of a SQL transform will generate correct results only when long column usage is correct. Incorrect usage generates a database error upon execution of a SQL statement in the SQL_TEXT.

Here is a list of additional restrictions for the long data type in Data Integrator transforms

Transform restrictions.

Transforms	Long column data can NOT be used in:
Case	case condition
Hierarchy_Flattening	parent and child columns
History_Preserving	compare columns
Pivot	pivot
Table_Comparison	primary key columns or compare columns

Retrieve and store

Data Integrator supports the retrieval and storage from-and-to Oracle, ODBC tables, and a delimited file format when they contain a long column.

Column adding and mapping

Using the Data Integrator Designer you can add and set a long data type. To add or map a column of the long data type:

1. From the target schema section of the query transform window, right-click and select **New Column Output** (to add) or **Properties** (to map) a column.

   The Column Properties window appears.

2. In the Column Properties window, select **Long** from the **Data Type** list.

General restrictions

These restrictions do not reflect limitations in Data Integrator, but represent Oracle or ODBC functionality.

For example, ODBC requires that applications place long data columns at the end of the SELECT statement. Otherwise, the operation involving a long column produces an error.

You can use LONG columns only for storage and retrieval of long strings. Long columns cannot be used in Data Integrator as:

- Join columns
- SQL functions—for example, substr
- Expressions and conditions
- SELECT lists of queries containing GROUP BY clauses
- SELECT lists of queries with the Distinct Rows option enabled
- GROUP BY, ORDER BY, or WHERE clauses
- Input or output parameters of Data Integrator functions—for example, lookup.
- Data Integrator variable data types
- Work flow and data flow input and output parameters

There are additional restrictions for batch loads.

Auto correct load restriction

The auto-correct load option does not support Oracle and ODBC target tables that contain a long column. Data Integrator ignores the option. Data Integrator resets the option and issues a warning message at runtime to indicate that the auto-correct load option has been disabled.

LOAD ROWS PER COMMIT RESTRICTION

To reduce memory contention, Data Integrator sets the value for a loader's Rows per commit option to 1 for Oracle or ODBC target tables that contain a long column. As a result, array or batch load functions are not available for an Oracle or ODBC target table that contains a long column. Loading in bulk may resolve the problem.

OPTIMIZER AND PUSH DOWN LOAD OPERATION TO READER RESTRICTION

A load operation that uses a long column is not pushed down when the following conditions hold:

- The datastore is an Oracle database
- The SELECT list contains a long column

Typically, Data Integrator optimization pushes load operations into the database when all of the following conditions, combined, hold:

- Both source and destination share the same datastore
- The datastore is a database
- There is only one SQL loader (or database) in the data flow

Optimization, as a result, combines the operation into a single INSERT statement in which a sub-query reads the data from the source table. However, since a long column cannot appear in the select list of the sub-query of an INSERT statement when the datastore is an Oracle database, the load operation is executed as a separate step.

numeric

The decimal data type and the numeric data type are identical in Data Integrator. See "decimal" on page 190 for data type information.

real

The real data type defines a 4-byte floating point value, with radix, exponent range, and precision of the platform on which Data Integrator is running.

The real value can have a plus or minus sign indicating a positive or negative value. The sign can appear before or after the value with any number of blanks between the value and the sign. Unsigned values are considered to be nonnegative.

This data type allows NULL values.

Databases store real values as a 32-bit approximation of the number. Because of this approximation, comparison results are unpredictable when a real value is used in an equality or inequality comparison.

Therefore, Business Objects recommends that you do not use a real value in a WHERE clause. Real values appear in WHERE clauses that Data Integrator generates when a column of type real is used:

- As a compare column in the Table_Comparison transform
- In the Map_Operation transform with an opcode of update or delete
- Explicitly in the WHERE clause of a Query transform

In some cases, columns of type real might unintentionally appear in the WHERE clause of these transforms. For example, when no compare columns are specified in a Table_Comparison transform, the transform uses all columns of the table as compare columns. Similarly, if the source of a Map_Operation transform does not have primary key specified and the opcode is update or delete, the transform uses all source columns in the WHERE clause of the UPDATE or DELETE statement.

Use caution when using the real data type in these transforms.

For more information, consult the appropriate reference material for Windows NT.

time

The time data type defines times of the day, with no calendar date.

Data Integrator manages time operations in the format used by your database manager. Conversion operations to or from times require you to specify the format of the time value. To specify a time format, generate a string from the following codes and other literal strings or punctuation.

Time format code	Description
`HH24`	2-digit hour of the day (0-23)
`MI`	2-digit minute (0-59)
`SS`	2-digit second (0-59)

The following examples show the use of time formats with Data Integrator functions. The value of the variable `MyTime` is 25 minutes after 8 in the evening.

Example	Output
`to_char($MyTime, 'HH24:MI:SS.FF')`	`20:25:00`
`to_char($MyTime, 'HH24:MI')`	`20:25`

You can add and subtract date, datetime, interval, and time values; see "Date arithmetic" on page 207.

This data type allows NULL values.

timestamp

The timestamp data type supports the timestamp (with no zone information) data type in Oracle 9i. The timestamp data type incorporates up to a 9-digit sub-second.

ARITHMETIC

Add or subtract timestamp values. The resulting data type from addition or subtraction operations depends on the operation and data types involved. See Date arithmetic for details.

CONVERSION BETWEEN TIMESTAMP AND CHARACTER STRINGS

You can convert between timestamp values and character values using the to_date and to_char functions. These functions have a format called FF which indicates the sub-second digits. For example, valid function calls are:

```
to_date ('2002.02.26 01234004 09:03:25','yyy.mm.dd ff hh24:mi:ss')

to_char (timestamp_column, 'yyyy.mm.dd hh24:mi:ss.ff')
```

Use the FF format for datetime columns to access sub-seconds. For example, a DB2 timestamp column is mapped to datetime in Data Integrator. This column contains micro-second. You can access these sub-seconds using the FF format.

LIMITATIONS

You cannot use timestamp columns in the SQL transform or in an Oracle stored procedure.

To use a timestamp column in the SQL transform, convert the timestamp column in the select list of the SQL transform to a character format using the to_char function and convert it back to timestamp using the to_date function.

To use a timestamp column in an Oracle stored procedure, convert input and output timestamp parameters in the stored procedure to char, using the to_char function and convert the output parameter back to timestamp in Data Integrator using the to_date function. Alternatively, you can convert the input parameter back to timestamp in the stored procedure using the Oracle to_timestamp function.

varchar

The varchar data type defines a variable-length character string with a specified maximum length.

When specifying a varchar data type in Data Integrator, you indicate the following characteristic of the type:

length — Number of characters that the variable or column can hold. *Length* must be greater than zero. There is no maximum allowable value for the *length*.

Character strings longer than the number of characters defined for the column or variable are truncated on the right to the length of the data type. Only the required number of characters is used to store strings shorter than *length*.

When reading from sources with string data types, Data Integrator removes trailing blanks. If a string value contains all blanks, the resulting varchar is length one with a single blank.

If you concatenate a varchar value with a NULL varchar value, the result is NULL.

Data Integrator provides functions to convert values to and from strings; to join strings together, use the concatenation operator (||). This data type allows NULL values.

Data Integrator supports reading, transforming, and loading National Language Supported (NLS) data from different language locales using the varchar data type. National character-set data types in the following databases are supported:

Database	Version	National character-set data type
DB2	7.0 and higher	`graphic`, `vargraphic`
MS SQL Server	7.0 and higher	`nchar`, `nvarchar`
Oracle	9i and higher	`nchar`, `nvarchar2`

When Data Integrator encounters a national character-set data type in an expression, it binds the column with the data type.recommended by the database. Likewise, when using the Metadata Exchange command in the Designer, the data type used in the database (not Data Integrator's varchar data type) is passed on to a BusinessObjects universe.

The Data Integrator engine reads and loads national character-set data types seamlessly without the need for you to configure a locale for a database client and its datastore for the columns that use these data types. For more information, see "Processing with and without UTF-16 Unicode" on page 555.

Data type processing

This section discusses how Data Integrator processes various data types—conversions during arithmetic operations and between data types:

- Date arithmetic
- Type conversion

Date arithmetic

Data Integrator performs some implicit data type conversions on date, time, datetime, timestamp, and interval values when performing date arithmetic. The following table describes these conversions:

Operation	Return type
DATE + INTERVAL	DATE
TIME + INTERVAL	TIME
DATETIME + INTERVAL	DATETIME
INTERVAL + INTERVAL	INTERVAL
DATE - DATE	INTERVAL
DATE - INTERVAL	DATE
TIME - TIME	INTERVAL
TIME - INTERVAL	TIME
INTERVAL - INTERVAL	INTERVAL
DATETIME - DATE	INTERVAL
DATETIME - TIME	INTERVAL
DATETIME - DATETIME	INTERVAL
DATETIME - INTERVAL	DATETIME
TIMESTAMP + TIMESTAMP	INTERVAL
TIMESTAMP - TIMESTAMP	INTERVAL
TIMESTAMP + INTERVAL	TIMESTAMP
TIMESTAMP - INTERVAL	TIMESTAMP

Type conversion

There are many opportunities for data type conversion in Data Integrator applications:

- Conversion to/from Data Integrator internal data types
- Conversion of data types within expressions
- Conversion among number data types
- Conversion between explicit data types

Conversion to/from Data Integrator internal data types

Data Integrator performs data type conversions when it imports metadata for external sources or targets into the repository and when it loads data into an external table or file. These conversions are performed using Data Integrator conversion functions rather than the conversion functions specific to the database or application that is the source of the data.

When encountering a column assigned to an unsupported data type, Data Integrator does not import the metadata for the column and indicates an error. The file `errorlog.txt` in the `Log` directory of the Data Integrator installation contains an entry indicating the column ignored.

To include the column in your Data Integrator application, convert the data type to one supported by Data Integrator before importing metadata for the table.

Data Integrator strips trailing blanks from string data types. A string that contains only blanks becomes a single blank.

The following table lists the conversions from database and application data types to Data Integrator internal data types.

Conversions from external sources to Data Integrator

Source	External data type	Is converted to
DB2	bigint	int **NOTE:** Because int is only four bytes, data is lost during the conversion.
	character	varchar
	clob	Converted to long only if imported using an ODBC datastore. For more information, see "long" on page 194.
	date	date
	decimal	decimal
	double	double
	float	double
	graphic	varchar
	integer	int
	long varchar	Converted to long only if imported using an ODBC datastore. For more information, see "long" on page 194.
	real	real
	smallint	int
	time	time
	timestamp	datetime
	varchar	varchar
	vargraphic	varchar
DETAIL_DB2	date	datetime
	decimal	decimal
	fixchar	varchar
	float	double
	integer	int

Conversions from external sources to Data Integrator (Continued)

Source	External data type	Is converted to
	smallint	int
	time	time
	timestamp	datetime
	varchar	varchar
DETAIL_NRDB	bin	not supported
	char	varchar
	num16	real
	num16u	real
	num32	real
	num32u	real
	num8u	real
	numchar	real
	packed	decimal
		NOTE: The packed type permits larger values than the Business Objects decimal type
	upacked	decimal
	uzoned	decimal
	varbin	not supported
	varchar	varchar
	zoned	decimal

Conversions from external sources to Data Integrator (Continued)

Source	External data type	Is converted to
Informix	byte	not supported
	char	varchar
	character	varchar
	character varying	not supported
	clob	Converted to long only if imported using an ODBC datastore. For more information, see "long" on page 194.
	date	date
	datetime	datetime
	dec	decimal
	decimal	decimal
	double	double
	float	double
	int	integer
	integer	integer
	money	decimal
	numeric	decimal
	real	real
	serial	not supported
	smallfloat	double
	smallint	integer
	text	varchar
	varchar	varchar

Conversions from external sources to Data Integrator (Continued)

Source	External data type	Is converted to
Microsoft SQL Server	binary	unsupported
	bit	int
	char	varchar
	datetime	datetime
	decimal	decimal
	double	double
	float	double
	image	unsupported
	int	int
	money/smallmoney	decimal
	nchar	varchar
	numeric	decimal
	nvarchar	varchar
	real	real
	smalldatetime	datetime
	smallint	int
	text	Converted to long only if imported using an ODBC datastore. For more information, see "long" on page 194.
	timestamp	unsupported
	tinyint	int
	varbinary	unsupported
	varchar	varchar

Conversions from external sources to Data Integrator (Continued)

Source	External data type	Is converted to
ODBC	bigint	int
	char	varchar
	datalink	unsupported
	date	date
	decimal	decimal
	double	double
	float	double
	graphic	unsupported
	int	int
	lob	unsupported
	numeric	decimal
	real	real
	sql_longvarchar	long
	time	time
	timestamp	datetime
	tinyint	int
	user-defined	unsupported
	varchar	varchar

Conversions from external sources to Data Integrator (Continued)

Source	External data type	Is converted to
Oracle	char	varchar
	clob	Converted to long only if imported using an ODBC datastore. For more information, see "long" on page 194.
	date	datetime
	decimal	decimal
	doubleprecision	double
	float	double
	label	unsupported
	long	long
	long row	unsupported
	nchar	varchar
	number	int: If scale is 0 and precision is < 9 decimal: All other
	nvarchar2	varchar
	real	double
	row	unsupported
	rowid	unsupported
	string	varchar
	timestamp	timestamp
	varchar	varchar
	varchar2	varchar

Conversions from external sources to Data Integrator (Continued)

Source	External data type	Is converted to
Sybase	binary	unsupported
	bit	int
	char	varchar
	datetime	datetime
	decimal	decimal
	double	double
	float	double
	image	unsupported
	int	int
	money	decimal(20,4)
	numeric	decimal
	real	real
	smalldatetime	datetime
	smallint	int
	smallmoney	decimal(12,4)
	text	Converted to `long` only if imported using an ODBC datastore. For more information, see "long" on page 194.
	timestamp	unsupported
	tinyint	int
	varbinary	unsupported
	varchar	varchar

Conversions from external sources to Data Integrator (Continued)

Source	External data type	Is converted to
Teradata	byteint	int
	char varying (n)	varchar(n)
	char [(n)]	varchar(n)
	date	date
	decimal	decimal
	double precision	float
	float	float
	int	int
	long varchar	varchar(64000)
	numeric	decimal
	real	float
	smallint	int
	time	time
	timestamp	datetime
	varchar(n)	varchar (n)

The following table lists the conversions from Data Integrator internal data types to external data types for particular databases and applications.

Internal data type	External data types						
	DB2	Informix	MS SQL Server	ODBC	Oracle	Sybase	Teradata
date	date	date	datetime	date	date	datetime	date
datetime	date	date	datetime	date	date	datetime	timestamp
decimal	decimal	decimal	decimal	decimal	decimal	decimal	decimal, numeric
double	double	float	float	double	double	float	N/A
int	int	int	int	int	int	int	int, smallint, byteint
interval	real	int	real	real	read	real	N/A
long	long varchar	clob	text	sql_longvarchar	long	text	long varchar
numeric	character	decimal	decimal	char	number	decimal	N/A
real	real	real	real	real	double	real	N/A
time	time	date	datetime	time	date	datetime	time
varchar	varchar graphic[a] vargrapic[a]	varchar	varchar nchar[b] nvarchar[b]	varchar	varchar2 nchar[c] nvarchar2[c]	varchar	varchar, char, char varying
timestamp	date	date	datetime	date	timestamp[c] date[d]	datetime	N/A

a. DB2 7.0

b. MS SQL Server 7.0

c. Oracle 9i

d. Oracle 8

Conversion of data types within expressions

When possible, Data Integrator optimizes data flows by pushing expressions down to an underlying database manager. In a single transaction, Data Integrator can push down expressions so that they are performed by the underlying database manager. However, when Data Integrator evaluates an expression which includes operands of more than one data type, Data Integrator attempts to convert the operands to the same data type first. (Except for national character-set data types which can be pushed down while others in an expression are not. For more information about supported national character-set data types, see "varchar" on page 205).

When a conversion is required, Data Integrator provides a message at validation.

If the conversion is illegal, Data Integrator provides an error and you must remove the mismatch before executing the job.

If the conversion is legal, Data Integrator provides a warning indicating that it will not interrupt job execution.

NOTE: When Data Integrator converts a data type to evaluate an expression, the results might not be what you expect. To avoid legal but incorrect conversions, always validate before executing and examine the circumstance of any data type conversion warnings.

Conversion among number data types

Data Integrator uses a type-promotion algorithm to evaluate expressions that contain more than one number data type. Number data types are ranked from highest to lowest, as follows:

- decimal
- double
- real
- int

If Data Integrator encounters expressions that have more than one number data type among the operands, it converts all of the operands to the data type of the operand with the highest ranking type.

For example, if A is an `int` and B is a `double`, the expression `A+B` is evaluated by first converting A to `double` and then adding the two `double` values. The result is type `double`.

If in addition to A and B, you multiply the result by a `decimal` number C, then `(A+B)*C` is evaluated by first converting `(A+B)` to `decimal`, and then performing the indicated operations on the two `decimal` values. The result is type `decimal`.

For addition, subtraction, and multiplication, the operation result will be equal to the higher of the two operands. For example:

```
int + double = double
```

For division, Data Integrator uses the following algorithm:

Numerator data type	**Denominator data type**			
	int	real	double	decimal(p,s1)
int	double	double	double	decimal(p,s1)
real	double	double	double	decimal(p,s1)
double	double	double	double	decimal(p,s1)
decimal(p,s2)	decimal(p,s2)	decimal(p,s2)	decimal(p,s2)	decimal(p,max(10,s1,s2))

Conversion among decimals of different scale or precision

If decimals of two different scales are included in a single expression, Data Integrator uses the higher of the two scales. For example:

```
decimal(5,4) * decimal(7,2) = decimal(7,2)
```

Expect a loss of precision when operating on two `decimals` of different scale values. For example, when adding a `decimal(28, 26)` to a `decimal(28,1)`, the resulting decimal value has the lower of the two scale values:

```
400000.5 + 40.00005 = 400040.5
```

The least scale for division involving a decimal is 10.

Conversions between strings and numbers

When Data Integrator encounters a string where a number would normally be expected (for example, in mathematical operations or functions that expect numeric arguments), it will attempt to convert the string to a number.

For multiplication and division operations, operands are converted to numbers. Other promotion algorithms are shown in the following table.

Provided data type	Data type required to evaluate the expression			
Addition				
	Number	Date/time	String	
Number	OK (promoted)	Number to interval	String to number	
Date/time	Number to interval	Illegal	String to interval	
String	String to number	String to interval	String to real	
Subtraction				
	Number	Date/time	String	Interval
Number	OK (promoted)	Illegal	String to number	Interval to number
Date/time	Number to interval	OK	String to interval	OK
String	String to number	String to datetime	String to real	String to number
Interval	Number to interval	Illegal	String to interval	OK
Comparison				
	Number	Date/time	String	
Number	OK (promoted)	Illegal	String to number	
String	Illegal	OK	String to datetime	
Interval	Interval to number	Illegal	Interval to number, string to number	

Conversions between strings and dates

For Oracle, if you load datetime data from Data Integrator into a char field in an Oracle table, Oracle puts the data in its default datetime format—which includes only date values—and loses the time from the value.

Conversion between explicit data types

You can use Data Integrator functions to convert data from one type to another:

- interval_to_char
- julian_to_date
- num_to_interval
- to_char
- to_date
- to_decimal

You can also import database-specific functions to perform data type conversions.

5 Transforms

A transform is a step in a data flow that acts on a data set. Built-in Data Integrator transforms are available through the object library. This chapter discusses built-in transforms.

This chapter contains the following sections:

Operation codes

Data Integrator maintains operation codes that describe the status of each row in each data set described by the inputs to and outputs from objects in data flows. The operation codes indicate how each row in the data set would be applied to a target table if the data set were loaded into a target. The operation codes are as follows:

Operation Code	Description
NORMAL	Creates a new row in the target. All rows in a data set are flagged as NORMAL when they are extracted by a source table or file. If a row is flagged as NORMAL when loaded into a target table or file, it is inserted as a new row in the target. Most transforms operate only on rows flagged as NORMAL.
INSERT	Creates a new row in the target. Rows can be flagged as INSERT by the Table_Comparison transform to indicate that a change occurred in a data set as compared with an earlier image of the same data set. The Map_Operation transform can also produce rows flagged as INSERT. Only History_Preserving and Key_Generation transforms can accept data sets with rows flagged as INSERT as input.
DELETE	Is ignored by the target. Rows flagged as DELETE are not loaded. Rows can be flagged as DELETE only by the Map_Operation transform.
UPDATE	Overwrites an existing row in the target table. Rows can be flagged as UPDATE by the Table_Comparison transform to indicate that a change occurred in a data set as compared with an earlier image of the same data set. The Map_Operation transform can also produce rows flagged as UPDATE. Only History_Preserving and Key_Generation transforms can accept data sets with rows flagged as UPDATE as input.

Descriptions of transforms

The following transforms are available from the object library on the **Transforms** tab. This chapter describes these transforms in detail.

Transform	Description
Case	The Case transform simplifies branch logic in data flows by consolidating case or decision making logic in one transform. Paths are defined in an expression table.
Date_Generation	Generates a column filled with date values based on the start and end dates and increment that you provide.
Effective_Date	Generates an additional "effective to" column based on the primary key's "effective date."
Hierarchy_Flattening	Flattens hierarchical data into relational tables so that it can participate in a star schema. Hierarchy flattening can be both vertical and horizontal.
History_Preserving	Converts rows flagged as UPDATE to UPDATE plus INSERT, so that the original values are preserved in the target. You specify in which column to look for updated data.
Key_Generation	Generates new keys for source data, starting from a value based on existing keys in the table you specify.
Map_CDC_Operation	While commonly used to support Oracle or mainframe changed-data capture, this transform supports any data stream if its input requirements are met. Sorts input data, maps output data, and resolves before- and after-images for UPDATE rows.
Map_Operation	Allows conversions between operation codes.
Merge	Unifies rows from two or more sources into a single target.
Pivot (Columns to Rows)	Rotates the values in specified columns to rows. (Also see Reverse Pivot.)
Query	Retrieves a data set that satisfies conditions that you specify. A query transform is similar to a SQL SELECT statement.
Reverse Pivot (Rows to Columns)	Rotates the values in specified rows to columns.
Row_Generation	Generates a column filled with int values starting at zero and incrementing by one to the end value you specify.
SQL	Performs the indicated SQL query operation.
Table_Comparison	Compares two data sets and produces the difference between them as a data set with rows flagged as INSERT and UPDATE.

NOTE: For all transforms, to refresh a target schema after making changes to transform options, choose **View** > **Refresh** or press **F5**.

Case

Specifies multiple paths in a single transform (different rows are processed in different ways).

The Case transform simplifies branch logic in data flows by consolidating case or decision making logic in one transform. Paths are defined in an expression table.

Data Inputs

Only one data flow source is allowed.

Options

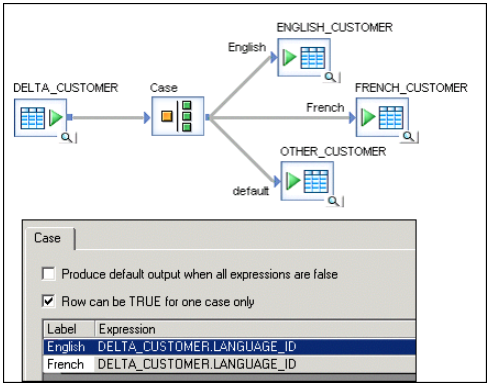
Label — Name of the connection description indicating where data will go if the corresponding CASE condition is true.

Expression — CASE expression for the corresponding label. DEFAULT is the expression used when all other CASE expressions evaluate to false. To enable DEFAULT, select the **Produce output when all expressions are false** check box.

Default — The expression used when all other CASE expressions evaluate to false. To enable, select the **Produce default output with label:** check box and enter a label in the corresponding field.

True — If the **Row can be TRUE for one case only** option is enabled, the row is passed to the first case whose expression returns TRUE. Otherwise, the row is passed to all the cases whose expression returns TRUE.

For jobs created in the 6.0 release or earlier, this option is set to FALSE. When a Case transform is created in 6.1, this option defaults to TRUE.



EDITOR

The Case transform editor includes an expression table and a smart editor.

➢ To use the Case transform editor

1. Use the buttons, or right-click the expression table to insert or delete cases.

   While using this table, the window also allows you to:

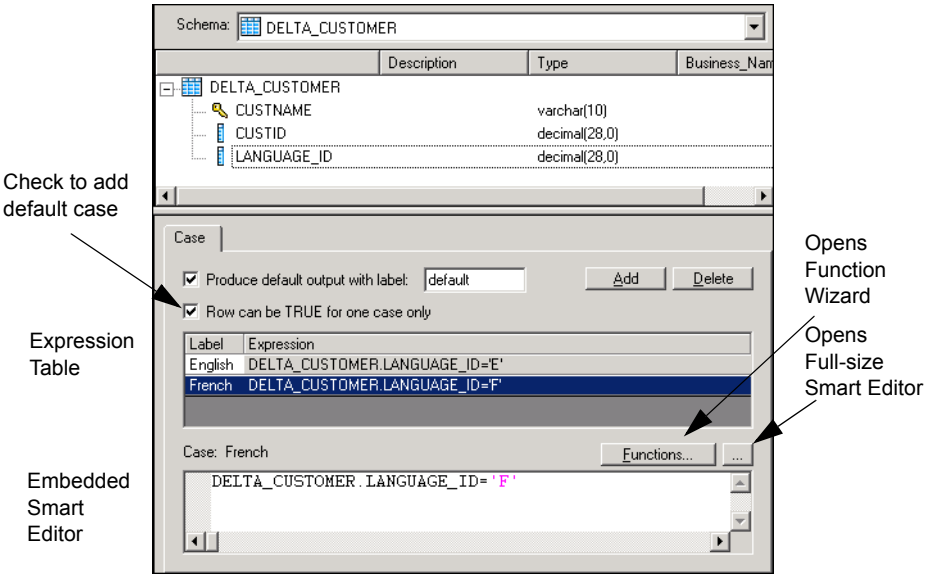
   - Select multiple rows
   - Apply delete functionality to a multiple selection
   - Press Delete or Insert keys to delete or add an expression

2. In the expression table, click a label to rename it.
3. Enter the expression in the editor (drag columns from the input schema).

   For large expressions, open the smart editor. Both the smart editor and the function wizard can assist you with expression creation.

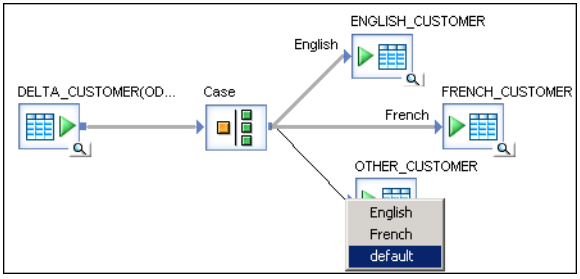
4. To add a DEFAULT case, select the **Produce default output with label:** check box.

   When you add a DEFAULT case, Data Integrator will send rows to this case when all other case conditions are false.



Data Outputs

Connect the output of the Case transform with another object in the workspace. Choose a case label from a pop-up menu. Each label represents a case expression (WHERE clause) created in the Case editor.



The connections between the Case transform and objects used for a particular case must be labeled. Each output label in the Case transform must be used at least once.

To delete a case connection, right-click the connection label and select **Delete**.

The Case transform can be used to implement IF-THEN-ELSE logic rather than using a conditional flow. However:

- Conditionals operate at the work flow level
- Case transforms operate within data flows

Date_Generation

Produces a series of dates incremented as you specify.

Use this transform to produce the key values for a time dimension target. From this generated sequence you can populate other fields in the time dimension (such as day_of_week) using functions in a query.

Data Inputs

None.

Options

Start date — The first date in the output sequence. Specify this date using the following format:

`YYYY.MM.DD`

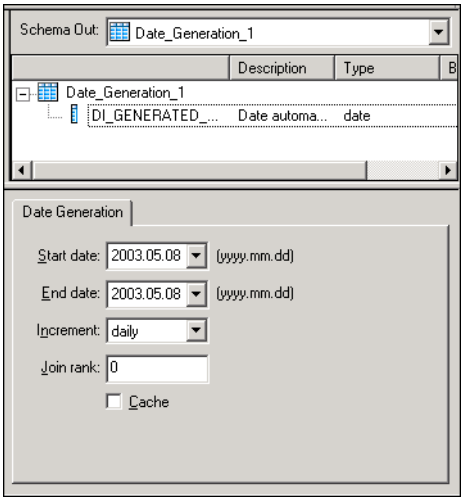
where `YYYY` a year, `MM` is a month value, and `DD` is a day value.

End date — The last date in the output sequence. Use the same format used for **Start date** to specify this date.

Increment — The interval between dates in the output sequence. Select **Daily**, **Monthly**, or **Weekly**.

Join rank — A positive integer indicating the weight of the output data set if the data set is used in a join. Sources in the join are accessed in order based on their join ranks. The highest ranked source is accessed first to construct the join.

Cache — Select this check box to hold the output from the transform in memory to be used in subsequent transforms. Select **Cache** only if the resulting data set is small enough to fit in memory.



DATA OUTPUTS

A data set with a single column named `DI_GENERATED_DATE` containing the date sequence. The rows generated are flagged as INSERT.

The Date_Generation transform does not generate hierarchical data.

Generated dates can range from 1900.01.01 through 9999.12.31.

EXAMPLE

To create a time dimension target with dates from the beginning of the year 1997 to the end of the year 2000, place a Date_Generation transform, a query, and a target in a data flow. Connect the output of the Date_Generation transform to the query, and the output of the query to the target. Inside the Date_Generation transform, specify the following Options:

- **Start date**: 1997.01.01
- **End date**: 2000.12.31

- **Increment**: Daily

Inside the query, create two target columns and the field name, and define a mapping for these time dimension values:

- Business quarter: `BusQuarter`
  Function: `quarter(Generated_date)`
- Date number from start: `DateNum`
  Function: `julian(generated_date) - julian(1997.01.01)`

Effective_Date

Calculates an "effective-to" value for data that contains an effective date. The calculated effective-to date and an existing effective date produce a date range that allows queries based on effective dates to produce meaningful results.

DATA INPUTS

Data that has an effective date column.

Effective dates allow you to indicate changes to information over time. The effective date value in each row of a data set indicates the date from which the data in the row is valid. As changes are made to the information, more rows are included to describe the information as it changes over time. Each row describing the set of information is distinguished from the others by the effective date of the row.

An example input data set might include a column that identifies the information being described (Project), a column that changes over time (Status), and an effective date:

Project	Effective date	Status
Cherry Lake	1999.06.22	Proposal
Cherry Lake	2003.01.12	Case
Hetch Hetchy Reservoir	1999.08.02	Proposal
Hetch Hetchy Reservoir	2003.05.06	Case

This transform description uses the term "related rows" to refer to a set of rows that describe the same information as it changes over time. There are two sets of related rows in the example above, described by the values in the Project column.

If the input data set allows duplicate effective dates, it might contain an effective sequence column to distinguish between related rows that also have the same effective date:

Project	Effective date	Effective sequence	Status
Cherry Lake	1999.06.22	0	Proposal
Cherry Lake	2002.01.12	0	Case
Hetch Hetchy Reservoir	1999.08.02	0	Proposal
Hetch Hetchy Reservoir	2002.10.17	0[a]	Proposal
Hetch Hetchy Reservoir	2002.10.17	1[a]	Case
Hetch Hetchy Reservoir	2003.05.06	0	Case

a. Project statuses are distinguished by the effective sequence.

In the example, only the row with the largest sequence number is effective-dated by this transform. A query that selects the status of project "Hetch Hetchy Reservoir" on 2002.12.31 will return 'case' as a result. The input data set can contain only rows flagged as NORMAL.

The input data set can contain hierarchical data. The transform operates only on the rows at the top-level of the input data set, and passes nested data through to the output without change. Columns containing nested schemas cannot be used as transform parameters.

OPTIONS

Effective date column

A column in the input data set of type date that contains the effective date. This column name is entered automatically if Data Integrator finds a column named `EFFDT` in the source. The column appears in the output schema with the name `EFFDT`.

This field is required.

Effective sequence column

A column in the input data set that indicates the order in time of related rows that have duplicate effective dates. If no related rows also share effective dates, the sequence numbers are the same ('0' for example). If related rows do share the same effective dates, the sequence numbers are incremented as rows with conflicting effective dates are added. This transform returns only the row containing the maximum sequence number if there are related rows with the same effective date.

This field is required only if the input data set allows duplicate effective dates.

Effective to column

The name of a date column added to the output schema that contains the effective-to date.

The effective-to date for a row is equal to the effective date of the related row with the closest greater effective date. If no such row exists, the **Default effective to date** is used.

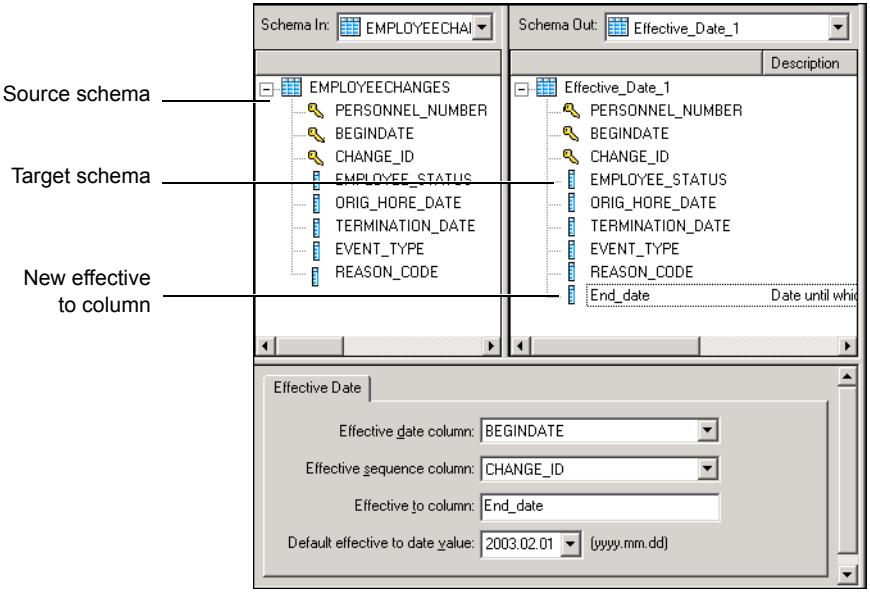
Default effective to date

A date assigned as the effective-to date for those rows with the highest effective date among related rows.

EDITOR

The Effective_Date transform editor includes a source and target schema and transform options. You can drag column names from the source schema to fill in values for the **Effective Date column**, and **Effective sequence** option.

The target schema is generated in response to the values you choose in the transform options. To refresh the target schema after you make a change to the options, refresh the target schema by choosing **View** > **Refresh** or pressing **F5**.



DATA OUTPUTS

The transform output includes all of the columns from the source schema and the calculated effective-to date column. For example, given a default effective-to date of January 1, 2999, the input described in the data input section is transformed as follows:

Transform output data set

Project	Effective date	Effective-to date	Status
Cherry Lake	1999.06.22	2003.01.12	Proposal
Cherry Lake	2003.01.12	2999.01.01[a]	Case
Hetch Hetchy Reservoir	1999.08.02	2002.10.17	Proposal
Hetch Hetchy Reservoir	2002.10.17	2003.05.06	Case
Hetch Hetchy Reservoir	2003.05.06	2999.01.01[a]	Case

a. The default effective-to date is used to close the effective date range.

In the case where an effective sequence column is necessary to produce a unique key—related rows contain the same effective date—the output from the Effective_Date transform includes a single row where the input had more than one. The row returned contains the largest sequence number:

Transform output data set with sequence numbers

Project	Effective date	Effective to date	Effective sequence	Status
Cherry Lake	1999.06.22	2003.01.12	0	Proposal
Cherry Lake	2003.01.12	2999.01.01	0	Case
Hetch Hetchy Reservoir	1999.08.02	2002.10.17	0	Proposal
Hetch Hetchy Reservoir	2002.10.17	2003.05.06	1[a]	Case
Hetch Hetchy Reservoir	2003.05.06	2999.01.01	0	Case

a. Data from the row with sequence 0 is omitted.

After the range of effective dates is generated for a set of data, you can use the effective-to date to filter appropriate records. For example, you can extract the subset of records valid as of today by selecting only those records whose effective-to column is later than today's date and effective-from column is earlier than today's date.

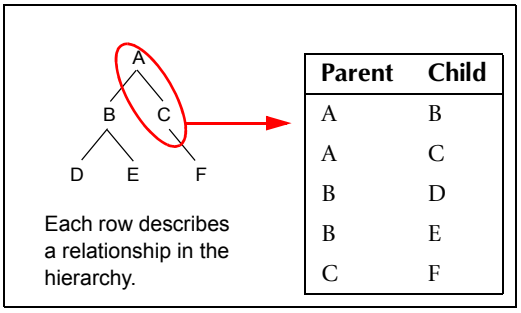
Nested schemas in the input are passed through without change.

Hierarchy_Flattening

Constructs a complete hierarchy from parent/child relationships, then produces a description of the hierarchy in vertically or horizontally flattened format.

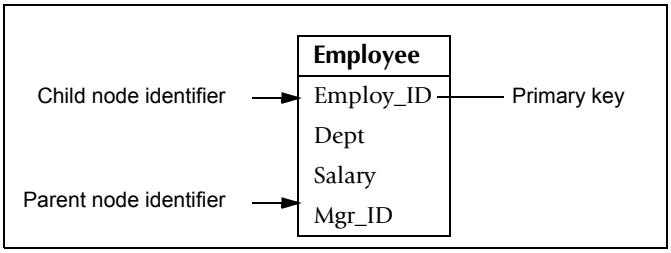
DATA INPUTS

Rows describing individual parent-child relationships.



Each row must contain two columns that function as the keys of the parent and child in the relationship. The input can also include columns containing attributes describing the parent and/or child.

An example of an input data set is an Employee Master hierarchy description in which each row represents the relationship between an employee (child node) and the employee's manager (parent node).



The input data set cannot include rows with operation codes other than NORMAL.

The input data set can contain hierarchical data.

OPTIONS

Parent column Identifies the column in the source data that contains the parent identifier in each parent-child relationship. You can drag this column from the source schema into the **Parent column** box.

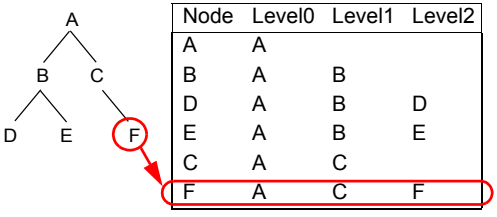
You cannot specify columns including nested schemas as the parent.

Child column Identifies the column in the source data that contains the child identifier in each parent-child relationship. You can drag this column from the source schema into the **Child column** box.

You cannot specify columns including nested schemas as the child.

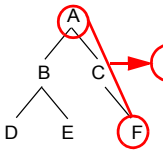
Flattening type Indicates how the hierarchical relationships are described in the output. Choose from two options:

- **Horizontal** — Each row of the output describes a single node in the hierarchy and the path to that node from the root. This mode requires that you specify the maximum path length through the tree as the **Maximum depth**.



  If there is more than one path to a node, all paths are described in the result.

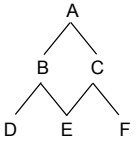
- **Vertical** — Each row of the output describes a single relationship between ancestor and descendent and the number of nodes the relationship includes. There is a row in the output for each node and all of the descendents of that node. Each node is considered its own descendent and therefore is listed one time as both ancestor and descendent.



Ancestor	Descendent	Depth	Root_Flag	Leaf_Flag
A	A	0	1	0
A	B	1	1	0
A	C	1	1	0
A	D	2	1	1
A	E	2	1	1
A	F	2	1	1
B	B	0	0	0
B	D	1	0	1
B	E	1	0	1
C	C	0	0	0
C	F	1	0	1
D	D	0	0	1
E	E	0	0	1
F	F	0	0	1

Use maximum length paths

(This option only applies to vertical flattening.) Indicates whether longest or shortest paths are used to describe relationships between descendents and ancestors when the descendent has more than one parent. The option only affects the DEPTH column in the output.



The depth of node E is the same for both paths to A

The depth of node M is “1” or “2” depending on the path to J

Maximum depth (This option only applies to horizontal flattening.) Indicates the maximum depth of the hierarchy. The root node (level 0) has a depth of 0; the first level has a depth of 1, and so on.

If you do not know the number of levels in your hierarchy, set **Maximum depth** to 1. When you execute the job, a warning message will appear in the execution log indicating that the **Maximum depth** is less than the actual depth of the hierarchy. Reset **Maximum depth** to the actual value reported in the warning message.

Parent attribute list

Identifies a column or columns that are associated with the parent column. You can drag columns from the source schema into the **Parent attribute list**. The column name appears in the target schema with a prefix that identifies the column as a parent attribute. The following table shows the result of adding a column named `POPULATION` to the parent attribute list.

Flattening mode	Source column	Target column
Vertical	POPULATION	P_POPULATION
Horizontal	POPULATION	P_L1_POPULATION
		P_L2_POPULATION
		(one column for each hierarchy level)

You can specify columns including nested schemas as the parent attribute.

Child attribute list

Identifies a column or columns that are associated with the child column. You can drag columns from the source schema into the **Child attribute list**. The column name appears in the target schema with a prefix that identifies the column as a child attribute. The following table shows the result of adding a column named `POPULATION` to the child attribute list.

Flattening mode	Source column	Target column
Vertical	POPULATION	C_POPULATION
Horizontal	POPULATION	C_L1_POPULATION
		C_L2_POPULATION
		(one column for each hierarchy level)

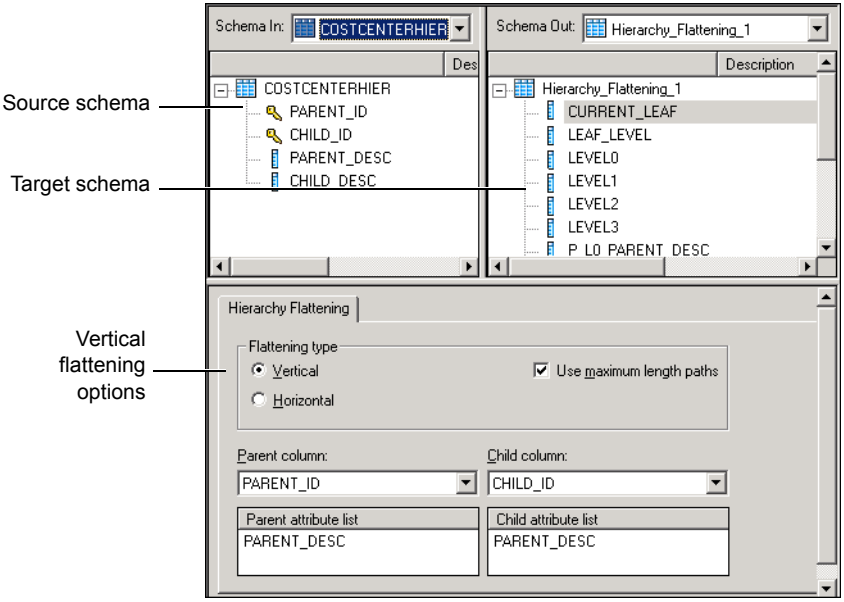
You can specify columns including nested schemas as the child attribute.

EDITOR

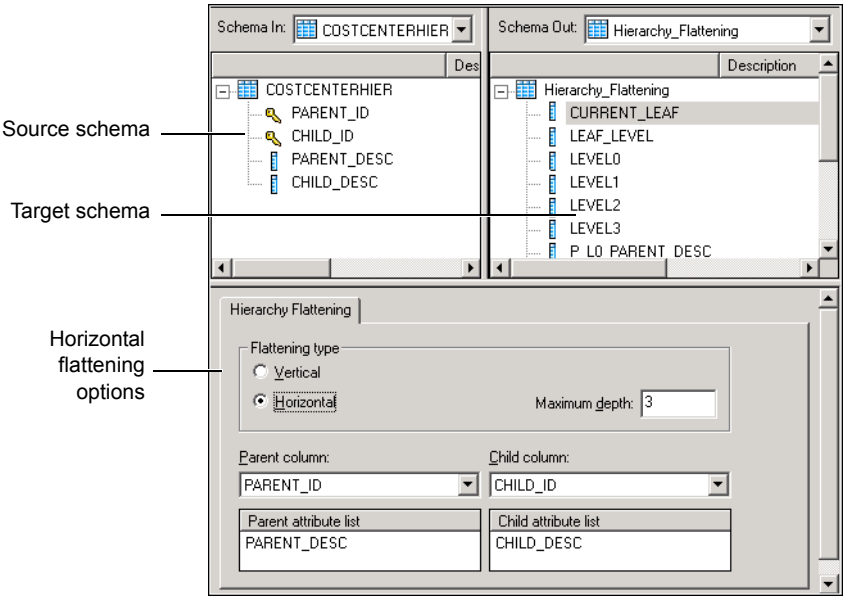
The Hierarchy_Flattening editor includes the source schema, the target schema, and transform options. You can drag column names from the source schema to fill in values for the **Parent column**, **Child column**, **Parent attribute list**, and **Child attribute list** options.

The target schema is generated in response to the values you choose in the transform options. To refresh the target schema after you make a change to the options, choose **View > Refresh** or press F5.

The following picture shows the editor with values for vertical flattening:



The following picture shows the editor with values for horizontal flattening:



Data Outputs

Horizontal flattening represents each level in the hierarchy as a column in the output, with the root listed in the first column and the outermost leaf listed in the last column. Represented horizontally, the number of levels in the hierarchy and the distance between a given node and the root node is clear.

Horizontal flattening produces the following target columns:

Column name	Description
CURRENT_LEAF	The end node described.
LEAF_LEVEL	The number of levels down from the root node where Current_leaf is found The root node has Leaf_level of 0.
LEVEL0	The descriptor for the top level node.
LEVEL1	The descriptor for the first level node If Leaf_level is 0, this value is NULL.
LEVEL*n*	The descriptor for the *n*th level node where *n* is the number of levels in the hierarchy If Leaf_level is *n*-1 or less, this value is NULL.
P_L0_*attribute_column*	Attribute column associated with the node described in Level0.
C_L1_*attribute_column*	Attribute column associated with the node described in Level1 when that node is the child node. If Leaf_level is 0, this value is NULL.
P_L1_*attribute_column*	Attribute column associated with the node described in Level1 when that node is the parent node. If Leaf_level is 0, this value is NULL.
C_L*n-1*_*attribute_column*	Attribute column associated with the child node described in Level*n-1* where *n* is the number of levels in the hierarchy. If Leaf_level is *n*-2, this value is NULL.
P_L*n-1*_*attribute_column*	Attribute column associated with the child node described in Level*n-1* where *n* is the number of levels in the hierarchy. If Leaf_level is *n*-2, this value is NULL.
C_L*n*_*attribute_column*	Attribute column associated with the child node described in Level*n* where *n* is the number of levels in the hierarchy. If Leaf_level is *n*-1, this value is NULL.

The following table shows an example of the target schema and data for horizontal flattening with a two-level hierarchy including country as the root node, state at level one, and city at level 2 (leaf nodes). The parent and child attributes are the same, a population value.

The table shows the three rows in the output; however, the format folds the row data onto two rows. The headings at the top of the table describe the first half of each row of data. The headings at the bottom of the table describe the second half of each row of data.

CURRENT_LEAF	LEAF_LEVEL		LEVEL0		LEVEL1		LEVEL2	
US	0		US		NULL		NULL	
		272,583,805		NULL		NULL		NULL
CA	1		US		CA		NULL	
		272,583,805		30,866,851		NULL		NULL
SanFrancisco	2		US		CA		SanFrancisco	
		272,583,805		30,866,851		30,866,851		723,959
		P_L0_POP		C_L1_POP		P_L1_POP		C_L2_POP

In a typical data flow including a Hierarchy_Flattening transform with the same attribute for parent and child nodes, you may follow the transform with a query that would filter the duplicated attribute values. The query would pass the P_L0_*attribute_column* and the C_L*n*_*attribute_column* through as they are, then keep either parent or child attribute for the intermediate levels.

Vertical flattening produces the following target columns:

Column name	Description
ANCESTOR	The node closer to the root node in the relationship described by this row.
DESCENDENT	The node farther from the root node in the relationship described by this row.
DEPTH	Number of levels between the Ancestor and Descendent.
ROOT_FLAG	Identifies the value in the Ancestor column as the top node of the hierarchy. Root_flag is 1 if Ancestor is the root node. Otherwise, Root_flag is 0.
LEAF_FLAG	Identifies the value in the Descendent column as the bottom node of the hierarchy. Leaf_flag is 1 if Descendent is the leaf node. Otherwise, Leaf_flag is 0.
P_*attribute_column*	Column from the source that you associate with the parent (can be more than one P_*attribute_column*). If Leaf_flag is 1, this value is NULL.
C_*attribute_column*	Column from the source that you associate with the child (can be more than one C_*attribute_column*). If Root_flag is 1, this value is NULL.

The following table shows an example of the target schema and data for vertical flattening with a two-level hierarchy. The hierarchy includes a country as the root node, state at level one, and city at level 2 (leaf nodes). The parent and child attributes are the same, a population value.

ANCESTOR	DESCENDENT	DEPTH	ROOT _FLAG	LEAF _FLAG	P_POP	C_POP
US	CA	1	1	0	272,583,805	30,866,851
US	SanFrancisco	2	1	1	272,583,805	723,959
CA	SanFrancisco	1	0	1	30,866,851	723,959
US	US	0	1	0	272,583,805	NULL
CA	CA	0	0	0	30,866,851	30,866,851
SanFrancisco	SanFrancisco	0	0	1	NULL	723,959

Each node is listed one time as both ancestor and descendent. The Parent attribute is null for a row describing the relationship between a leaf node and itself. Likewise the Child attribute is null for a row describing the relationship between a root node and itself.

The transform ignores any hierarchical data unless a nested schema is identified as a parent or child attribute. An attribute column containing nested data is passed through the transform without change.

Error Conditions

If the hierarchy represented by the input data set is cyclic—some node is its own ancestor—Data Integrator produces a run-time error.

No errors are produced if the input data source describes multiple root nodes.

History_Preserving

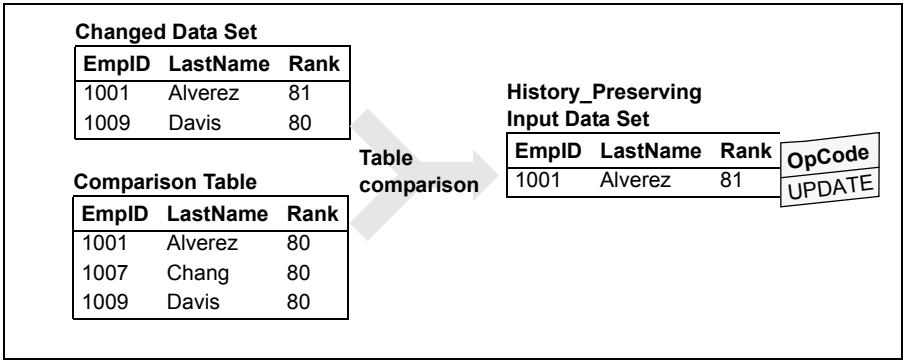
The History_Preserving transform allows you to produce a new row in your target rather than updating an existing row. You can indicate in which columns the transform identifies changes to be preserved.

If the value of certain columns change, this transform creates a new row for each row flagged as UPDATE in the input data set.

Data Inputs

A data set that is the result of a comparison between two images of the same data in which changed data from the newer image are flagged as UPDATE rows and new data from the newer image are flagged as INSERT rows.

For example, a target table that contains employee job rankings information is updated periodically from a source table. In this case, the table comparison flags changed data for employee Alverez and discards unchanged data for employee Davis. The result is a single row flagged with the UPDATE operation code.



The input data set can contain hierarchical data. The transform operates only on the rows at the top-level of the input data set, and passes nested data through to the output without change. Columns containing nested schemas cannot be used as transform parameters.

Use caution when using columns of data type real in this transform. Comparison results are unpredictable for this data type. See "real" on page 200 for more information.

Options

Date columns - Valid from

A date or datetime column from the source schema. Specify a **Valid from** date column if the warehouse uses an effective date to track changes in data.

This value is used as the new value in the **Valid from** date column in the new row in the warehouse added to preserve history of an existing row. It is also used to update the **Valid to** date column in the previously current row in the warehouse.

Data Integrator validates that both the **Valid from** date column and the **Valid to** date column have been specified if either is specified.

Date columns - Valid to

A date or datetime column from the source schema. Specify a **Valid to** date column if the warehouse uses an effective date to track changes in data and if you specified a **Valid from** date column.

This value is used as the new value in the **Valid to** date column in the new row added to the warehouse to preserve history of an existing row.

The **Valid to** date column cannot be the same as the **Valid from** date column.

Date columns - Valid to date value

A date value specified as a four-digit year followed by a period, followed by a two-digit month, followed by a period, and followed by a two-digit day value.

The **Valid to date value** is used as the new **Valid to** date column value in the new row added to the warehouse to preserve history of an existing row.

Current flag - Column

A column from the source schema that identifies the current valid row from a set of rows with the same primary key.

The flag column indicates whether a row is the most current data in the warehouse for a given primary key.

Column cannot be the same value as the **Valid from** date column or the **Valid to** date column.

Data Integrator validates that a current flag **Column**, **Set value**, and **Reset value** are specified, if any are specified.

C**olumn** cannot contain a nested schema.

Current flag - Set value

An expression that evaluates to a value with the same data type as the current flag **Column**. This value is used to update the current flag **Column** in the new row in the warehouse added to preserve history of an existing row.

You must enter a value in the Current flag **Column** box to enable the **Set value**.

Current flag - Reset value

An expression that evaluates to a value with the same data type as current flag **Column**. This value is used to update the current flag **Column** in an existing row in the warehouse that included changes in one or more of the compare columns.

You must enter a value in the current flag **Column** box to enable the **Flag reset value**.

Compare columns

The column or columns in the input data set for which this transform compares the before and after images to determine if there are changes.

If the values in each image of the data match, the transform flags the row as UPDATE. The result updates the warehouse row with values from the new row. The row from the before image is included in the output as UPDATE to effectively update the date and flag information.

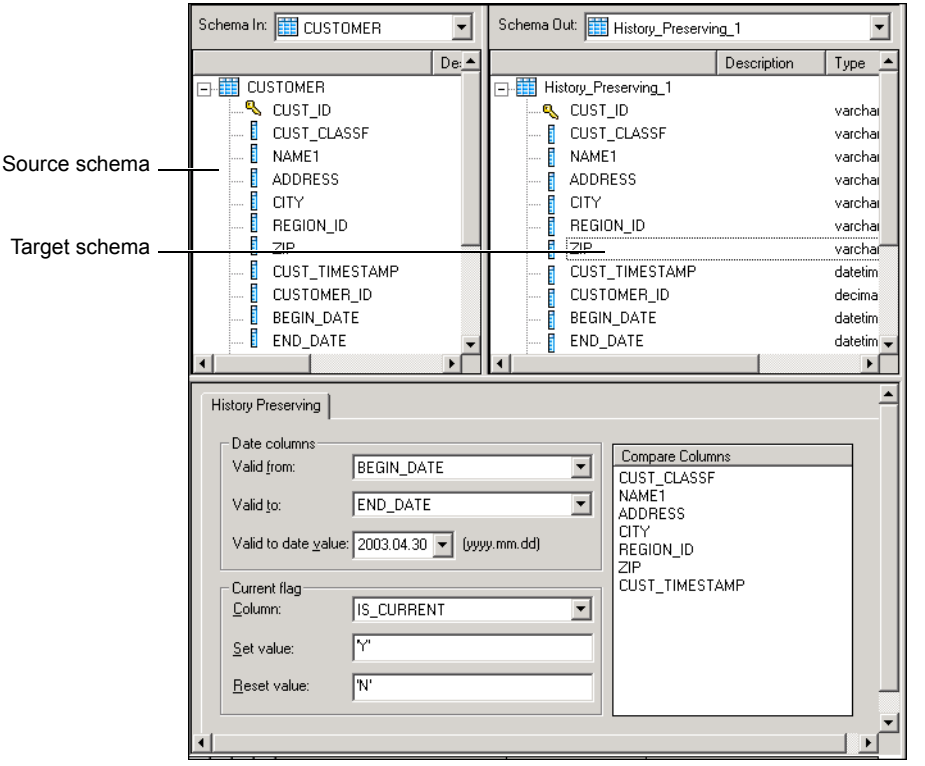
If the values in each image do not match, the row from the after image is included in the output of the transform flagged as INSERT. The result adds a new row to the warehouse with the values from the new row.

Compare columns cannot contain nested schemas.

EDITOR

The History_Preserving transform editor includes the source schema, the target schema, and transform options. You can drag column names from the source schema to fill in values for the current flag **Column**, **Valid from** date column, **Valid to** date column, and **Compare columns** options.

The target schema is generated in response to the values you choose in the transform options. To refresh the target schema after you make a change to the options, choose **View > Refresh** or press **F5**.



DATA OUTPUTS

A data set with rows flagged as INSERT or UPDATE.

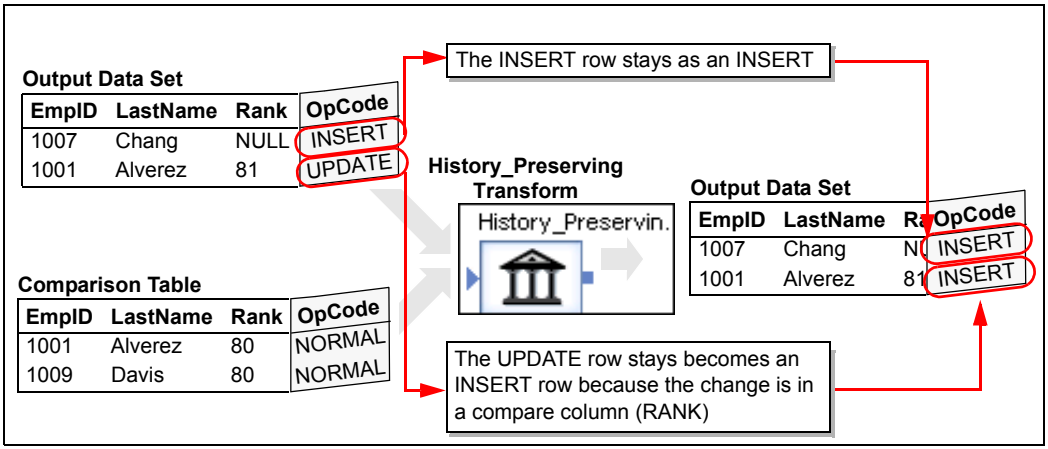
For each row in the input data set, there are two possible outcomes from the transform:

- An INSERT row

  A new row must be added to the comparison table if:

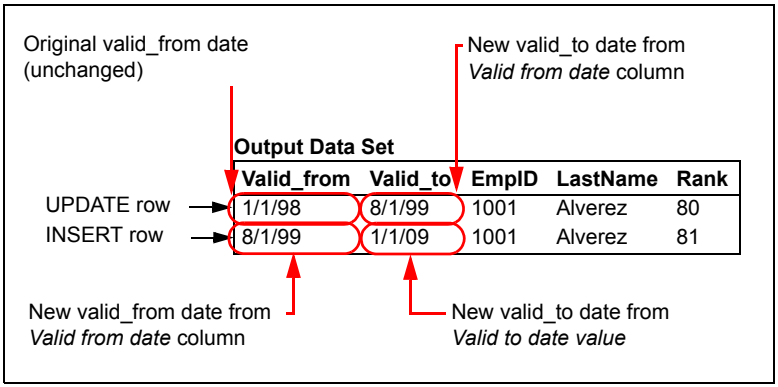
- Given an input row flagged as UPDATE — A value in a compare column from the input data set does not match a corresponding value in the comparison table.
- Given an input row flagged as INSERT — The primary key from the input data set does not appear in the comparison table.

The transform produces an INSERT row with the values from the input data set row.



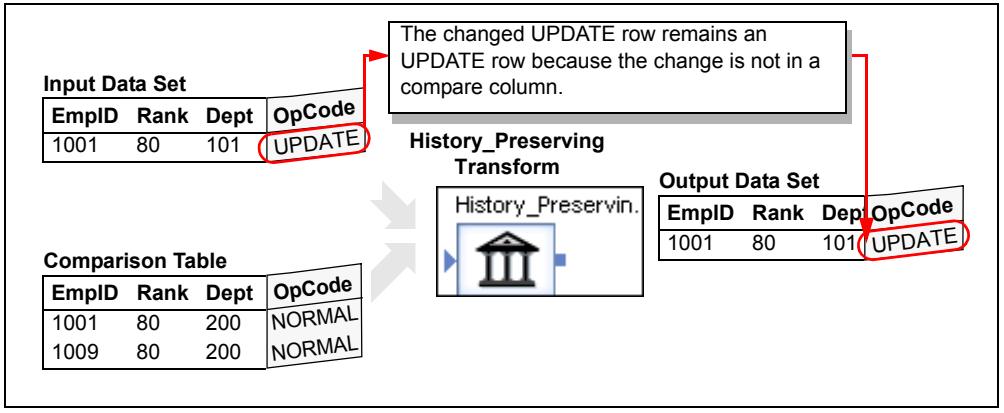
If you specified flag values for the History_Preserving transform, Data Integrator includes the **Flag set value** in the INSERT row. In addition Data Integrator includes an UPDATE row to update the previously current row in the warehouse with the **Flag reset value**.

If you specified effective date columns (**Valid to** date column and **Valid from** date column), Data Integrator includes this data as well.



- An UPDATE row

  Input rows flagged as UPDATE contain changes, but not in the compare columns. The transform produces an UPDATE row with the values from the input data set row.



Nested schemas in the input are passed through without change.

Key_Generation

Generates new keys for new rows in a data set.

When it is necessary to generate artificial keys in a table, the Key_Generation transform looks up the maximum existing key value from a table and uses it as the starting value to generate new keys. The transform expects the generated key column to be part of the input schema.

Data Inputs

A data set that is the result of a comparison between two images of the same data in which changed data from the newer image are flagged as UPDATE rows and new data from the newer image are flagged as INSERT rows.

The data set includes a column into which generated keys are added.

The input data set can contain hierarchical data. The transform operates only on the rows at the top-level of the input data set, and passes nested data through to the output without change. Columns containing nested schemas cannot be used as transform parameters.

OPTIONS

Table name The fully qualified name of the source table from which the maximum existing key is determined (key source table). This table must be already imported into the repository. **Table name** is represented as "DATASTORE.OWNER.TABLE" where DATASTORE is the name of the datastore Data Integrator uses to access the key source table; OWNER, if required, depends on the database type associated with the table:

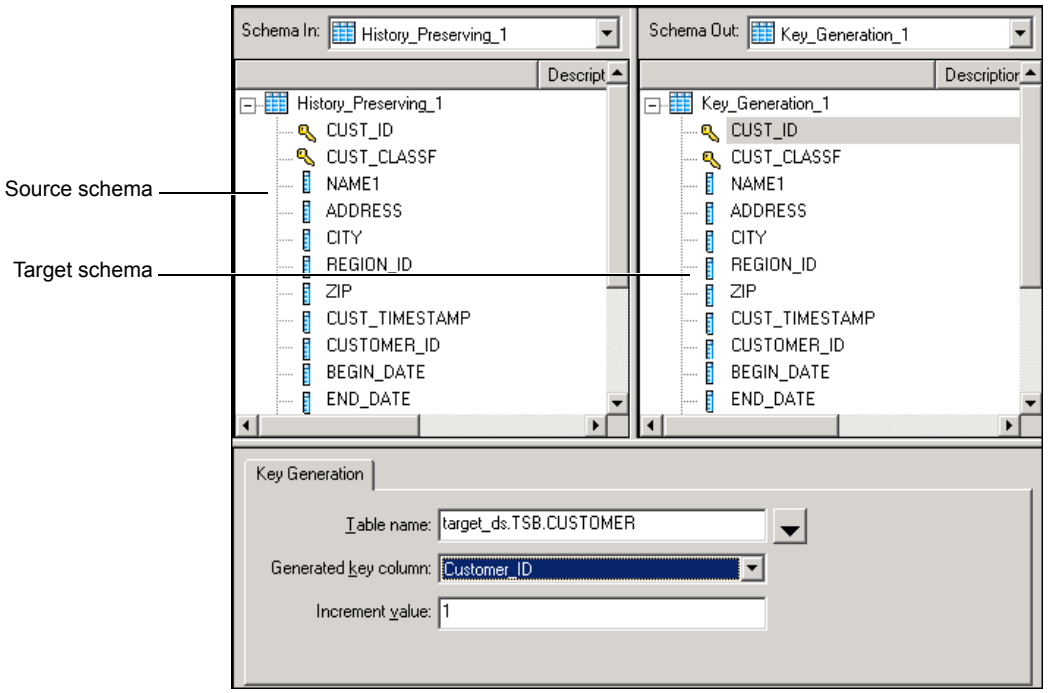
Database type	Owner value
DB2	Data source dependent
Informix	Informix-defined user name
Microsoft SQL Server	User name
ODBC	Data source dependent
Oracle	User name
Sybase	User name

Generated key column
The column in the key source table containing the existing keys values. A column with the same name must exist in the input data set; the new keys are inserted in this column.

Increment value The interval between generated key values.

EDITOR

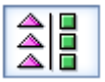
The Key_Generation transform editor includes the source schema, the target schema, and transform options. To refresh the target schema after you make a change to the options, choose **View** > **Refresh** or press **F5**.



DATA OUTPUTS

The input data set with the addition of key values in the generated key column, for input rows flagged as INSERT.

Use the Key_Generation transform to produce keys that distinguish rows that would otherwise have the same primary key. For example, suppose the History_Preserving transform produces rows to add to a warehouse and these rows have the same primary key as rows that already exist in the warehouse. In this case, you can add a generated key to the warehouse table and fill new key values using the Key_Generation transform.



Map_CDC_Operation

Using its input requirements (values for the **Sequencing column** and a **Row operation column**), performs three functions:

- Sorts input data based on values in **Sequencing column** box and (optional) the **Additional Grouping Columns** box.
- Maps output data based on values in **Row Operation Column** box. Source table rows are mapped to INSERT, UPDATE, or DELETE operations before passing them on to the target.
- Resolves missing, separated, or multiple before- and after-images for UPDATE rows.

While commonly used to support Oracle or mainframe changed-data capture, this transform supports any data stream as long as its input requirements are met.

This transform is typically the last object before the target in a data flow because it produces INPUT, UPDATE and DELETE operation codes. Data Integrator produces a warning if other objects are used.

Data Input

All rows in the input data are set to NORMAL. This is an internal Data Integrator operation code.

The input data set can contain hierarchical data. Nested schemas in the input are passed through without change.

Options

Sequencing column

(Required) Specifies an integer used to order table rows.

If you are using an Oracle or mainframe CDC source table, the `DI_SEQUENCE_NUMBER` column is automatically selected as the **Sequencing column**.

Input already sorted by sequencing column

This transform by default assumes that the input data is already sorted based on the value selected in the **Sequencing column** box. If you deselect this check box, Data Integrator will re-sort the input data using the value in the **Sequencing column** box.

Use the re-sort capability of this transform only when necessary as it impacts job performance.

Additional grouping columns

In addition to the **Sequencing column**, you can sort input on additional columns by dragging them into this box from the input schema. Sorts are prioritized based first on the sequencing column and then on the order of the columns added to this box.

Row operation column

(Required) Specifies a column with one of the following output operation codes for each row:

`I` for INSERT

B for before-image of an UPDATE

`U` for after-image of an UPDATE

`D` for DELETE

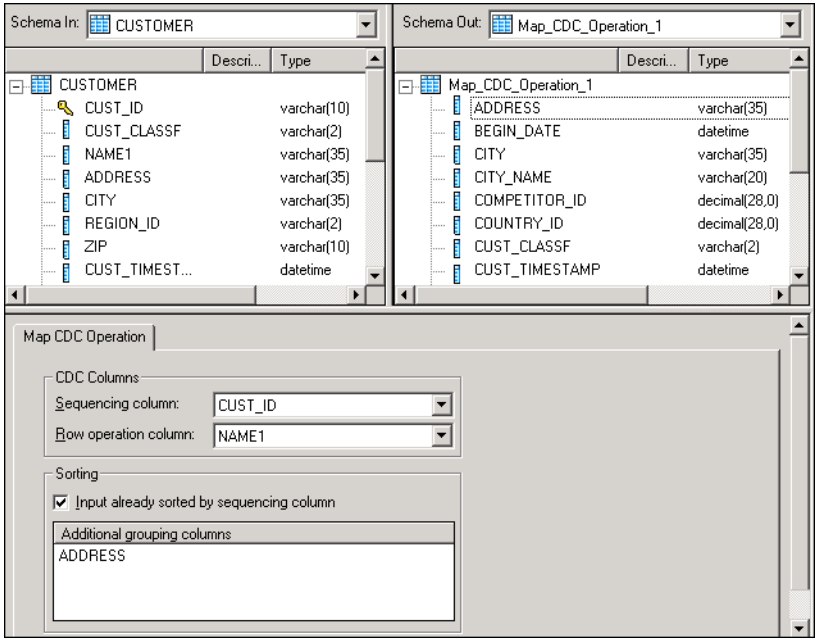
If you are using an Oracle or mainframe CDC source table, the `DI_OPERATION_TYPE` column is automatically selected as the **Row operation column**.

EDITOR

Use the list boxes in the **CDC columns** section to select the required columns in your input schema used to sequence the input rows and map operations to the output.

If you want to sort the input using additional columns, click a column in the input schema and drag it into the **Additional grouping columns** box.

Map_CDC_Operation transform editor set to default values for an Oracle or mainframe CDC table



If you are using an Oracle or mainframe CDC source table, the DI_SEQUENCE_NUMBER and DI_OPERATION_TYPE columns appear in the input schema and are automatically removed in the output schema. To propagate these columns to the output, create additional columns that map to them prior to coming into the Map_CDC_Operation transform. For more information, see "Importing CDC data from Oracle" on page 490.

Output Data

A data set with rows flagged as specified by the values in the column selected as the **Row operation column**.

Rows in the input data set all use NORMAL as their internal Data Integrator operation code.

Rows in the output data set can have any of the following operation codes:

- INSERT
- DELETE
- UPDATE

In addition, the DISCARD option is assigned under some conditions. Discarded rows are not passed through to the output of the transform. For information about when a DISCARD is assigned, see "Rules for resolving before- and after-image pairs" on page 267.

Sorting CDC data

When you apply changed data to a loader, it is important that the order of the rows is preserved. For example, if the following operations are applied to an empty target:

- INSERT into TAB1 values ("Bob', 'Boat', 3500)
- INSERT into TAB1 values ('Jane', 'BMW Roadster', 24000)

- UPDATE TAB1 set toy = 'Motorcycle', price = 12000 where name = 'Bob'
- DELETE from TAB1 where name = 'Bob'

The table TAB1 will be left with one row:

```
'Jane', 'BMW Roadster', 24000
```

If these operations are applied out of order, for example, if the DELETE occurs before the UPDATE operation, then database consistency is no longer preserved. In this example:

- The table has two rows (Bob and Jane)
- The last UPDATE statement fails because there is no row on which to perform an UPDATE

By ordering the input rows using the sequencing column, the order of the original set of operations is preserved.

The sequencing column values are also useful if you are using before- and after-images for update rows because it is possible that before- and after-image pairs may be separated, multiplied or lost depending on the design of your data flow. You can re-sort input columns as needed by using the sequencing column and any number of additional columns.

The before- and after-images of an UPDATE row have the same sequence value. Thus correctly sorted before- and after-image rows are listed in pairs.

Rules for resolving before- and after-image pairs

For an introduction to the use of before-and after-images in Data Integrator, see "Using before-images" on page 497

The Map_CDC_Operation transform uses the following rules to process and resolve before- and after-images:

- When constructing UPDATE rows, the value in the **Row Operation Column** is used. If there are before-images in the input stream, before- (B) and after-image (U) row pairs are combined into one UPDATE row.

  For example, given the following sample input of six rows:

Sequencing Column	Operation Column	Internal Operation Code
1	I	NORMAL
2	B	NORMAL
2	U	NORMAL
3	D	NORMAL
4	B	NORMAL
4	U	NORMAL

  The following four rows will be the output:

Sequencing Column	Operation Column	Internal Operation Code
1	I	INSERT
2	U	UPDATE (before- and after-images)
3	D	DELETE
4	U	UPDATE (before- and after-images)

- If there are no before-images (B)in the input stream, the after-images (U) alone produce UPDATE rows.

  Given the following sample input rows:

Sequencing Column	Operation Column	Internal Operation Code
1	I	NORMAL
2	U	NORMAL
3	D	NORMAL
4	U	NORMAL

The following four rows will be the output:

Sequencing Column	Operation Column	Internal Operation Code
1	I	INSERT
2	U (no before)	UPDATE
3	D	DELETE
4	U (no before)	UPDATE

- If a before-image (`B`) row is followed by additional `B` rows, the subsequent `B` rows are ignored until an after-image (`U`) row is encountered.

  For example, given the following six input rows:.

Sequencing Column	Operation Column	Internal Operation Code
1	U	NORMAL
1	B	NORMAL
2	B	NORMAL
3	B	NORMAL
4	B	NORMAL
2	U	NORMAL

  Two UPDATE rows are output:

Sequencing Column	Operation Column	Internal Operation Code
1	U	UPDATE (before- and after-images)
2	U	UPDATE (before- and after-images)

  - The first two rows are processed as one UPDATE row.
  - The third and sixth row are processed as a pair. One UPDATE row is output.
  - The remaining rows are DISCARDED. For more information, see the following rule.

- If after a series of `B` rows, either no `U` rows remain or another row type is encountered, `B` rows are discarded.

  For example, given the following sample input of five rows:

Sequencing Column	Operation Column	Internal Operation Code
1	U	NORMAL
1	B	NORMAL
2	B	NORMAL
3	B	NORMAL
4	I	NORMAL

  Two rows are output:

Sequencing Column	Operation Column	Internal Operation Code
1	U	UPDATE (before- and after-images)
4	I	INSERT

Map_Operation

Allows conversions between data manipulation operations.

The Map_Operation transform allows you to change operation codes on data sets to produce the desired output. For example, if a row in the input data set has been updated in some previous operation in the data flow, you can use this transform to map the UPDATE operation to an INSERT. The result could be to convert UPDATE rows to INSERT rows to preserve the existing row in the target.

Data Integrator can push Map_Operation transforms to the source database.

Data Inputs

A data set with rows flagged with any operation codes.

The input data set can contain hierarchical data.

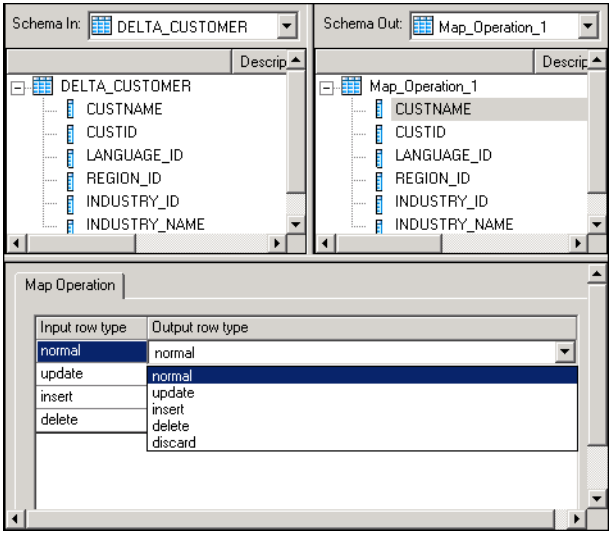
Use caution when using columns of data type real in this transform. Comparison results are unpredictable for this data type. See "real" on page 200 for more information.

Options

Output row type Indicate the new operations desired for the input data set. Choose from the following operation codes: INSERT, UPDATE, DELETE, NORMAL, or DISCARD.

DATA OUTPUTS

A data set with rows flagged as specified by the mapping operations.



Rows in the input data set may have any of the following operation codes:

- NORMAL
- INSERT
- DELETE
- UPDATE

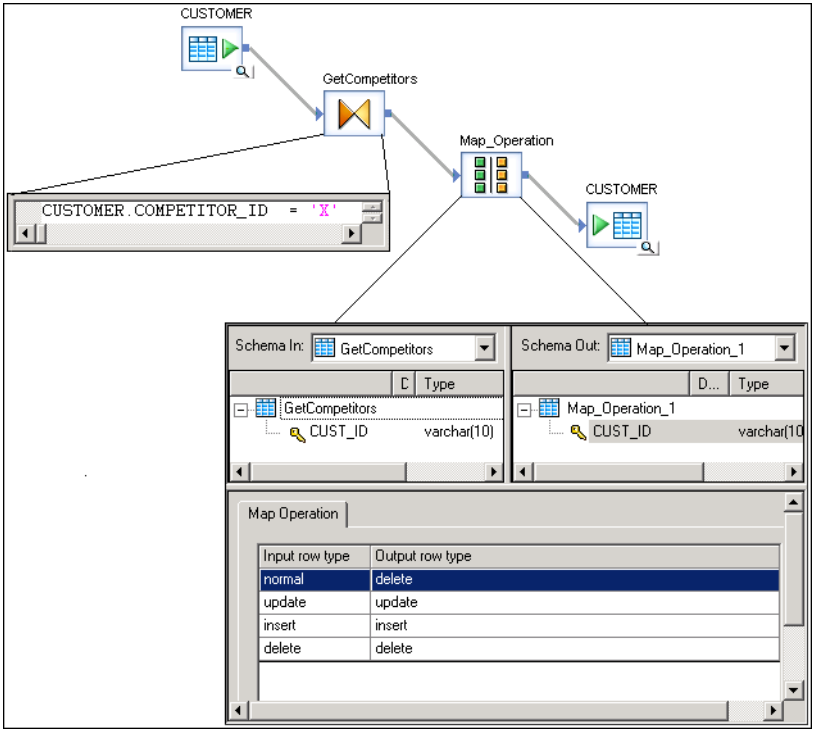
Any of these operation codes can be mapped to:

- NORMAL
- INSERT
- DELETE
- UPDATE

In addition, the DISCARD option can be assigned. Discarded rows are not passed through to the output of the transform.

By default, every input operation type maps to itself. For each specified mapping, every row in the input data set that matches the input mapping operation is converted to the specified output operation.

Nested schemas in the input are passed through without change. In the following example, the Map_Operation is used to delete all customers who have also been indicated as competitors:



Merge

Combines incoming data sets, producing a single output data set with the same schema as the input data sets.

Data Inputs

A data set from two or more sources with rows flagged as any operation code.

All sources must have the same schema, including:

- The same number of columns
- The same column names
- The same data types of columns

If the input data set contains hierarchical data, the names and data types must match at every level of the hierarchy.

Options

None.

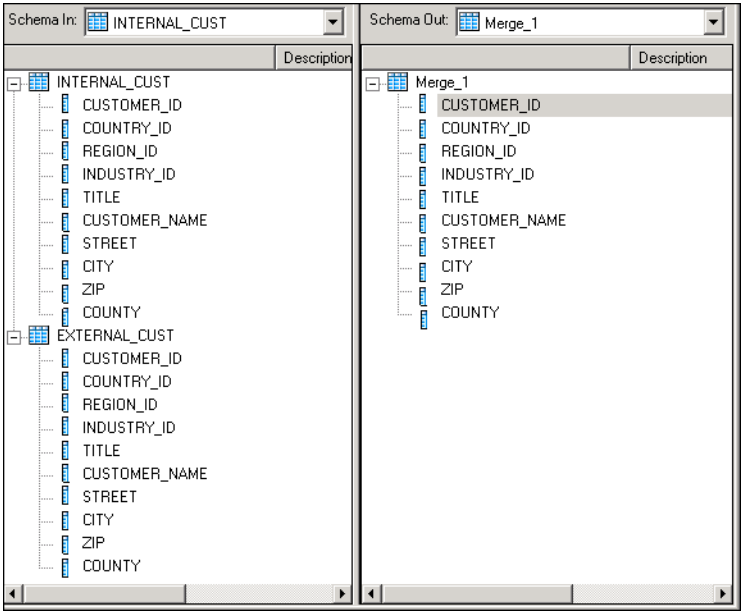
Data Outputs

A data set consisting of rows from all sources, with any operation codes. The output data has the same schema as the source data, including nested schemas.

The output data set contains a row for every row in the source data sets. The transform does not strip out duplicate rows. If columns in the input set contain nested schemas, the nested data is passed through without change.

If the data types of columns in the sources do not match the target, add a query in the data flow before the Merge transform. In the query, apply a data type conversion to the columns with data types that do not match the target column data types.

You must apply other operations such as DISTINCT in a query following the Merge transform.



Pivot (Columns to Rows)

Creates a new row for each value in a column that you identify as a pivot column.

The Pivot transform allows you to change how the relationship between rows is displayed. For each value in each pivot column, Data Integrator produces a row in the output data set. You can create pivot sets to specify more than one pivot column.

Data Inputs

A data set with rows flagged as NORMAL.

Options

Pivot sequence column
: The name you assign to the sequence number column. For each row created from a pivot column, Data Integrator increments and stores a sequence number. Data Integrator resets the sequence to 1 when creating a row from an original row. For example, if the row corresponds to the first column pivoted, the sequence number for the row is 1.

Non-pivot columns
: The columns in the source that are to appear in the target without modification.

Pivot set
: The number that identifies a pivot set. For each pivot set, you define a group of pivot columns, a pivot data field, and a pivot header name. Each pivot set must have a unique **Data field column** and the **Header column**. Data Integrator automatically saves this information.

Data field column

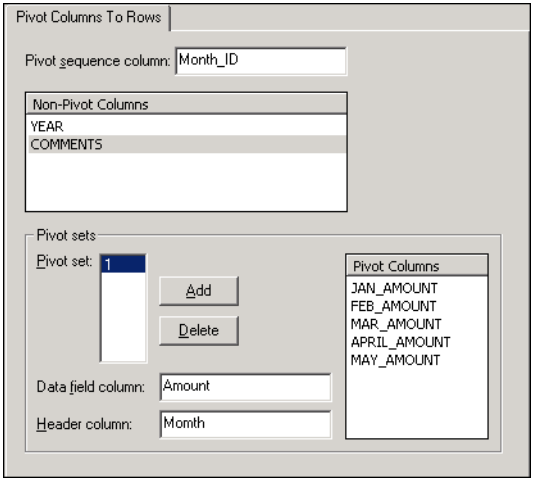
The name of the column that contains the pivoted data. This column contains all of the **Pivot columns** values.

The data type of this column is determined by the **Pivot columns** data type. If two or more **Pivot columns** contain different data types, Data Integrator converts the columns to a single data type—the data type of the first column you add to the pivot set.

Header column The name of the column that contains the pivoted column names. This column lists the names of the columns where the corresponding data originated.

Pivot columns A set of columns to be rotated into rows. Describe these columns in the **Pivot header name**. Describe the data in these columns in the **Pivot data field**.

EDITOR



Data Outputs

A data set with rows flagged as NORMAL. This target includes the nonpivoted columns, a column for the sequence number, the pivot data column, and the pivot header column.

Example

Suppose you have a table containing rows for an individual's expenses, broken down by expense type.

Emp_name	Mgr_ID	Internal _Expense	Travel _Expense	Misc _Expense
AAA	1234	2000.00	5000.00	100.00
BBB	9876	3000.00	0.00	1000.00
CCC	5555	4800.00	800.00	0.00

This source table has expense numbers in several columns, so you might have difficulty calculating expense summaries. The Pivot transform can rearrange the data into a more manageable form, with all expenses in a single column, without losing category information.

Set the Pivot transform options to pivot the data such that all of the expenses are in the same column. Note that you only need one pivot set in this example.

Options	Value
Pivot sequence column	Sequence
Nonpivot columns	Emp_name
Pivot set	1
Pivot data column	Expense
Header column	Expense_Type
Pivot columns	Internal_Expense Travel_Expense Misc_Expense

Select **View** > **Refresh** or press **F5** to update the output schema for the Pivot transform.

The output data set includes the employee name (not pivoted) and new columns for the pivot sequence, expense type (pivot header), and actual expense data. The manager ID column is not listed in either the pivot or the nonpivot column lists, so it is not included in the output.

The result is a single column of expense values that can be summarized easily.

Emp_name	Sequence	Expense_Type	Expense
AAA	1	Internal_Expense	2000.00
AAA	2	Travel _Expense	5000.00
AAA	3	Misc _Expense	100.00
BBB	1	Internal_Expense	3000.00
BBB	2	Travel _Expense	0.00
BBB	3	Misc _Expense	1000.00
CCC	1	Internal_Expense	4800.00
CCC	2	Travel _Expense	800.00
CCC	3	Misc _Expense	0.00

Suppose that instead of containing one type of data—expenses—your source table contains two types of data—expenses and days traveling both domestically and internationally—for two months.

Emp_name	Dom_Exp	Int_Exp	Dom_Day	Int_Day
AAA	2000.00	5000.00	10	5
BBB	3000.00	0.00	0	0
CCC	4800.00	800.00	15	1

You want to create a target table that has the data in two columns: expenses and days. Therefore, you want to create two pivot sets. Create one pivot set to pivot on the expense columns.

Options	Value
Pivot sequence column	Seq
Nonpivot columns	Emp_name
Pivot set	1
Data field column	Expense
Header column	Expense_Type
Pivot columns	Dom_Exp Int_Exp

Create a second pivot set to pivot on the day columns.

Options	Value
Pivot sequence column	Seq
Nonpivot columns	Emp_name
Pivot set	2
Data field column	Num_Days
Header column	Day_Type
Pivot columns	Dom_Day Int_Day

In this case, the output data set includes the employee name (not pivoted) and new columns for the pivot sequence, expense type, expense data, day type, and day data. Because you linked the pivot data, domestic and international data are contained in unique rows.

Emp_name	Seq	Expense_Type	Expenses	Day_Type	Num_Days
AAA	1	Dom_Exp	2000.00	Dom_Day	10
AAA	2	Int_Exp	5000.00	Int_Day	5
BBB	1	Dom_Exp	3000.00	Dom_Day	12
BBB	2	Int_Exp	0.00	Int_Day	0
CCC	1	Dom_Exp	4800.00	Dom_Day	15
CCC	2	Int_Exp	800.00	Int_Day	1

When working with multiple pivot sets, pivoted columns cannot contain a different number of rows.

If the example target table contained additional expenses (internal plus miscellaneous expenses), but only had days traveled to match domestic and international travel expenses, the expense data set would be larger than the days traveled data set. In that case, you would have to add a new artificial column containing NULL values to the input data set, and associate the day columns with those additional expenses.

Emp_name	Seq	Expense_Type	Expenses	Day_Type	Num_Days
AAA	1	Dom_Exp	2000.00	Dom_Day	10
AAA	2	Int_Exp	5000.00	Int_Day	5
AAA	3	Internal_Exp	500.00	NULL	NULL
AAA	4	Misc_Exp	75.00	NULL	NULL
BBB	1	Dom_Exp	3000.00	Dom_Day	12
BBB	2	Int_Exp	0.00	Int_Day	0
BBB	3	Internal_Exp	350.00	NULL	NULL
BBB	4	Misc_Exp	140.00	NULL	NULL
CCC	1	Dom_Exp	4800.00	Dom_Day	15
CCC	2	Int_Exp	800.00	Int_Day	1
CCC	3	Internal_Exp	1000.00	NULL	NULL
CCC	4	Misc_Exp	55.00	NULL	NULL

Reverse Pivot (Rows to Columns)

Creates one row of data from several existing rows.

The Reverse Pivot transform allows you to combine data from several rows into one row by creating new columns. For each unique value in a pivot axis column and each selected pivot column, Data Integrator produces a column in the output data set.

Data Inputs

A data set with rows flagged as NORMAL.

Options

Non-pivot columns
The columns in the source table that will appear in the target table without modification.

Pivoted columns The columns containing data you want rotated into the same row. A set of columns will be created for each unique value in the **Pivot axis column**.

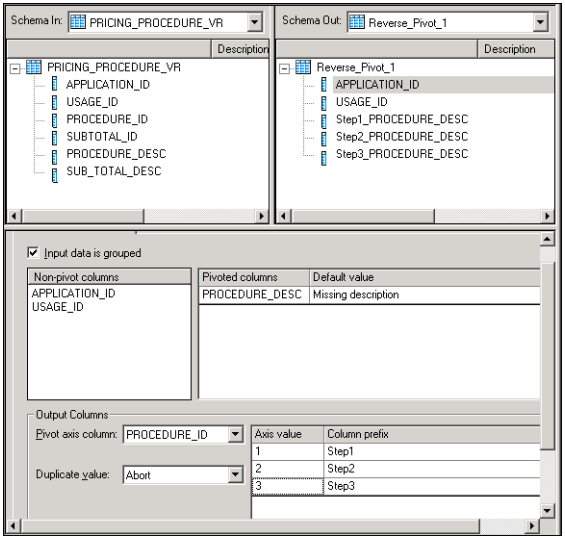
Default value
The value stored when a rotated column has no corresponding data. The default is "null" if you do not enter a value. Do not enter a blank.

Pivot axis column
The column that determines what new columns are needed in the output table. At run time, a new column is created for each **Pivoted column** and each unique value in this column.

Duplicate value	Action taken when a collision occurs. A collision occurs when there is more than one row with the same key and value in the **Pivot axis column**. In this case, you can store either the first row or the last row, or you can abort the transform process.
Axis value	The value of the pivot axis column that represents a particular set of output columns. A set of **Pivoted columns** is generated for each axis value. There should be one **Axis value** for each unique value in the **Pivot axis column**.
Column Prefix	Text added to the front of the **Pivoted column** names when creating new column names for the rotated data. If this field is blank, then the axis value is used. An underscore separates the prefix name from the pivoted column name.

DATA OUTPUTS

A data set with rows flagged as NORMAL. This target includes the nonpivoted columns and a column for the combination of each pivot column and each pivot axis.



EXAMPLE

Suppose you had a table containing contact information for each employee. Each row in the table contains data for a particular employee and contact type.

EmpNo	Type	Name	Address	Phone
100	emergency	Andrew	404 Hallam St	555-4450
100	home	Pat	125 Mercury St	555-6035
100	work	Sean	8400 Page Mill Rd	555-5000
200	emergency	Linda	126 River Rd	555-1087
200	home	David	479 Mill St	555-6914
300	work	Joanne	9500 Page Mill Rd	555-8500

Because the table can have several rows for each employee, finding information, such as a missing contact, for a particular employee may be difficult. The Pivot on Rows transform can rearrange the data into a more searchable form without losing the category information.

Set the Pivot on Rows transform options to pivot the data such that all of the contact information for a particular employee is in the same row.

Option	Value		
Non-pivot columns	EmpNo		
Pivoted columns	Name Phone		
Default value	Null		
Pivot axis column	Type		
Duplicate value	Abort		
Axis Value	emergency	home	work
Column Prefix	emergency	home	work

The output data set includes the employee number field (not pivoted) and two fields—name and phone—for each pivot axis. In this case, there are three pivot axes (emergency, home, and work). Therefore, there are six additional fields. In cases where there is no data for a field in the initial source, the Pivot on Rows transform stores a null value.

The result is a single row for each employee, which you can use to search easily for missing contact information.

EmpNo	Emerg_ Name	Emerg_ Phone	Home_ Name	Home_ Phone	Work_N ame	Work_ Phone
100	Andrew	555-4450	Pat	555-6035	Sean	555-5000
200	Linda	555-1087	David	555-6914	Null	Null
300	Null	Null	Null	Null	Joanne	555-8500

Query

Retrieves a data set that satisfies conditions that you specify. A query transform is similar to a SQL SELECT statement.

DATA INPUTS

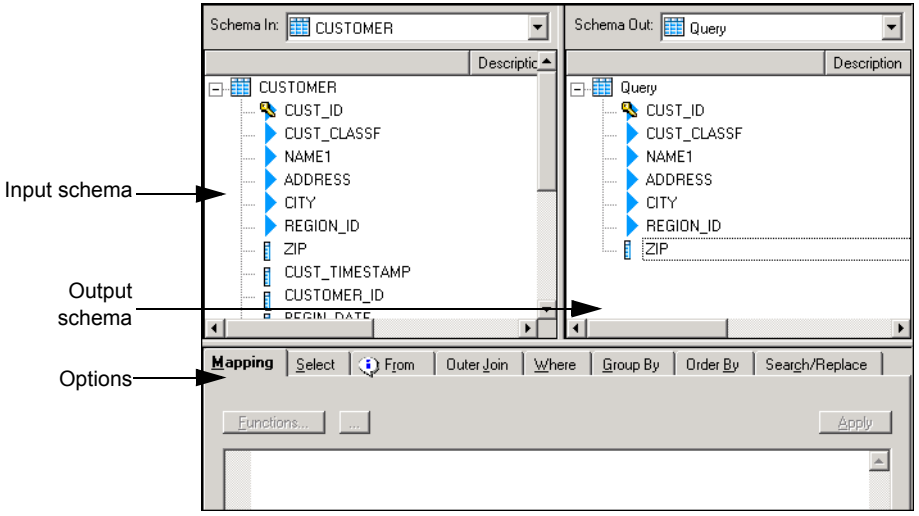
A data set from one or more sources with rows flagged as any operation code.

Use caution when using columns of data type real in this transform. Comparison results are unpredictable for this data type. See "real" on page 200 for more information.

EDITOR

Use the Query transform editor to specify:

- Input schema
- Output schema
- Options



You can resize the different areas by dragging split bars. To change the width of properties displayed in the input and output schema areas, click and drag the header border. To change the display sequence of properties, drag one property header over another.

Input schema

The input schema area displays all schemas input to the Query transform as a hierarchical tree. Each input schema can contain zero or more of the following elements:

- Columns
- Nested schemas

Icons preceding columns are combinations of the following graphics:


Primary key


Column that is not used in output mapping


Column that is used in output mappings

The **Input** list at the top left of the query editor indicates the schema that is currently selected. As you select schemas or columns in the input schema area, the **Input** list displays the corresponding schema. Conversely, you can select a schema in the **Input** list to move easily to a required schema.

You can right-click elements in the input schema area and select the following menu commands:

Command	Applicable elements	Effect
Copy	Columns, schemas	Stores a copy of the selected elements in the clipboard, leaving the elements in the input schema area.
Find	Anywhere in input schema area	Locates an output element with the name or description you enter.
Refresh	Anywhere in input schema area	Refreshes the display of the input schema area.
Parent	Columns	Selects the parent schema of the selected column.
Collapse	Columns, schemas	Collapses a selected schema or a selected column's parent schema (to facilitate viewing/navigation).
Properties	Columns, schemas	Displays the properties of the selected element. You *cannot* modify the properties. See the Chapter 2, "Data Integrator Objects" for descriptions of object properties.

Output schema

The output schema area displays the schema output from the Query transform as a hierarchical tree. The output schema can contain one or more of the following elements:

- Columns
- Nested schemas
- Functions

Icons preceding columns are combinations of the following graphics:


Primary key.


Column that has a simple mapping. A simple mapping is either a single column or an expression with no input column (that is, an expression that does not vary with input).

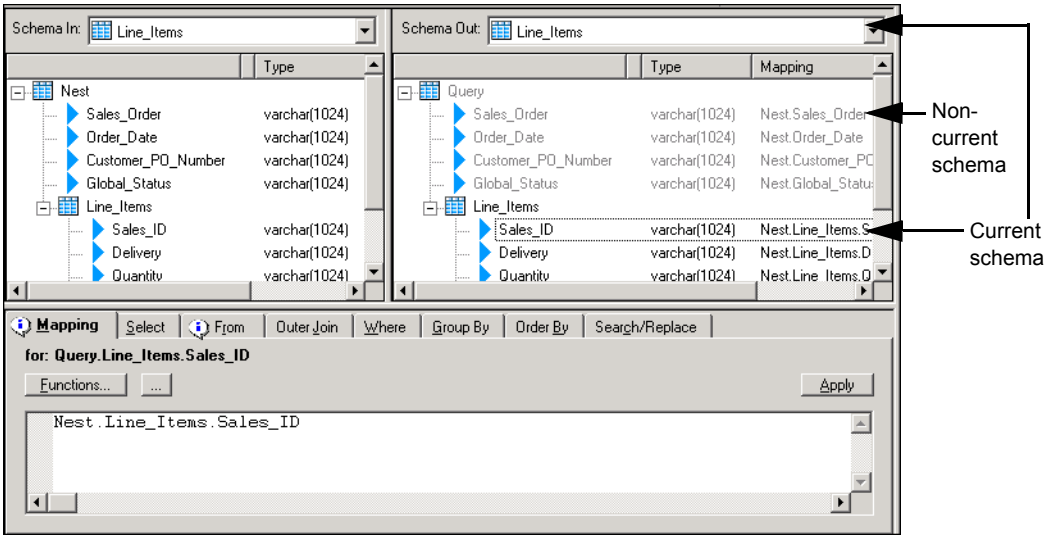

Column that has a complex mapping. A complex mapping is any mapping that is not simple.


(Red cross superimposed on any icon) Incorrect mapping. Data Integrator does not perform a complete validation during design, so the editor might not flag an incorrect mapping. For a complete validation, select **Debug** > **Validate**.

The **Output** list at the top right of the query editor indicates the current schema. The current schema determines:

- The output elements that you can modify (add, map, or delete)
- The scope of the **Select** through **Order by** tabs in the options area (see "Options" on page 296).

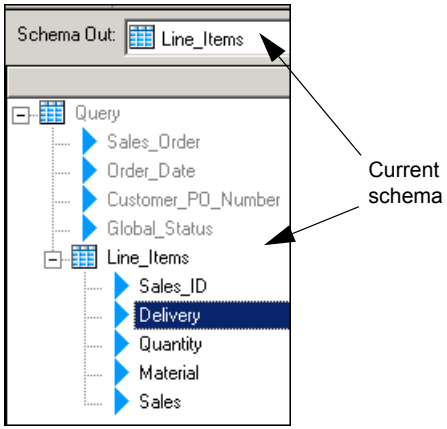
Non-current schemas appear dimmed in the output schema area.



There are several ways to change the current schema:

- Select a schema from the **Output** list.
- Right-click a schema, column, or function in the output schema area and select **Make Current**.
- Double-click one of the non-current (dimmed) elements in the output schema area.

The **Output** list and output schema area shows the selected schema.



When you connect a target table to a query with an empty output schema, Data Integrator automatically fills the query's output schema with the columns from the target table, without mappings.

Data Integrator only fills the target schema in the output of a query when you connect a target table to a query with an empty output schema. If the output schema contains any column mappings, Data Integrator does not overwrite those mappings. Similarly, if you connect a query to one target, then disconnect that target and connect to another target, the output schema will show the columns from the first target connected.

Data Integrator does not fill the output schema from a file, an XML message, an IDoc, or any other target.

There are several techniques to change the output schema:

- Drag and drop (or copy and paste) columns or nested schemas from the input schema area to the output schema area (this provides simple column mappings).

  If you drop a column on an existing column, you can remap that column. Select **Remap Column** to update only the column mapping or select **Remap with Data Type** to update the column mapping and data type. Alternatively, you can select **Insert Above** or **Insert Below** to add the column as a new mapping or **Cancel** if you do not want to add the column to the output schema.

- Right-click the current schema and select **New Output Column** or **New Output Schema**. You can provide simple column mappings by dragging input columns over the new output columns. For complex mappings, use the options area (see "Options" on page 296).

- Right-click the current schema and select **New Function**. The function must already be imported into the Data Integrator repository. You can add adapter functions and SAP R/3 functions (if you have the SAP license extension). These functions return multiple columns (in contrast to the functions used in mappings and Where clauses, which return single values). In the Define Input Parameter(s) window, map all first-level input parameters for the function to the input parameters of the query.

- Right-click columns in the current schema to assign and reverse primary key settings on output columns. A key icon indicates primary keys.

- Right-click the current schema and select **Unnest** to flatten output schemas. Use this command when a job has a source with a nested schema (such as an XML file), and you map columns from this source to a flat target table schema.

You can right-click elements in the output schema area and select commands. Generally, the elements must be within the current schema.

Right-click commands available in output schema area

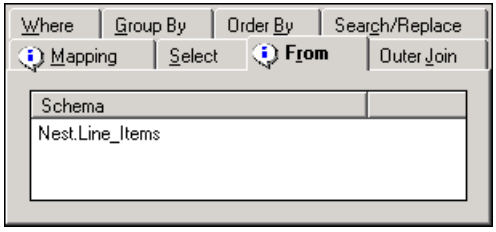
Command	Applicable elements	Effect
Cut	All	Removes the selected elements from the output schema area and stores a copy of the elements in the clipboard.
Copy	All	Stores a copy of the selected elements in the clipboard, leaving the elements in the output schema area.
Paste	All	Inserts the elements stored in the clipboard at the current cursor location (this must be within the current schema). Only visible when the clipboard contains something. If the cursor overlaps an existing column, you are prompted to insert above, insert below, remap column, or cancel. **NOTE**: Copying an input element and pasting it in the output schema area provides a simple mapping from the input element to the output element. You can also do this by *dragging* the input element to the output schema area.
Delete	All	Removes the selected elements from the output schema area (without making a copy).
Find...	All	Locates an output element with the name or description you enter.
Make Current	All outside the current schema	Makes the selected schema, or the schema of the selected element, the current schema.
New Output Column...	Schemas	Adds an output column to the current schema with the name and properties you enter.
New Output Schema...	Schemas	Adds a nested schema to the current schema with the name you enter.

Right-click commands available in output schema area (Continued)

Command	Applicable elements	Effect
New Function Call...	Schemas	Adds a function or stored procedure call to the current schema. The function or procedure must already be imported into the Data Integrator repository. For information about adding stored procedures, see "Query output schemas" on page 505. You can add adapter functions and (with the SAP license extension) SAP R/3 functions. These functions return multiple columns (in contrast to the functions used in mappings and Where clauses that return single values). In the Define Input Parameter(s) window, map all first-level input parameters for the function to the input parameters of the query.
Modify Function	Functions	Allows you to modify the selected function. In the Define Input Parameter(s) window, map all first-level input parameters for the function to the input parameters of the query.
Unnest	Nested schemas	Toggles to flatten or re-nest the selected schema into the parent schema in the query output. An unnested schema will not appear in the succeeding transform or target; only its columns appear. Unnested schemas appear in the Query output schema area as table icons with a black arrow that points to the left.
Primary Key	Columns	Toggles the primary key attribute of the column on (check mark appears next to command) or off (no check mark appears next to command). A key icon indicates a column is a primary key.
Generate DTD...		Generates a DTD format that corresponds to the structure of the selected schema (either NRDM or relational). Generates all data types as varchar.
Generate XML Schema...		Generates an XML Schema that corresponds to the structure of the selected schema (either NRDM or relational). All data types match those of the selected schema. For more information, see "Generating DTDs and XML Schemas from an NRDM schema" on page 233 of the *Data Integrator Designer Guide*.
Create File Format...		Creates a file format from a relational table schema. All data types match those of the original table schema.
Properties	All	Displays the properties of the selected element.

Options

The options area contains several tabs where you enter information to specify the data you want retrieved. Specifying information on these tabs is similar to specifying a SQL SELECT statement. Tabs containing entries are flagged by a special icon.



When you drag and drop (or copy and paste) input columns to the output schema, Data Integrator inserts values in the **Mapping** and **From** tabs. For simple mappings, this is often sufficient.

For more complex mappings, complete the appropriate tabs.

Tab	Description
Mapping	Specifies how the selected output column will be derived (or mapped).
Select	Specifies whether to output only distinct rows (discarding any identical duplicate rows).
From	Specifies all input schemas that are used in the current schema.
Outer Join	Specifies an inner table and outer table for any joins (in the **Where** tab) that are to be treated as outer joins.

Where	Specifies conditions that determine which rows are output.
	Enter the conditions in SQL syntax, like a WHERE clause in a SQL SELECT statement. For example:
	`TABLE1.EMPNO = TABLE2.EMPNO`
	`AND TABLE1.EMPNO > 1000`
	`OR TABLE2.EMPNO < 9000`
	Use the buttons above the smart editor for help building expressions.

Group By	Specifies how the output rows are combined (if required).
Order By	Specifies how the output rows are sorted (if required).
Search	Enables you to search for a specific word or term in the input and/or output schemas.

NOTE: Use the **Select** through **Order By** tabs to specify additional constraints for the current schema, similar to SQL SELECT statement clauses.

- **Mapping** tab

  Use the **Mapping** tab to specify how the selected output column is derived (or mapped). You can specify any valid expression. See Chapter 3, "Smart Editor".

  Most commonly, mapping expressions contain table columns and functions:

  - Type input column names or drag columns from the input schema and drop them in the box on the **Mapping** tab.

- Insert functions by entering them directly, using the smart editor, or by clicking the **Functions** button to open the function wizard.

- **Select** tab

  Use the **Select** tab to output only distinct rows (discarding any identical duplicate rows).

  To discard duplicate rows, select the **Distinct rows** check box. This is similar to specifying a SELECT DISTINCT SQL statement.

- **From** tab

  Use the **From** tab to specify the input schemas used in the current output schema. Type input schema names or drag schemas from the input schema and drop them in the box on the **From** tab. This list is similar to the FROM clause in a SQL SELECT statement.

- **Outer Join** tab

  Use the **Outer Join** tab to specify an inner table and an outer table for joins that you want treated as outer joins. You specify join conditions on the **Where** tab. An outer join shows all rows that meet the join condition, plus all rows in the outer table that do not meet the join condition. Therefore, an outer join produces all rows from the "outer table" along with all "inner table" data that satisfies the join condition.

  To enter schema names, right-click in the editor space, select **Insert**, and then select an input schema from the drop-down list. To delete a schema, right-click and select **Delete**. For more information about outer joins, see "Outer join details" on page 304.

- **Where** tab

  Use the **Where** tab to set conditions that determine which rows are output.

  Enter the conditions in SQL syntax, like a WHERE clause in a SQL SELECT statement.You can specify:

  - Join conditions; for example:

    ```
    TABLE1.EMPNO = TABLE2.EMPNO
    ```

  - Data set filters; for example:

    ```
    TABLE1.EMPNO > 1000
    ```

  - Multiple conditions using logical operators; for example:

    ```
    TABLE1.EMPNO = TABLE2.EMPNO
    AND TABLE1.EMPNO > 1000
    OR TABLE2.EMPNO < 9000
    ```

  You can specify any valid expression. See Chapter 3, "Smart Editor". To enter conditions, you can:

  - Type expressions in the editor.
  - Drag columns from the input schema area to the editor.
  - Use the **Functions** button.

    See Chapter 6, "Functions and Procedures" for more information about functions.

    Use the pushdown_sql function to have Data Integrator create WHERE clauses dynamically based on data rather than prespecifying the clause (see "pushdown_sql" on page 446 for more information).

  - Use the **Propose Join** button.

    Click the **Propose Join** button to have Data Integrator generate a join expression. See "Propose join details" on page 303 for more information.

- **Group By** tab

  Use the **Group By** tab to specify a list of columns for which you want to combine output. For each unique set of values in the group by list, Data Integrator combines or aggregates the values in the remaining columns. For example, you might want to group sales order records by order date to find the total sales ordered on a particular date.

  To add a column to the group by list, select the column in the input schema area and drag it to the box on the **Group By** tab. To remove a column, right-click the column and select **Delete**.

  To group by complex expressions (rather than by specific column values), use another query to produce a single column containing the grouping expression. Insert the new query immediately before this transform in your data flow, and specify the created column on the **Group By** tab.

  If you specify a group by list, then all columns in the output schema must be either in the group by list or mapped to an aggregate function, such as avg, count, max, min, or sum.

  This tab is similar to the GROUP BY clause in a SQL SELECT statement.

- **Order By** tab

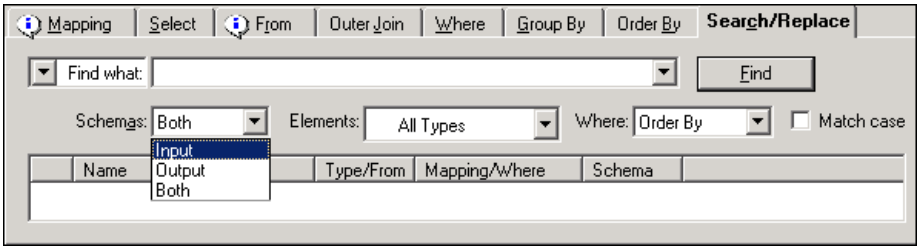
  Use the **Order By** tab to specify the columns you want used to sort the output data set. Add columns in the order you want them sorted. To add a column, select the column in the input schema area and drag it to the box on the **Order By** tab. The Designer adds the column to the bottom of the list. Select **Ascending** or **Descending** from the adjacent drop down box. The first column listed is used for primary sorting, the second column listed is used for secondary sorting, and so forth.

- **Search** tab

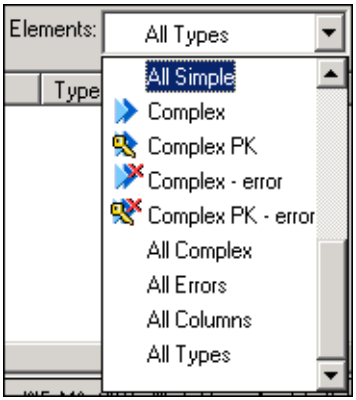
  Use the **Search** tab to search for a specific word or term in the input schema or the output schema.

➢ To search in an input schema or output schema

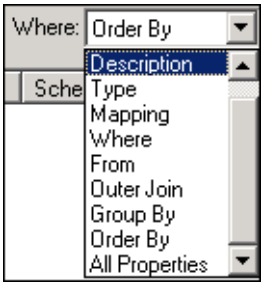
1. Enter the search term in the **Find what** box or select from previous search terms in the drop-down list.
2. In the **Schemas** list, choose the schemas in which to search.



3. In the **Mapping** list, choose the types of mappings in which to search.



4. In the **Where** list, choose the properties to search within.



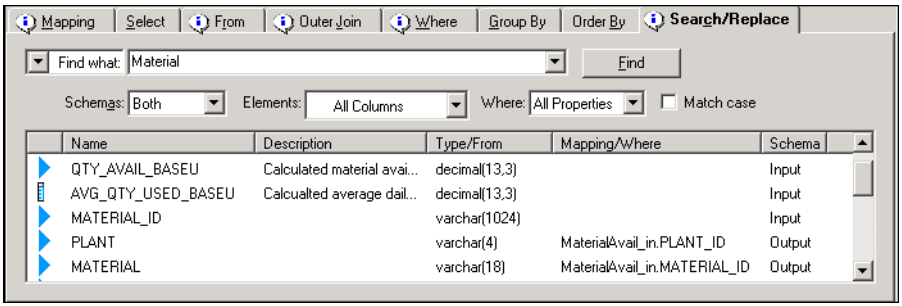
NOTE: You can search within one or all properties, but not within two or three specific properties at a time.

5. Select the **Match case** check box to constrain your search to the capitalization entered.
6. Click **Find**.

   The Designer searches the query for the word(s) you specified within the constraints you defined.

   NOTE: The Designer searches for columns loaded into memory. If columns are not loaded into memory, you must expand the schema to load the columns into memory before clicking **Find** and searching for the columns.

All matches are shown in the box below the find constraints. When you click to select a table or column name, the table or column is automatically highlighted and shown in the corresponding input or output schema area.



Initially, the Designer lists the matching columns in the order they appear within the schemas. If you are searching both schemas, the Designer lists the first match found in the input schema first and the last match found in the output schema last. You can sort the list of matches by property. Each time you click a property heading, the Designer resorts the matches, cycling through original order, ascending order, and descending order.

Arrow icons confirm column and sort type. For example, if you sort the data by the **Description** property and in ascending order, an "up" arrow appears next to the **Description** heading. Click that heading again and a "down" arrow appears to indicate that the data is now sorted in descending alpha-numeric order. Click again and the match list returns to its original sort order.

Data Outputs

A data set based on the conditions you specify and using the schema specified in the output schema area.

Propose join details

When you click the **Propose Join** button on the **Where** tab, Data Integrator generates a join expression. For each pair of sources, the generated expression includes a join condition based on primary and foreign keys and column names:

- If a foreign key relationship exists, Data Integrator adds a join condition to the expression for the columns related through keys. For example, if foreign key K2 of table T2 references primary key K1 of table T1, Data Integrator includes the join condition: T1.K1=T2.K2

- If no foreign key relationship exists, Data Integrator adds a join condition to the expression for columns with the same name where at least one column is part of a primary key. For example, suppose there is no foreign key relationship between tables T and S, that both tables contain column A, and that column A is part of the primary key in table S. In this case, Data Integrator includes the join condition: T.A = S.A
- If neither condition is satisfied, Data Integrator proposes no join condition for that pair of sources.

For outer joins, the number of joins proposed is limited by the join relationship. See "Propose join and outer joins" on page 310.

Because Data Integrator proposes as many joins as are possible given the inputs, you must review and edit the generated expression to reflect the conditions you want. Edit the resulting expression as necessary to reflect the required join relationship between the sources. Each time you click **Propose Join**, Data Integrator writes the join expression in the WHERE clause.

The join ranks set in the source objects determine the order that Data Integrator uses for each data set in a join. See "Join ordering" on page 24 of the *Data Integrator Performance Optimization Guide*.

Outer join details

An outer join between two data sets shows you all of the rows that meet the join condition, plus all of the rows in one data set that did not meet the join condition. Thus all rows from one data set—referred to as the "outer table"—are reproduced at least once in the result. Only the data from the "inner table" that satisfy the join condition appear in the result.

For rows from the outer table that do not have corresponding data from the inner table, the missing values are filled in as null. In a normal join between the tables, the same rows would be absent in the result.

Sources joined in a query produce different results based on the join:

Employee Table

EmpID	LastName
1008	Alverez
1009	Davis
1010	Tanaka
1011	Laprais

An **Inner Join** produces 3 rows

EmpID	LastName	Bonus
1008	Alverez	1000
1009	Davis	1500
1011	Laprais	1000

Bonus Table

EmpID	Bonus
1008	1000
1009	1500
1011	1000

An **Outer Join** produces 4 rows where the extra row is filled with NULLs

EmpID	LastName	Bonus
1008	Alverez	1000
1009	Davis	1500
1010	Tanaka	NULL
1011	Laprais	1000

For example, in Oracle, you can produce an outer join for this data with the following query:

```
SELECT Employee.EmpID, Employee.LastName,
   Bonus.Bonus
FROM Employee, Bonus
WHERE Employee.EmpID = Bonus.EmpID (+);
```

The outer table is `Employee` and the inner table is `Bonus`.

The outer join predicate is:

```
Employee.EmpID = Bonus.EmpID (+);
```

In Microsoft SQL Server and DB2, the query is:

```
SELECT Employee.EmpID, Employee.LastName,
   Bonus.Bonus
   FROM Employee Left Outer JOIN Bonus
   ON (Employee.EmpID = Bonus.EmpID)
```

The outer table is `Employee` and the inner table is `Bonus`.

In Data Integrator, the query is constructed as follows:

Input to query:	`Employee` and `Bonus`
Outer Join tab:	Outer Table = `Employee` Inner Table = `Bonus`
Where tab:	`Employee.EmpID = Bonus.EmpID`

Comparison with lookup function

You can produce a similar data set using the lookup function. However, the lookup function:

- Returns only one column value for each comparison
- Can be used against only one source at a time
- Cannot be used to produce the same results as non-equality joins (for example, `A.x<B.y`)
- Permits default values other than nulls
- Can be cached when desired

An outer join:

- Returns all column values for each comparison
- Provides similar capability to multiple lookup calls
- Allows non-equality joins

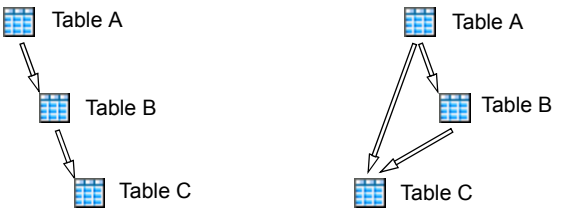
Sources in an outer join query must be joined in a hierarchical order:

- A source can only be the inner table of one outer join
- A source cannot be "outer joined" with itself in a single query transform

Outer join construction

Data Integrator follows the SQL standard for constructing outer joins. This standard imposes the following restrictions for producing outer joins:

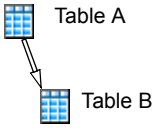
- A query that includes an outer join cannot also include an inner join (that is, a "normal join").
- A source object cannot be an inner table to more than one outer table in the same query.



Correct outer join relationship
Each table is an inner table to only one outer table.

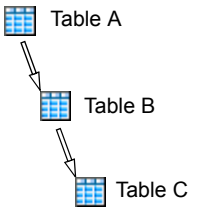
Incorrect outer join relationship
Table C is an inner table for both Table A and Table B.

NOTE: The diagrams used to illustrate the relationship between sources included in an outer join use a double-shafted arrow to indicate the join relationship. The source at the butt of the arrow is the outer table in the join; the source at the head of the arrow is the inner table in the join. Although the illustrations use the term "table," the join can include files, tables, or (in real-time jobs) messages as sources.

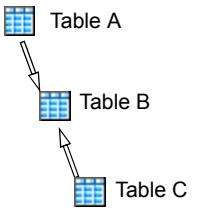


This notation indicates an outer join between Table A and Table B where:
Table A is the outer table
Table B is the inner table

- All sources except one must be an inner table.



Correct outer join relationship
Only Table A is not an inner table to any other table.



Incorrect outer join relationship
Both Table A and Table C are not inner tables.

- Every source read by the query must be included in the WHERE clause.

  This rule combined with the first rule (cannot have an inner and an outer join in the same query) requires that all of the input tables be part of the outer join description.

- If a source is included in more than one outer join in the same query, the WHERE clause cannot contain terms that compare any column of the source with a constraint.

  Columns from the outermost table or the innermost table can be constrained.

  NOTE: When all of the sources used as inputs to the query are Oracle tables, and if a column is constrained in the outermost table, Data Integrator does not push the operation down to Oracle to perform the join.

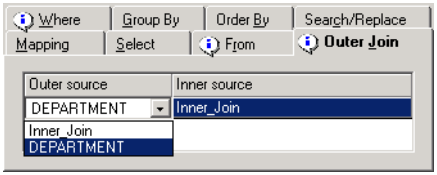
- The WHERE clause of a query including an outer join cannot contain the OR operator.

You are unlikely to encounter these situations in normal day-to-day data movement activities. However, if you do have situations that would conflict with these rules, you can use more than one query to produce the desired results.

➢ To construct a query with an outer join

1. Connect the sources to a query in a data flow.
2. Open the query editor.
3. Click the **Outer Join** tab.

   The tab contains two columns in which to specify the source names for the outer join.



4. Right-click and choose **Insert** to add a set of drop-down boxes under each column.
5. In the **Outer Table** list, select the source you want to use as the outer table in the join. In the **Inner Table** list, select the source you want to use as the inner table in the join.
6. To add another table to the join, right click and choose **Insert** to add another set of drop-down boxes under each column.

   NOTE: You cannot select a source as the **Inner Table** more than once.

7. Go to the **Where** tab to specify the outer join predicate.
8. Click **Propose Join**.

   You can also specify the join conditions by entering them manually.

9. Determine the output schema by dragging the columns from the input schema area to the output schema area.
10. Validate the data flow that contains the Query transform by choosing **Debug** > **Validate**.

Propose join and outer joins

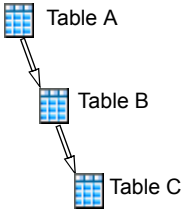
If you click the **Propose Join** button after filling in the **Outer Join** tab, Data Integrator generates a join expression for each pair of sources related through the outer join. (Whereas, for inner joins, Data Integrator generates a join expression for *every* pair of sources.) Data Integrator uses the same algorithm to produce the join expression for outer joins that it uses for inner joins—an algorithm based on foreign keys or primary keys and column names (see "Propose join details" on page 303).

How join ranks are used in outer joins

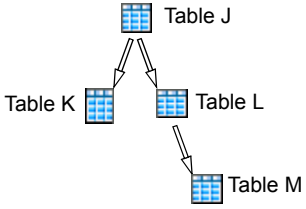
Data Integrator reads sources in preparation for the join in an order determined by a "depth first traversal" of a tree starting from the root node. Controlling this order—referred to as the join order in this description—can often have a profound effect on the performance of producing the join result. The join order has no effect on the actual result produced. This section describes how join ranks can affect the join order.

Each row in the **Outer Join** tab represents a branch of the tree.

Consider these examples without join ranks set:



In this join tree, there is only one branch to traverse. Table A is read then joined with Table B. The results from that join are joined with Table C.

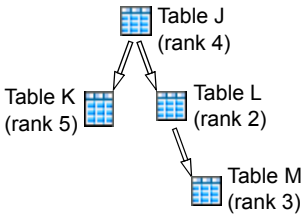


In this join tree, the longest branch is traversed first. Table J is read, then joined with Table L. The results are joined with Table M. The results are then joined with Table K.

You can use join ranks to control the order in which the sources are joined. Data Integrator joins the child nodes in the order of decreasing join ranks. The join rank associated with a node is the maximum join rank of any node further down the branch.

For example, suppose you are joining four tables in an outer join—tables J through M—and you assign each a different join rank.

In this join tree, the join rank of Table L is considered 3 because the join rank of Table M is greater than the original join rank of Table L. However, the join rank of Table K (5) is still larger than the join rank of Table L. The join order is as follows: Table J is read and joined with Table K. The results are joined with Table L. The result of that join is joined with Table M.



For more information about maximizing performance in join situations, see "Join ordering" on page 24 of the *Data Integrator Performance Optimization Guide*.

Outer join examples

Expanding on the original outer join example with employees and their bonuses, it is possible to use outer joins to produce a result showing all of the departments in a company, the employees included in them, and their bonuses. In this case, the Department table would be the outermost join source, the Employee table the next, and the Bonus table the innermost join source.

Department Table

DeptID	Department
01	Accounting
02	Finance
03	Sales
04	Marketing

Employee Table

EmpID	LastName	DeptID
1008	Alverez	01
1009	Davis	02
1010	Tanaka	01
1011	Laprais	01

Bonus Table

EmpID	Bonus
1008	1000
1009	1500
1011	1000

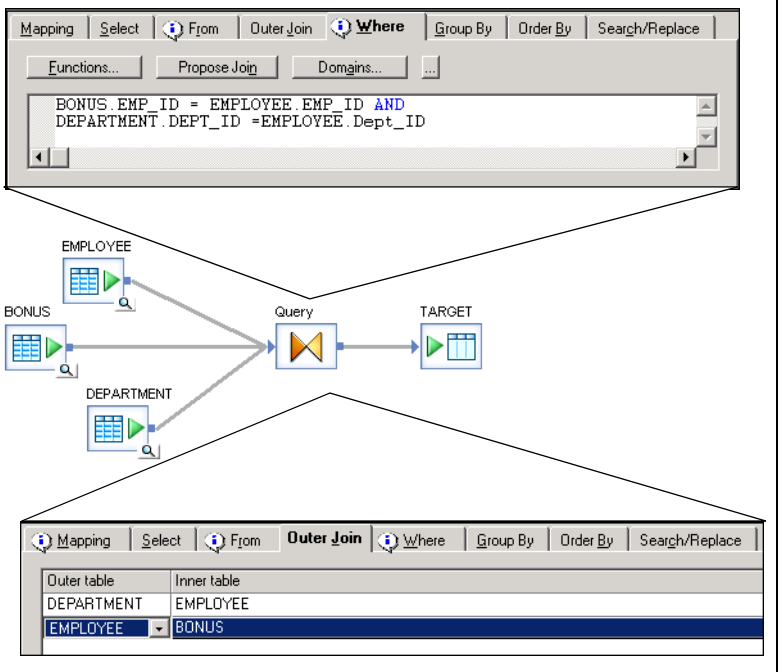
An **Outer Join** produces 6 rows where the extra rows are filled with NULLs

Outer Join Results

DeptID	Department	LastName	Bonus
1	Accounting	Alverez	1000
1	Accounting	Tanaka	NULL
1	Accounting	Laprais	1000
2	Finance	Davis	1500
3	Sales	NULL	NULL
4	Marketing	NULL	NULL

The query contains the outer join list and the WHERE clause for the three sources.

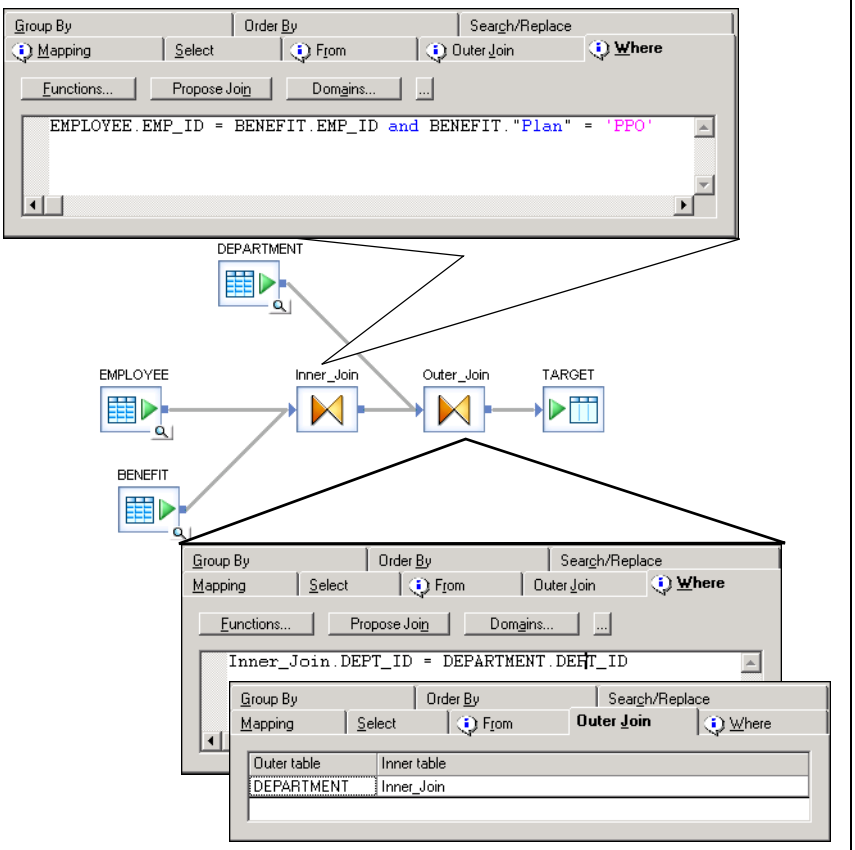
More than two sources can be included in the outer join



In some cases, you must use more than one query object to obtain your desired result because of outer join restrictions. Suppose you want a list of all departments that have employees who subscribe to a particular benefit plan. You need an outer join and an inner join. An outer join between the Department table and the Employee table produces the list of all departments and the corresponding employees. An inner join between the Employee and Benefit tables with a constraint on the benefit

plan produces only the employees who subscribe to the plan identified by the constraint.

An inner and outer join require two queries



This combination requires two queries because a single query cannot describe both inner and outer joins. An inner join is required to eliminate the Employee rows that do not have the benefit plan identified by the constraint.

Row_Generation

Produces a data set with a single column. The column values start from zero and increment by one to a specified number of rows.

DATA INPUTS

None.

OPTIONS

Row count — A positive integer indicating the number of rows in the output data set.

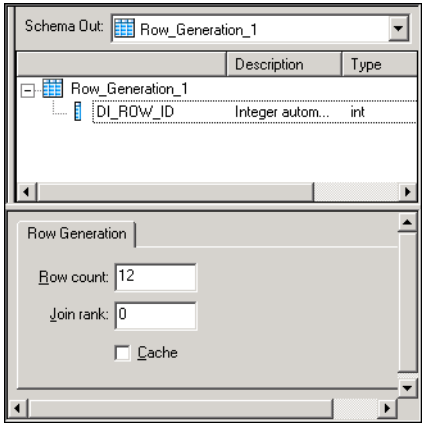
Join rank — A positive integer indicating the weight of the output data set if the data set is used in a join. Sources in the join are accessed in order based on their join ranks. The highest ranked source is accessed first to construct the join.

Cache — Select this check box to hold the output from the transform in memory to be used in subsequent transforms.

Select **Cache** only if the resulting data set is small enough to fit in memory.

EDITOR

The Row_Generation transform editor includes the target schema, and transform options.



DATA OUTPUTS

The Row_Generation transform produces a data set with a single column and the number of rows specified in the **Row** option. The rows contain integer values in sequence starting from zero and incrementing by one in each row.

SQL

Performs the indicated SQL query operation.

Use this transform to perform standard SQL operations for things that cannot be performed using other built-in transforms.

Data Inputs

None. This transform does not allow an input data set.

Options

Datastore — The name of the datastore that Data Integrator uses to access the tables referred to in **SQL text**.

Join rank — A positive integer indicating the weight of the output data set if the data set is used in a join. Sources in the join are accessed in order based on their join ranks. The highest ranked source is accessed first to construct the join.

Cache — Select this check box to hold output from the transform in memory for use in subsequent transforms.

Select **Cache** only if the resulting data set is small enough to fit in memory.

SQL text — The text of the SQL query. This string is passed to the database server.

You do not need to put enclosing quotes around the SQL text. You can put enclosing quotes around the table and column names, if required by the syntax rules of the DBMS involved.

Update schema — Click this option to automatically calculate and populate the output schema for the SQL SELECT statement.

DATA OUTPUTS

There are two ways of defining the output schema for a SQL transform if the SQL submitted is expected to return a result set:

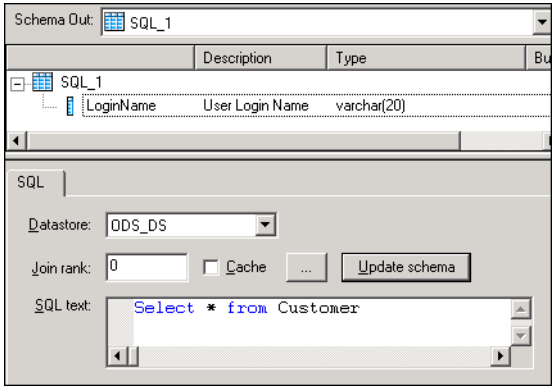
- **Automatic** — After you type the SQL statement, click **Update schema** to execute a described select statement against the database which obtains column information returned by the select statement and populates the output schema.
- **Manual** — Output columns must be defined in the output portion of the SQL transform if the SQL operation is returning a data set (for example, if the SQL operation is a SELECT).

  The number of columns defined in the output of the SQL transform must equal the number of columns returned by the SQL query.

  The column names and data types of the output columns need not match the column names or data types in the SQL query. Data Integrator conversion rules apply. (When possible, Data Integrator optimizes data flows by pushing expressions down to an underlying database manager. In a single transaction, Data Integrator can push down expressions so that they are performed by the underlying database manager. However, when Data Integrator evaluates an expression which includes operands of more than one data type, Data Integrator attempts to convert the operands to the same data type first. (Except for national character-set data types which can be pushed down while others in an expression are not. For more information about supported national character-set data types, see "varchar" on page 205).) Errors are flagged for illegal conversion.

  The output data set cannot contain hierarchical data.

Exercise care when specifying the output columns. Typically the column data types of the two sets of columns should be an exact match. If you choose to have different data types, you need to ensure that they are compatible—if they are not, you will get a runtime error from the underlying database manager



Table_Comparison

Compares two data sets and produces the difference between them as a data set with rows flagged as INSERT or UPDATE.

The Table_Comparison transform allows you to detect and forward changes that have occurred since the last time a target was updated.

For information about how these operation codes (INSERT and UPDATE, for example) are used, see "Operation codes" on page 224.

Data Inputs

- The data set from a source or the output from another transform. Only rows flagged as NORMAL are considered by the transform. This data is referred to as the input data set.
- The specification for a database table to compare to the input data set. This table is referred to as the comparison table.

If the input data set contains hierarchical data, only the top-level data is included in the comparison, and nested schemas are not passed through to the output.

Use caution when using columns of data type real in this transform. Comparison results are unpredictable for this data type. See "real" on page 200 for more information.

OPTIONS

Table name The fully qualified name of the source table from which the maximum existing key is determined (key source table). This table must be already imported into the repository. **Table name** is represented as "DATASTORE.OWNER.TABLE" where DATASTORE is the name of the datastore Data Integrator uses to access the key source table; OWNER, if required, depends on the database type associated with the table:

Database type	Owner value
DB2	Data source dependent
Informix	Informix-defined user name
Microsoft SQL Server	User name
ODBC	Data source dependent
Oracle	User name
Sybase	User name

Generated key column

(Optional) A column in the comparison table. When there is more than one row in the comparison table with a given primary key value, this transform compares the row with the largest generated key value of these rows and ignores the other rows.

If there is more than one row in the comparison table with the same primary key value and generated key value, the transform arbitrarily chooses which row of these rows to compare.

If you do not specify a **Generated key column**, and your comparison table contains rows with the same primary keys, the transform arbitrarily chooses which row of these rows to compare.

If the input data set and the comparison table both have the column you specified in **Generated key column**, the transform does not compare the values for this column.

Comparison method

Select the method for accessing the comparison table:

- **Row-by-row select** — Select this option to have the transform look up the target table using SQL every time it receives an input row. This option is best if the target table is large compared to the number of rows the transform will receive as input.
- **Cached comparison table** — Select this option to load the comparison table into memory. In this case, queries to the comparison table access memory rather than the actual table. However, the table must fit in the available memory. This option is best when the table fits into memory and you are comparing the entire target table.
- **Sorted input** — Select this option to read the comparison table in the order of the primary key column(s) using sequential read.

  This option improves performance because the comparison table is read only once.

  To take advantage of this option, the input data set order must match the **primary key column(s)** order in the Table_Comparison transform.

If this is already the case, drag the primary key column(s) from the input schema in the Table_Comparison transform into the **Input primary key columns** box. Data Integrator will read the comparison table in the order of the primary key column(s) using a sequential read.

If you need to pre-sort the input data, add a query between the source and the Table_Comparison transform. Then, from the query's input schema, drag the primary key columns into the **Order By** box of the query. The query's columns, in the **Order By** box, must match the order of the primary key columns that you drag into the **Primary Key Column(s)** box of the Table_Comparison transform.

In this way, you explicitly add a sort operation (ORDER BY) to the input data set of the query to ensure that the order of the input data set will match the order of the read from the comparison table data in the transform.

Input primary key column(s)

The column(s) in the input data set which uniquely identify each row. These columns must be present in the comparison table with the same column names and data types.

Drag the column(s) from the input schema into the **Input primary key columns** box.

The transform selects rows from the comparison table that match the values from the primary key columns in the input data set.

If values from more than one column are required to uniquely specify each row in the table, add more than one column to the **Input primary key columns** box.

You cannot include nested schemas in the **Input primary key columns** list.

Compare columns

(Optional) Improves performance by comparing only the sub-set of columns you drag into this box from the input schema.

If no columns are listed, all columns in the input data set that are also in the comparison table are used as compare columns.

Data Outputs

A data set containing rows flagged as INSERT or UPDATE. This data set contains only the rows that make up the difference between the two input sources. The schema of the output data set is the same as the schema of the comparison table. Note that no DELETE operations are produced.

The transform compares two data sets, one from the input to the transform (input data set), and one from a database table specified in the transform (the comparison table). The transform selects rows from the comparison table based on the primary key values from the input data set. The transform compares columns that exist in the schemas for both inputs.

If a column has a date data type in one table and a datetime data type in the other, the transform compares only the date section of the data. The columns can also be time and datetime data types, in which case Data Integrator only compares the time section of the data.

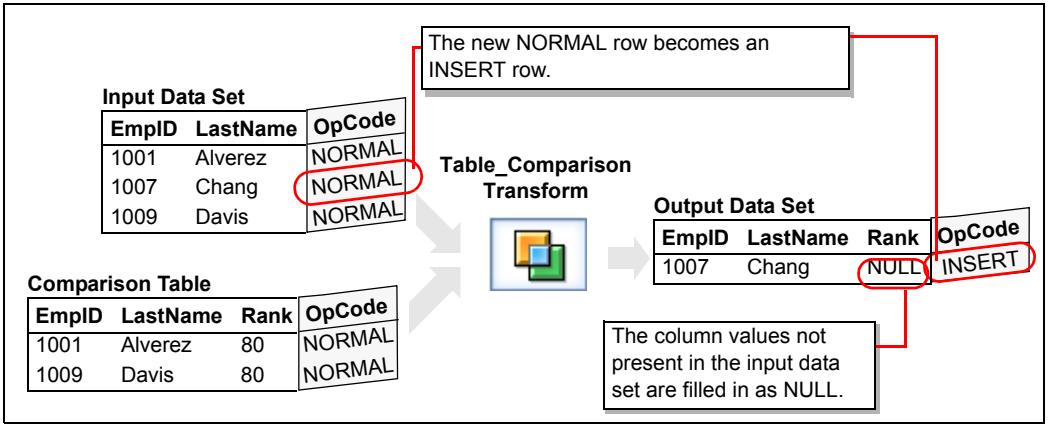
The transform generates a data set consisting of rows with INSERT and UPDATE operation codes. No DELETE rows are produced: if a primary key value in the comparison table is not present in the input data set, no corresponding row appears in the output.

For each row in the input data set, there are three possible outcomes from the transform:

- An INSERT row.

  The primary key value from the input data set does not match a value in the comparison table. The transform produces an INSERT row with the values from the input data set row.

  If there are columns in the comparison table that are not present in the input data set, the transform adds these columns to the output schema and fills them with NULL values.

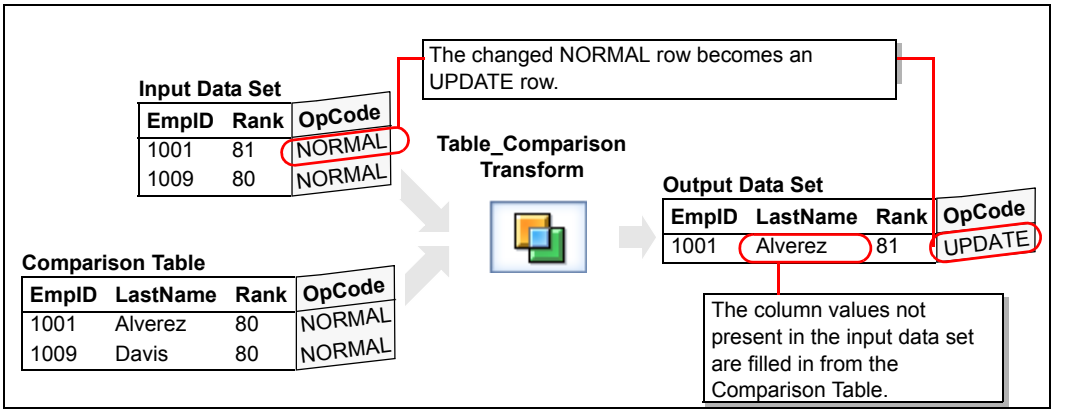


- An UPDATE row.

  The primary key value from the input data set matches a value in the comparison table, and values in the non-key compare columns differ in the corresponding rows from the input data set and the comparison table.

The transform produces an UPDATE row with the values from the input data set row.

If there are columns in the comparison table that are not present in the input data set, the transform adds these columns to the output schema and fills them with values from the comparison table.



- The row is ignored.

  The primary key value from the input data set matches a value in the comparison table, but the comparison does not indicate any changes to the row values.

A generated key column can indicate which row of a set containing identical primary keys is to be used in the comparison. For an UPDATE, the output data set will contain the largest generated key found for a given primary key. For an INSERT, the output data set will contain a NULL value for the generated key. To produce a new generated key, direct the output of the Table_comparison transform into a Key_Generation transform. For information, see "Key_Generation" on page 259.

6 Functions and Procedures

In Data Integrator, functions take input values and produce a return value. Procedures take input values and perform a set of operations with returning a specific value. Input values can be parameters passed into a data flow, values from a column of data, or variables defined inside a script. This chapter discusses functions and procedures, including detailed descriptions of built-in functions—the input parameters and required syntax, and the return values and data types.

This chapter contains the following sections:

About functions

This section discusses general information about functions. Specifically, this section discusses:

- Functions compared with transforms
- Operation of a function
- Arithmetic in date functions
- Including functions in expressions
- Kinds of functions you can use in Data Integrator

Functions compared with transforms

Some functions can produce the same or similar values as transforms. However, functions and transforms operate in a different scope:

- Functions operate on single values, such as values in specific columns in a data set.
- Transforms operate on data sets, creating, updating, and deleting rows of data.

Data Integrator does not support functions that include tables as input or output parameters, except functions imported from SAP R/3.

Operation of a function

The function's operation determines where you can call the function. For example, the lookup function operates as an iterative function. The lookup function can cache information about the table and columns it is operating on between function calls. By contrast, conversion functions, such as to_char, operate as stateless functions. Conversion functions operate

independently in each iteration. An aggregate function, such as max, requires a set of values to operate. Neither the lookup function (iterative) nor the max function (aggregate) can be called from a script or conditional where the context does not support how these functions operate.

The function type determines where a function can be used:

Type	Description
Aggregate	Generates a single value from a set of values. Aggregate functions, such as max, min, and count, use the data set specified by the expression in the **Group By** tab of a query. Can be called only from within a Query transform—not from custom functions or scripts.
Iterative	Maintains state information from one invocation to another. The life of an iterative function's state information is the execution life of the query in which they are included. The lookup function is an iterative function. Can be called only from within a Query transform—not from functions or scripts.
Stateless	State information is not maintained from one invocation to the next. Stateless functions such as to_char or month can be used anywhere expressions are allowed.

Arithmetic in date functions

Data Integrator performs some implicit data type conversions on date, time, datetime, and interval values. For a table describing these conversions, see "Date arithmetic" on page 207.

Including functions in expressions

In Data Integrator, you can use functions in the following:

- Transforms (Query, Case, SQL)
- Script objects
- Conditionals
- Other custom functions

Before you use a function, you need to know if the function operation makes sense in the expression you are creating.

For example:

- The max function cannot be used in a script or conditional where there is no collection of values on which to operate.
- Parameters can be output by a work flow but not by a data flow.

Data Integrator includes two editors you can use to add an existing function to an expression. These are:

- Smart editor

  Embedded in other editor windows like the Script Editor, Conditional Editor, and Query Editor, the smart editor offers color coded syntax, a right-click menu, keyboard short cuts, and a list of available variables, data type formats, and functions that you can use to define a function. For more information, see "Smart Editor" on page 173.

- Function wizard

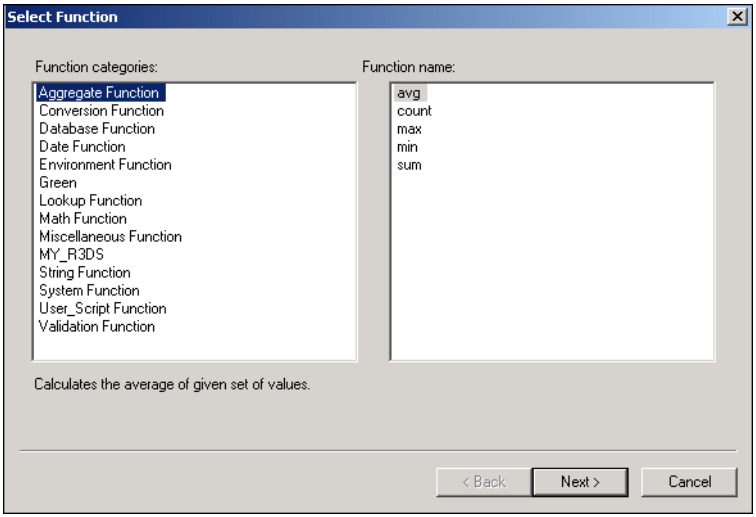
  You can use the function wizard to define the parameters for an existing function. The function wizard offers the most help when defining complex functions. The function wizard is described in detail below.

➢ To create an expression that includes an existing function

1. Go to the script, query, or conditional editor in which you will include the expression.
2. Enable the smart editor and begin entering your expression.
3. When you want to include the function, click the **Functions** button.

The Designer opens the Select Function window of the function wizard.

4. Select a category in the **Function categories** box.



A list of functions in that category appears in the **Function name** box. The functions shown depend on the object you are using. For example, the functions available for R/3 data flows are a subset of those available for data flows.

In some cases, it does not make sense to use a function even though it is available. For example, the SQL function can be called in a mapping expression or a WHERE clause, but it would result in a SQL statement inside the SQL statement Data Integrator generates to execute a data flow.

5. Select a specific function in the **Function name** list.

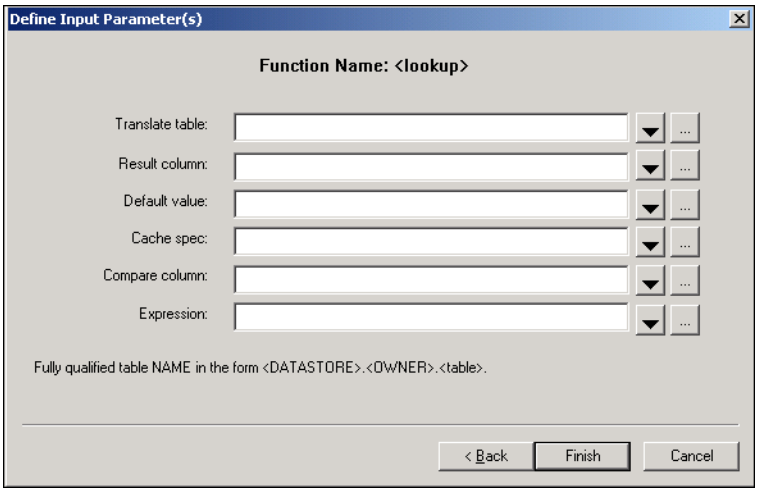
   A description of the function appears below the boxes.

6. Click **Next**.

7. Enter the values required by the function in the text boxes.

   The page shown for each function is unique. Each page is designed to help you construct the current function. The example below shows the most common layout, however more complex functions may use different layouts.

   Click in a box to see a description of the parameter at the bottom of the window. Use the arrow button to select input parameters. Use the smart editor button to see a larger input box.



8. Click **Finish**.

   The function and the parameters appear in the smart editor.

Kinds of functions you can use in Data Integrator

Data Integrator provides a set of built-in functions (see "Descriptions of built-in functions" on page 342). In addition, you can use:

- Custom functions — Functions you define yourself
- Database and application functions — Functions specific to your DBMS

In Data Integrator, database and application functions, custom functions, and most built-in functions can be executed in parallel within the transforms in which they are used. For more information, see "Degree of parallelism and functions" on page 71 of the *Data Integrator Performance Optimization Guide*.

Custom functions

You can create your own functions by writing script functions in Data Integrator scripting language (see Chapter 7, "Data Integrator Scripting Language" for more information about the scripting language and to see example functions) using the smart editor. Saved custom functions appear in the function wizard and the smart editor under the **User_Script** category. They also are displayed in the object library under the **Custom Functions** tab. You can edit and delete custom functions from the object library.

Consider these guidelines when you create your own functions:

- Functions can call other functions.
- Functions cannot call themselves.
- Functions cannot participate in a cycle of recursive calls. For example, function A cannot call function B, which calls function A.
- Functions return a value.
- Functions can have parameters for input, output, or both. However, data flows cannot pass parameters of type output or input/output.

Before creating a custom function, you must know the input, output, and return values and their data types. The return value is predefined to be `Return`.

➢ To create a custom function

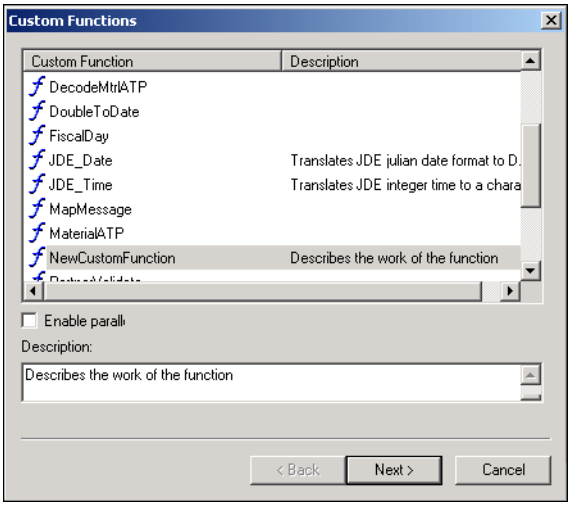
1. Choose **Tools** > **Custom Functions**.

2. In the **Custom Function** list, right-click and select **New**.

   Alternatively, from the object library, right-click and select **New** in the **Custom Functions** tab.

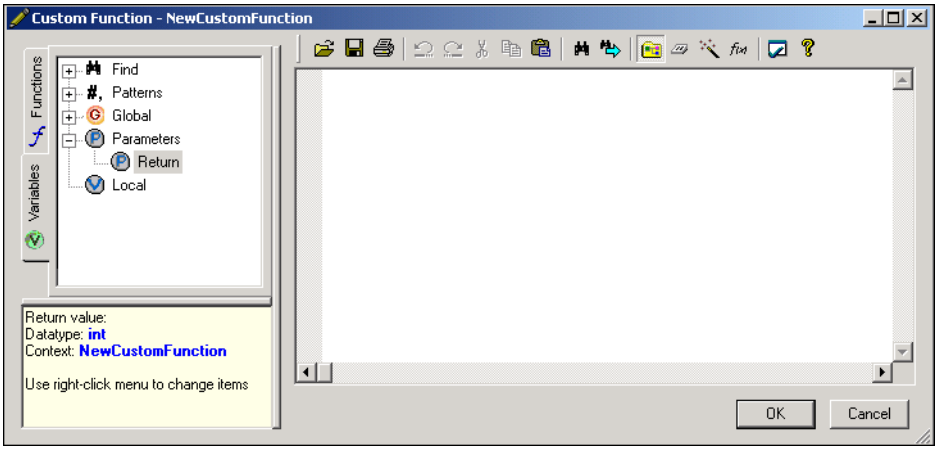
3. Enter the name of the new function.

4. Enter a description for your function.

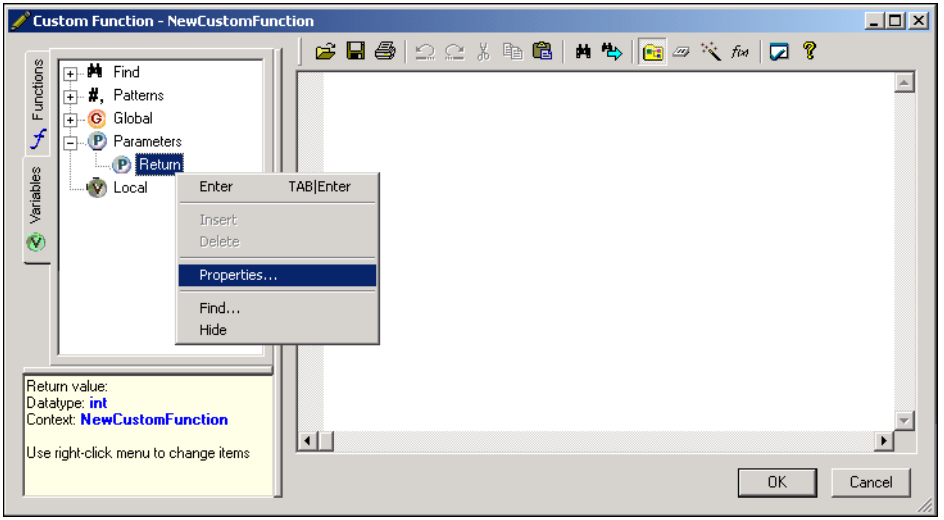


5. Click **Next** to open the smart editor.

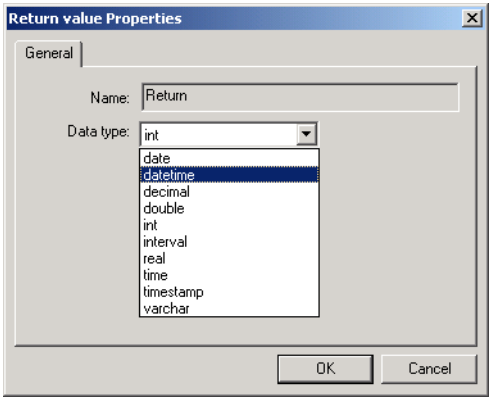
In the smart editor, you can define the return type, parameter list, and any local variables to be used in the function.



6. In the **Variables** tab, right-click **Return** and select **Properties...**



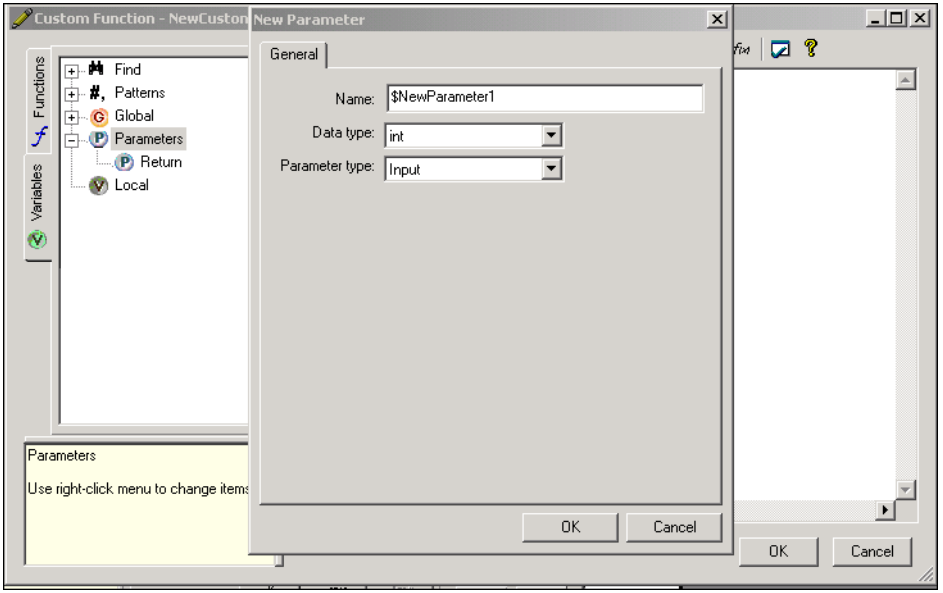
By default, the return data type is set to `int`. To change this, select another return data type from the Data type list. Click **OK**.



7. In the **Variables** tab, right-click **Parameters** and choose **Insert**.
8. Define parameter properties by choosing a **Data type** and a **Parameter type** (Input, Output, or Input/Output).

   NOTE: Data Integrator data flows cannot pass variable parameters of type **output** and **input/output**.

9. Click **OK**.

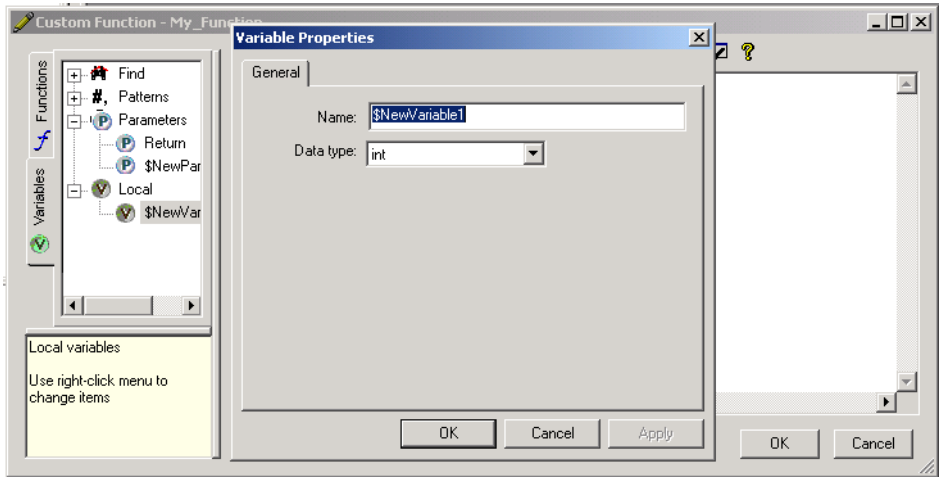


Repeat steps 7 - 9 for each parameter required in your function. After you add one parameter, the right-click menu allows you to choose where to insert each new one. The following right-click menu is available:

Use this menu to create, delete, or edit variables and parameters.

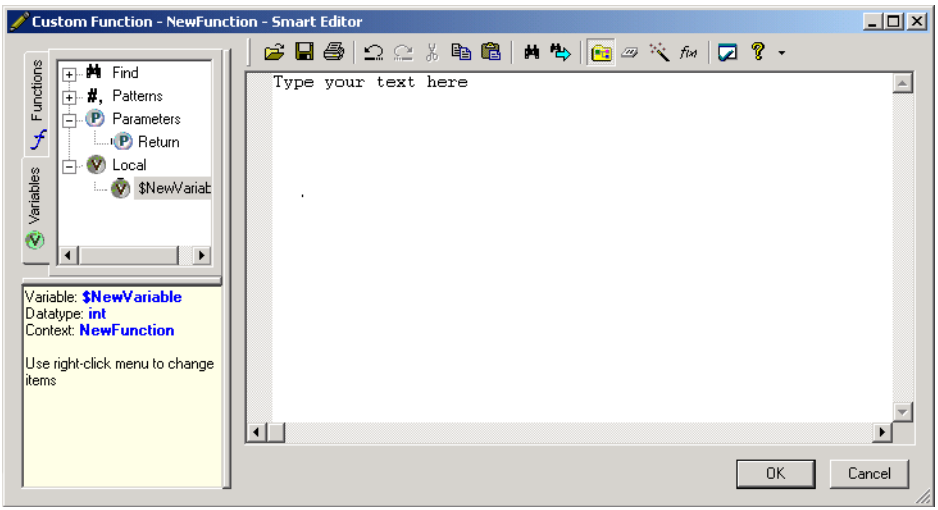
10. To define variables used by the function, but not passed outside the function, right-click **Local** and choose **Insert**.

11. Choose a data type in the Variable Properties window and click **OK**.



Repeat this step for each variable required in your function.

12. Complete the text for your function. For more information about smart editor options, see "Smart Editor" on page 173.



13. Click the **Validate** icon to validate your function.

    If your function contains syntax errors, Data Integrator displays a listing of those errors in an embedded pane below the editor.

14. To see where the error occurs in the text, double-click on an error.

    The smart editor redraws to show the location of the error.

15. When your function is valid, click **OK** to save the function, and the variables and parameters inside.

    Variables and parameters for an existing custom function are local to each function. Therefore, they are not displayed in the Variables and Parameters window (accessible from **Tools** > **Variables**). Variables and parameters for custom functions can be viewed in the smart editor library, under the **Variables** tab, when you edit the custom function.

➢ To edit an existing function

1. Choose **Tools** > **Custom Functions**, select the function and select **Next**.

   The smart editor opens with the function displayed and the variables and parameters that exist for this function shown under the **Variables** tab.

   Alternatively:

   - Go to the **Custom Function** tab in the object library and double-click the function you want to edit.
   - Go to the **Custom Function** tab in the object library, right-click the function, and select **Edit**.

2. See steps 5 through 15 from To create a custom function for editing details.

➢ To replicate a custom function

1. From the object library, right-click a custom function and select **Replicate**.

   The Custom Function editor opens with only the **Function name** field enabled.

2. Enter a new name for the function.

   The name must be valid and different from the original name.

3. Click **Finish**.

   The new custom function appears in the object library.

➢ To delete a custom function

Go to the **Custom Function** tab in the object library, right-click the function, and select **Delete**.

If you delete a function from the list of custom functions, you must remove references to the function from expressions in scripts, conditionals, queries, and other custom functions.

Database and application functions

You can import the metadata for database and application functions and use them in Data Integrator applications. You can also import stored functions and procedures. For more information, see "About procedures" on page 491.

At run time, Data Integrator passes the appropriate information to the database or application from which the function was imported. For information about importing functions, see "Imported stored function and procedure information" on page 91 of the *Data Integrator Designer Guide*.

The metadata for a function includes the input and output parameters and their data types. If there are restrictions on data passed to the function, such as requiring uppercase values or limiting data to a specific range, you must enforce these restrictions in the input. You can either test the data before extraction or include logic in the data flow that calls the function.

Descriptions of built-in functions

This section describes each built-in function available in Data Integrator.

The following table lists the names and descriptions of functions, as well as the function's category in the function wizard and smart editor:

Functions

Function	Category	Description
abs	Math	Returns the absolute value of an input number.
add_months	Date	Adds a given number of months to a date.
avg	Aggregate	Calculates the average of a given set of values.
ceil	Math	Returns the smallest integer value greater than or equal to an input number.
concat_date_time	Date	Returns a datetime from separate date and time inputs.
count	Aggregate	Counts the number of values in a table column.
dataflow_name	Miscellaneous	Returns the data flow name in which this call exists. If the call is not in a data flow, returns NULL.
datastore_field_value	Miscellaneous	Retrieves the value of a specified datastore field.
date_diff	Date	Returns the difference between two dates or times.
date_part	Date	Extracts a component of a given date.
day_in_month	Date	Determines the day in the month on which the given date falls.
day_in_week	Date	Determines the day in the week on which the given date falls.
day_in_year	Date	Determines the day in the year on which the given date falls.
exec	System	Sends a command to the operating system for execution.
file_exists	Miscellaneous	Checks to see if a given file or directory exists.
fiscal_day	Date	Converts a given date into an integer value representing a day in a fiscal year.

Functions (Continued)

Function	Category	Description
floor	Math	Returns the largest integer value less than or equal to an input number.
gen_row_num	Miscellaneous	Returns an integer value beginning with 1 then incremented sequentially by 1 for each additional call. This function can be used to generate a column of row IDs.
get_domain_description	Miscellaneous	Returns the description of a value when given the domain name and the value.
get_env	Environment	Returns a value for the specified environmental variable.
host_name	Miscellaneous	Returns the name of the computer on which the job is executing.
ifthenelse	Miscellaneous	Allows conditional logic in mapping and selection operations.
index	String	Returns the index of a given word in a string.
interval_to_char	Conversion	Returns a string representation of the interval.
is_set_env	Environment	Verifies if the specified environment variable is set.
is_valid_date	Validation	Indicates if an expression can be converted into a valid date value.
is_valid_datetime	Validation	Indicates if an expression can be converted into a valid datetime value.
is_valid_decimal	Validation	Indicates if an expression can be converted into a valid decimal value.
is_valid_double	Validation	Indicates if an expression can be converted into a valid double value.
is_valid_int	Validation	Indicates if an expression can be converted into a valid integer value.
is_valid_real	Validation	Indicates if an expression can be converted into a valid real value.
is_valid_time	Validation	Indicates if an expression can be converted into a valid time value.
isempty	Miscellaneous	Indicates if a nested table contains data.
isweekend	Date	Indicates that a date corresponds to Saturday or Sunday.

Functions (Continued)

Function	Category	Description
job_name	Miscellaneous	Returns the name of the job in which the call to this function exists.
julian	Date	Converts a date to its integer Julian value, the number of days between the start of the Julian calendar and the date.
julian_to_date	Conversion	Converts a Julian value to a date.
key_generation	Database	Generates keys for the specified table, after determining the appropriate starting value.
last_date	Date	Returns the last date of the month for a given date.
length	String	Returns the number of characters in a given string.
ll_error	Miscellaneous	Passes connection errors for LiveLoad databases to the LiveLoad Monitor. See "Control functions" on page 23 of the *Data Integrator LiveLoad User's Guide*.
ll_switch	Miscellaneous	Triggers the switch between live and load databases controlled by LiveLoad. See "Control functions" on page 23 of the *Data Integrator LiveLoad User's Guide*.
lookup	Lookup	Finds a value in one table or file based on values in a second table or file.
lookup_ext	Lookup	Finds data from a database table, flat file, or memory datastore table.
lookup_seq	Lookup	Finds a value in one table based on values in a second table or file, and ensures that the sequence matches.
lower	String	Changes the characters in a string to lowercase.
lpad	String	Pads a string with characters from a specified pattern.
lpad_ext	String	Pads a string with logical characters from a specified pattern.
ltrim	String	Removes specified characters from the start of a string.
ltrim_blanks	String	Removes blank characters from the start of a string.

Functions (Continued)

Function	Category	Description
ltrim_blanks_ext	String	Removes blank and control characters from the start of a string.
mail_to	System	Sends the specified e-mail message.
max	Aggregate	Returns the maximum value from a list.
min	Aggregate	Returns the minimum value from a list.
month	Date	Determines the month in which the given date falls.
num_to_interval	Conversion	Converts a numeric value to an interval.
nvl	Miscellaneous	Replaces NULL values.
print	String	Prints the given string to the trace log.
pushdown_sql	Miscellaneous	Allows you to create dynamic WHERE clauses.
quarter	Date	Determines the quarter in which the given date falls.
raise_exception	Miscellaneous	Calling this function causes an exception to be generated.
raise_exception_ext	Miscellaneous	Same as raise_exception, but takes a second parameter for an exit code.
rand	Math	Returns a random number between 0 and 1.
replace_substr	String	Returns a string where every occurrence of a given search string in the input is substituted by the given replacement string.
repository_name	Miscellaneous	Returns a database connection string and owner name. For example: `beq-local.DBUser.` This is the ID for the repository from which the job is run.
round	Math	Rounds a given number to the specified precision.
rpad	String	Pads a string with characters from a given pattern.
rpad_ext	String	Pads a string with logical characters from a given pattern.
rtrim	String	Removes given characters from the end of a string.
rtrim_blanks	String	Removes blank characters from the end of a string.

Functions (Continued)

Function	Category	Description
rtrim_blanks_ext	String	Removes blank and control characters from the end of a string.
set_env	Environment	Sets an environmental variable temporarily to a specified value.
sleep	Miscellaneous	Suspends the execution of the data flow or work flow from where it is called.
sql	Database	Runs a SQL operation in the specified database.
substr	String	Returns a specific portion of a string starting at a given point in the string.
sum	Aggregate	Calculates the sum of a given set of values.
sysdate	Date	Returns the current date as listed by the Job Server's operating system.
system_user_name	Miscellaneous	Returns the user name used to log into the Job Server's operating system.
systime	Time	Returns the current time as listed by the operating system.
table_attribute	Miscellaneous	Retrieves the value of a specified table attribute.
to_char	Conversion	Converts a date to a string.
to_date	Conversion	Converts a string to a date.
to_decimal	Conversion	Converts a varchar to a decimal.
total_rows	Database	Returns the number of rows in a particular table in a datastore.
trunc	Math	Truncates a given number to the specified precision.
truncate_table	Miscellaneous	Allows you to explicitly expunge data from a memory table.
upper	String	Changes the characters in a string to uppercase.
week_in_month	Date	Determines the week in the month in which the given date falls.
week_in_year	Date	Determines the week in the year in which the given date falls.

Functions (Continued)

Function	Category	Description
WL_GetKeyValue	String	Returns the value of a given keyword in Web log search strings.
word	String	Returns one word out of a string.
word_ext	String	Returns the word identified by its position in a delimited string.
workflow_name	Miscellaneous	Returns the work flow in which this call exists. Returns the name of the inner most work flow in cases where several work flows enclose this function call. If no work flow is found, returns job name.
year	Date	Determines the year in which the given date falls.

abs

Returns the absolute value of a number.

SYNTAX

`abs(`*num*`)`

RETURN VALUE

decimal, double, int, or real
: The absolute value of the given number, *num*. The type of the return value is the same as the type of the original number.

WHERE

num — The source number.

EXAMPLE

Function	Results
`abs(12.12345)`	`12.12345`
`abs(-12.12345)`	`12.12345`

add_months

Adds a given number of months to a date.

SYNTAX

`add_months`(*original_date, months_to_add*)

RETURN VALUE

date

WHERE

original_date

months_to_add Number of months to add to the original date.

DETAILS

The *months_to_add* can be any integer. If *original_date* is the last day of the month or if the resulting month has fewer days than the day component of *original_date*, then the result is the last day of the resulting month. Otherwise, the result has the same day component as *original_date*.

EXAMPLE

Function	Results
`add_month('1990.12.17', 1)`	`'1991.01.17'`
`add_month('2001.10.31',4)`	`'2002.2.28'`

avg

Calculates the average of a given set of values.

SYNTAX

`avg(`*value_list*`)`

RETURN VALUE

decimal, double, int, or real
: The calculated average of *value_list*. The average is calculated to the same precision as the input value.

WHERE

value_list
: The source values for which to calculate an average, such as a table column.

EXAMPLE

To calculate the average of values in the salary column of a table, use the avg function in a query:

- In the **Mapping** tab of the query editor, enter:

  ```
  avg(SALARY)
  ```

- In the **Group By** tab in the query editor, specify the columns for which you want to group the salary, such as the department column. For each unique set of values in the group by list, such as each unique department, Data Integrator calculates the average salary.

ceil

Returns the smallest integer value greater than or equal to a number.

SYNTAX

`ceil(`*num*`)`

RETURN VALUE

decimal, double, int, or real
The indicated integer, cast as the same type as the original number, *num*.

WHERE

num The source number.

EXAMPLE

Function	Results
`ceil(12.12345)`	`13.00000`
`ceil(12)`	`12`
`ceil(-12.223)`	`-12.000`

concat_date_time

Returns a datetime from separate date and time inputs.

SYNTAX

```
concat_date_time(date, time)
```

RETURN VALUE

datetime	The datetime value obtained by combining the inputs.

EXAMPLE

```
concat_date_time(MS40."date",MS40."time")
```

count

Counts the number of values in a table column.

SYNTAX

`count(`*column*`)`

RETURN VALUE

int	The number of rows in the column that have a non-NULL value.

WHERE

column	The column in the input table in which to count values.

EXAMPLE

To determine the number of customers located in a specific sales region, use the count function with a filter defined in the **Where** tab of the query editor. The following WHERE clause selects the rows in the REGION column with the value `TX`:

```
REGION = "TX"
```

With the target column selected, enter the count function in the **Mapping** tab of the editor:

```
count(REGION)
```

dataflow_name

Returns the data flow name in which this call exists. If the call is not in a data flow, returns NULL

SYNTAX

```
dataflow_name()
```

RETURN VALUE

varchar

EXAMPLE

```
print('Data Flow Name: [dataflow_name()]')
```

datastore_field_value

Retrieves the value of a specified datastore field.

SYNTAX

`datastore_field_value`(*ds_name, field_name*)

RETURN VALUE

varchar

WHERE

ds_name	This is the name of the datastore.
field_name	The name of the field.

DETAILS

The *field_name* should match the name seen in the language of the datastore. In the datastore editor click the **Show ATL** button to see the valid field names. If a specified field is not found or the datastore is invalid, NULL is returned. If the *field_name* is 'password' NULL is returned.

EXAMPLE

Function	Results
`datastore_field_value('mssql', 'sql_server_database')`	`'DBUser'`

date_diff

Returns the difference between two dates or times.

SYNTAX

`date_diff`(*date1*, *date2*, *fmt_str*)

RETURN VALUE

int

WHERE

date1, *date2*

fmt_str The string describing the format of the dates. Choose from the following values.

D	Day
H	Hours
M	Minutes
S	Seconds

DETAILS

This function is equivalent to interval_to_char(date1 - date2, 'fmt_str').

EXAMPLE

Function	Results
`date_diff(start_date,sysdate(),'D')`	The number of days between the date in the column start_date and today's date.
`date_diff(start_time,systime(),'M')`	The number of minutes between the time in the column start_time and the current time.

date_part

Extracts a component of a given date.

SYNTAX

`date_part`(*in_date, fmt_str*)

RETURN VALUE

int

WHERE

in_date	The input date.
fmt_str	The string describing the format of the extracted part of the date. Choose from the following values. YY Year MM Month DD Day HH Hours MI Minutes SS Seconds

DETAILS

This function takes in a datetime and extracts the component requested as an integer.

NOTE: *Year* is displayed as four digits, not two.

EXAMPLE

Function	Results
`date_part('1990.12.31', 'YY')`	`1990`
`date_part('1991.01.17 23:44:30', 'SS')`	`30`

day_in_month

Determines the day in the month on which the input date falls.

SYNTAX

`day_in_month(`*date1*`)`

RETURN VALUE

int	The number from 1 to 31 that represents the day in the month that *date1* occurs.

WHERE

date1	The source date.

This function extracts the day component from the date value.

EXAMPLE

Function	Results
`day_in_month(to_date('Jan 22, 1997','mon dd, yyyy'))`	22
`day_in_month(to_date('02/29/1996','mm/dd/yyyy'))`	29
`day_in_month(to_date('1996.12.31','yyyy.mm.dd'))`	31

day_in_week

Determines the day in the week on which the input date falls.

SYNTAX

`day_in_week(`*date1*`)`

RETURN VALUE

int	The number from 1 (Monday) to 7 (Sunday) that represents the day in the week that *date1* occurs.

WHERE

date1	The source date.

This function allows you to categorize dates according to the day of the week the date falls upon. For example, all dates for which this function returns a "3" occur on Wednesday.

EXAMPLE

Function	Results
`day_in_week(to_date('Jan 22, 1997','mon dd, yyyy'))`	`3` (Wednesday)
`day_in_week(to_date('02/29/1996','mm/dd/yyyy'))`	`4` (Thursday)
`day_in_week(to_date('1996.12.31','yyyy.mm.dd'))`	`2` (Tuesday)

day_in_year

Determines the day in the year on which the input date falls.

SYNTAX

```
day_in_year(date1)
```

RETURN VALUE

int	The number from 1 to 366 that represents the day in the year that *date1* occurs.

WHERE

date1	The source date.

EXAMPLE

Function	Results
`day_in_year(to_date('Jan 22, 1997','mon dd, yyyy'))`	`22`
`day_in_year(to_date('02/29/1996','mm/dd/yyyy'))`	`60`
`day_in_year(to_date('1996.12.31','yyyy.mm.dd'))`	`366` (1996 was a leap year.)

exec

Sends a command to the operating system for execution. With this function, you can add a program to a Data Integrator job.

SYNTAX

`exec`(*command_file, parameter_list, flag*)

RETURN VALUE

Varchar(1020)	Returns up to 1020 characters that depend on the value of *flag*.

WHERE

command_file	A string that indicates the location and file name to execute. This string is relative to the Job Server location. It can be an absolute or relative path. The files and directories in the path must be accessible from the Job Server's computer. The *command_file* can be a Windows batch file, a UNIX shell script, or a binary executable. To run other interpreted scripts, the *command_file* must be the name of the command interpreter (e.g., 'perl') and the script must be the first parameter in the *parameter_list*.
parameter_list	A string that lists the values to pass as arguments to the command file. Separate parameters with spaces. When passing no parameters to an executable, enter an empty string (' ').
flag	An integer that specifies what information appears in the return value string, and how to respond upon error—how to respond if *command_file* cannot be executed or exits with a nonzero operating system return code.

exec function flags

Flag	If successful, returns:	On error:	Notes:
0	Standard output	Raises an exception: `System function failure.`	
1	NULL string	Raises an exception: `System function failure.`	Use this flag to track error states in either of the following cases: • The command never produces output • The calling job does not need output
2	Standard output	NULL string	Use this flag if you do not intend to track the status of the command other than the presence or absence of output.
3	NULL string	NULL string	
4	Standard output	DATA INTEGRATOR error message string	See "Details" on page 365.
5	NULL string	DATA INTEGRATOR error message string	See "Details" on page 365.
8	The concatenation of the return code and the combined stdout and stderr (standard error).	Returns the concatenation of the return code and the combined stdout and stderr (standard error).	See "Details" on page 365.

exec function flags (Continued)

Flag	If successful, returns:	On error:	Notes:
256	NULL string	NULL string	Use this flag if you want your program to run independently of DATA INTEGRATOR. Unlike flags 0-8, if you use flag 256, DATA INTEGRATOR will *not* wait until the command (executable program) completes before continuing with job processing. In this case, the command runs independently of DATA INTEGRATOR and stdout, stderr, and return code cannot be returned. Raises an exception (`System function failure`) if the program cannot be launched (e.g., program file not found).

DETAILS

- The program that this function executes must not wait for any user input (e.g., a prompt for a password). For flags 0-8, DATA INTEGRATOR waits for the program to complete, therefore if the program hangs for input DATA INTEGRATOR will hang also. For flag 256, DATA INTEGRATOR will continue if the program hangs for input.
- For flags 4 and 5, the return value format for a *Data Integrator error message string* is:

  ```
  'error-number: error-message-text'
  ```

  where the first field is exactly 7 characters wide, and the second begins at index 10. If the program cannot be executed, the error number is 50307. If the program exits with a non-zero return code, the error number is 50306. The text is from Business Objects' errormessage.txt. For example:

  ```
  '  50306: Function <exec> failed to execute
  program 'foo.exe'. Program terminated with
  exit code 3.'
  ```

- For flag 8, the return value format is:

  ```
  'return-code: stdout-and-stderr'
  ```

  where the first field is exactly 7 characters wide and the second begins at index 10. The return code is produced by the program. Zero indicates success. Consult your program's documentation to determine the meaning of other codes.

  For example:

  - `'      0: 8 file(s) copied.'`
  - `'      1: The system cannot find the file specified.'`
  - `'      1: a.tmp -> /usr/tmp/a.tmp cp: *.lcl: The system cannot find the file specified.'`
  - `'     -2: manmix(): fatal application error.'`

  The 7-character format enables you to easily extract the first field (the return code from the executed command) as a string of digits (which DATA INTEGRATOR automatically converts to an integer wherever necessary), and the second field as a regular string. For example:

  - In a script:

    ```
    $foo = exec('foo.bat', ' ', 8);
    $foo_rc = substr($foo, 1, 7);
    $foo_txt = substr($foo, 10, 1020);
    ```

  - In a data flow, map

    ```
    exec('foo.bat', '', 8)
    ```

    to an output column "foo" in a query. Then in a subsequent query, refer to that column's components in a mapping or WHERE clause. For example:

    ```
    substr(query.foo, 1, 7);
    substr(query.foo, 10, 1020);
    ```

USE OF REMOTE SHELLS

Use a remote shell to run a command elsewhere on the network:

- The *command_file* named in an exec call can be 'rsh' on either NT or UNIX systems ('remsh' on HP-UX systems) to invoke the remote shell facility. This is a means of running a command on a machine elsewhere on the network. For example:
  - `exec('rsh', 'RemoteMachineName CommandToRunRemotely CmdArg1 CmdArg2', 0);`
  - `exec('rsh', 'RemoteBox -lRemoteUser RemoteCommand CmdArg', 3);`

  Invoke the remote shell facility sparingly, as the remote connection setup, remote authentication, and increased message traffic reduce performance.
- For *flag* values 4, 5, and 8, the return code which DATA INTEGRATOR receives is that of the rsh (or remsh) command (i.e., 0 if it successfully gets a remote connection and authorization, nonzero otherwise). There is no relation between this and the return value of the remote command (this is inherent in the remote shell mechanism on all the operating systems). To work around this, wrap the remote command in a .bat file (NT) or shell script (UNIX) which will get the command's return code (%errorlevel% if NT, $? if UNIX), and print it to stdout or stderr. For example:
  - `exec('rsh', 'RemoteMachineName remcmdWrapper.bat CmdArg1 CmdArg2', 8);`
  - `exec('rsh', 'RemoteBox -lRemoteUser /usr/acta/remcmdWrapper CmdArg', 4);`

- The system administrator of the remote machine must set up access for the DATA INTEGRATOR user. The `.rhosts` and/or the `hosts.equiv` file must have an entry allowing this access.
  - If the remote machine is NT, the Remote Shell Service must be running on it.
  - If the remote machine is UNIX, the Remote Shell daemon rshd (remshd on HP-UX) must be running on it.

  Consult your operating system documentation for more information.

EXAMPLES

The examples below can be used with NT or UNIX. If you were using the first two examples for UNIX, substitute 'sh', 'csh', 'ksh', 'bash' or 'tcsh' for 'cmd'. Also, the first two examples call 'cmd' rather than the program directly. You need to use 'cmd' (or its equivalent) if either:

- The "command" is a built-in of the shell (e.g., 'DIR' is not a program in NT)
- Piping (a single '|' in an argument) occurs

In either NT or UNIX, the vertical bar symbol sends the output of one command to another command. Only use a vertical bar inside quotes. In DATA INTEGRATOR, the double vertical bar symbol (||) concatenates strings. Only use a double vertical bar outside quotes.

Also, remember that the '\', '/' symbols are interchangeable when using NT. However, only '/' is accepted as a directory separator on UNIX.

```
exec('cmd', 'dir ' || $filename, 8);

exec('cmd', 'x:/bin/program1.exe |
x:/bin/postprocess.bat', 4);
```

```
exec(SRC.PROGNAME, ARGTBL.ARGS || ' lastArg',
2);

exec('c:\Data
Integrator\bin\clone_and_rename.bat',
TBL.FNAME, 1);

exec('C:\Perl5\bin\perl.exe',
'C:\sandbox\stats.pl 20 50 3000', 0);
```

file_exists

Checks to see if a given file or directory exists.

SYNTAX

`file_exists(`*file_path*`)`

RETURN VALUE

int	Returns 1 if a file or directory is present on the disk (even if 0 bytes long), 0 otherwise.

WHERE

file_path	The file name and path, relative to where the Job Server is running. It can be an absolute or relative path.

EXAMPLES

Invoke sleep for one second when the file temp.msg exists in the directory called "c:".

```
while (file_exists('c:/temp.msg') = 1)
begin
  sleep(1000);
end
```

Set a variable to a file name and use the function to check whether the file exists:

```
$unix_file = '/tmp/t.cpp';
if (file_exists($unix_file)) $type = 'unix';
```

Set a variable based on the value of the function:

```
$i = file_exists('c:/autoexec.bat')
```

fiscal_day

Converts a given date into an integer value representing a day in a fiscal year.

SYNTAX

`fiscal_day`(*start_year_date, in_date*)

RETURN VALUE

int

WHERE

start_year_date	The first month and day of a fiscal year. Use this format: 'mm.dd'.
in_date	The date you want to convert. Use any valid datetime.

EXAMPLE

Function	Results
`fiscal_day('03.01', '1999.04.20')`	50

floor

Returns the largest integer value equal to or less than a number.

SYNTAX

`floor(`*num*`)`

RETURN VALUE

decimal, double, int, or real
: The indicated integer, cast as the same type as the original number, *num*.

WHERE

num
: The source number.

EXAMPLE

Function	Results
`floor(12.12345)`	`12.00000`
`floor(12)`	`12`
`floor(-12.223)`	`-13.000`

gen_row_num

Returns an integer value beginning with 1 then incremented sequentially by 1 for each additional call. This function can be used to generate a column of row IDs.

SYNTAX

```
gen_row_num()
```

RETURN VALUE

int

DETAILS

Each occurrence of the function in a data flow is a unique instance, resulting in a unique sequence. Two call instances return values independent of each other. The first time an instance of this function is called, the function returns a value of 1. Subsequent calls of the same instance return the previous value incremented by 1 (i.e., 2, 3, 4…).

Each time a data flow is called, all instances are reinitialized, starting at 1.

EXAMPLES

Use the function in a query's mapping expression to add a column of row IDs to a target.

```
gen_row_num()
```

get_domain_description

Returns the description of a value when given the domain name and the value.

SYNTAX

`get_domain_description(`*domain_name*`,`*domain_value*`)`

RETURN VALUE

varchar	The description is returned as a quoted string. If the value is not in the domain, then a NULL is returned.

WHERE

domain_name	Fully qualified domain name, including the database owner if required. For example: *datastorename*.*owner*.*domain* *datastorename*..*domain*
domain_value	The constant value for which you want to return a description.

EXAMPLE

Function	Results
`get_domain_description('psoft..ACTION','ADL')`	`"Additional"`

get_env

Returns a value for the specified system environment variable.

SYNTAX

`get_env('variable_name')`

RETURN VALUE

varchar(255)	Returns the value of the environment variable. Returns NULL if the environment variable is not set. You can use the is_set_env function to determine whether a variable is set. See "is_set_env" on page 382.

WHERE

variable_name	The name of the environment variable. The name must be surrounded by single quotes.

EXAMPLE

Function	Results
`getenv('TMP')`	`C:\Temp`

host_name

Returns the name of the computer on which the job is executing.

SYNTAX

```
host_name()
```

RETURN VALUE

varchar

EXAMPLE

```
print('Host Name: [host_name()]');
```

ifthenelse

Allows conditional logic in expressions.

SYNTAX

`ifthenelse(`*condition*`, `*true_branch*`, `*false_branch*`)`

RETURN VALUE

true_branch or *false_branch*
Returns one of the values provided, based on the result of *condition*. The data type of the return value is the data type of the expression in *true_branch*. If the data type of *false_branch* is not convertible to the data type of *true_branch*, DATA INTEGRATOR produces an error at validation. If the data types are convertible but don't match, a warning appears at validation.

WHERE

condition	An expression that evaluates to TRUE or FALSE.
true_branch	An expression that the function returns if *condition* evaluates to TRUE.
false_branch	An expression that the function returns if *condition* evaluates to FALSE.

To improve performance, DATA INTEGRATOR pushes this function to the database. Thus, the database evaluates the IFTHENELSE logic rather than the DATA INTEGRATOR engine.

Use this function to apply conditional logic when mapping columns or selecting columns in a query. For more flexible control over conditions in a script, use the IF keyword in the DATA INTEGRATOR scripting language.

EXAMPLE

Function	Results
`ifthenelse (ZIP < 94000, 'SOUTH', 'NORTH')`	If the value in the column `ZIP` is less than 94000, the value returned is `SOUTH`. If `ZIP` is greater than 94000, then the value returned is `NORTH`.

index

Returns the index of a given character sequence in a string.

SYNTAX

`index(`*input_string*`,` *index_string*`,` *start*`)`

RETURN VALUE

int	Specifies the first location of the indicated character sequence.

WHERE

input_string	The source string.
index_string	The character sequence sought in *input_string*.
start	The position where the function starts searching in *input_string* for the character sequence contained in *index_string*. *Start* should be a positive number between 1 and the length of *input_string*.

DETAILS

The function searches for the *index_string* beginning at the *start* position in the input_string. If *start* is 0, it is reset to 1; if *start* is greater than the number of characters in *input_string*, the function returns NULL.

If *index_string* is not found in *input_string*, the function returns NULL. The characters in *index_string* must match exactly the sequence of characters in *input_string*. The search is case-sensitive.

EXAMPLE

Function	Results
`index('Accounting Department', 'DEPARTMENT', 1)`	`NULL`
`index('Accounting Department', 'Department', 1)`	`12`

interval_to_char

Converts an interval value to a string.

SYNTAX

`interval_to_char(`*input_interval, interval_type*`)`

RETURN VALUE

varchar — The converted string.

WHERE

input_interval — The value of type interval to be converted.

interval_type — A string describing the format of the interval. Choose from the following values:

- D Day
- H Hours
- M Minutes
- S Seconds

EXAMPLE

Function	Results
`interval_to_char(start_date - sysdate(), 'd')`	The number of days between the date in the column `start_date` and today's date.
`interval_to_char(start_time - systime(), 'm')`	The number of minutes between the time in the column `start_time` and the current time.

is_set_env

Verifies if the specified system environment variable is set.

SYNTAX

`is_set_env(`*variable_name*`)`

RETURN VALUE

int	Returns 1 if the environment variable is set, otherwise, returns 0.

WHERE

variable_name	The name of the environment variable. The name must be surrounded by single quotes.

EXAMPLE

Function	Results
`is_set_env('MODE')`	Returns 1 if the MODE variable has already been set; returns 0 if the MODE variable has not been set.

is_valid_date

Indicates if an expression can be converted into a valid calendar date value. For example the following will return a negative result:

```
is_valid_date ('01/34/2002', 'mm/dd/yyyy')
```

This expression returns 0 because there is no such date as January 34th."

SYNTAX

`is_valid_date(`*input_expression, date_format*`)`

RETURN VALUE

int	Returns 1 if the string can be converted into a date; returns 0 if not.

WHERE

input_expression	The expression to be validated. If the expression does not resolve to a value of data type varchar, you will see a warning that the value has been converted to a varchar.
date_format	The string identifying the date format of the input string. Construct the date format using the following codes and other literal strings or punctuation: DD 2-digit day of the month

MM	2-digit month
MONTH	Full name of month
MON	3-character name of month
YY	2-digit year
YYYY	4-digit year

For more information about date formats, see "date" on page 187.

EXAMPLE

Function	Results
`is_valid_date (Orders.SubmitDate,'mm/dd/yyyy')`	Tests whether the string `Orders.SubmitDate` can be converted to a calendar date with the `mm/dd/yyyy` date format.

is_valid_datetime

Indicates if an expression can be converted into valid calendar date and time values. For example the following will return a negative result:

```
is_valid_datetime ('01/14/2002 26:56:09',
'mm/dd/yyyy hh24:mi:ss')
```

This expression returns 0 because there is no such hour as "26", even on the 24 hour clock.

SYNTAX

`is_valid_datetime(`*input_expression, datetime_format*`)`

RETURN VALUE

int	Returns 1 if the expression can be converted into a valid datetime; returns 0 if not.

WHERE

input_expression	The expression to be validated.
datetime_format	The string identifying the datetime format of the input expression. Construct the datetime format using the following codes and other literal strings or punctuation:

DD	2-digit day of the month
MM	2-digit month
MONTH	Full name of month
MON	3-character name of month

YY	2-digit year
YYYY	4-digit year
HH24	2-digit hour of the day (0-23)
MI	2-digit minute (0-59)
SS	2-digit second (0-59)

For more information about datetime formats, see "datetime" on page 189.

EXAMPLE

Function	Results
`is_valid_datetime (Orders.Received,'mm/dd/yyyy hh24:mi:ss')`	Tests whether the string `Orders.Received` can be converted to the `mm/dd/yyyy hh24:mi:ss` datetime format.

is_valid_decimal

Indicates if an expression can be converted into a valid decimal value.

SYNTAX

`is_valid_decimal(`*input_expression, decimal_format*`)`

RETURN VALUE

int	Returns 1 if the string can be converted into a valid decimal; returns 0 if not.

WHERE

input_expression	The expression to be validated.
decimal_format	A string indicating the decimal format of the input expression. Use pound characters (#) to indicate digits and a decimal indicator. If necessary, include commas as thousands indicators. For example, to specify a decimal format for numbers smaller than 1 million with 2 decimal digits, use the following string: `'#,###,###.##'`

EXAMPLE

Function	Results
`is_valid_decimal (Orders.Price,'###,###.##')`	Tests whether the string `Orders.Price` can be converted to decimal format.

is_valid_double

Indicates if an expression can be converted into a valid double value.

SYNTAX

`is_valid_double(`*input_expression, double_format*`)`

RETURN VALUE

int	Returns 1 if the string can be converted into a valid double; returns 0 if not.

WHERE

input_expression	The expression to be validated.
double_format	A string indicating the double format of the input expression. Use pound characters (#) to indicate digits and a decimal indicator. If necessary, include commas as thousands indicators. For example, to specify a double format for numbers smaller than 1 million with 2 decimal digits, use the following string: `'#,###,###.##'`

EXAMPLE

Function	Results
`is_valid_double (Product.Weight,'###.###')`	Tests whether the string `Product.Weight` can be converted to double format.

is_valid_int

Indicates if an expression can be converted into a valid integer value.

SYNTAX

`is_valid_int(`*input_expression, int_format*`)`

RETURN VALUE

int	Returns 1 if the string can be converted into a valid integer; returns 0 if not.

WHERE

input_expression	The expression to be validated.
int_format	The format specifying the thousands separator of the input expression. For example, to specify an integer format, use the following string: `'#.###.###'`

EXAMPLE

Function	Results
`is_valid_int (QuarterResults.Volume,'###.###')`	Tests whether the string `QuarterResults.Volume` can be converted to the `###.###` integer format.

is_valid_real

Indicates if an expression can be converted into a valid real value.

SYNTAX

`is_valid_real(`*input_expression*`, `*real_format*`)`

RETURN VALUE

int	Returns 1 if the string can be converted into a valid real; returns 0 if not.

WHERE

input_expression	The expression to be validated.
real_format	A string indicating the real format of the input expression. Use pound characters (#) to indicate digits and a decimal indicator. For example, to specify a real format for numbers smaller than 1 million with 2 decimal digits, use the following string: `'#,###,###.##'`

EXAMPLE

Function	Results
`is_valid_real (QuarterResults.Mean,'#,###.#####')`	Tests whether the string `QuarterResults.Mean` can be converted to real format.

is_valid_time

Indicates if an expression can be converted into a valid time value.

SYNTAX

`is_valid_time(`*input_expression, time_format*`)`

RETURN VALUE

int	Returns 1 if the string can be converted into a valid time; returns 0 if not.

WHERE

input_expression	The expression to be validated.
time_format	The string identifying the time format of the input expression. Construct the time format using the following codes and other literal strings or punctuation:

HH24	2-digit hour of the day (0-23)
MI	2-digit minute (0-59)
SS	2-digit second (0-59)

EXAMPLE

Function	Results
`is_valid_time (Orders.ReceivedTime,'hh24:mi:ss')`	Tests whether the string `Orders.ReceivedTime` can be converted to the `hh24:mi:ss` datetime format.

isempty

Indicates if a nested table contains data.

SYNTAX

`isempty`(*table_name*)

RETURN VALUE

int	The result of the content test: returns 1 if the table does not contain data; returns 0 if the table does contain data.

WHERE

table_name	The fully qualified name of the nested table to test. A fully qualified name contains the parent table names up to the top level of the table in the current context. If you only specify a table name, DATA INTEGRATOR looks for the table among the tables available through the FROM clause of the current context. If you specify a partially qualified table name (only part of the table hierarchy), DATA INTEGRATOR looks for the table among the tables available in the FROM clause of the context indicated by the partial qualification.

When performing operations on hierarchical data, the isempty function allows you to exclude rows in a higher-level table based on whether a lower-level table contains data.

DATA INTEGRATOR determines that a nested table is empty when the table contains no rows. If the nested table contains even one row with null values in all columns, the isempty function indicates that the table has content. If the nested table is empty except for another nested table, and the second nested table does contain data, then the first nested table is not empty.

EXAMPLE

You can use the isempty function to determine if there are line items associated with a sales order. For example, if the sales order is the input data set to a Query transform and you want the query to exclude orders without line items, include the following expression in the WHERE clause of the top-level context of the query:

```
isempty (order_table.line_items_table)
```

isweekend

Indicates if a date corresponds to Saturday or Sunday.

SYNTAX

`isweekend(`*date1*`)`

RETURN VALUE

int	The result of the date test: returns 1 if the date is a Saturday or Sunday; returns 0 if not.

WHERE

date1	The value of type date or datetime to be tested.

EXAMPLE

Function	Results
`isweekend(hire_date)`	Tests whether the date in `hire_date` is a Saturday or Sunday.
`isweekend(SYSDATE)`	Tests whether today is a Saturday or Sunday.

job_name

Returns the name of the job in which the call to this function exists.

SYNTAX

```
job_name()
```

RETURN VALUE

varchar

EXAMPLE

```
print('Starting execution of Job: [job_name()]
as user: [system_user_name()]');
```

julian

Converts a date to its integer Julian value, the number of days between the start of the Julian calendar and the date.

SYNTAX

`julian(`*date1*`)`

RETURN VALUE

int	The Julian representation of the date.

WHERE

date1	The source value of type date or datetime.

EXAMPLE

Function	Results
`julian(to_date('Apr 19, 1997', 'mon dd, yyyy'))`	`729436`

julian_to_date

Converts a Julian value to a date.

SYNTAX

`julian_to_date(`*input_julian*`)`

RETURN VALUE

date	The date that corresponds to the Julian value inputted.

WHERE

input_julian	An integer representing the Julian value to be converted.

EXAMPLE

Function	Results
`julian_to_date(Julian_Date)`	Converts the number indicated by `Julian_Date` to its date value.

key_generation

Generates the next value in a series, after determining the last value in the series.

The key_generation function determines the maximum existing key value in a given column in the table in the database manager and uses that value as a starting point to generate key values for the target schema.

SYNTAX

`key_generation (`*table*`, `*key_column*`, `*key_increment*`)`

RETURN VALUE

int	The column value found to meet the function requirements.

WHERE

table	The full path name of the file or full database specification of the table in which the *key_column* is located. Enclose this value in single quotation marks.
key_column	A column with existing keys from which this function determines the largest existing key value. Enclose this value in single quotation marks.
key_increment	The integer increment used between key values this function generates.

EXAMPLE

Function	Results
`key_generation('target_ds.dbo.sales','order_number',1)`	Looks for the last key value in the `order_number` column in the sales database and returns the largest value plus one.

last_date

Returns the last date of the month for a given date.

SYNTAX

`last_date`(*in_date*)

RETURN VALUE

date

WHERE

in_date	The date for which the last date of the month is to be calculated.

EXAMPLE

Function	Returns
`last_date('1990.10.01')`	`'1990.10.31'`

length

Returns the number of characters in a given string.

SYNTAX

`length(`*value*`)`

RETURN VALUE

integer	The number of characters in *value*.

WHERE

value	A string indicating the column name, variable, or other element whose length is calculated.

EXAMPLE

In the **Mapping** box of a query, you can use the length function to return the number of characters in each row of a column. For example, with the OUTPUT field selected in the target schema of a query, entering the following statement in the **Mapping** box:

```
length(dal_emp.ename)
```

produces the following results:

Source column (dal_emp.ename)	Target column (output)
jones	5
nguyen	6
tanaka	6

ll_error

Sets ERROR state for a loaded data cache (LOAD) defined as *datastore*. All requests for the loaded data cache are suspended until the state is manually changed for the datastore. (Manually change state back to LOAD from the Data Integrator's monitor tab.) See "Control functions" on page 23 of the *Data Integrator LiveLoad User's Guide* for more information. Also see "LiveLoad operation cycle" on page 3 of the *Data Integrator LiveLoad User's Guide*.

SYNTAX

`ll_error(`*datastore*`)`

RETURN VALUE

integer	Returns 1 if no problems occur. Returns 0 to indicate switch failure.

WHERE

datastore	A string containing the name of the LiveLoad-enabled datastore.

ll_switch

Sets switch state for a LiveLoad-enabled datastore. The LIVE data cache becomes available for RECONCILIATION and the LOAD data cache becomes LIVE. See "Control functions" on page 23 of the *Data Integrator LiveLoad User's Guide* for more information. Also see "LiveLoad operation cycle" on page 3 of the *Data Integrator LiveLoad User's Guide*.

SYNTAX

`ll_switch(`*datastore, post_switch_command*`)`

RETURN VALUE

integer	Returns 1 if no problems occur. Returns 0 to indicate switch failure.

WHERE

datastore	A string containing the name of the LiveLoad-enabled datastore.
post_switch_command	A string that indicates the location and batch file name to execute immediately after switching occurs. This string is relative to the Job Server location. It can be an absolute or relative path. The files and directories in the path must be accessible from the Job Server computer. The *post_switch_command* can be a Windows batch file, a UNIX shell script, or a binary executable.

lookup

Retrieves a value in a table or file based on the values in a different source table or file.

SYNTAX

```
lookup (translate_table, result_column, default_value,
cache_spec, compare_column, expression)
```

RETURN VALUE

any type	The value in the *translate_table* that meets the lookup requirements. The return type is the same as *result_column*.

WHERE

translate_table	The table or file that contains the result or value you are looking up (*result_column*). The *compare_column* is also located in this table. Use a fully qualified table name, including the datastore, owner, and table name. For example: `oracle_ds.TIGER.sales` **NOTE**: You might need to put the owner in quotes, particularly if you use lower case letters.
result_column	The column containing the values you want to retrieve. This column is in the *translate_table*.
default_value	The value returned when there is no matching row in the *translate_table*.

cache_spec The caching method the lookup operation uses. List within single quotes.

There are three possible settings:

- NO_CACHE

  Reads values from the *translate_table* for every row without caching values.
- PRE_LOAD_CACHE

  Loads the *result_column* and *compare_column* into memory after applying filters but before executing the function.

  Select this option if the number of rows in the table is small or you expect to access a high percentage of the table values.
- DEMAND_LOAD_CACHE

  Loads *result_column* and *compare_column* values into memory as the function identifies them.

  Select this option if the number of rows in the table is large and you expect to access a low percentage of the table values frequently.

  Select this option when you use the table in multiple lookups and the compare conditions are highly selective, resulting in a small subset of data.

compare_column The column in the *translate_table* that the function uses to find a matching row.

expression The value that the function searches for in the *compare_column*. This can be a simple column reference, such as a column found in both a source and the *translate_table*. This can also be a complex expression given in terms of constants and input column references.

When *expression* refers to a unique source column, you do not need to include a table name qualifier. If *expression* is from another table or is not unique among the source columns, you need a table name qualifier.

NOTE: You can specify more than one *compare_column* and *expression* pair—simply add additional pairs at the end of the function statement. The values must match for all specified pairs in order for the lookup function to find a matching row.

The lookup function uses a value you provide (*expression*) to find a corresponding value in a file or different table. Specifically, the function searches for the row in the *translate_table* where the value in the *compare_column* matches the value in *expression*. The function returns the *result_column* value from this matching row.

For example, if your source schema uses a customer ID to identify each row, but you want the customer name in your target schema, you can use the lookup function to return the customer name given the customer ID.

In SQL terms, the lookup function evaluates *expression* for each row, then executes the following command:

```
SELECT result_column
   FROM translate_table
   WHERE compare_column = expression
```

The value returned by this SELECT statement is the result of the lookup function for the row.

You can specify multiple *compare_column* and *expression* pairs to uniquely identify the *result_column* value. However, the function wizard only provides fields for one pair; add extra *compare_column* and *expression* pairs to the output that the wizard generates.

When there are no matching rows in the *translate_table*, the lookup function returns the *default_value*. When multiple matching rows exist in the *translate_table*, the row returned is based on whether the translate table is a standard RDBMS table, an SAP R/3 table, or a flat file:

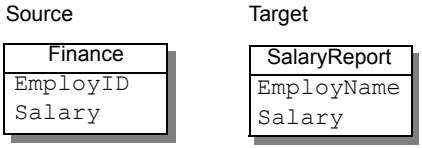
- For standard RDBMS tables, the lookup function will find the matching row with the maximum value in the *result_column* and return that value.
- For SAP R/3 tables or flat files, the lookup function randomly selects a matching row and returns the value in the *result_column* for that row.

NOTE: To avoid random row selection when the *translate_table* is an SAP R/3 table or a flat file, Acta recommends you use the lookup_seq() function.

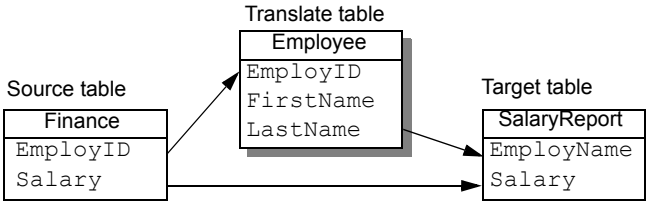
To enhance performance, you can configure the lookup function to hold the values from the *translate_table* in memory. To do so, use the *cache_spec* setting. The optimal setting depends on the number of rows the function must read, the number of rows in the table, and the available memory.

EXAMPLES

You can use the lookup function to return a text value given a numerical identifier. For example, suppose you have a source table containing a numerical identifier, such as an employee number, and you want to use the employee's name in your target.

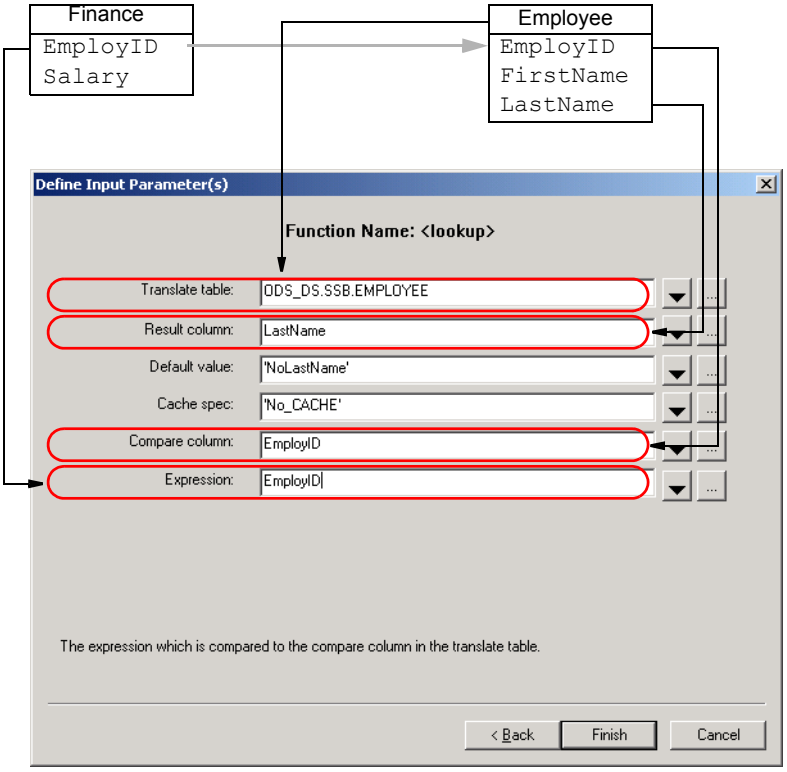


You can use the lookup function to return the employee name based on the employee number. The lookup function uses a third table that translates the values from the source table to the desired values in the target table.



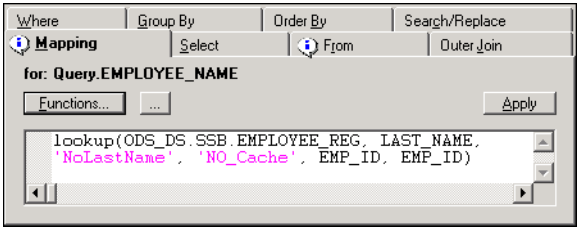
To produce the desired target column, select the column that you want to translate in the target schema. Click the **Functions** button, located over the **Mapping** text box. The function wizard opens. Under **Function categories**, select **Miscellaneous_Function**, then under **Function name**, select **lookup**. Click **Next**. The Define Input Parameters window opens.

Enter the function parameters as follows:



The expression value refers to a column in the source file or table and therefore does not require qualification with a table name. If this expression was from another table or was not unique among the source columns, it would require a table name qualifier.

The function wizard automatically produces the mapping text.



You can create a lookup function with two *expression* and *compare_column* pairs:

```
lookup(sap_ds..VBUP, GBSTA, 'none',
'NO_CACHE', VBELN, VBAK.VBELN, POSNR,
VBAP.POSNR)
```

This function returns the value from the GBSTA column in the VBUP table that corresponds to the VBELN value in the VBAK table and the POSNR value in the VBAP table. When no corresponding value is found, the function returns "none."

lookup_ext

The most powerful of the lookup functions, lookup_ext is associated with its own graphic editor in the function wizard. Not only can this function retrieve a value in a table or file based on the values in a different source table or file, but it also provides extended functionality allowing you to:

- Return multiple columns from a single lookup
- Choose from more operators to specify a lookup condition
- Specify a return policy for your lookup
- Perform multiple (including recursive) lookups
- Call lookup_ext in scripts and custom functions (which also lets you reuse the lookup(s) packaged inside scripts)
- Define custom SQL, using the *SQL_override* parameter, to populate the lookup cache, narrowing large quantities of data to only the sections relevant for your lookup(s)
- Use lookup_ext to dynamically execute SQL
- Call lookup_ext, using the function wizard, in the query output mapping to return multiple columns in a Query transform
- Design jobs to use `lookup_ext` without having to hard-code the name of the translation file at design time.
- Use lookup_ext with memory datastore tables

NOTE: You cannot call this function from an R3 data flow.

SYNTAX

lookup_ext *([translate_table, cache_spec, return_policy], [return_column_list], [default_value_list], [condition_list], [orderby_column_list], [output_variable_list], [sql_override])*

RETURN VALUE

any type	The return type is the first lookup column in *return_column_list*.

WHERE

translate_table	The table, file, or memory datastore that contains the result(s) or value(s) you are looking up (*result_column_list*). If the *translate_table* is a database table, use the *datastore.owner.table* format. For example:
	
	`ERP_ds.OWNER.EMPLOYEES`
	
	If the *translate_table* is a flat file, use the *file_ds.filename* format. For example:
	
	`delim."c:/temp/employees"`
	
	NOTE: To substitute a variable for a file name, replace the data inside the double quotes. For example: `delim."$employees"` The variable used to store a file name can be a local or global variable, or a parameter passed to a work flow or a data flow. If the caching mode is `NO_CACHE`, Data Integrator can pass in a different file name each time `lookup_ext` is called. For example, you can call `lookup_ext` in a while loop and assign a different file name to the variable passed as lookup file. If the caching mode is `PRE_LOAD_CACHE` or `DEMAND_LOAD_CACHE`, only the first file name passed is used. All file names passed during subsequent calls are ignored.
	
	If the *translate_table* is a memory datastore table, use the *memory_ds.table* format. For example:
	
	`mem_ds..employees`

cache_spec

The caching method the lookup_ext operation uses.

There are three possible cache specifications:

- 'NO_CACHE'

  Reads values from the *translate_table* for every row without caching values.
- 'PRE_LOAD_CACHE'

  Loads the *return_column_list*, *compare_column* (see *condition_list*), and *orderby_column_list* into memory after applying constant filters and before executing the function.

  Select this option if the number of rows in the table is small or you expect to access a high percentage of the table values.
- 'DEMAND_LOAD_CACHE'

  Loads *return_column_list*, *compare_column* (see *condition_list*), and *orderby_column_list* into memory as the function identifies them.

  Select this option if the number of rows in the table is large and you expect to frequently access a low percentage of table values.

  Select this option when you use the table in multiple lookups and the compare conditions are highly selective, resulting in a small subset of data.

return_policy

This optional parameter specifies whether the return columns should be obtained from the smallest or the largest row based on values in the orderby columns. The value for this parameter

can be MIN or MAX, and when left blank defaults to MAX. Use *return_policy* when you expect duplicate rows and want output data from the one of the selected rows.

return_column_list

A comma-separated list containing the names of output columns in the *translate_table*.

default_value_list

A comma-separated list containing the default expressions for the output columns. When no rows match the lookup condition, the default values are returned for the output column.

Each default expression type must be compatible with the corresponding output column type such that if the types are not exactly the same, automatic conversion is still possible.

If *default_value_list* is empty or has fewer expressions than the number of output columns, NULL will be used as default. You cannot have more default expressions than the number of output columns.

condition_list A list of triplets that specify lookup conditions. Each set in a triplet contains *compare_column*, *compare_operator* (<,<=,>, >=, =, !=, IS, IS NOT), and *compare_expression*.

The *compare_column* is from the *translate_table*. It is compared against *compare_expression* to compute the output row.

The *compare_expression* is written in terms of constants, variables, and columns in the calling data flow or scripts. While it cannot contain column reference from the *translate_table*, it can

be a simple constant, variable, or column reference, or a complex expression involving arithmetic operations and function calls.

Use compare operators IS and IS NOT to examine *compare_column* against the NULL constant. When you use IS or IS NOT as the compare operator, *compare_expression* must contain the NULL constant. When you compare other operators to a *compare_expression* containing a NULL, the lookup condition return value will always return FALSE.

If you create more than one set of triplets, all triplets are implicitly AND'd together to computer the final lookup condition.

EXAMPLE: [c1, '=',10,c2,'<',query.a,c3,'>=',lower(query.name)]

orderby_column_list

A comma-separated list of column names from the *translate_table*. Working together with *return_policy*, the *orderby_column_list* is used to determine which duplicate row to return the output when more than one row satisfies the lookup condition. When duplicate rows occurs, the list of duplicates is sorted based on columns from the *orderby_column_list* and choosing a row to return using the MIN/MAX *return_policy*.

The *orderby_column_list* is optional. If left empty, the orderby columns match the output columns.

EXAMPLES:
[c1,c3,c4] — Sort the duplicate rows using column values in c1, c3, c4.
[ ] — Empty list is a placeholder for specifying subsequent parameters.

output_variable_list

A comma-separated list of output variables. When more than one output column is specified in the function call, the output variables are used to receive output returns. Variables and output columns are matched by position.

This parameter is optional unless more than one output column appears in the *return_column_list*. When this occurs, output variables must be equal in number to output columns.

To enable conversion, the variable data type must be compatible with the corresponding output column. You do not need to specify output variables if the function is called using the function wizard to map output columns in the query window.

EXAMPLE: [$a,$b,$c]

sql_override

This parameter, available as a button called **Custom SQL** in the function wizard, must contain a valid, single-quoted SQL SELECT statement or a $variable of type varchar to populate the lookup cache when the cache specification is PRE_LOAD_CACHE. This parameter replaces the SQL SELECT statement generated internally by the function for populating the cache. The SELECT statement must select at least those columns referenced in *return_column_list*, *condition_list*, and *orderby_column_list*.

Any valid SQL SELECT statement is permitted and may contain references to other tables besides the *translate_table* to specify inner and outer joins. This parameter can only be specified

when the *translate_table* is a database table.

If this parameter is specified when the cache specification is NO_CACHE, the sql_override query is executed each time the function is called.

If this parameter is specified when the cache specification is PRE_LOAD_CACHE, only the first sql_override query is executed to populate the lookup cache. All subsequent SQL statements are ignored after the lookup cache is built.

If this parameter is specified when the cache specification is DEMAND_LOAD_CACHE, the caching mode will be converted to PRE_LOAD_CACHE and behave as if the PRE_LOAD_CACHE mode were specified.

EXAMPLE:
['select out1, out2,compare1,compare2,orderby1,orderby2 from lookuptbl,othertbl where c1=10 and lookuptbl.c2=othertbl.c2']

Also, when you use *sql_override* in NO_CACHE mode, lookup_ext will dynamically execute the SQL if you pass in a dynamic SQL statement in the form of a variable.

EXAMPLE: LOOKUP REPLACEMENT

In this example, retrieve the name of an employee whose `empno` is equal to `1`.

The following statements show how you can do this with both lookup and lookup_ext:

```
Lookup(ds.owner.emp, empname, 'no body', 'NO_CACHE', empno, 1);
Lookup_ext[(ds.owner.emp, 'NO_CACHE','MAX'], [empname], ['no body']. [empno, '=', 1]
```

➢ To simulate the lookup function using lookup_ext

1. Specify a *translate_table*, *cache_spec*, and one default value in *default_value_list*.
2. Specify one lookup column in the *return_column* list.
3. Specify your lookup condition in the condition_list, using only the equality comparison operator (=).
4. Do not complete other lookup_ext parameters as they do not exist in the lookup function.

If multiple rows are selected by the lookup condition, the row with the maximum lookup column is returned. This is the same behavior as the lookup function and is also the default *return_policy* for lookup_ext.

EXAMPLE: LOOKUP_SEQ REPLACEMENT

In this example, retrieve the name of an employee who works for department 20 and is the eldest employee, less or equal to 30 years old.

The following statements show how you can do this with both lookup_seq and lookup_ext:

```
Lookup_seq(ds.owner.emp, empname, 'no body', age, 30, deptno, 20);
Lookup_ext[(ds.owner.emp, 'PRE_LOAD_CACHE', 'MAX'], [empname], ['no body'],
[deptno, '=', 20, age, '<=', 30], [age]
```

NOTE: The return policy, MAX is optional since MAX is already the default behavior.

➢ To simulate the lookup_seq function using lookup_ext

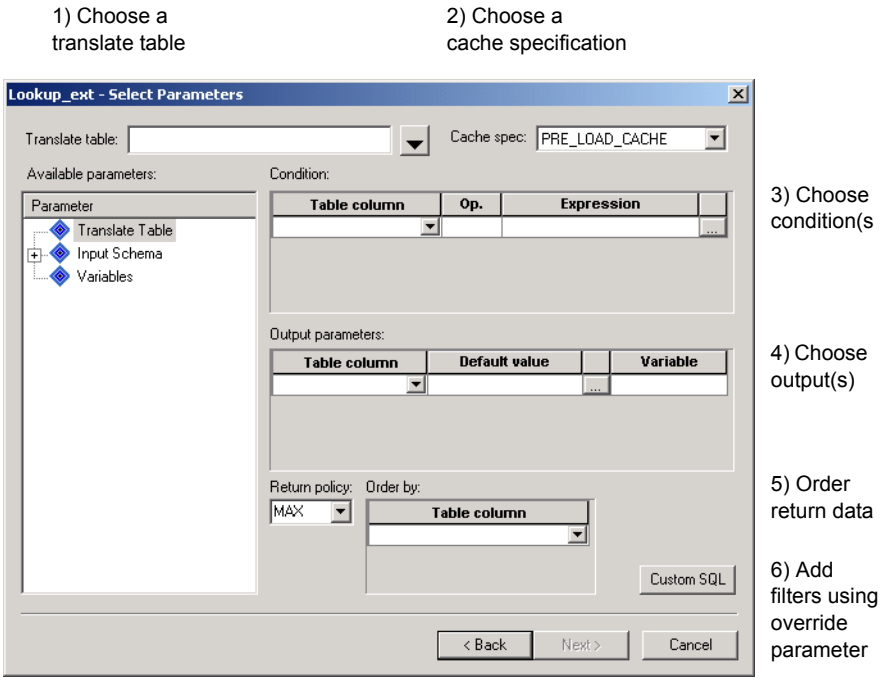
1. Specify a *translate_table*, *cache_spec*, and one default value in *default_value_list*.
2. Specify one lookup column in the *return_column* list.

3. Specify your lookup condition in the *condition_list*, using only the equality comparison operator "=".
4. Specify the *sequence_column* in the *orderby_column_list*.
5. Specify the *sequence_column* and the *sequence_expression* in the *condition_list* using the "<=" operator.
6. Do not complete other lookup_ext parameters as they do not exist in the lookup_seq function.

If multiple rows are selected by the lookup condition, the row with the maximum lookup column is returned. This is the same behavior as the lookup_seq function and is also the default *return_policy* for lookup_ext.

LOOKUP_EXT EDITOR

The lookup_ext function wizard makes it easy to select a translate table, a cache specification, condition(s), output(s), order return data, and if needed, create a custom SQL filter.



ADDITIONAL CONSIDERATIONS

- When calling lookup_ext in jobs or work flows, the caching mode is always NO_CACHE because Data Integrator cannot determine when to release the cache after executing the function. Also, Data Integrator does not keep track of how many times this function is called. This should not be a problem as lookup in jobs and work flows is generally called to perform one-time initializations.

- In all parameters of lookup_ext, you can reference any Data Integrator-supported data type except the LONG and NRDM data types. So, you cannot look up a LONG column, or specify a column or expression of type LONG in *default_value_list*, *orderby_column_list*, or *condition_list*.
- If an optional parameter is missing, an empty placeholder ([]) must occupy that spot if other optional parameters that follow the missing parameter are to be specified in the function call.
- Business Objects recommends that for best performance, you use the equality operator to specify lookup condition. If the caching mode is NO_CACHE and the lookup_ext is against a database table, the underlying DBMS typically has fast access methods such as an index to retrieve data based on an indexed key. If the caching method is PRE_LOAD_CACHE, using equal comparison will result in more efficient memory lookup than any other comparison operators.

  If the caching method is PRE_LOAD_CACHE, any lookup condition involving a constant expression will be pushed down to the database resulting in a smaller lookup cache than the current lookup.

lookup_seq

Retrieves a value in a table or file based on the values in a different source table or file and a particular sequence value.

SYNTAX

```
lookup_seq (translate_table, result_column, default_value,
sequence_column, sequence_expression, compare_column,
expression)
```

RETURN VALUE

any type	The value in the *translate_table* that meets the lookup_seq requirements. The return type is the same as *result_column*.

WHERE

translate_table	The table or file that contains the result or value you are looking up (*result_column*). The *sequence_column* and *compare_column* are also located in this table. Use a fully qualified table name, including the datastore, owner, and table name. For example: `ERP_ds.OWNER.EMPLOYEES` The *translate_table* is cached automatically for the operation of the function.
result_column	The column containing the values you want to retrieve. This column is in the *translate_table*.
default_value	The value returned when there is no matching row in the *translate_table*.

sequence_column The column in *translate_table* that indicates the sequence of the row. This column often contains a date that indicates when new values were added to the row. For example, in some source tables, *sequence_column* is the `EFFDT` column, which indicates when the data in the row became effective.

sequence_expression
The value the function searches for in the *sequence_column* to find a matching row. For example, if you are looking up values from a slowly changing dimension table and are interested in only those rows in which the data is current as of today, you could use the return value from the sysdate function for *sequence_expression*.

compare_column The column in the *translate_table* that the function uses to find a matching row.

expression The value that the function searches for in the *compare_column*. This can be a simple column reference, such as a column found in both a source and the *translate_table*. This can also be a complex expression given in terms of constants and input column references.

When *expression* refers to a unique source column, you do not need to include a table name qualifier. If *expression* is from another table or is not unique among the source columns, you need a table name qualifier.

NOTE: You can specify more than one *compare_column* and *expression* pair—simply add additional pairs at the end of the function statement. The values must match for all specified pairs in order for the lookup function to find a matching row.

The lookup_seq function uses a value you provide (*expression*) to find a corresponding value in a different file or table (*translate_table*). When multiple rows match, the function uses the row's sequence to determine the matching row.

More specifically, the function searches for the rows in the *translate_table* where the value in the *compare_column* matches the value in *expression*. When the function finds multiple matching rows, it searches in the *sequence_column* for the row with the closest value less than or equal to the *sequence_expression*. If no row has a value less than or equal to the *sequence_expression*, the function finds the row with the closest value to the *sequence_expression*. For the matching row, the function returns the value in the *result_column*.

For example, if your source schema uses an employee ID to identify each row, and you want the employee's salary at the end of the previous year in your target schema, you can use the lookup_seq function to return the employee salary given the employee ID and the effective date of the salary. The salary returned will be the value corresponding to the latest effective date less than or equal to the value of *sequence_expression*.

In SQL terms, the lookup_seq function evaluates *expression* for each row, then determines which sequence column value meets the requirements:

```
SELECT MAX(sequence_column)
   FROM translate_table, source_table
   WHERE sequence_column <= sequence_expression
      AND compare_column =
         source_table.expression)
```

Suppose this query stores the *sequence_column* value returned as *sequence_result*. Next, the function uses the *sequence_result* to find the proper *result_column*:

```
SELECT result_column
   FROM translate_table, source_table
   WHERE sequence_column = sequence_result
      AND compare_column =
         source_table.expression
```

The value returned by these queries is the result of the lookup_seq function for the row.

You can specify multiple *compare_column* and *expression* pairs to uniquely identify the *result_column* value. However, the function wizard only provides fields for one pair; add extra *compare_column* and *expression* pairs to the output that the wizard generates.

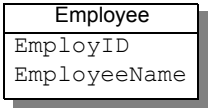
DATA INTEGRATOR always caches the comparison table when performing a lookup_seq function.

If the lookup_seq function does not find the value of *expression* in *compare_column*, then the function evaluates and returns the *default_value*.

EXAMPLE

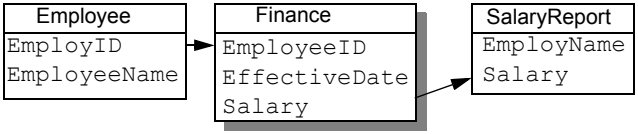
You can use the lookup_seq function to return a value from a slowly changing dimension table given an identifier. For example, suppose you have a source table that contains a numerical identifier, such as an employee number, and you want to retrieve the employee's salary at a specific time in the past. You can use the lookup_seq function to return the employee's salary on a particular date based on the employee number.

The source table contains the employee number and employee name.

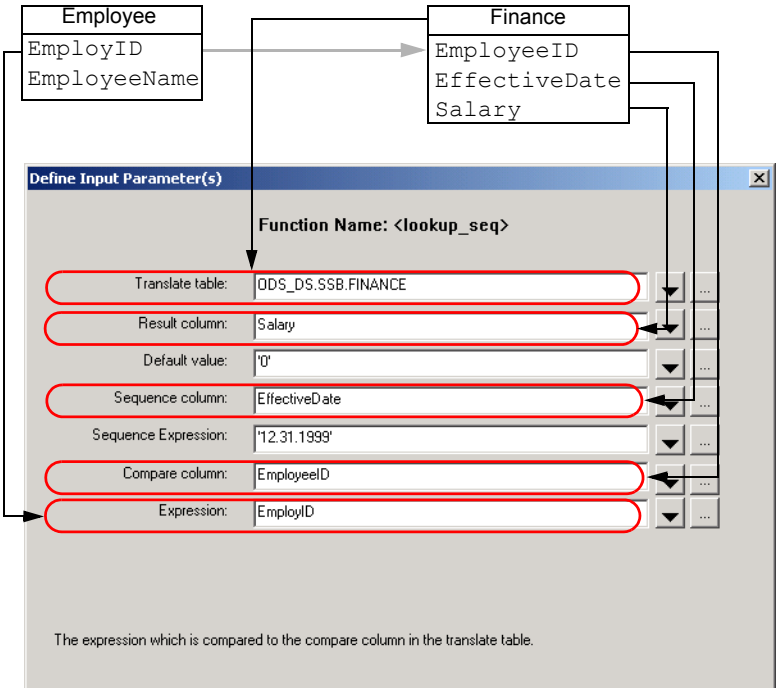


You want the target table to contain the employee name and salary.

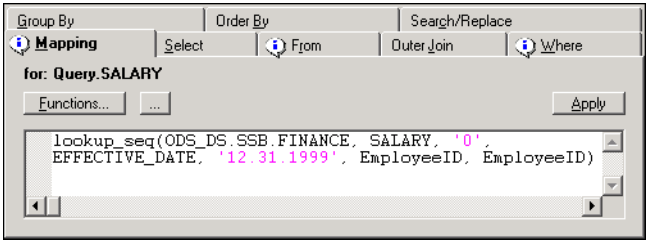
Use the lookup_seq function to translate the values from the source table to the desired values in the target table. The lookup_seq function uses a third, "translation," table.



To produce the desired target column, select the column in the target schema. Next, click the **Functions** button, located over the **Mapping** text box. In the function wizard, select **Miscellaneous_functions** under **Function categories**, then select **lookup_seq** under **Function name**. Enter the function parameters as follows:



The function wizard automatically produces the mapping text.



For each employee, this lookup_seq function returns the value from the salary column for that employee that is the most recent before December 31, 1999.

lower

Changes the characters in a string to lowercase.

SYNTAX

`lower(`*value*`)`

RETURN VALUE

varchar	The lowercase string. The return type is the same as *value*. Any characters that are not letters are left unchanged.

WHERE

value	The string to be modified.

EXAMPLE

Function	Results
`lower('Accounting101')`	`'accounting101'`
`upper((LastName,1,1))|` `  lower(substr(LastName,2,LENGTH(LastName)))`	The value in column `LastName` with the first letter uppercase and the rest of the value lowercase. Note that this example does not account for two-word last names.

lpad

Pads the string with characters in the left from a given pattern

This function repeats the pattern at the beginning of the input string until the final string is the appropriate length. If the input_string is already longer than the expected length, then this function truncates the string.

SYNTAX

`lpad(`*input_string, size, pad_string*`)`

RETURN VALUE

varchar	The modified string. The return type is the same as *value*. Any characters that are not letters are left unchanged.

WHERE

input_string	The string source.
size	An integer value indicating the number of characters in the return string.
pad_string	A character or set of characters that this function concatenate to *input_string*.

EXAMPLE

Function	Results
lpad('Tanaka', 15, ' ')	' Tanaka'
lpad(last_name, 25, ' ')	The value in the column last_name, padded with spaces to 25 characters on the left or truncated to 25 characters.

lpad_ext

Pads the left side of the string with logical characters from a given pattern.

NOTE: These logical characters prohibit this function from getting pushed down to the database.

This function repeats the pattern at the beginning of the input string until the final string is the appropriate length. If the input_string is already longer than the expected length, then this function truncates the string.

SYNTAX

`lpad_ext(`*input_string, size, pad_string*`)`

RETURN VALUE

varchar	The modified string. The return type is the same as *value*. Any characters that are not letters are left unchanged.

WHERE

input_string	The string source.
size	An integer value indicating the number of characters in the return string.
pad_string	A logical character or set of logical characters that this function concatenates to the *input_string*.

EXAMPLES

Function	Results
lpad_ext('Tanaka', 15, ' ')	' Tanaka'
lpad_ext(last_name, 25, ' ')	The value in the column last_name, padded with spaces to 25 characters on the left or truncated to 25 characters.

The extended function yield results different from the basic function when "char1" or "char2" contains multibyte characters. For instance, let "¶" be a Chinese ideograph, a double-byte character which occupies two cells on a display device or printer, then:

Function	Input	Output
lpad	("abc¶¶", 10)	" ¶¶abc"
lpad	("abcd", 10, "¶")	"¶¶¶abcd"
lpad	("ab¶¶", 4)	"¶ab"
lpad_ext	("abc¶¶", 10)	" ¶¶abc"
lpad_ext	("abcd", 10, "¶")	"¶¶¶¶¶abcd"
lpad_ext	("ab¶¶", 4)	"¶¶ab"

ltrim

Removes specified characters from the start of a string.

SYNTAX

`ltrim(`*input_string*`,` *trim_string*`)`

RETURN VALUE

varchar	The modified string. The return type is the same as *input_string*.

WHERE

input_string	The string to be modified.
trim_string	The characters to remove from *input_string*.

The ltrim function is case-sensitive.

The function scans *input_string* left-to-right removing all characters that appear in *trim_string* until it reaches a character not in *trim_string*.

EXAMPLE

Function	Results
`ltrim('Marilyn', ' ')`	`'Marilyn'`
`ltrim('ABCABCD', 'ABC')`	`'D'`
`ltrim('ABCABCD', 'EFG')`	`'ABCABCD'`
`ltrim('ABCDEABCDE', 'ABC'`	`'DEABCDE'`

To remove all leading blanks in a string, use ltrim as follows:

```
ltrim(EMPLOYEE.NAME, ' ')
```

where `EMPLOYEE.NAME` specifies the `NAME` column in the `EMPLOYEE` table.

ltrim_blanks

Removes blank characters from the start of a string.

SYNTAX

`ltrim_blanks(`*input_string*`)`

RETURN VALUE

varchar	The modified string. The return type is the same as *input_string*.

WHERE

input_string	The string to be modified.

EXAMPLE

Function	Results
`ltrim_blanks('  Marilyn')`	`'Marilyn'`
`ltrim_blanks(last_name)`	The value contained in the column last_name, with all leading blanks and control characters removed.

ltrim_blanks_ext

Removes blank and control characters from the start of a string.

SYNTAX

```
ltrim_blanks_ext(input_string)
```

RETURN VALUE

varchar	The modified string. The return type is the same as *input_string*.

WHERE

input_string	The string to be modified.

EXAMPLE

Function	Results
`ltrim_blanks_ext('  Marilyn')`	`'Marilyn'`
`ltrim_blanks_ext(last_name)`	The value contained in the column last_name, with all leading blanks removed.

mail_to

Captures the specified number of lines in the trace log and error log, packages the information as e-mail, and uses your Job Server computer's mail client to send e-mail messages to your local mail server for standard e-mail processing.

SYNTAX

`mail_to(`*recipients_list*`,` *subject*`,` *message*`,`
number_of_trace_lines`,` *number_of_error_lines*`)`

RETURN VALUE

int	Returns 0 if function succeeds. Returns a non-zero integer if function fails.

WHERE

recipients_list	A string containing one or more recipient e-mail addresses separated by commas (,). This string cannot be empty and must contain valid, qualified e-mail address information.
subject	A string containing the subject of the e-mail. This string can be empty.
message	A string containing the e-mail message. This string can be empty.
number_of_trace_lines	The number of lines from the end of the trace log file to append to the end of the e-mail. This input cannot be empty.
number_of_error_lines	The number of lines from the end of the error log file to append at the end of the e-mail. This input cannot be empty.

Only use this function within a script.

To use this function, a mail client must be installed and running on the Job Server computer that calls the function. The login account for the mail client must have the same user name and password as the DATA INTEGRATOR service. The type of client varies by the operating system:

- If the Job Server is on a computer running the Windows operating system, then the mail client must comply with MAPI (message application programming interface). In addition, the mail client must be configured as the default mail client. For example, Microsoft Outlook is a MAPI-based mail client.
- If the Job Server is on a computer running the UNIX operating system, then the mail client must be mailx-compliant.

NOTE: If you do not have the required mail client, contact Business Objects Customer Support for assistance.

EXAMPLE

Function	Results
`$myvar = mail_to('admin@company.com', 'Out of memory error in the SalesFact job. Please fix the error before running recovery job.',' ', 10, 10)`	The message is sent to one recipient.
`$myvar = mail_to('admin@company.com, manager@company.com', 'Out of memory error:' || systime(), 'Out of memory error while running the data flow:' || $dataflow_name || ' in the job:' || $job_name || '.', 10, 20)`	The message is sent to two recipients. The job name and data flow names are included in the text of the message as variables. Note that Data Integrator trims blank spaces from the end of strings; this example includes a blank on the beginning of the next string. You can also concatenate a string with a single blank. In the script, type "$a = ;" where $a is the local integer variable defined in the work flow. Put the cursor just ahead of the semicolon before clicking the **Functions** button to construct a mail_to statement.

NOTE: Often, you list e-mail addresses as nicknames in your mail service address book. If your mail system is compatible with the Data Integrator mail_to function, you can use these nicknames (comma separated) as values in the *recipients_list*. In this case, the Data Integrator mailer program searches your e-mail address book for the nickname and uses the corresponding qualified e-mail address for message routing.

max

Returns the maximum value from a list.

SYNTAX

`max(`*value_list*`)`

RETURN VALUE

any type	The maximum value of the column values. The return type is the same as the values in *value_list*.

WHERE

value_list	The source values for which to identify a maximum.

EXAMPLE

To calculate the maximum value in the salary column of a table, use the max function in a query:

- In the **Mapping** tab of the query editor, enter:

  ```
  max(SALARY)
  ```

- In the **Group By** tab in the query editor, specify the columns for which you want to find the maximum salary, such as the department column. For each unique set of values in the group by list, such as each unique department, Data Integrator calculates the maximum salary.

min

Returns the minimum value from a list.

SYNTAX

`min(`*value_list*`)`

RETURN VALUE

any type	The minimum value of the column values. The return type is the same as the values in *value_list*.

WHERE

value_list	The source values for which to identify a minimum.

EXAMPLE

To calculate the minimum value in the salary column of a table, use the min function in a query:

- In the **Mapping** tab of the query editor, enter:

  ```
  min(SALARY)
  ```

- In the **Group By** tab in the query editor, specify the columns for which you want to find the minimum salary, such as the department column. For each unique set of values in the group by list, such as each unique department, Data Integrator calculates the maximum salary.

month

Determines the month in which the given date falls.

SYNTAX

`month(`*date1*`)`

RETURN VALUE

int	The number from 1 to 12 that represents the month component of *date1*.

WHERE

date1	The source date.

EXAMPLE

Function	Results
`month(to_date('Jan 22, 1997', 'mon dd, yyyy'))`	1
`month(to_date('3/97', 'mm/yy'))`	3

num_to_interval

Converts a numeric value to an interval.

SYNTAX

`num_to_interval(`*number1*, *format*`)`

RETURN VALUE

interval	The converted interval.

WHERE

number1	The value of type int, real, decimal, or numeric to be converted.
format	A string describing the format of the interval. Choose from the following values: D Days H Hours M Minutes S Seconds

EXAMPLE

Function	Results
`num_to_interval(elapsed_days, 'D')`	The value from the column `elapsed_days` converted to an interval of days.
`start_time + num_to_interval(elapsed_seconds, 'S')`	A time indicating the number of seconds from the column `elapsed_seconds` from the time in the column `start_time`.

nvl

Replaces NULL values.

SYNTAX

`nvl(`*expression1*`,` *replacement_value*`)`

RETURN VALUE

any type	The value of *expression1* if not NULL, otherwise, the value of *replacement_value*.

WHERE

expression1	The value to be tested for NULL.
replacement_value	The value to replace *expression1* if *expression1* is NULL. *replacement_value* must be the same data type as *expression1*.

EXAMPLE

Function	Results
`nvl(modification_date, sysdate())`	If the column `modification_date` for a row hasn't been set, this function inserts today's date.
`nvl(lookup(r3..vbpa, kunnr, 'NULL', vbeln, vbak.vbeln, posnr, vbap.posnr, parvw, 'RE'), lookup(r3..vbpa, kunnr, 'NULL', vbeln, vbak.vbeln, posnr, vbap.posnr, parvw, 'RG'))`	Both expressions are determined by the result of lookup functions.

print

Prints the given string to the trace log.

SYNTAX

`print(`*input_string*`)`

RETURN VALUE

int	Value is 0 when the string contains valid data. Value is NULL and no string prints when the string contains NULL data.

WHERE

input_string	The message to be written to the trace log.

EXAMPLE

Function	Results
`print('Reached decision point for running full or incremental data flows')`	Writes "Reached decision point for running full or incremental flows" to trace log and returns 1.
`print('The date is: [$start_date]')`	Writes "The date is 2000.06.03" to trace log and returns 1.
`print('[$month_sal*12]')`	Writes "48000" to trace log and returns 1.
`print('Total Sal is: [$month_sal*12]');`	Writes "Total Sal is: 48000" to trace log and returns 1.
`print('The return value from the SQL() function is > [$y]');`	Writes "The return value from the SQL() function is > 23456" to trace log and returns 1.

pushdown_sql

Allows you to create dynamic WHERE clauses.

SYNTAX

```
pushdown_sql (datastore, input_string)
```

RETURN VALUE

None.

WHERE

datastore	The name of the datastore containing the data you want to retrieve. Data Integrator creates a WHERE clause and pushes it down to this database. Surround the datastore name by single quotes.
input_string	A character string that forms the WHERE clause. Surround the character string by single quotes. Typically, this is a column from another source to the query, such as an XML message. Delimit columns or parameters with curly braces. For example, {XML_IN.STATUS_QUERY}. If the string contains a curly brace, use the backslash escape key to delimit the curly brace.

DETAILS

The pushdown_sql function allows you to create WHERE clauses that change based on data input. With the pushdown_sql function, the WHERE clause need not be pre-specified. The pushdown_sql function is particularly useful in real-time jobs, if you want to select data based on input from an XML message.

Unlike other functions, the pushdown_sql function can only be used in the WHERE clause of a query transform. You cannot use the pushdown_sql function in other places, such as in a query's mapping, in a conditional, or in a script.

Data Integrator must be able to push the WHERE clause that it creates from this function to the database. This function works best, therefore, when used in a query transform where the immediate input is the table source where you want to push the WHERE clause.

Data Integrator does not parse the SQL contained in the input string. Therefore, the input must be well-formed with correct syntax.

EXAMPLE

Suppose the datastore EC_DS contains the table BIKES, which stores information about different models. And suppose the QUERY_REQUEST column in the XML_IN message contains requests for information from this table. For example, a value in the QUERY_REQUEST column might be:

```
TYPE = 'MOUNTAIN' and PRICE < 1500
```

In a data flow used in a real-time job, you can use the pushdown_sql function in a query to select data from the BIKES table based on the data in the XML_IN message. You can return the data in another XML message.

Function	Results
`pushdown_sql ('EC_DS', '{XML_IN.QUERY_REQUEST}')`	Data Integrator includes the following WHERE clause in the SQL SELECT statement: `WHERE TYPE = 'MOUNTAIN' and PRICE < 1500`

quarter

Determines the quarter in which the given date falls.

SYNTAX

`quarter(`*date1*`)`

RETURN VALUE

int	The number from 1 to 4 that represents the quarter component of *date1*.

WHERE

date1	The source date.

EXAMPLE

Function	Results
`quarter(to_date('Jan 22, 1997', 'mon dd, yyyy'))`	1
`quarter(to_date('5/97', 'mm/yy'))`	2

raise_exception

Calling this function causes an exception to be generated.

SYNTAX

`raise_exception`(*error_msg*)

RETURN VALUE

int	Returns '1' always.

WHERE

error_msg	The string which will be written to the Job Server's error log.

DETAILS

The work flow or job may or may not be terminated based on whether a try-catch block surrounds the call.

EXAMPLES

```
ifthenelse(sal < 1000000, 0,
raise_exception('Salary exceeds 1 million
dollars.'))
```

raise_exception_ext

Calling this function causes an exception to be generated with and exit code.

SYNTAX

`raise_exception_ext`(*error_msg*),(*exit_code*)

RETURN VALUE

int	Returns '1' always.

WHERE

error_msg	The string which will be written to the Job Server's error log.
exit_code	Code the job exits with, if the exception is not caught in a try/catch block. Use a number in range 1 to 255 (zero means "success" to all operating systems).

DETAILS

The work flow or job may or may not be terminated based on whether a try-catch block surrounds the call.

EXAMPLES

```
ifthenelse(sal < 1000000, 0,
raise_exception_ext('Salary exceeds 1 million
dollars.', sal/1000000 + 1))
```

rand

Returns a random number between 0 and 1.

SYNTAX

```
rand()
```

RETURN VALUE

real	The random number. The return value is between 0 and 1.

EXAMPLE

Function	Results
`100 * rand()`	A random number between 0 and 100.

replace_substr

Takes an input string, replaces each occurrence of a specified substring with a specified replacement, and returns the result.

SYNTAX

`replace_substr`(*in_str, search_str, replace_str*)

RETURN VALUE

varchar

WHERE

in_str	The input string that you are changing. If NULL, return will be NULL.
search_str	String to search for. If *search_string* is NULL, varchar is returned and will be the same as in_str.
replace_str	If *replace_string* is omitted or NULL, all occurrences of *search_string* are removed.

EXAMPLE

Function	Result
`replace_substr('a penny saved is a penny earned', 'penny', 'million')'`	`'a million saved is a million earned'`

repository_name

Returns a database connection string and owner name. For example: `beq-local.DBUser`. This is the ID for the repository from which the job is run.

SYNTAX

```
repository_name()
```

RETURN VALUE

varchar

EXAMPLE

```
print('Repository Name: [repository_name()]')
```

round

Rounds a given number to the specified precision.

SYNTAX

`round(`*num1*`, `*precision*`)`

RETURN VALUE

decimal, double, int, or real
: The rounded number. The return type is the same as the original number, *num1*.

WHERE

num1
: The source number.

precision
: An integer indicating the number of decimals in the result. If *precision* is negative, digits left of the decimal point are rounded.

EXAMPLE

Function	Results
`round(120.12345, 2)`	`120.12`
`round(120.12999, 2)`	`120.13`
`round(120, -2)`	`100`
`round(120.123, 5)`	`120.12300`

rpad

Pads a string with characters from a given pattern.

The function repeats the pattern at the end of the input string until the final string is the appropriate length. If the input string is already longer than the expected length, this function truncates the string.

SYNTAX

`rpad(`*input_string*`,` *size*`,` *pad_string*`)`

RETURN VALUE

varchar	The new string.

WHERE

input_string	The source string.
size	An integer value indicating the number of characters in the resulting string.
pad_string	A character or set of characters that this function concatenates to *input_string*.

EXAMPLE

Function	Results
`rpad('Tanaka',15,' ')`	`'Tanaka          '`
`rpad(last_name,25,' ')`	The value in the column `last_name`, padded with spaces to 25 characters, or truncated to 25 characters.

rpad_ext

Pads a string with logical characters from a given pattern.

NOTE: These logical characters prohibit this function from getting pushed down to an Oracle database.

The function repeats the pattern at the end of the input string until the final string is the appropriate length. If the input string is already longer than the expected length, this function truncates the string.

SYNTAX

`rpad_ext(`*input_string*`,` *size*`,` *pad_string*`)`

RETURN VALUE

varchar	The new string.

WHERE

input_string	The source string.
size	An integer value indicating the number of characters in the resulting string.
pad_string	A character or set of characters that this function concatenates to *input_string*.

EXAMPLES

Function	Results
`rpad_ext('Tanaka',15,' ')`	`'Tanaka '`
`rpad_ext(last_name,25,' ')`	The value in the column `last_name`, padded with spaces to 25 characters, or truncated to 25 characters.

The extended function yield results different from the basic function when "char1" or "char2" contains multibyte characters. For instance, let "¶" be a Chinese ideograph, a double-byte character which occupies two cells on a display device or printer, then:

Function	Input	Output
rpad	("abc¶¶", 10)	"abc¶¶ "
rpad	("abcd", 10, "¶")	"abcd¶¶¶"
rpad	("ab¶¶", 4)	"ab¶"
rpad_ext	("abc¶¶", 10)	"abc¶¶ "
rpad_ext	("abcd", 10, "¶")	"abcd¶¶¶¶¶"
rpad_ext	("ab¶¶", 4)	"ab¶¶"

rtrim

Removes specified characters from the end of a string.

SYNTAX

`rtrim(`*input_string*`,` *trim_string*`)`

RETURN VALUE

varchar	The modified string. The return type is the same as *input_string*.

WHERE

input_string	The string to be modified.
trim_string	The characters to remove from *input_string*.

The function scans *input_string* right-to-left removing all characters that appear in *trim_string* until it reaches a character not in *trim_string*.

To remove all trailing blanks in a string, use the rtrim_blanks function.

EXAMPLE

Function	Results
`rtrim('Marilyn  ', ' ')`	`'Marilyn'`
`rtrim('ZABCABC', 'ABC')`	`'Z'`
`rtrim('ZABCABC', 'EFG')`	`'ZABCABC'`

rtrim_blanks

Removes blank characters from the end of a string.

SYNTAX

`rtrim_blanks(`*input_string*`)`

RETURN VALUE

varchar	The modified string. The return type is the same as *input_string*.

WHERE

input_string	The string to be modified.

EXAMPLE

Function	Results
`rtrim_blanks('Marilyn  ')`	`'Marilyn'`
`rtrim_blanks(last_name)`	The value contained in the column `last_name` with trailing blanks removed.

rtrim_blanks_ext

Removes blank and control characters from the end of a string.

SYNTAX

`rtrim_blanks_ext(`*input_string*`)`

RETURN VALUE

varchar	The modified string. The return type is the same as *input_string*.

WHERE

input_string	The string to be modified.

EXAMPLE

Function	Results
`rtrim_blanks('Marilyn  ')`	`'Marilyn'`
`rtrim_blanks(last_name)`	The value contained in the column `last_name` with trailing blanks and control characters removed.

set_env

Sets a system environment variable to a specified value for the duration of a job.

SYNTAX

`set_env('variable_name', variable_value)`

RETURN VALUE

int	Returns 1 if successful, otherwise, 0.

WHERE

variable_name	The name of an environment variable. The name must be surrounded by single quotes.
variable_value	The value you want assigned to the environment variable. If the value is text, you must surround the text by single quotes. Data Integrator only sets the variable to this value for the duration of the current job.

Use the get_env and set_env functions to set and retrieve variables across operations in a job. Also refer to "get_env" on page 375.

EXAMPLE

Function	Results
`setenv('TMP','C:\Data Integrator\Temp')`	Sets the value of the TMP environment variable to "C:\Data Integrator\Temp" and returns a 1.

sleep

Suspends the execution of the calling data flow or work flow.

SYNTAX

`sleep`(*num_millisecs*)

RETURN VALUE

int	Returns '1', always.

WHERE

num_millisecs	The number of milliseconds to "sleep".

DETAILS

Calling this function causes the thread that executes this function to halt operations for the given number of milliseconds. To force a job to halt operations (until a condition becomes true), call this function in a work flow, not in a data flow.

EXAMPLE

The following example invokes sleep for one second when a file exists in a directory called 'c'.

```
while (file_exists('c:/temp.msg') == 0)
begin
sleep(1000);
end
```

sql

Runs a SQL operation against tables in the specified database.

SYNTAX

`sql(`*datastore*`, `*sql_command*`)`

RETURN VALUE

varchar(1020)	Returns the first 1020 characters from the query's output. Typically, if *sql_command* is a SELECT statement, the return value is the first row value of the first column. If *sql_command* is not a SELECT statement, the return value is typically NULL. You must remember this if you assign the value returned to a variable.

WHERE

datastore	A string containing the name of the datastore where the tables involved in the SQL operation reside. This name is the name you specified when you created the datastore in Data Integrator. Include this string in single quotation marks.
sql_command	A string containing the text of the SQL command to execute. This string must be enclosed in single quotation marks ('). If the string contains quoted values, the internal quotation marks must be single quotation marks preceded by the escape character, backslash (\).

Data Integrator makes column and table names uppercase when sending the *sql_command* to Oracle to resolve. To specify a lowercase column or table name from an Oracle database, enclose the name with double quotation marks (").

EXAMPLES

Function	Results
`sql('source_ds', 'SELECT EmpID FROM Emp WHERE sal > 100000')`	Runs the SQL command against the database connected through the datastore `source_ds`.
`sql('source_ds','SELECT customer.LastName FROM customer WHERE customer.State = \'CA\'')`	Returns the last names of the customers in California from the customer table. Note that the quotation marks around the state value require a backslash to indicate that the single quotation marks are considered part of the *sql_command* string.
`sql('oracle_ds', ('SELECT "start_timestamp" FROM "status_table" WHERE "extract_name" = \'DF_RecoverDim\' AND "stop_timestamp" = NULL')`	Returns the timestamp from a status table for the completed data flow. An Oracle datastore requires double quotation marks around the lowercase column and table names.
`$start_date=sql('warehouse_ds', 'SELECT finish_timestamp FROM time_table WHERE table_name= "Component_Orders" ');`	In this script example, because the function returns a varchar value, it is not possible to assign the return value to a date variable directly. Modify your statement to the next example.

Function	Results
`$temp_char=sql('warehouse_ds', 'SELECT finish_timestamp FROM time_table WHERE table_name= "Component_Orders" ');` `$start_date=to_date($temp_char,'dd-mon-yyyy');`	This script example assumes the database returns the date in `dd-mon-yyyy` format. If you are unsure of the format the database returns, then force it to return the date in a specific format by doing a conversion. To accomplish this, use the to_char function in Oracle or the convert function in MS SQL.

substr

Returns a specific portion of a string starting at a given point in the string.

SYNTAX

```
substr(input_string, start, length)
```

RETURN VALUE

varchar	The modified string. The return type is the same as *input_string*.

WHERE

input_string	The string to be modified.
start	The position of the first character in the new string. The function counts characters from the beginning of *input_string*. In normal data flows, the first character is position number 1. If *start* is 0, the new string begins with the first character (position 1). If *start* is negative, the new strings begins with the first character in the string (position 1). If *start* is greater than the number of characters in *input_string*, the function returns a NULL string.
length	The number of characters in the resulting string. If *length* is 0 or negative, the function returns a NULL string. If *length* is greater than the number of characters remaining in *input_string* after *start*, the function returns only the remaining characters.

EXAMPLE

Function	Results
`substr('94025-3373', 1, 5)`	`'94025'`
`substr('94025-3373', 7, 4)`	`'3373'`
`substr('94025', 7, 4)`	`NULL`
`substr('Dr. Schultz', 4, 18)`	`'Schultz'`
`substr('San Francisco, CA',-4, 18)`	`'San Francisco, CA'`

sum

Calculates the sum of a given set of values.

SYNTAX

`sum(`*value_list*`)`

RETURN VALUE

decimal, double, int, or real
: The total of the values. The return type is the same as the values in *value_list*.

WHERE

value_list : The source values to sum.

EXAMPLE

To calculate the sum of values in the salary column of a table, use the sum function in a query:

- In the **Mapping** tab of the query editor, enter:

  ```
  sum(SALARY)
  ```

- In the **Group By** tab in the query editor, specify the columns for which you want to find the total salary, such as the department column. For each unique set of values in the group by list, such as each unique department, Data Integrator calculates the sum of the salary.

sysdate

Returns the current date as listed by the Job Server's operating system.

NOTE: The value that the sysdate function returns is a datetime value. Internally Data Integrator reads both the date and the time when it runs a sysdate function. The data that is used by the job depends on the data type of a particular column. For example, if the data type of a column in a query is `date`, Data Integrator only uses the date for calculations. The time data is ignored. If you change the data type to `datetime`, both a date and a time are used.

SYNTAX

```
sysdate()
```

RETURN VALUE

date Today's date.

EXAMPLE

Function	Results
`isweekend(sysdate())`	Tests whether today is a Saturday or Sunday.
`to_char(sysdate(), 'yyyy.mm.dd')`	Converts the sysdate function's `datetime` value to a string that displays only the date. For example, you can use this to exclude part of the `datetime` data by only providing a format for the data you want to display in a report. To convert a `datetime` value to a string containing only the date, use this expression and change the column's data type to `varchar`.

system_user_name

Returns the user name used to log into the Job Server's operating system.

SYNTAX

```
system_user_name()
```

RETURN VALUE

varchar

EXAMPLE

```
print('Starting execution of Job: [job_name()]
as user: [system_user_name()]');
```

systime

Returns the current time as listed by the Job Server's operating system.

SYNTAX

```
systime()
```

RETURN VALUE

time The current time.

EXAMPLE

Function	Results
`$timestamp = sql('my_datastore',('UPDATE status_table SET job_start_time = \' '||to_char(systime(),'hh24:mi:ss.ff'))||'\'');`	This expression updates the `job_start_time` column of the `status_table` with the current time. It also formats the time data.
`to_char(systime(),'hh24:mi:ss.ff'`	Trims date data from the systime() function in cases where it is added by default. Set the column that contains this expression to the data type `varchar`. The data type for a column that calls the systime() function should be `time`. If the data type is set to `datetime`, Data Integrator will add the default date for the `datetime` data type (1900:01:01) because systime() does not read dates.

table_attribute

Retrieves the value of the specified table attribute.

SYNTAX

`table_attribute`(*table_name, attribute_name*)

RETURN VALUE

varchar	The value of the table attribute. If the specified attribute does not exist, NULL is returned.

WHERE

table_name	Use the following format: '`datastore.owner.table`'. If a valid table by this name does not exist, NULL is returned.
attribute_name	The name of a table attribute. Valid attributes for a table are listed in the table's Properties window.

EXAMPLE

Function	Result
`table_attribute('mssql.awez.CUSTOMER', 'Number_Of_Inserts')`	'1788'

to_char

Converts a date to a string. Also supports the Oracle 9i timestamp data type. Its precision allows up to 9 digits for sub-seconds. For more information, see "timestamp" on page 203.

SYNTAX

`to_char(`*date1*`, `*format*`)`

RETURN VALUE

varchar	A formatted string describing *date1*.

WHERE

date1	The source date, time, or datetime value.
format	A string indicating the format of the generated string. Choose from the following codes:

DD	2-digit day of the month
MM	2-digit month
MONTH	Full name of month
MON	3-character name of month
YY	2-digit year
YYYY	4-digit year
HH24	2-digit hour of the day (0-23)
MI	2-digit minute (0-59)
SS	2-digit second (0-59)
FF	Up to 9-digit sub-seconds

Other values included in *format* appear in the result unchanged.

EXAMPLE

Function	Results
`to_char(call_date, 'dd-mon-yy hh24:mi:ss.ff')`	The date value from the `call_date` column formatted as a string, such as: `28-FEB-97 13:45:23.32`

The hyphens and spaces in *format* in the example are reproduced in the result; all the other characters are recognized as part of a parameter string in the table above and substituted with appropriate current values.

to_date

Converts a string to a date. Also supports the Oracle 9i timestamp data type. Its precision allows up to 9 digits for sub-seconds. For more information, see "timestamp" on page 203.

SYNTAX

`to_date(`*input_string*`,` *format*`)`

RETURN VALUE

date, time, or datetime
: A date, time, or both representing the original string.

WHERE

input_string — The source string.

format — A string indicating the format of the source string. Choose from the following codes:

DD	2-digit day of the month
MM	2-digit month
MONTH	Full name of month
MON	3-character name of month
YY	2-digit year
YYYY	4-digit year
HH24	2-digit hour of the day (0-23)
MI	2-digit minute (0-59)
SS	2-digit second (0-59)
FF	Up to 9-digit sub-seconds

EXAMPLE

Function	Results
`to_date('Jan 8, 1968', 'mon dd, yyyy')`	`1968.01.08` stored as a date.

to_decimal

Converts a varchar to a decimal.

SYNTAX

`to_decimal`(*in_str, decimal_sep, thousand_sep, scale*)

RETURN VALUE

decimal	Uses a precision of 28 and the given scale.

WHERE

in_str	The number string. Null implies a NULL return.
decimal_sep	The character that separates the decimal component from the whole number component.
thousand_sep	The character that separates thousands from hundreds in the whole number component.
scale	The number of digits to the right of the decimal point in the returned value.

DETAILS

Takes a string that represents a number and converts it to a decimal. If the input string is invalid, a 0 is returned.

EXAMPLE

Function	Result
`to_decimal('99,567.99', '.', ',', 3)`	`99567.990`

total_rows

Returns the number of rows in a particular table in a datastore. This function can be used with any type of datastore.

SYNTAX

`total_rows(`*datastore.owner.table_name*`)`

or for a memory datastore:

`total_rows(`*datastore..table_name*`)`

RETURN VALUE

int	The number of rows in the table.

WHERE

datastore	The name of the datastore containing the table.
owner	The name of the datastore owner. Not used for memory tables.
table_name	The name of the database table or memory table containing the rows you want to count.

EXAMPLE

Function	Results
total_rows (ora_ds.scott.emp_table)	Retrieves the total number of rows from an Oracle table
total_rows (mem_ds..bigtable)	Retrieves the total number of rows from a memory table

trunc

Truncates a given number to the specified precision, without rounding the value.

SYNTAX

`trunc(`*num1*`, `*precision*`)`

RETURN VALUE

decimal, double, int, or real
: The truncated number. The return type is the same as the original number, *num1*.

WHERE

num1
: The source number.

precision
: An integer indicating the number of decimals in the result. If *precision* is negative, digits to the left of the decimal point are truncated and the value is padded with zeros.

EXAMPLE

Function	Results
`trunc(120.12345, 2)`	`120.12`
`trunc(120.12999, 2)`	`120.12`
`trunc(180, -2)`	`100`
`trunc(120.123, 5)`	`120.12300`

truncate_table

Allows you to explicitly expunge data from a memory table. This provides finer control than the active job has over your data and memory usage. Use this function only with memory tables.

SYNTAX

`trunc(`*memds..tab_name*`)`

RETURN VALUE

int	The return value is always 1.

WHERE

memds	The datastore containing the memory table.
tab_name	The name of the memory table from which you want to expunge data.

EXAMPLE

Function	Results
`truncate_table (mem_ds..bigtable)`	`Truncates rows from the memory table.`

upper

Changes the characters in a string to uppercase.

SYNTAX

`upper(`*value*`)`

RETURN VALUE

varchar	The uppercase string. The return type is the same as *value*. Any characters that are not letters are left unchanged.

WHERE

value	The string to be modified.

EXAMPLE

Function	Results
`upper('Accounting101')`	`'ACCOUNTING101'`
`upper(substr(LastName,1,1))|` `  lower(substr(LastName,2,LENGTH(LastName)))`	The value in column `LastName` with the first letter uppercase and the rest of the value lowercase. Note that this example does not account for two-word last names.

week_in_month

Determines the week in the month in which the given date falls.

This function considers the first week of the month to be first seven days. The day of the week is ignored when calculating the weeks.

SYNTAX

`week_in_month(`*date1*`)`

RETURN VALUE

int	The number from 1 to 5 that represents which week in the month that *date1* occurs.

WHERE

date1	The source date.

EXAMPLE

Function	Results
`week_in_month(to_date('Jan 22, 1997', 'mon dd, yyyy'))`	`4`
`week_in_month(to_date('Jan 21, 1997', 'mon dd, yyyy'))`	`3`

week_in_year

Determines the week in the year in which the given date falls.

This function considers the first week of the year to be the first seven days. The day of the week is ignored when calculating the weeks.

SYNTAX

`week_in_year(`*date1*`)`

RETURN VALUE

int	The number from 1 to 53 that represents which week in the year that *date1* occurs.

WHERE

date1	The source date.

EXAMPLE

Function	Results
`week_in_year(to_date('Jan 22, 1997','mon dd, yyyy'))`	`4`
`week_in_year(to_date('Jan 21, 1997','mon dd, yyyy'))`	`3`
`week_in_year(to_date('1996.12.31', 'yyyy.mm.dd'))`	`53` (1996 was a leap year.)

WL_GetKeyValue

Returns the value of a given keyword in Web log search strings.

SYNTAX

```
WL_GetKeyValue(string, keyword)
```

EXAMPLE

If you search for 'BusinessObjects' on Google,

This appears in a Web log as:

```
GET
"http://www.google.com/search?hl=en&lr=&saf
e=off&q=BusinessObjects&btnG=Google+Search"
```

`WL_GetKeyValue('http://www.google.com/searc h?hl=en&lr=&safe=off&q=BusinessObjects&btnG =Google+Search','q')` returns `'BusinessObjects'`.

word

Returns one word out of a string.

SYNTAX

`word(`*input_string*`,` *word_num*`)`

RETURN VALUE

varchar	A string containing the indicated word. The return type is the same as *input_string*.

WHERE

input_string	The source string.
word_num	A nonnegative integer specifying the index of the target word in the string. The first word in a string is word number 1. If *word_num* is 0 or greater than the number of words in *input_string,* then the word function returns a NULL string.

A word is defined to be any string of consecutive non-white space characters terminated by white space, or the beginning and end of *input_string*. White space characters are the following:

- Space
- Horizontal or vertical tab
- Newline
- Linefeed

EXAMPLE

Function	Results
`word('Accounting Department', 1)`	`'Accounting'`
`word('Accounting', 1)`	`'Accounting'`
`word('Accounting', 2)`	`NULL`

word_ext

Returns the word identified by its position in a delimited string.

This function is useful for parsing Web log URLs or file names.

SYNTAX

`word_ext` (*string, separator(s), word_number*)

RETURN VALUE

varchar	A string containing the indicated word. The return type is the same as *string*.

WHERE

string	The source string.
separator (s)	Any character specified.
word_number	A nonnegative integer specifying the index of the target word in the string. The first word in a string is word number 1. If *word_number* is 0 or greater than the number of words in *string,* then the word function returns a NULL string.

A word is defined to be any string of consecutive non-white space characters terminated by white space, or the beginning and end of *string*. White space characters are the following:

- Space
- Horizontal or vertical tab
- Newline
- Linefeed

EXAMPLE

Function	Results
`word_ext('www.acta.com',2,'.')`	`'acta'`
`word_ext('www.cs.wisc.edu', '.', -2)`	`'wisc' *`
`word_ext('www.cs.wisc.edu', '.', 5)`	`NULL`
`word_ext('aaa+=bbb+=ccc+zz=dd', '+=', 4)`	`'zz' **`
`word_ext(',,,,,aaa,,,,bb,,,c ', ',', 2)`	`'bb' ***`

* A negative word number means count from right to left

**If 2 separators are specified (+=), the function looks for either one.

***This function skips consecutive delimiters.

workflow_name

Returns the name of the current work flow.

SYNTAX

```
workflow_name()
```

RETURN VALUE

varchar

DETAILS

In cases where several work flows enclose this function, the function returns the name of the inner most work flow. If no work flow is found, the function returns the job name.

EXAMPLE

```
print('Work Flow Name: [workflow_name()]')
```

year

Determines the year in which the given date falls.

SYNTAX

`year(`*date1*`)`

RETURN VALUE

int	The number that represents the year component of *date1*.

WHERE

date1	The source date.

EXAMPLE

Function	Results
`year(to_date('Jan 22, 1997','mon dd, yyyy'))`	1997
`year(to_date('03/97', 'mm/yy'))`	1997
`year(to_date('03/19', 'mm/yy'))`	2019

About procedures

Data Integrator supports the use of stored procedures for Oracle, Microsoft SQL Server, Sybase, and DB2 databases. You can call stored procedures from the jobs you create and run in Data Integrator

This section discusses:

Overview

A stored procedure is a generic term used to describe an executable object, or a named entity that is stored in a database and can be invoked using input and output parameters. Generally, a stored procedure is one or more precompiled SQL statements. By calling a stored procedure from within Data Integrator, you can invoke business logic you have already coded thus enabling you to quickly and conveniently develop data extraction and data management tasks. Stored procedures can also be used to:

- Maintain business logic rules and provide a single point of control to ensure rules are accurate and enforced
- Significantly reduce network overhead with client/server applications because:
  - Procedures are stored on the database server

- Compiled execution plans for procedures are retained in the data dictionary

Data Integrator supports stored procedures for Oracle, MS SQL Server, Sybase and DB2 databases. Data Integrator also supports stored functions and packages for Oracle databases. Queries, scripts, conditionals, and custom functions can all be configured to include stored procedures, stored functions, and packages.

Stored procedures must exist in the database before you can use the procedure in Data Integrator. Create a stored procedure in a database using the client tools provided with the database, such as Oracle SQL *Plus. After it is created, a stored procedure can be called by users who have execution privileges for the procedure. After they are imported into Data Integrator, stored procedures can be used like functions in Data Integrator jobs.

Stored procedures include parameters. Each parameter has a name, a data type, and a mode (`IN`, `INOUT`, or `OUT`). A stored procedure can use a NULL or default parameter value for its input and can produce more than one output parameter value.

Requirements

To use stored procedures with Data Integrator, the following requirements must be met:

- The client and server versions must match.
- Only user-defined stored procedures can be used. Data Integrator does not support stored procedures provided by a database system.
- User-defined stored procedures or stored functions must meet the following additional requirements:
  - The return type (the data type of the result value) must be a Data Integrator supported data type, such as VARCHAR, DATE, or INT.

    NOTE: This release does not support the LONG data type in stored procedures.

  - The name of the stored procedure—the combination of the datastore name, owner name, and procedure name—must be unique. Data Integrator only imports the first procedure or function with a particular name.

    Using Oracle for example, if you have multiple procedures or functions with the same name in a package and want to use all of them, you must rename the procedures, giving each a unique name. A procedure is overloaded when multiple versions exist in a particular package. Overloading is not supported in Data Integrator.

- Data Integrator validates the user name specified in the datastore which was used to import the stored procedure. This ensures that the data flow that calls a stored procedure enforces restrictions associated with that procedure, such as:
  - Data input restrictions
  - Execution privileges

Creating stored procedures in a database

This section provides:

- An example scenario for using a stored procedure
- Tips for creating stored procedures on each database
- Example syntax for Oracle, SQL Server, Sybase, and DB2 databases based on the example scenario

Since syntax varies between databases, the SQL statements needed to create a stored procedure for any scenario vary. The client tools used to pass the SQL statements to the database server also vary. Refer to your database documentation for more detailed descriptions and examples regarding how to create stored procedures.

In the following example, the source database has a stored procedure `Get_emp_rec` that retrieves an employee's name and hire date from the Employee table using a given employee number. The stored procedure takes an input parameter `Emp_number` and returns two values `Emp_name` and `Hire_date` via output parameters.

The schema of the database table Employee is as follows:

Employee table schema

Name	Null?	Type
empno	Not Null	Integer
ename	Null	Varchar(20)
job	Null	Varchar(9)
mgr	Null	Integer
hiredate	Null	Date
sal	Null	Decimal(7,2)
comm	Null	Decimal(7,2)
deptno	Null	Integer

Creating stored procedures in Oracle

Oracle supports both stored procedures and stored functions. Any stored procedure that returns a value is called a stored function. Oracle is the only database to allow return values with data types other than an integer.

In Oracle, stored procedures are created using the `CREATE [OR REPLACE] PROCEDURE` statement, and stored functions are created using the `CREATE [OR REPLACE] FUNCTION` statement. The `OR REPLACE` option allows you to override an existing definition of a procedure or function with the same name.

Oracle supports IN, OUT, and INOUT parameters. An INOUT parameter allows you to pass in a parameter, modify it, and return the modified value. You can use the DEFAULT keyword or the assignment operator to give an IN parameter a default value. When an IN parameter has a default value, you can omit the parameter from the argument list when you call the procedure. If you do specify an argument value in the call, the specified value overrides the default value. INOUT and OUT parameters must be specified.

Example Syntax

The Oracle syntax for the stored procedure described in on page 493 is:

```
CREATE OR REPLACE PROCEDURE
get_emp_rec (Emp_Number IN NUMBER,
Emp_Name OUT VARCHAR, Emp_Hiredate OUT DATE) AS

BEGIN

SELECT ename, hiredate
INTO Emp_Name, Emp_Hiredate
FROM Employee
WHERE empno = Emp_Number;

END;
/
```

In this example, the parameters are declared as IN and OUT, but Oracle also supports an INOUT parameter type, which allows you to pass in a parameter, modify it, and return the modified value. INPUT parameters can take default values. This means that the parameter can be omitted from the actual parameter list when you call the procedure. INOUT and OUT parameters must be specified.

An Oracle package is an encapsulated collection of related program objects (e.g., procedures, functions, variables, constants, cursors, and exceptions) stored together in the database. Data Integrator allows you to import procedures or functions created within packages and use them the same way as top-level procedures or functions.

Data Integrator does not support overloading of procedures or functions. Overloading a procedure means creating multiple procedures with the same name in the same package, each taking arguments of a different number or datatype. If you have multiple procedures or functions with the same name in the same package (and you want to use all of them), you need to rename the procedures or functions so that they all have distinct names. Otherwise, Data Integrator will only import the first procedure or function by that name.

Creating stored procedures in MS SQL Server or Sybase

In Microsoft SQL Server and Sybase, a stored procedure may have a group number (between 1 and 32767), which is an optional integer used to group procedures of the same name so they can be dropped together with a single `DROP PROCEDURE` statement.

The first procedure you create with a name automatically is assigned group number 1. While you can execute this procedure without specifying the group number, other procedures with the same procedure name must be called with the group number. For example, `execute orderproc;2` and so on. Data Integrator imports the procedure with the group number 1, i.e., the first procedure, unless you specify the name including the group number, such as `orderproc;2`.

A parameter can be either an input parameter (default) or an output parameter. A parameter can not be both input and output. In the following sample, the first parameter `@Emp_Number` is an input parameter and the other two parameters, `@Emp_Name` and `@Emp_Hiredate`, are output

parameters. Output parameters are specified using the keyword OUTPUT. Input parameters can take default values (database defaults). If a default value is defined, the procedure can be executed without specifying a value for that parameter. You specify a parameter name with "@" as the first character.

An optional integer return value is automatically added to each stored procedure to store its return status. In the following example, if the SELECT statement is successful, "0" is returned as the return value. The parameter name of the return value is RETURN_VALUE. Microsoft SQL Server and Sybase stored procedures can be used like Oracle stored functions in Data Integrator because they include this return value.

A user defined function is a new feature of SQL Server 2000. Data Integrator supports user defined stored procedures not user defined functions at this time.

Example

The MS SQL Server and Sybase syntax for the stored procedure described in on page 493 is:

```
CREATE PROCEDURE GET_EMP_REC @Emp_Number int,
@Emp_Name varchar(20) OUTPUT, @Emp_Hiredate
datetime OUTPUT AS

SELECT @Emp_Name = ename, @Emp_Hiredate =
hiredate
FROM Employee
WHERE empno = @Emp_Number
RETURN 0
```

Creating stored procedure in DB2

In DB2, a parameter can be IN, OUT or INOUT. Any SQL data type can be used as a data type of a parameter. User defined data types are not supported.

Example

The DB2 syntax for the stored procedure described on page 493 is:

```
CREATE PROCEDURE GET_NAME_USING_ID (IN NID
INTEGER, OUT outVar varchar(20))

language SQL

reads sql data

BEGIN

select first_name into outVar FROM CONTACT
where id = NID;

END
```

`Reads sql data` and `language SQL` are two of the many options that DB2 stored procedures support. Refer to the DB2 documentation for the DB2 stored procedure options and their meaning.

Importing metadata for stored procedures

Use the **Import by Name** command to import stored procedures into Data Integrator. You must know the name of the stored procedure, stored function, or package you want to import. Use the name of the stored procedure, the package name only, or, if you are using an MS SQL Server or Sybase database, use the name of the stored procedure and the group number, such as `orderproc;2`. See "To import by name" on page 93 of the *Data Integrator Designer Guide* for a step-by-step procedure.

Stored procedures are created and structured very much like functions. Imported stored procedures appear on the **Datastores** tab of the object library. Each is listed in the **Functions** category of the database datastore from which the procedure was imported. It is displayed with the following syntax:

<datastore>.<owner>.<proc_name>.

If a stored procedure was defined within an Oracle package, the name of the procedure is displayed as <datastore>.<owner>.<package>.<proc_name>.

To view the signature of the procedure, right-click the imported stored procedure and select **Open** (or simply double-click the procedure).

- A stored procedure has a signature that consists of its defined parameters including two output parameters added by DATA INTEGRATOR during import, AL_SP_RETCODE and AL_SP_ERRMSG. For more information about AL_SP_RETCODE and AL_SP_ERRMSG, see "Checking execution status" on page 509.
- If the database you are using provides a return code for a stored procedure, it is displayed in the signature as AL_SP_RETURN.
- Each parameter has a name, a data type, and a mode (IN, INOUT, or OUT). The mode is indicated by icons shown before the parameter name.
- Oracle stored functions have a signature that consists of parameters and a return data type. When you call a function, the return value is named AL_SP_RETURN.

NOTE: If you change the signature of a stored procedure, for example you add a new parameter, re-import the metadata for the procedure from the database.

Structure of a stored procedure

Imported information for a stored procedure includes:

- Name and owner
- Return data type if the stored procedure has a return value or code
- Parameters

For each parameter, Data Integrator imports the following information:

- Name
- Mode (`IN`, `OUT`, or `INOUT`)
- data type (length, precision and scale if applicable)

  NOTE: In an Oracle stored procedure declaration, it is illegal to constrain CHAR and VARCHAR2 parameters with a length, or to constrain NUMBER parameters with a precision and/or scale. These constraints are taken from the procedure arguments. Therefore, the length of a CHAR or VARCHAR and the precision and scale of a NUMBER data type parameter are not stored in the data dictionary.

  However, Data Integrator does not allow the length of a VARCHAR data type or the precision of a NUMBER data type to be 0. (Data Integrator converts CHAR data types in Oracle to VARCHAR.) When importing a stored procedure, Data Integrator fills in the default length for a VARCHAR or the precision and scale for a NUMBER parameter. For VARCHAR, Data Integrator sets the length to 256. For NUMBER, Data Integrator sets precision to 28 and scale to 6. At runtime, the length of VARCHAR and the precision and scale of NUMBER come from the procedure arguments.
- Position within the parameter list
- Nullable (whether a default value exists)

Data Integrator omits a parameter if any of the following data type conditions is true:

- Is not a Data Integrator supported data type or is LONG (see Chapter 4, "Data Types," in the *Data Integrator Reference Guide*)
- Data type is missing in the database dictionary
- Is a composite data type (e.g., table, record, or cursor)

After Data Integrator imports a stored procedure, the Designer reports warnings about any omitted parameters.

NOTE: Even if a stored procedure has an omitted parameter, the procedure is callable if the parameter has a default value defined. If the omitted parameter does not have a default value, the procedure is not callable and Data Integrator throws a runtime error.

If a stored procedure has a return value, its result type (the datatype of the result value) must be an Data Integrator supported database server data type. Otherwise, the procedure cannot be imported and used within Data Integrator applications.

For each stored procedure, Data Integrator automatically adds two optional output parameters, `AL_SP_RETCODE` and `AL_SP_ERRMSG`. These parameters allow you to check the execution status of the stored procedure. See "Checking execution status" on page 509.

Calling stored procedures

This section discusses calling stored procedures:

- In general
- From queries
- Without the function wizard

In general

You can call a stored procedure just as you call custom functions you create in Data Integrator.

- When you call a stored procedure inside a script, conditional or custom function, Data Integrator evaluates the call as an independent SQL statement.
- When you call a stored procedure in a query's column mapping or WHERE clause, (Oracle only) Data Integrator can combine the SQL for the stored procedure with the SQL for the query (see "From queries" on page 504).
- You can also call a stored procedure in a SQL transform. You must use proper syntax. Data Integrator does not validate SQL in a SQL transform.
- A stored procedure can be used within an expression. For example:

```
ora_ds.acta_user.get_emp_sal
(EMPLOYEE.NAME) + 10
ora_ds.acta_user.get_emp_sal
(EMPLOYEE.NAME) < 100
```

Stored procedures can require input values for some parameters but not others. You must determine the requirements of the stored procedure and prepare the appropriate inputs. In addition, Data Integrator supports the following:

- Input parameters can be constants or expressions
- If an input parameter has a default value, you can omit the parameter in the call by leaving the parameter unspecified in the function wizard. The database will use the parameter's default value when evaluating the stored procedure call.
- All `OUT` or `INOUT` parameters must be specified except the two Data Integrator adds at import AL_SP_ERTCODE and AL_SP_ERRMSG.

- All OUT or INOUT parameters must be variables (except in a query output schema; see "Query output schemas" on page 505). A variable must be declared before use.
- If a stored procedure returns a value (such as an Oracle function or a MS SQL Server or Sybase stored procedure), you can use the procedure in an expression.
- You can call a stored procedure as a step inside: an output schema of a query, script, or custom function. For example:

  As an Oracle user you create two Oracle stored procedures: GET_EMP_SALARY and SET_EMP_SALARY

```
CREATE OR REPLACE FUNCTION
GET_EMP_SALARY(Emp_Name IN VARCHAR)

RETURN NUMBER AS

Return_value NUMBER;

BEGIN

SELECT sal
INTO Return_value
FROM Employee
WHERE ename = Emp_Name;
RETURN Return_value;

END;

CREATE OR REPLACE PROCEDURE
SET_EMP_SALARY(Emp_Name IN VARCHAR,Emp_Sal
IN NUMBER) AS

BEGIN

UPDATE Employee
SET sal = Emp_Sal
WHERE ename = Emp_Name;

END;
```

After importing the stored procedures into an Oracle datastore, you can use the stored procedures in a script. For example:

```
$new_sal =
ora_ds.bodi_user.get_emp_salary('MARTIN') +
100;

ora_ds.bodi_user.set_emp_salary('MARTIN',
$new_sal);
```

From queries

You can call stored procedures from Query transforms. In queries, you can use stored procedures in:

- Mapping and WHERE clauses
- Query output schemas

Mapping and WHERE clauses

In a query's column mapping or WHERE clause, you can use a stored procedure in an expression, subject to the same constraints as other expressions (see "In general" on page 502). However, when stored procedures are used in a query's column mapping or WHERE clause, Data Integrator can combine the SQL statement for the stored procedure call with the SQL statement for the query and submit one request to the database.

For Data Integrator to combine the SQL statements, the stored procedure must meet the following conditions:

- It must be from an Oracle database
- It must be a stored function.
- All its parameters must be IN parameters; none can be an OUT or INOUT parameter.
- All parameter values must be specified. You cannot leave a parameter unspecified, such as to use the default value.

- All parameter values must be constants or expressions with only constants or database table column names; none can have Data Integrator functions or operations.
- The data types of the procedure's parameters and the return type (the data type of its result value) must be Data Integrator supported data types. Data Integrator must not omit any of the parameters when importing the procedure.
- Data Definition Language (DDL) operations like creating tables, views etc. and transaction control statements such as `COMMIT`/`ROLLBACK` cannot be performed inside the stored procedure.

In addition, when importing the function, you must select the **Callable from SQL expression** check box on the Import Function By Name window (see "Importing metadata for stored procedures" on page 498).

Data Integrator will push down a stored procedure call to Oracle if the first five conditions are meet and the **Callable from SQL expression** check box is selected. If a stored procedure call contains any DDL statements, it ends the current transaction with `COMMIT` or `ROLLBACK`, or it issues any `ALTER SESSION` or `ALTER SYSTEM` commands. Therefore, you should deselect the **Callable from SQL expression** option on the Import Function By Name window to ensure that the function call will not be pushed down to database.

When a stored procedure call cannot be pushed down to database, it is submitted to database as a separate SQL statement.

Query output schemas

When a stored procedure can provide at least one return value or has any `INOUT` or `OUT` parameters other than AL_SP_RETCODE or AL_SP_ERRMSG, you can call the procedure by including it as an object in the output schema of a query.

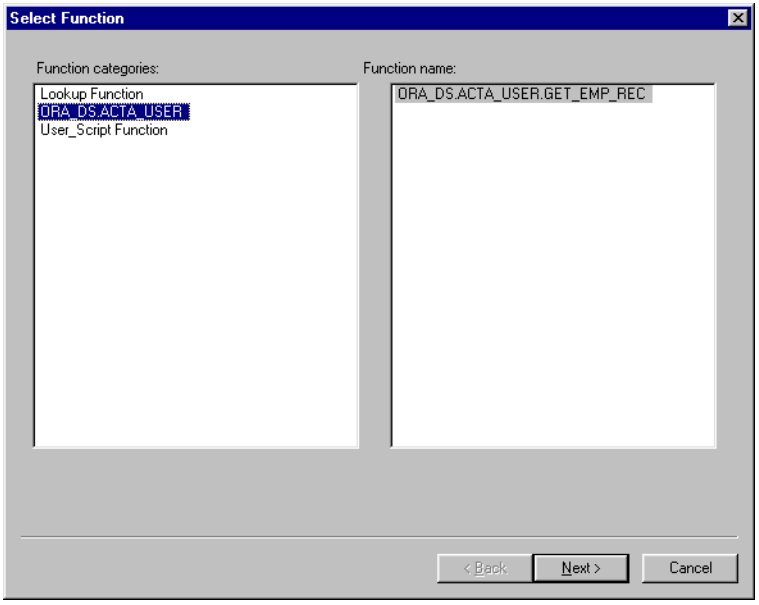
When using a stored procedure in a query output schema, you must map `INOUT` or `OUT` parameters and return values to a column of the output schema. In this case, you cannot map these parameters to variables.

➢ To include a stored procedure in a query output schema

1. In a data flow diagram, click a query name to open the query editor.
2. In the query editor, right-click the query name in the output schema and select **New Function Call**.

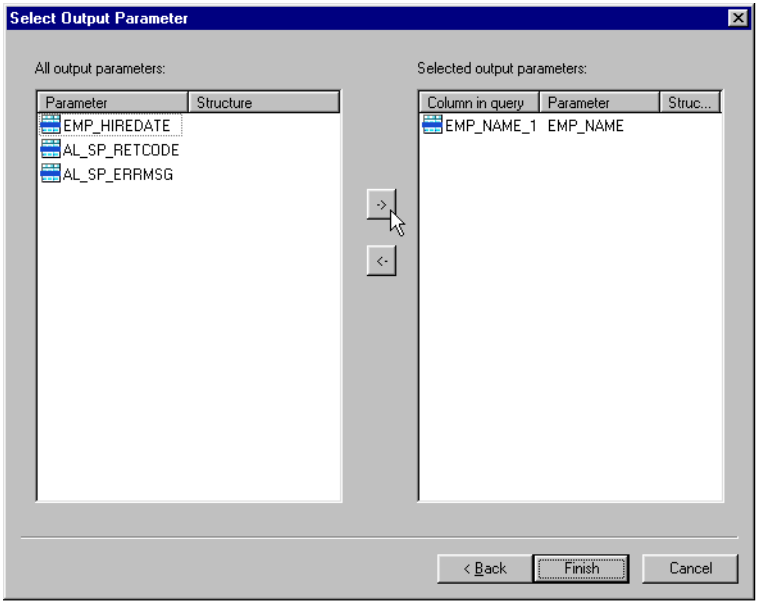
   Alternately, you can right-click an existing stored procedure object in the output schema area of the query editor and select **Modify Function**.

   Data Integrator provides a function wizard to help you include input and output parameters and the return value for a procedure call. The imported stored procedures are listed for each datastore on the first page of the function wizard.



3. Select a function category. Notice that the function names change depending on the category you select.
4. Under **Function name**, select a stored procedure, then click **Next**.
5. In the Define Input Parameter(s) window, enter the input parameters for the stored procedure, then click **Next**.
6. In the Select Output Parameter window, select the procedure's output parameters that you want to map to the query output.

   Under **All output parameters**, select the parameters you want to map, and click the arrow key to move them under **Selected output parameters**.



   You can map more than one `INOUT` or `OUT` parameter and the return value as output columns from a single procedure call.

7. Click **Finish**.

Data Integrator adds those parameters to the query's output schema. In this example, the output schema of the Query transform would have two columns, `EMP_NAME` and `EMP_HIREDATE`.

Without the function wizard

You can enter an existing stored procedure call on the **Mapping** tab or the **Where** tab in a query editor.

When entering a stored procedure call without using the function wizard:

- Match the parameter values you enter to the signature of the stored procedure in Data Integrator. Values must be specified in the same order that the parameters are defined.
- Use `AL_UNSPECIFIED_PARAM` in place of a missing parameter, such as when you want to use the default value for an `IN` parameter.
- Use the correct naming convention for the stored procedure:
  - `<datastore>.<owner>.<proc_name>`
  - `<datastore>.<owner>.<package>.<proc_name>`.

  If the name is case-sensitive in the database (and not all uppercase), enter the name as it appears in the database and use double quotation marks (") around the name to preserve the case.

The two output parameters added by Data Integrator during import, `AL_SP_RETCODE` and `AL_SP_ERRMSG`, are optional. You do not need to provide arguments for these two parameters if you are not interested in the values stored in them. See "Checking execution status" on page 509 for more information.

Checking execution status

To allow you to monitor the execution status of a stored procedure, Data Integrator adds two parameters to the signature of each stored procedure when importing the procedure:

- AL_SP_RETCODE, varchar(256). There are three possible values:
  - `ACTA_SP_OK` — The stored procedure call succeeded.
  - `ACTA_SP_DB_CONN_EXCEPTION` — The database connection cannot be created because of a connection error, invalid user, password, or host name.
  - `ACTA_SP_CALL_ERROR` — The connection completes but the call fails in the database. Details for the cause of the error are available in AL_SP_ERRMSG.
- `AL_SP_ERRMSG`, varchar(1024)

Data Integrator adds these two parameters to the end of the signature, following any user-defined parameters.

With these parameters, there are two techniques you can use to handle errors if a stored procedure fails during execution:

- Throw an error that stops the job immediately. An error message is logged in the `errorlog.txt` file.

  To use this method, do not map as output parameters the two parameters DATA INTEGRATOR adds to imported stored procedures.
- Save the Oracle error messages into the `AL_SP_ERRMSG` parameter. This method allows the query, script, conditional, or custom function that contains the stored procedure call to continue its execution.

To use this method, define parameters for AL_SP_RETCODE and AL_SP_ERRMSG. Use the execution status stored in AL_SP_RETCODE to control your application logic. For example, if you store the value of AL_SP_ERRMSG in a variable, you can use the print function to print the variable to the job log. With this method, the error message is not logged in the errorlog.txt file.

7 Data Integrator Scripting Language

This chapter describes the Data Integrator scripting language. You can use the language to write scripts and custom functions (also known as user-script functions). The scripting language can also be used to write expressions such as complex column mapping expressions and WHERE clause conditions.

The language is also used internally by Data Integrator to save the objects you create into repository tables. The saved objects can be exported to a file (.atl) and later imported to another repository. This technique is commonly used when migrating to a new environment or when backing up a repository before initiating an upgrade.

➢ To use this language

1. Open an editor for an object in the Designer workspace such as a script, conditional, or transform.
2. Inside any of these editors, use the smart editor to enter a script or expression using the language syntax described in this chapter.

This chapter contains the following sections:

Language syntax

With the DATA INTEGRATOR scripting language, you can assign values to variables, call functions, and use standard string and mathematical operators. The syntax used in the Data Integrator scripting language can be used in an expression as well as in a script.

Syntax for statements in scripts

Jobs and work flows can use scripts to define detailed steps in the flow of logic. A script can run functions and assign values to variables, which can then be passed to other steps in the flow.

Statements in a script object or custom function must end with a semicolon (`;`). Comment lines must start with a # character.

To view examples of scripts, see "Sample scripts" on page 525.

Syntax for column and table references in expressions

Expressions are a combination of constants, operators, functions, and variables that evaluate to a value of a given data type. Expressions can be used inside script statements or added to data flow objects. Because expressions can be used inside data flow objects, they often contain column names.

The Data Integrator Scripting Language recognizes column and table names without special syntax. For example, you can indicate the `start_date` column as the input to a function in the **Mapping** tab of a query as follows:

```
to_char(start_date, 'dd.mm.yyyy')
```

The column `start_date` must be in the input schema of the query.

If there is more than one column with the same name in the input schema of a query, indicate which column is included in an expression by qualifying the column name with the table name. For example, indicate the column `start_date` in the table `status` as follows:

```
status.start_date
```

Note that column and table names as part of SQL strings may require special syntax based on the DBMS that the SQL is evaluated by. For example, select all rows from the `LAST_NAME` column of the `CUSTOMER` table as follows:

```
sql('oracle_ds','SELECT CUSTOMER.LAST_NAME
   FROM CUSTOMER')
```

Because the column name is all upper case in Oracle, no special syntax is required. However, select all rows from the `start_date` column in the Oracle table `Status` as follows:

```
sql('oracle_ds','SELECT "Status"."start_date"
   FROM "Status"')
```

The table name and column are defined in Oracle to have a special case. To preserve this case, enclose the identifiers in double quotation marks.

Strings

This section contains the following information about strings:

- Quotation marks
- Escape characters
- Trailing blanks

For information about NULLs and empty strings, see "NULL values and empty strings" on page 519.

Quotation marks

The type of quotation marks to use in strings depends on whether you are using identifiers or constants:

Identifier	Name of an object (for example, table, column, data flow, or function).
Constant	A fixed value used in computation. There are two types of constants: • String constants (for example, 'Hello World' or '1995.01.23') • Numeric constants (for example, 2.14)

Identifiers need quotation marks if they contain special (non-alphanumeric) characters. For example, you need a double quote for `"compute large numbers"` because it contains blanks.

Use single quotes for string constants.

Escape characters

If a constant contains a single quote (`'`) or backslash (`\`) or another special character used by the Data Integrator scripting language, then those characters need to be escaped. For example, the following characters must be preceded with an escape character to be evaluated properly in a string. Data Integrator uses the backslash (\) as the escape character.

Character	Example
`single quote (')`	`'World\'s Books'`
`backslash (\)`	`'C:\\temp'`

Trailing blanks

Data Integrator strips trailing blanks from strings. To include a blank at the end of a string, concatenate a string with a single blank to the end of the string, or include the blank at the beginning of the following string. For example, to leave space between text and a variable name, construct the string as follows:

```
'The name of the data flow is:' || ' ' ||
$DataFlowName
```

where the double bar (||) is the concatenation operator.

Variables

Variable names must be preceded by a dollar sign ($).

- Local variables used in a script or expression must be defined in the job or work flow that calls the script using the Variables and Parameters window. See Chapter 12, "Variables and Parameters," in the *Data Integrator Designer Guide*.
- Global variables used in a script or expression must be defined at the job level using the Variables and Parameters window.
- The return value must be passed outside the function using the following statement:

  `RETURN`(*expression*)

  where *expression* defines the value to be returned.
- Local variables used in a custom function must be defined using the smart editor. For more information see, "To create a custom function" on page 333.
- Existing variables and parameters displayed in the smart editor are filtered by the context from which the smart editor is opened.

Variable interpolation

You can embed expressions within constant strings, and Data Integrator will evaluate the variables and substitute the value into the string—the concatenation operator (||) is not required.

Here is an example using the concatenation operator:

```
$st_date=sql('warehouse_ds','SELECT
max(start_timestamp)
   FROM dw_process_log
   WHERE table_name=’\"||$Table_Name||'\"
print('The value of the start date
   is:'||sysdate()+5);
```

The statement can be simplified to the following:

```
$st_date=sql('warehouse_ds','SELECT
max(start_timestamp)
   FROM dw_process_log
   WHERE table_name={$Table_Name}'
print('The value of the start date
   is:[sysdate()+5]');
```

Curly braces ({}) and square brackets ([]) enclose the embedded expressions:

- The square brackets ([]) indicate that the value of the expression should be substituted.
- The curly braces ({}) indicate that the value of the expression should be quoted with single quotation marks.

Strings including curly braces or square brackets cause a processing error. You can avoid the error by preceding the braces or brackets with a backslash (\).

Functions and stored procedures

A script, expression or custom function can call most built-in or custom functions. A function cannot call itself or call another function that would lead to a recursive call. For example, function A cannot call function A, nor can function A call function B if function B calls function A.

You can also use functions and stored procedures imported from a DBMS. Use the exact case for names of imported functions and stored procedures

- Built-in functions are listed in the function wizard and smart editor grouped by category.
- Custom functions are listed in the function wizard and smart editor under the **User_Script** node.

  To create a custom function, see "To create a custom function" on page 333.
- Imported functions and stored procedures are listed in the function wizard and smart editor under the datastore used to import them.

  To import a function see, "Importing metadata for stored procedures" on page 498.

Operators

The operators you can use in scripts and expressions are listed in the following table, in order of precedence. Note that when operations are pushed to a DBMS to perform, the precedence is determined by the rules of the DBMS.

Operator	Description
+	Addition
–	Subtraction
*	Multiplication
/	Division
=	Assignment, comparison
<	Comparison, less than
<=	Comparison, less than or equal to
>	Comparison, greater than
>=	Comparison, greater than or equal to
!=	Comparison, not equal to
||	Concatenate
AND	Logical AND
OR	Logical OR
NOT	Logical NOT

When you use the smart editor to add operators, you can type in operators or use the built-in key pad. Access the key pad from the smart editor's tool bar.

NULL values

Indicate NULL values using the keyword NULL. For example, you can check whether a column (COLX) is null or not:

```
COLX IS NULL

COLX <> NULL
```

Data Integrator does not check for NULL values in data columns. Use the function `nvl` to remove NULL values (see "nvl" on page 444).

NULL values and empty strings

Data Integrator uses the following two rules with empty strings:

- When you assign an empty string to a variable, Data Integrator treats the value of the variable as `NULL`.
- As a constant (`''`), Data Integrator treats the empty string as `NULL`.

Data Integrator uses the following three rules with `NULLs` and empty strings in conditionals:

- When you compare against a `NULL` constant (`NULL` or `''`), it is treated as a test for `NULL`. This includes comparing a variable that has a value of `NULL` against a `NULL` constant.

  For example, assuming a variable is assigned: `$var1 = '';`

Condition	Translates to	Returns
`If (NULL = NULL)`	`NULL` is `NULL`	`TRUE`
`If (NULL != NULL)`	`NULL` is not `NULL`	`FALSE`
`If ('bbb' = '' )`	`bbb` is `NULL`	`FALSE`
`If ('bbb' != '')`	`bbb` is not `NULL`	`TRUE`
`If ($var1 = NULL)`	`NULL` is `NULL`	`TRUE`
`If ($var != NULL)`	`NULL` is not `NULL`	`FALSE`

- When you compare against a variable that has the value of `NULL`, it is treated as a comparison against `NULL`. The result is indeterminate which means it is not `TRUE` so the `FALSE` branch will execute. For example, assuming a variable is assigned: `$var1 = '';`

Condition	Translates to	Returns
`If($var1='bbb')`	`bbb = NULL`	`FALSE`
`If ($var1 != 'bbb')`	`NULL != NULL`	`FALSE`

- When comparing two variables, always test for NULL. In this scenario, you are not testing a variable with a value of NULL against a NULL constant (as in the first rule). Either test each variable and branch accordingly or test in the conditional as shown in the second row of the following table.

  For example, assuming two variables are assigned as follows:

  ```
  $var1 = '';

  $var2 = '';
  ```

Condition	Recommendation
`if($var1 = $var2)`	Do not compare without explicitly testing for NULLs. Business Objects does not recommend using this logic because any relational comparison to a NULL value returns FALSE.
`if ( (($var1 = NULL) AND ($var2 = NULL)) OR ($var1 = $var2))`	Will execute the TRUE branch if both $var1 and $var2 are NULL, or if neither are NULL but are equal to each other.

Debugging and Validation

If you are using the smart editor to create a script or a custom function, select the **Validate** icon in the tool bar (or right-click and select **Validate**). Errors are listed in the smart editor window under the text box. Double-click each error to see where it occurred in your script.

Otherwise:

- Select the **Debug > Validate > Current View** command to check the syntax of expressions used in the definition of the current object. Errors are displayed in the Output window.
- Select the **Debug > Validate > All Objects in View** command to check the syntax of expressions used in the current object definition and the definitions of all objects called by the current object. Errors are displayed in the Output window.

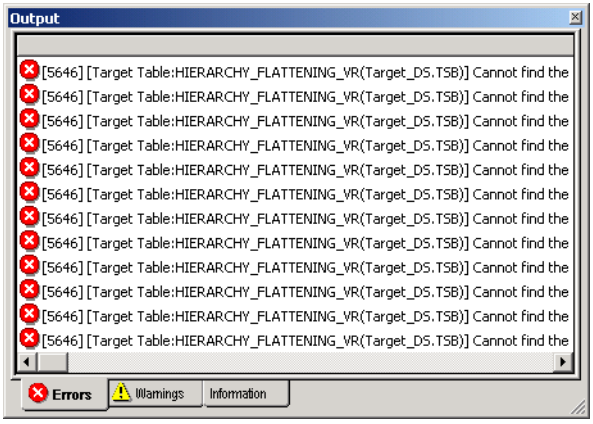
By default, Data Integrator lists ten errors before it aborts parsing. By reporting ten errors for each validation, Data Integrator allows you to shorten the edit-validate-test cycle. You can repair several errors in each iteration of the cycle.

If you want to change this default setting, select **Tools** > **Options** > **Job Server** > **General** and set the options as follows:

Section	Enter "Parser"
Key	Enter "NumErrors"
Value	Enter a positive number

For each error, Data Integrator provides a description of the problem.

Data Integrator also provides three menu options for errors.



Right-click an error to use these menu options:

View	The **View** option allows you to view the error text in a smaller, separate window for easy reading.

Go To Error	The **Go to Error** option places an error icon near the line that caused the error. You can also double-click the error to place this icon. This option not available for all errors, however.
Copy	The **Copy** option copies the error text on to the clipboard so you can place it elsewhere.

When you execute a job, you may still encounter errors in an expression that are returned from the operating system or the DBMS. When executing a job, Data Integrator sends as many operations as possible to the DBMS to evaluate. The DBMS may evaluate part of the expression and return a value that contributes to evaluating the rest of the expression.

Keywords

The following sections describe the keywords in the scripting language. To view the list of reserved words, see "Reserved Words" on page 575.

Keywords are listed in the selection list of the smart editor filtered by the context from which the smart editor is opened.

BEGIN

Indicates the beginning of the code that becomes the function, script, or other construct. BEGIN and END statements are added automatically to function, transform, and script definitions.

CATCH

Indicates a catch for a try/catch block. If an error occurs while executing any of the statements between the TRY and the CATCH statements, Data Integrator executes the statements defined by the CATCH. Use the CATCH keyword as shown below or use `CATCH(ALL)`.

```
BEGIN
TRY
   BEGIN
      script_step;
      script_step;
   END
CATCH (exception_number)
   BEGIN
      catch_step;
      catch_step;
   END
CATCH (exception_number)
   BEGIN
      catch_step;
      catch_step;
   END
END
```

ELSE

Defines the second branch for an IF statement. If no ELSE follows the IF, no action is taken if the condition is not met.

END

Indicates the end of the code that becomes the function, script, or other construct. BEGIN and END statements are added automatically to function, transform, and script definitions.

IF

Indicates a conditional step in the code. An IF statement can be constructed with or without an ELSE step. Use the IF keyword as follows:

```
IF (condition) script_step; ELSE script_step;
```

or

```
IF (condition) script_step;
```

where *condition* is an expression that evaluates to True or False. *script_step* indicates the set of instructions to execute based on the value of *condition*. If *script_step* contains more than one statement, enclose these statements in BEGIN and END statements. An example of this syntax is shown under "RepeatString function" on page 525.

RETURN

Indicates the value to be returned by a function. Use the RETURN keyword as follows:

```
RETURN (expression);
```

where *expression* defines the value to be returned.

TRY

Indicates the start of a try/catch block. For more information, see "CATCH" on page 522.

WHILE

Defines a set of statements to execute until a condition evaluates to FALSE. Use the WHILE keyword as follows:

```
WHILE (condition) script_step;
```

where *condition* is an expression that evaluates to True or False. *script_step* indicates the set of instructions to execute based on the value of *condition*. If *script_step* contains more than one statement, enclose each statement in BEGIN and END statements.

Sample scripts

The following examples show how a script would be formatted for the Square and Repeat String functions.

Square function

This function accepts an integer and returns the square of the input value. To define this script, supply the following values to Data Integrator:

Description	Value	Arguments
Function name	Square	
Return value	Return	int
Input value	input_int	int, input

The text of the function script is as follows:

```
RETURN($input_int * $input_int);
```

RepeatString function

This function accepts a one-character string and an integer indicating the number of times to repeat the input character. It outputs the created string or, if the input character or repeat number is NULL, it outputs a NULL value. It uses a variable to keep track of the string components as it is being generated.

To define this script, supply the following values to Data Integrator:

Description	Value	Arguments
Function name	RepeatString	
Return value	Return	VARCHAR (255),allow NULL
Input string	InputCharacter	VARCHAR (1), input, allow NULL
Input integer	RepeatNumber	INT, input, allow NULL
Internal variable	Partial	VARCHAR (255), allow NULL

The text of the function is as follows:

```
IF ($InputCharacter = NULL)
   BEGIN
   $Partial = NULL;
   RETURN NULL;
   END
IF ($RepeatNumber = NULL)
   RETURN NULL;
IF ($RepeatNumber > 255)
   $RepeatNumber = 255;
IF ($RepeatNumber < 0)
   $RepeatNumber = 0;
   BEGIN
   WHILE ($RepeatNumber != 0)
      BEGIN
      $Partial = ($Partial || $InputCharacter);
      $RepeatNumber = ($RepeatNumber - 1);
      END
   RETURN $Partial;
   END
```

8 Metadata in repository tables and views

Data Integrator provides full access to its repository metadata used by the applications you create. To access this metadata to analyze applications, either manually enter SQL SELECT statements or open the metadata reporting tool.

This chapter describes the tables and views in the repository that are useful for metadata analysis. This chapter also provides example SQL SELECT statements for creating reports using a SQL editor.

This chapter contains the following sections:

- Imported metadata
- Internal metadata
- Operational metadata

See Chapter 17, "Metadata Reporting," in the *Data Integrator Designer Guide* for information about using the metadata reporting tool to generate repository reports.

Imported metadata

This set of tables (prefixed by AL_) and views (prefixed by ALVW_) capture information about the metadata imported into Data Integrator from external databases and applications (such as Oracle, PeopleSoft, and SAP R/3).

AL_INDEX

This table contains index information for tables.

Column Name	Description
TABLEKEY	The table ID associated with the index. The TABLEKEY relates to table information in ALVW_TABLEINFO.
NAME	The index name.
COLNAME	The name of the table's index column.
COLPOSITION	Position of the column in the index.

The following query returns the index information associated with a table EMPLOYEE in datastore ORA_DS:

```
SELECT NAME, COLNAME, COLPOSITION
   FROM AL_INDEX, ALVW_TABLEINFO
   WHERE AL_INDEX.TABLEKEY =
         ALVW_TABLEINFO.TABLE_ID
      AND TABLE_NAME = 'EMPLOYEE'
      AND DATASTORE = 'ORA_DS'
```

AL_PCOLUMN

This table contains partition information for tables.

Column Name	Description
TABLEKEY	The partitioned table ID. The TABLEKEY relates to table information in ALVW_TABLEINFO.
COLNAME	The name of the table's partition column.
COLPOSITION	Position of the column in the partition.

AL_PKEY

This table contains primary key information for tables.

Column Name	Description
TABLEKEY	The table ID associated with a primary key. The TABLEKEY relates to table information in ALVW_TABLEINFO.
COLNAME	Name of the table's primary key column.
COLPOSITION	Position of the primary key column.

The following query returns the primary key associated with a table EMPLOYEE in datastore ORA_DS:

```
SELECT COLNAME, COLPOSITION
   FROM AL_PKEY, ALVW_TABLEINFO
   WHERE AL_PKEY.TABLEKEY =
         ALVW_TABLEINFO.TABLE_ID
      AND TABLE_NAME = 'EMPLOYEE'
      AND DATASTORE = 'ORA_DS'
```

ALVW_COLUMNATTR

This view contains information about column attributes.

Column Name	Data type	Description
TABLE_NAME	Varchar(64)	Name of the table.
TABLE_OWNER	Varchar(64)	Owner of the table.
DATASTORE	Varchar(64)	The datastore to which this table belongs.
COLUMN_NAME	Varchar(64)	Name of the column.
COLUMN_ATTR	Varchar(64)	Name of the attribute (property of this table). The name of the attribute will be the same as seen when viewing the attributes of a table in the Designer's interface. Examples: Description, Business_Name, Date_last_loaded, Date_created, Total_Number_Of_Rows_Processed.
COLUMN_ATTR_VALUE	Varchar(255)	Value of this attribute.

The following query returns all columns and their descriptions for table EMP in datastore HR:

```
SELECT COLUMN_NAME, COLUMN_ATTR_VALUE
   FROM ALVW_COLUMNATTR
   WHERE TABLE_NAME = 'EMP'
      AND COLUMN_ATTR = 'Description'
      AND DATASTORE = 'HR'
```

ALVW_COLUMNINFO

This view contains information about the columns in a table.

Column Name	Data type	Description
TABLE_NAME	Varchar(64)	Name of the table.
TABLE_OWNER	Varchar(64)	Owner of the table.
DATASTORE	Varchar(64)	The datastore to which this table belongs.
COLUMN_NAME	Varchar(64)	Name of the column.
COLUMN_DATATYPE	Varchar(20)	Data type of this column. Examples: `integer`, `datetime`, `decimal`, `real`.
COLUMN_LENGTH	Int	Length of this column in bytes.
COLUMN_PRECISION	Int	Precision of this column (applies only to columns with the data type `decimal`).
COLUMN_SCALE	Int	Scale for this column (applies only to columns with the data type `decimal`).
COLUMN_IS_NULLABLE	Int	One (`1`) indicates this column will accept NULL values, `0` indicates not.
COLUMN_ID	Int	The ID for this column within the repository. It can be used to join with other tables containing additional column information.

The following query returns all columns in table `EMP` in datastore `HR`:

```
SELECT COLUMN_NAME
   FROM ALVW_COLUMNINFO
   WHERE TABLE_NAME = 'EMP'
      AND DATASTORE = 'HR'
```

ALVW_FKREL

This view contains information about the primary and foreign key relationships among tables.

Column Name	Description
TABLEKEY	The table ID.
PKCOLUMN	Primary key column in this table.
FKTABLE	The table where this column is referenced.
FKCOLUMN	The column in the 'foreign' table which is the same as the primary column.

The following query returns the primary and foreign key information associated with table EMPLOYEE in datastore ORA_DS.

```
SELECT FKTABLE, FKCOLUMN, PKCOLUMN
   FROM ALVW_FKREL, ALVW_TABLEINFO
   WHERE ALVW_FKREL.TABLEKEY =
         ALVW_TABLEINFO.TABLE_ID
      AND TABLE_NAME='EMPLOYEE'
      AND DATASTORE='ORA_DS'
```

ALVW_MAPPING

The ALVW_MAPPING view joins the AL_COLMAP and the AL_COLMAP_TEXT tables and can be used for impact analysis. This view contains information about target tables and columns, the sources used to populate target columns, and the transforms Data Integrator applies to sources before applying them to targets. To ensure mappings are current, see "Refresh report content" on page 456 of the *Data Integrator Designer Guide*.

Column Name	Data type	Description
DF_NAME	varchar(64)	Data flow that populates the target table.
TRG_TAB_NAME	varchar(64)	Name of the target table.
TRG_TAB_ID	int	ID for this table within the repository.
TRG_TAB_DESC	varchar(100)	Description of the target table.
TRG_OWNER	varchar(64)	Owner of the target table.
TRG_DS	varchar(64)	Datastore of the target table.
TRG_TYPE	varchar(64)	Type of target. Examples: table, BW transfer structure.
TRG_USAGE	varchar(65)	Usage of the target table. Examples: fact, dimension, lookup. Currently set to NULL.
TRG_COL_NAME	varchar(65)	Column name in the target.
TRG_COL_ID	int	ID for this column in the repository.
TRG_COL_DESC	varchar(100)	Description of this column.
SRC_TAB_NAME	varchar(64)	Name of the source table used to populate the target.
SRC_TAB_ID	int	ID for this table within the repository.
SRC_TAB_DESC	varchar(100)	Description of the source table.
SRC_OWNER	varchar(64)	Owner of the source table.
SRC_DS	varchar(64)	Datastore of the source table.
SRC_TYPE	varchar(64)	Type of source. Examples: table, file.
SRC_COL_NAME	varchar(65)	Name of the source column.
SRC_COL_ID	int	ID for this column in the repository.
SRC_COL_DESC	varchar(100)	Description of this column.
MAPPING_TYPE	varchar(65)	Types of source to target mapping. Examples: direct, computed, lookup.
MAPPING_TEXT	varchar(255)	The expression used to map the source to the target column.

Example use case

The following query returns target tables and columns populated from the column EMPID in table EMP (in datastore HR):

```
SELECT TRG_TAB_NAME, TRG_COL_NAME
   FROM ALVW_MAPPING
   WHERE SRC_TAB_NAME = 'EMP'
      AND SRC_COL_NAME = 'EMPID'
      AND SRC_DS = 'HR'
```

Mapping types

The AL_COLMAP_TEXT table contains information qualifying the mapping relationships. This information, stored in the MAPPING_TYPE column, can have the following values:

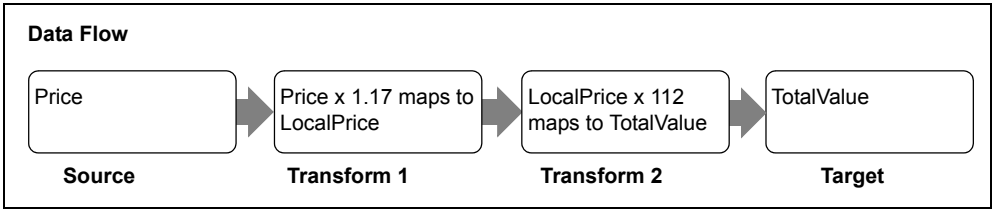
Mapping Type	Description
Direct	The target column is mapped directly from a source column with no expression to transform it. For example, EMPID (employee ID) mapped directly from source to target.
Computed	There is an expression associated with the target column. For example, NAME is LAST_NAME||','||FIRST_NAME.
Generated	There is no source column associated with the target column. For example, the target table is mapped to a constant or a function, such as sysdate, or is obtained from a transform, such as Date_Generation.
LookedUp	A lookup function is used in the expression.
Merged	Two data streams are merged to populate the target table. The two expressions mapped to the target table are separated by AND.
Not mapped	The column in the target table is not being populated by the data flow.
Unknown	Data Integrator is unable to identify the expression used to map the target column. This happens only under unusual error conditions.

How mappings are computed

When a data flow processes information, it performs potentially complex transformations on data in preparation for loading it into one or more target tables. Typical operations include:

- Reading data from the appropriate sources
- Processing data using query transforms or other transforms
- Splitting the data stream and then merging it again

Consider the following example, where two transformations operate against a value from one column of a source table.



The information is captured in AL_COLMAP_TEXT as follows:

Target column	Source column	Mapping expression
Total_value	Price	`((Price x 1.17) x 112)`

This kind of information becomes more valuable as transformation complexity increases. Consider the following example:

- Data flow DF_1 reads three columns (a, b, c) from source table S.
- Table S is connected to a query transform Q1.
- The query transform output has four columns (Qa, Qb, Qc, and Qd) whose mapping expressions are S.a, S.b, S.c and S.a – S.b.

- The output of Q1 is connected to query transform Q2, which has two columns Q2y and Q2z whose expressions are Qa – Qb and Qc – Qd.
- The output of Q2 is loaded into target table T, which has two columns: T1 and T2.

The mapping expressions for target columns T1 and T2 are computed by starting from the end point (the target object) and "walked" back through the list of transforms, with columns of a transform written in terms of expressions from the previous transform.

When processing is started on data flow DF_1, it starts with column T1 of target table T.

The expression for T1 is Q2y, which in turn is Qa – Qb, which can be written as S.a – S.b. Therefore the mapping expression is S.a – S.b and column T1 has two source columns—it is mapped from S.a and S.b. The AL_COLMAP table contains two rows for the target column to describe the two source columns.

In the case of T2, it is mapped from Qc – Qd, which can be written as S.c – (S.a – S.b). In this case, there are three rows for this target column in the AL_COLMAP table, one for each source column.

Mapping complexities

If a data flow calls another data flow and then loads a target table, the mappings are expressed in terms of the tables and columns used within the other data flow. Information is generated by "drilling down" into the other data flow to continue the mapping process.

The situation in which the Merge transform is used within a data flow is a bit more complex, because when two data streams are merged, there are two ways to populate a target table. This possibility is captured by separating the mapping expressions with the keyword `AND`. For example, a target column could be populated from `S.a AND R.a`.

Transforms like Hierarchy_Flattening and Pivot also introduce complexities in the way columns are mapped.

It is also possible that some target columns are mapped by constants or expressions that do not use source columns. In this case there will be no rows in the AL_COLMAP table for the target column. The mapping expression in the AL_COLMAP_TEXT table will reflect this.

If a target column is populated with a call to the lookup function, then its source columns are both the looked up column and the key used to do the lookup.

ALVW_TABLEATTR

This view contains information about the attributes in a table.

Column Name	Data type	Description
TABLE_NAME	Varchar(64)	Name of the table.
TABLE_OWNER	Varchar(64)	Owner of the table. For SAP R/3, the value is `NULL`.
DATASTORE	Varchar(64)	The datastore to which this table belongs.
TABLE_ATTR	Varchar(64)	Name of the attribute (property of this table). The name of the attribute will be the same as seen when viewing the attributes of a table in the Designer's interface. Examples: Description, Business_Name, Date_last_loaded, Date_created, Total_Number_Of_Rows_Processed.
TABLE_ATTR_VALUE	Varchar(255)	Value of this attribute.

The following query returns when table `EMP` in datastore `HR` was last loaded:

```
SELECT TABLE_ATTR_VALUE
   FROM ALVW_TABLE_ATTR
   WHERE TABLE_NAME = 'EMP'
      AND TABLE_ATTR = 'Date_last_loaded'
      AND DATASTORE = 'HR'
```

ALVW_TABLEINFO

This view contains the list of tables imported into the repository.

Column Name	Data type	Description
TABLE_NAME	Varchar(64)	Name of the table.
TABLE_OWNER	Varchar(64)	Owner of the table. For SAP R/3, this table is NULL.
DATASTORE	Varchar(64)	The datastore to which this table belongs.
TABLE_ID	int	Internal ID of the table within the repository.

Internal metadata

This set of tables and views capture information about built-in metadata objects used by Data Integrator and the relationships between those objects.

- AL_LANG
- AL_ATTR
- AL_USAGE
- ALVW_FUNCINFO
- ALVW_PARENT_CHILD

AL_LANG

This table contains various Data Integrator objects. These objects are also displayed in Data Integrator's object library.

Column Name	Description
OBJECT_KEY	Internal ID of the object.
OBJECT_TYPE	Type of object (an integer). Query the AL_REPOTYPE_NAMES table to find the corresponding name for this type. Examples: data flow, table, datastore.
NAME	Name of the object.
VERSION	Indicates the number of times this object has been updated.
TYPE	Subtype of the object.
OWNER	For table objects, the owner.
DATASTORE	For table objects, the datastore to which they belong.
NORMNAME	Unique name for this object in this table.

The following query returns all the data flows in the repository:

```
SELECT OBJECT_KEY, NAME
   FROM AL_LANG A
   WHERE OBJECT_TYPE = 1
      AND VERSION =
         (SELECT MAX(VERSION) FROM AL_LANG B
         WHERE A.NORMNAME = B.NORMNAME)
```

AL_ATTR

This table contains attributes of repository objects provided by Data Integrator. For example, a description is an object attribute.

Column Name	Description
PARENT_OBJID	The internal ID of an object.
PARENT_OBJ_TYPE	Type of the object (integer).
ATTR_NAME	The attribute name for this object.
ATTR_VALUE	Value of the attribute.

The following query returns all the data flows and their descriptions in the repository:

```
SELECT OBJECT_KEY, NAME, ATTR_VALUE
   FROM AL_LANG O, AL_ATTR A
   WHERE O.OBJECT_TYPE = 1
      AND O.OBJECT_KEY = A.PARENT_OBJID
      AND ATTR_NAME = 'Description'
      AND O.VERSION =
         (SELECT MAX(VERSION) FROM AL_LANG B
         WHERE O.NORMNAME = B.NORMNAME)
```

AL_USAGE

This table is identical to ALVW_PARENT_CHILD except it captures the entire call hierarchy. For example, if a table is used in a data flow which is called from a work flow, then rows appear in this table that associate the work flow (parent) to the table (descendent). The Depth column is unique to this table.

NOTE: You need to populate this table explicitly by using the **Calculate Usage Dependencies** command, (accessible in the Designer and in the metadata reporting tool).

To query this table, use the following "Example use cases".

Column Name	Data type	Description
PARENT_OBJ	varchar(64)	Name of the calling object.
PARENT_OBJ_TYPE	varchar(64)	Type of the object. Examples: batch job or data flow.
PARENT_OBJ_DESC	varchar(255)	The description associated with this object.
PARENT_OBJ_KEY	int(4)	Key in the AL_LANG table of the parent object
DESCEN_OBJ	varchar(64)	Name of the descendant object. For transforms, the name of the output schema of the transform call (if the name of the transform is unique). If it is not unique, Data Integrator generates a unique numeric suffix and appends that to the given name.
DESCEN_OBJ_TYPE	varchar(64)	Type of the called object. Examples: data flow, table, function, file.
DESCEN_OBJ_DESC	varchar(255)	Description associated with the called object.
DESCEN_OBJ_USAGE	varchar(20)	(Applies only to tables and files) How the child is used. Examples: source, target, lookup table.
DESCEN_OBJ_KEY	int(4)	Key in AL_LANG of the descendant object.
DESCEN_OBJ_DS	varchar(64)	(Applies only to tables and files) The datastore of the child object.
DESCEN_OBJ_OWNER	varchar(64)	Owner of the child table.
DEPTH	int	Indicates the number of levels between objects in a parent/descendent relationship.

Example use cases

You can query the AL_USAGE table using SQL statements:

- To find out which jobs load a table
- To find out which objects depend on a source
- To produce a "where used" report for an object
- To show which jobs load what targets

For example, the following report shows a few columns and rows from AL_USAGE listing the objects that are related to the SALES_ORDER target table:

Parent_Obj	Parent_Obj_Type	Descen_Obj	Descen_Obj_Type	Descen_Obj_Desc	Descen_Obj_Usage
Build_Fact	DataFlow	SALES_ORDER	Table	Sales order target fact table	Target
Daily_Job	WorkFlow	SALES_ORDER	Table	Sales order target fact table	Target
DF_NewOrders	Job	SALES_ORDER	Table	Sales order target fact table	Target
Get_IDoc	Dataflow	SALES_ORDER	Table	Sales order target fact table	Target
IDoc_job	Dataflow	SALES_ORDER	Table	Sales order target fact table	Target
initial_Job	Job	SALES_ORDER	Table	Sales order target fact table	Target

➢ To find out which jobs load a table

The following query returns which jobs load a particular target:

```
SELECT PARENT_OBJ
   FROM AL_USAGE
   WHERE PARENT_OBJ_TYPE = 'Job'
      AND DESCEN_OBJ_TYPE = 'Table'
      AND DESCEN_OBJ_USAGE = 'Target'
      AND DESCEN_OBJ = 'targetTable'
```

➢ To find out which objects depend on a source

The following query returns which objects would be affected if you drop a source table:

```
SELECT PARENT_OBJ, PARENT_OBJ_TYPE
   FROM AL_USAGE
   WHERE DESCEN_OBJ_TYPE = 'Table'
      AND DESCEN_OBJ = 'targetTable'
```

➢ To produce a "where used" report for an object

The following query returns what objects call a given object. The given object in this case is the target table SALES_ORDER.

```
SELECT
        AL_USAGE.PARENT_OBJ,
        AL_USAGE.PARENT_OBJ_TYPE,
        AL_USAGE.DESCEN_OBJ,
        AL_USAGE.DESCEN_OBJ_DESC,
        AL_USAGE.DESCEN_OBJ_TYPE,
        AL_USAGE.DESCEN_OBJ_USAGE
   FROM AL_USAGE
   WHERE (AL_USAGE.DESCEN_OBJ_TYPE = 'Table'
        AND AL_USAGE.DESCEN_OBJ='SALES_ORDER')
```

The following table shows the result for an example repository:

Parent_Obj	Parent_Obj_Type	Descen_Obj	Descen_Obj_Type	Descen_Obj_Desc	Descen_Obj_Usage
Build_Fact	DataFlow	SALES_ORDER	Table	Sales order target fact table	Target
Daily_Job	WorkFlow	SALES_ORDER	Table	Sales order target fact table	Target
DF_NewOrders	Job	SALES_ORDER	Table	Sales order target fact table	Target
Get_IDoc	Dataflow	SALES_ORDER	Table	Sales order target fact table	Target
IDoc_job	Dataflow	SALES_ORDER	Table	Sales order target fact table	Target
initial_Job	Job	SALES_ORDER	Table	Sales order target fact table	Target

➢ To show which jobs load what targets

The following query returns all of the jobs in the repository and what targets they load:

```
SELECT
   AL_USAGE.PARENT_OBJ,
   AL_USAGE.PARENT_OBJ_TYPE,
   AL_USAGE.DESCEN_OBJ,
   AL_USAGE.DESCEN_OBJ_DESC,
   AL_USAGE.DESCEN_OBJ_TYPE,
   AL_USAGE.DESCEN_OBJ_USAGE
   FROM AL_USAGE
   WHERE (AL_USAGE.PARENT_OBJ_TYPE = 'Job'
      AND AL_USAGE.DESCEN_OBJ_TYPE = 'Table'
      AND AL_USAGE.DESCEN_OBJ_USAGE = 'Target')
```

This table shows the results for an example repository:

Parent_Obj	Parent_Obj_Type	Descen_Obj	Descen_Obj_Type	Descen_Obj_Desc	Descen_Obj_Usage
Daily_Job	Job	SALES_ORDER	Table	Sales order target fact table	Target
IDoc_job	Job	SALES_ORDER	Table	Sales order target fact table	Target
Initial_Job	Job	SALES_ORDER	Table	Sales order target fact table	Target

ALVW_FUNCINFO

This view contains a list of functions you defined in Data Integrator and those you imported into its repository.

Column Name	Description
FUNC_KEY	Internal ID for the function within the repository.
FUNC_NAME	The function name.
FUNC_OWNER	(Applies only to imported functions) Owner of the function.
DATASTORE	(Applies only to imported functions) Datastore to which the function belongs.

ALVW_PARENT_CHILD

This view contains information about objects (parents) that contain (or call) other objects (children).

Column Name	Description
PARENT_OBJ	Name of the calling object.
PARENT_OBJ_TYPE	Type of the object. Examples: batch job, real-time job, data flow.
PARENT_OBJ_DESC	The description associated with this object.
DESCEN_OBJ	Name of the called object.
DESCEN_OBJ_TYPE	Type of the called object. Examples: data flow, table, function, file.
DESCEN_OBJ_DESC	Description associated with the called object.
DESCEN_OBJ_USAGE	(Applies only to tables and files) How the child is used. Examples: source, target, lookup table.
DESCEN_OBJ_DS	(Applies only to tables and files) The datastore of the child object.
DESCEN_OBJ_OWNER	Owner of the child table.

You can query this table to:

- View which data flows load table `EMP`

```
SELECT PARENT_OBJ
   FROM ALVW_PARENT_CHILD
   WHERE DESCEN_OBJ_TYPE = 'Table'
      AND DESCEN_OBJ = 'EMP'
      AND DESCEN_OBJ_USAGE = 'Target'
```

- View the data flows called by job `HR_INITIAL_LOAD`

```
SELECT DESCEN_OBJ
   FROM ALVW_PARENT_CHILD
   WHERE PARENT_OBJ_TYPE = 'Job'
      AND PARENT_OBJ = 'HR_INITIAL_LOAD'
      AND DESCEN_OBJ_TYPE = 'Dataflow'
```

Operational metadata

These tables and views contain information about the run-time statistics of Data Integrator jobs and data flows.

- AL_HISTORY
- ALVW_FLOW_STAT

AL_HISTORY

This table contains information about the execution statistics of jobs.

Column Name	Description
OBJECT_KEY	Internal ID of the job within the repository.
INST_MACHINE	Computer on which the job was executed.
TYPE	Batch job or a real-time job.
SERVICE	Name of the job.
START_TIME	Time when the job was launched.
END_TIME	Time when the job completed.
EXECUTION_TIME	Difference between start time and end time in seconds.
STATUS	Status of the job upon completion. Examples: Error (E), Success (S).
HAS_ERROR	Displays a zero if there are no errors.

The following query returns the run statistics of all successfully executed jobs:

```
SELECT SERVICE, INST_MACHINE, START_TIME,
END_TIME, EXECUTION_TIME
   FROM AL_HISTORY A
   WHERE HAS_ERROR = 0
      AND SERVICE NOT IN
('di_job_al_mach_info', 'CD_JOB_d0cafae2')
      AND OBJECT_KEY = (SELECT MAX(OBJECT_KEY)
FROM AL_HISTORY b WHERE A.SERVICE = B.SERVICE)
```

ALVW_FLOW_STAT

This view contains information about the execution statistics of transforms within data flows.

Column Name	Description
JOB_NAME	Name of the executed job.
JOB_KEY	Internal ID for this job in the repository.
JOB_RUNID	Unique identifier for a particular execution of the job.
RUN_SEQ	Unique identifier for a particular sequence of an execution.
PATH	Position of the transform in the data flow.
OBJECT_NAME	Name of the transform.
OBJECT_TYPE	Type of the transform.
ROW_COUNT	Number of rows processed by this transform.
START_TIME	Time when transform started executing.
END_TIME	Time when transform stopped.
EXECUTION_TIME	The difference between start and end time.

9 Locales and Multi-Byte Functionality

Data Integrator supports the use of different locales in sources, the Job Server, and targets. It also supports single and multi-byte code pages. By combining these settings, you can control processing across different languages and allow for differences in capitalization, time and date formats, and character sets.

This chapter includes the following sections:

Definitions

This chapter uses the following terms:

Locale	Consists of three values related to world regions that control the format of data when it is stored, processed, or displayed. To specify a locale for the Job Server, you must select a **Language**, **Territory**, and **Code page** value. Datastore and file format locales do not require that you set the **Territory** value for a locale. For more information, see "Guidelines for setting locales" on page 558. Database sources and targets might not need locale values specified. For more information, see "Processing with and without UTF-16 Unicode" on page 555.
Language	Specifies the locale value for a human language. To view the list of supported languages, see "List of supported locales and encodings" on page 567.

Territory	Specifies the locale value for a geographical location (usually the country) where a locale language is used. The paring of a language with a territory determines factors such as date format, time format, decimal separator, currency format, and so on. Data Integrator uses territory values to process the following data types:
	
	`date` `interval`
	`datetime` `numeric`
	`decimal` `real`
	`double` `time`
	`int` `timestamp`

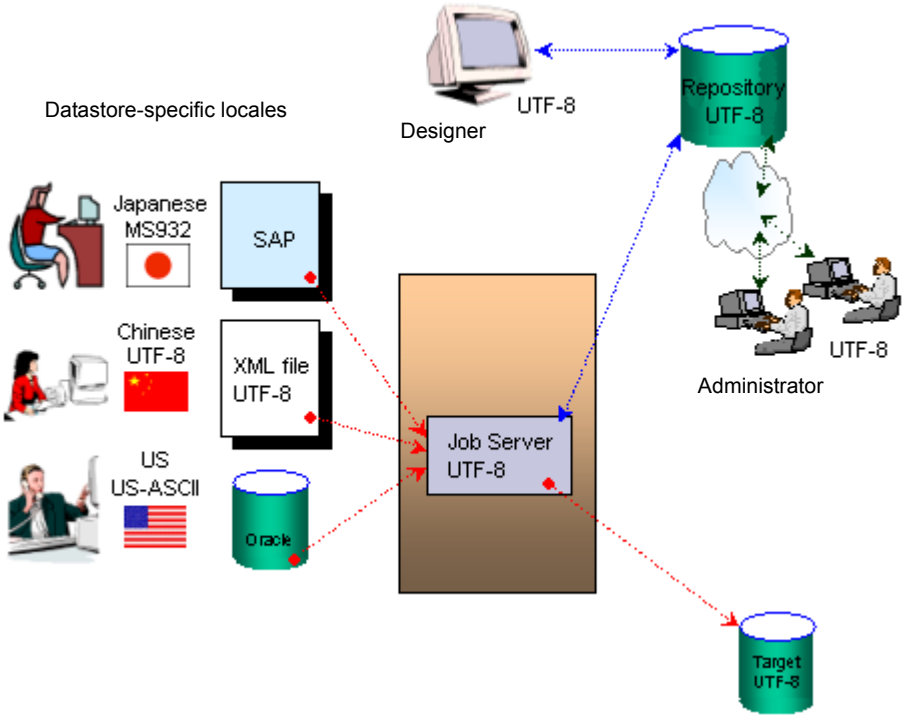
Code page	Specifies the locale value for the sequence of bits that defines a character. Data Integrator uses a code page value to transcode `varchar` data types. All code pages contain the ASCII characters as a subset of their entire character set. This means for example, that the Japanese code page MS932 also partially supports other languages such as English. However, use `Japanese` as the **Language** value, `Japan` as the **Territory** value, and `ms932` as the **Code page** to avoid possible data corruption in a Japanese locale. It is your responsibility to select corresponding values for locales. While Data Integrator validates that locale values are entered, it does not validate that they are realistic.
Single-byte	Allows one byte per character in a code page. Each character needs just a single byte of memory for processing.
Multi-byte	Allows one or more bytes per character in a code page. Some languages, like Korean and Chinese, can only be represented by multi-byte characters. Use multi-byte code pages to process characters for those languages. Note that this functionality can impact performance.
Encoding	Represents a code page in XML.
Transcode	Converts data from one code page to another. To support ETL environments in which sources with different locales are processed in the same job, Data Integrator supports transcoding. Note that transcoding can impact performance.

Unicode	Provides a unique number for every character and a method to implement ISO/ISE 10646. Data Integrator supports UTF-8 and UTF-16 Unicode transformation formats.
UTF-8	Serializes a Unicode code point as a sequence of one to four bytes depending on the complexity of the character (single-byte characters use one byte and multi-byte characters use up to four). Data Integrator allows you to select UTF-8 as a code page for the Job Server and connections to sources and targets.
UTF-16	Standardizes each Unicode code point at two bytes for each character. Allows access to 63k characters as 16-bit units. Data Integrator supports UTF-16 for: • An MS SQL Server database when its datastore code page is set to `utf8` or `utf16` • National character-set data types in the following databases: Oracle with `nchar` or `nvarchar2` MS-SQL with `nchar` or `nvarchar` DB2 with `graphic` or `vargraphic` When using UTF-16 support, you do not have to set locales for connections to database sources and targets. For more information, see "Processing with and without UTF-16 Unicode" on page 555.

Locale support

By supporting different locales Data Integrator allows you to configure an enterprise environment in which components process data in different human languages. For example:

- You can configure locales for the following sources:
  - Language `Japanese`, territory `Japan`, and code page `MS932`
  - Language `Simplified Chinese`, territory `China`, and code page `UTF-8`
  - Language `English`, territory `US`, and code page `US-ASCII`
- Then load data to a target database using a UTF-8 code page.



To use Data Integrator's locale support, set locales within each:

- Database (source, target, and repository) or application (SAP, PeopleSoft, Oracle, Siebel, JDE)
- Database client
- Datastore connection to a database or application

  Datastore locales must match the locales of source and target database clients. This allows the datastore to move data between Data Integrator and each database without possible data corruption. If the database locale differs from its database client locale, it is your responsibility to ensure that the database transcodes the data before it reaches or after it leaves Data Integrator.

  NOTE 1: To avoid the necessity of setting locales for a database client and a Data Integrator datastore, see "Processing with and without UTF-16 Unicode" on page 555.

  NOTE 2: All adapter datastores are automatically set by Data Integrator to the code page UTF-8. They are handled the same way as XML message sources and targets. For more information, see "File format locales" on page 561.

- File format (flat file, XML Schema, and DTD)

  Match the file format locale to that of each source or target file.

- Job Server

  Data Integrator uses the Job Server's locale for the engines it spawns. To avoid possible data corruption, use the same locale settings for the repository, its client, and the Job Server.

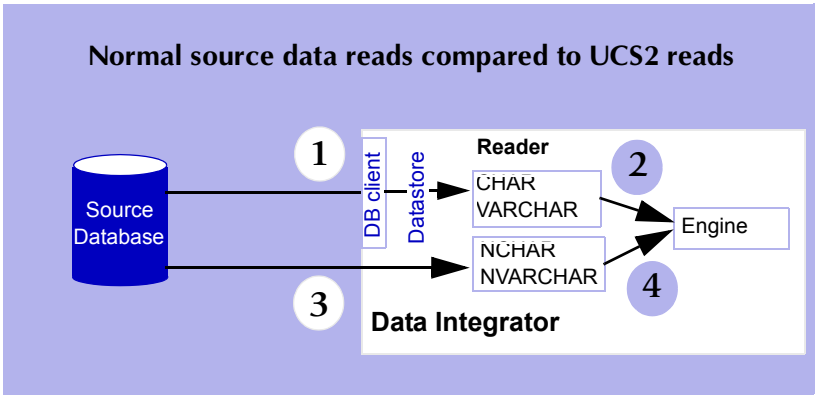
Code page support

The code page you select in each locale used by a job determines whether transcoding occurs during job processing. Data Integrator automatically transcodes between different code pages when necessary to support complex, multi-language data management in a single job.

However, it is your responsibility to set up the connections between Data Integrator and its repository, sources, and targets to avoid data corruption due to mismatched locale settings between Data Integrator and its connections to external systems.

Processing with and without UTF-16 Unicode

The Data Integrator Job Server supports reading and loading national character-set data types (`nchar`, `nvarchar`, `nvarchar2`, `graphic` and `var-graphic`). Data Integrator automatically handles these data types using UTF-16. If extracting or loading data that uses only these data types, you do not have to set locales for a database client and its datastore.



The figure above shows the path your data takes from a source database to a Data Integrator engine when data is read during a job run. The numbered paragraphs that follow describe each step.

The upper arrow shows a normal job run and the locales the job uses to support data integrity:

1. Data Integrator assumes the database transcodes data into the database client code page, as needed, before it uses its datastore code page to read data from the database. The client and datastore code pages must match otherwise Data Integrator cannot recognize the format of the data. Notice that the data types are not national character-set data types.
2. If the datastore code page is different from the Job Server code page, Data Integrator transcodes it then processes the job.

The lower arrow shows a national character-set data type job run:

3. Notice that the client and datastore are not in the path. Data Integrator reads the data formatted for national character-set data types using a UTF-16 code page without using client and datastore code pages.
4. The engine transcodes the data from UTF-16 to the Job Server's code page before it is processed.

When data continues to a target database, the processes are reversed. Data Integrator automatically transcodes the data in the national character-set data type path back into UTF-16 before loading it into a target. In a normal job run path, if the datastore code page differs from the Job Server code page, the engine transcodes the data to the target's code page before it is passed on to the database client. The datastore locale and the database client locale must match so that data is accurately sent into the database where it might again need to be transcoded into a different locale for storage.

Data Integrator support of national character-set data types is restricted to the following source and target databases:

- Oracle with `nchar` or `nvarchar2`
- MS-SQL with `nchar` or `nvarchar`
- DB2 with `graphic` or `vargraphic`

Data Integrator can also extract, transform, and load a single table with both national character-set data types and other data types. In this case, the data in the columns with the national character-set data types uses the UTF-16 path and the other data uses the datastore path through Data Integrator.

In the Designer, you can also assign a Unicode as a code page by creating an MS SQL Server datastore connection and selecting **UTF16** or **UTF8** as its code page.

National character-set data types help you avoid having to set locales for connections to database and application sources and targets. However, you still must set locales for the Job Server and for file formats (if you use files in your job).

Also, when Data Integrator imports a table with columns using any character data type (`nchar`, `nvarchar`, `varchar`, `char`, and so on), it imports the column size in number of characters (not bytes). Similarly, while creating a new column in Query objects, Data Integrator assumes the column size is in number of characters. As the number of bytes per character varies from code page to code page, at runtime, the Data Integrator engine allocates memory based on the Job Server's code page. For more information, see "Column Sizing" on page 566.

Minimizing transcoding in Data Integrator

As a rule, transcoding impacts job performance.

- Use the same locale for all components and use a single-byte code page if possible.
- If a datastore or file format and the Job Server use a different locale, Data Integrator automatically transcodes the data, which supports a multi-language enterprise environment.

Data Integrator minimizes the impact of transcoding for equivalent code sets such that transcoding between the following code page pairs does not impact performance.

Superset	Subset
MS1252	ISO88591 (LATIN1)
MS1250	ISO88592
MS1251	ISO88595
MS1253	ISO88597
MS1254	ISO88599
MS1255	ISO88598
MS1256	ISO88596
MS1257	ISO88594

For a list of all supported languages, territories, code pages, and encodings, see "List of supported locales and encodings" on page 567.

Guidelines for setting locales

Use the following guidelines when you set locales.

Job Server locale

When you install a Job Server, the installer provides a window with the default locale settings. The Job Server takes its default locale from the user session locale on the operating system of the host computer on which it is installed. The Job Server's locale is also used by the engines it spawns .

- If your jobs will run in a multi-language environment, set the Job Server's locale to a superset of all datastore and file format locales. UTF-8 is a superset for most code pages. However, if your system can use more specific locale definition, use the more specific locale. For example:

- Extracting from Chinese MS950, Korean MS949, and Japanese MS932 sources requires a code page that supports transcoding each of these into a single code page. UTF-8 would be an appropriate Job Server code page in this case.
- Extracting from German ISO88591 and German MS1252 sources could be done using German MS1252 as the Job Server's code page. Data Integrator minimizes transcoding between German MS1252 and ISO88591. For more information, see "Minimizing transcoding in Data Integrator" on page 557.

- If you do not have a multi-language environment, use a single-byte code page and use the same settings for all locale values or use only locales with code pages that minimize transcoding. This strategy ensures the best performance.

Also make sure that the locale of the repository's data base client (installed on the Designer's computer) matches the Job Server locale. The Designer uses the Job Server's locale to ensure that it passes accurate data to the repository

To complete the Job Server installation, you can:

- Accept the default settings.
- On Windows, select settings from a displayed list
- On UNIX, enter a value. For a list of possible values, see "List of supported locales and encodings" on page 567.

Database, database client, and datastore locales

Set a database and its database client locales using your database software.

Set a Data Integrator datastore to the same locale as the application or database client to which it connects. Data Integrator automatically sets each datastore locale to match that of the Job Server. However, if your sources or targets use different locales, use the **Set Locale** button in the Datastore Editor to edit datastore locales manually.

The **Territory** option is not available for configuring datastore locales in Data Integrator. Data Integrator transcodes code pages. However, in some cases Data Integrator requires the territory value. Consider the effects on a simple job containing a single data flow that includes a source loading directly into a target:

- The Job Server locale is `eng_us.UTF-8` and the source datastore locale is configured as `deu_de.ms1252` (German).
- Data Integrator transcodes from datastore code page MS1252 to Job Server code page UTF-8, but Data Integrator expects the decimal data with "." as decimal separator. As the datastore territory setting is set to German, the decimal separator is "," (comma).

In this case, Data Integrator does not change the decimal separator. The result is that you must configure the source database client locale to match the Job Server locale's territory settings (dot "." as a decimal separator).

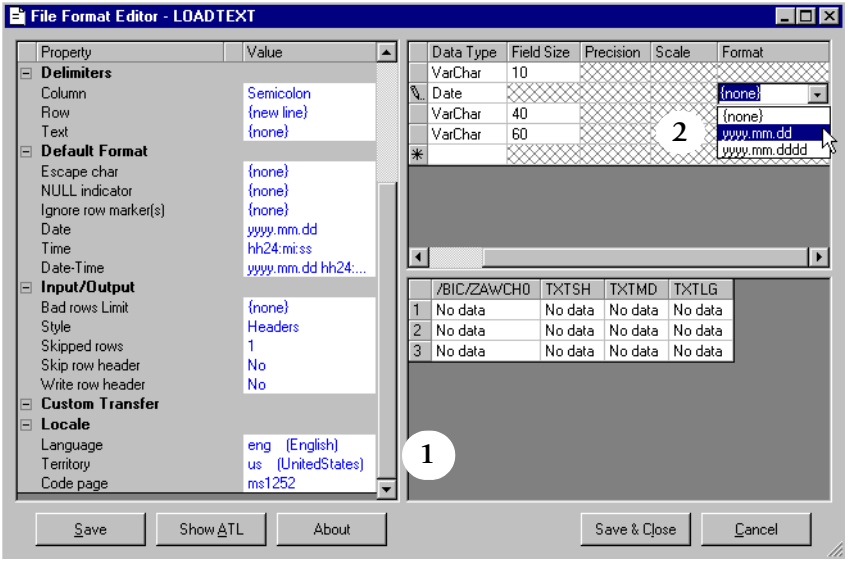
Locales apply for all profiles created from each datastore. For more information, see "Datastore" on page 34.

All adapter datastores are automatically set by Data Integrator to the code page UTF-8. They are handled the same way as XML message sources and targets. For more information, see "XML encodings" on page 562.

File format locales

For flat files, the default locale is automatically set to match that of the Job Server. However, you can use the **Locale** section on the file format editor to specify the language and code page that corresponds to the file data.

1. As in datastore locales, the **Territory** option for a flat file's locale is not active in Data Integrator.
2. However, you can change the data type for a flat file column and then enter the format that you want to use for a `date` or any numeric data type using the Column Attributes work area.



For more information, see "File format" on page 62.

XML encodings

For XML Schemas or DTDs used as sources, the default encoding (code page) for messages is assumed by Data Integrator to be UTF-8. Header information is ignored. Data Integrator transcodes inbound data to the Job Server code page (if necessary) before processing it. For XML files Data Integrator reads header information at run-time. If encoding information is not specified, Data Integrator assumes it is UTF-8. To edit the encoding for XML file sources, do so in the original file then rerun the job.

For XML Schemas or DTDs used as targets, the default encoding is automatically set to UTF-8.

- If you use an XML Schema or DTD as a file, you can change the encoding using the target editor. See "Target XML files, messages, and templates" on page 133.
- If you use an XML Schema or DTD as a message, the encoding cannot be changed. Data Integrator transcodes out-bound message configuration information (like global variable data) into UTF-8.

Locales Data Integrator automatically sets

The Administrator's locale is automatically set to UTF-8. By using UTF-8 (a Unicode that supports all languages), Data Integrator ensures data integrity in the Administrator.

Multi-byte support

A code page can support either single or multiple byte characters as well as add support to the language you select for your locale. Many Asian languages require multi-byte character support. In addition, Data Integrator provides support for two Unicodes: UTF-8 and UCS2. These are multi-byte code pages that support most of the world's languages.

Multi-byte string functions supported in Data Integrator

All Data Integrator string functions support multi-byte data.

For example, when using the functions INDEX, LENGTH, RPAD, and SUBSTR, the sizes and offsets used as arguments or return values are expressed as number of characters, not number of bytes. You can also see this logical, intuitive behavior when these functions are pushed down and evaluated by the database. Similarly, when Data Integrator evaluates the SQL predicate `LIKE`, the single-character wildcard "_" matches exactly one character, not one byte.

Numeric data types: assigning constant values

Use care when assigning a constant to a numeric variable or column in Data Integrator. Use one of the following methods:

- Assigning a value as a numeric directly
- Assigning a value in string format

Assigning a value as a numeric directly

If a numeric value is not within quotes—for example, `$comm = 123.45`—then Data Integrator stores the value as a numeric. If the value is stored as numeric, then dot (".") is the only recognized decimal separator in the Designer, regardless of the locale. While executing the job, however, the Data Integrator engine automatically converts the value to appropriate decimal separator for the locale.

Syntax	Comment
$VALUE = 123.45	Correct syntax
$VALUE = 123,45	Wrong syntax. Use a comma "," as a decimal separator inside Data Integrator only if the operating system's decimal separator is a comma "," and the value is within quotes.

Assigning a value in string format

If a numeric value is stored within quotes—for example, `$comm = '123,45'`—then the Designer stores the value in a string format and while executing the job, the Data Integrator engine automatically converts the value from the string to the appropriate numeric data type. When the Data Integrator engine converts a string to a numeric format, it uses the decimal separator for the specified Job Server locale.

For example, if the Job Server locale is set to German regional settings (`deu_de.ms1252`), then Data Integrator uses a comma (",") as the decimal separator when converting a value from a string to appropriate numeric data type.

Syntax	Comment
$VALUE = '123,45'	Correct syntax
$VALUE = '123.45'	Wrong syntax. If the operating system's decimal separator is a comma "," and the value is within quotes, then Data Integrator tries to process a dot (".") as a thousands separator.

If the Job Server locale is set to English regional settings (`eng_us.ms1252`), then Data Integrator uses a dot (".") as the decimal separator when converting a value from a string to an appropriate numeric data type.

Syntax	Comment
$VALUE = '123.45'	Correct syntax
$VALUE = '123,45'	Wrong syntax. If the operating system's decimal separator is a comma"," and the value is within quotes, then Data Integrator tries to process a dot (".") as a thousands separator.

NOTE: Using the incorrect decimal separator can generate incorrect results. Similarly, having a thousands separator can generate incorrect results. Business Objects recommends that you do not use a thousands separators when converting a value from string to numeric.

BOM Characters

The Unicode standard includes the use of BOM (Byte Order Mark) characters as a signature for file data.

Data Integrator supports BOM characters. When it reads data from a file, the Data Integrator engine trims BOM characters. The following BOM characters and their corresponding encodings are supported:

BOM Characters (Bytes)	Encoding Form
FE FF	UTF-16 big-endian
FF FE	UTF-16 little-endian
EF BB BF	UTF-8

Round-trip conversion

While transcoding, if Data Integrator encounters round-trip conversion conflicts, it transcodes to the first code point match (ascending order of the hexadecimal values) in the target code page. For example, if two different Japanese characters are mapped to different hexadecimal code points in the MS932 code page (EEFA and FFE4) but then mapped to the same UTF-16 code point (FFE4). When transcoding back from UTF-16 to MS932, Data Integrator converts them both to code point EEFA.

Column Sizing

The number of bytes per character may vary from one code page to another. For example, the "£Á" which represents a special "A" in the Korean MS949 code page needs 2 bytes to represent the character; while the UTF-8 code page needs 3 bytes to represent the same character.

	Value	Hex values	Bytes
MS949 Korean Code page	special A	A3 C1	2
UTF-8 code page	special A	EF BC A1	3

Data Integrator represents sizes in number of characters (except for fixed-width file formats which are represented in bytes). Internally, Data Integrator allocates enough memory to store multi-byte characters.

If the datastore code page is different from the Job Server code page, then transcoding must occur, which may result in the need for extra space allocation.

List of supported locales and encodings

This section lists values that you can select for locales and encodings.

Note that for a language, territory, code page or encoding you can also select `<default>`. For a Job Server locale, this means that the value is read from the operating system's locale for the current user session. For example, you log in to your Windows session as `Davis` with a user locale of `eng_gb.ms1252`. When you install a Job Server, it picks up the same locale and displays it as the default (`eng_gb.ms1252`). At this point you can:

- Accept these values. The Job Server will always use the `eng_gb.ms1252` locale.
- Use `default` for one or all values, for example by entering `<default>_<default>.<default>`. The Job Server's locale will always match the operating system's user session.
- Edit this locale to, for example `eng_us.ms1252`. The Job Server will always use the `eng_us.ms1252` locale.

Your choice will depend on how you want to set up your system.

Languages

Data Integrator uses the standard ISO 639-2/T three-letter abbreviations. Enter codes in lower case.

```
sqi Albanian          kor Korean
bul Bulgarian         lav Latvian
bel Byelorussian      lit Lithuanian
cat Catalan           mkd Macedonian
hrv Croatian          nor Norwegian
ces Czech             por Portuguese
dan Danish            ron Romanian
nld Dutch             rus Russian
eng English           srp Serbian
est Estonian          zho Simplified Chinese
fas Farsi             slk Slovak
fin Finnish           slv Slovenian
fra French            spa Spanish
deu German            swe Swedish
ell Greek             tha Thai
heb Hebrew            zho Traditional Chinese
hun Hungarian         tur Turkish
ita Italian           ukr Ukrainian
jpn Japanese          vie Vietnamese
```

Territories

Data Integrator uses standard ISO 3166-1 two-letter abbreviations. Enter codes in lower case.

```
al Albania              it Italy
ar Argentina            jp Japan
at Austria              kp Korea, Dem. Peo. Rep.
au Australia            kr Korea, Republic of
be Belgium              lt Lithuania
bg Bulgaria             lv Latvia
br Brazil               mk Macedonia
by Belarus              mx Mexico
ca Canada               nl Netherlands
ch Switzerland          no Norway
cl Chile                pt Portugal
cn China                ro Romania
cz Czech Republic       ru Russia
de Germany              sa Saudi Arabia
dk Denmark              se Sweden
ee Estonia              si Slovenia
es Spain                sk Slovakia
fi Finland              th Thailand
fr France               tr Turkey
gb United Kingdom       tw Taiwan
gr Greece               ua Ukraine
hr Croatia              us United States
hu Hungary              vn Vietnam
il Israel               yu Yugoslavia
ir Iran                 za South Africa
```

NOTE: The combination of language code “zho” and territory “cn” maps to Simplified Chinese, while “zho” plus “tw” maps to Traditional Chinese.

Code pages

Data Integrator supports the following code pages. Enter codes in lower case.

Data Integrator supported code pages

Code page Name	Multi-byte?	Vendor	Mapped from Unicode Version	Notes
iso88591		ISO/IEC 8859-1:1987	2.1	Western European
iso885910		ISO/IEC 8859-10:1992	2.1	Latin6, Nordic
iso885915		ISO/IEC 8859-15:1999	2.1	Latin9, Western European
iso88592		ISO/IEC 8859-2:1987	2.1	Latin2, Eastern European
iso88593		ISO/IEC 8859-3:1988	2.1	Latin3, Southeast European
iso88594		ISO/IEC 8859-4:1988	2.1	Latin4, Baltic
iso88595		ISO/IEC 8859-5:1988	2.1	Cyrillic
iso88597		ISO/IEC 8859-7:1987	2.1	Greek
iso88598		ISO/IEC 8859-8:1988	2.1	Hebrew
iso88599		ISO/IEC 8859-9:1989	2.1	Latin5, Turkish
japaneuc	Yes	Japanese EUC (JIS X 0201-1976, JIS X 0208-1990, JIS X 0212-1990)	2.1	
latin1		ISO/IEC 8859-1:1987	2.1	Western European
ms1250		Microsoft code page 1250	2.1	Latin-2 (Central Europe)
ms1251		Microsoft code page 1251	2.1	Cyrillic (Slavic)

Data Integrator supported code pages (Continued)

Code page Name	Multi-byte?	Vendor	Mapped from Unicode Version	Notes
ms1252		Microsoft code page 1252	2.1	Latin 1 (ANSI), ISO 8859-1 plus Microsoft extensions
ms1253		Microsoft code page 1253	2.1	Greek
ms1254		Microsoft code page 1254	2.1	Latin 5 (Turkish), ISO 8859-9 plus Microsoft extensions
ms1255		Microsoft code page 1255	2.1	Hebrew
ms1257		Microsoft code page 1257	2.1	Baltic Rim
ms1258		Microsoft code page 1258	2.1	Vietnamese
ms1361	Yes	Microsoft code page 1361		Korean Johab, KSC 5681-1992 plus Microsoft extensions Currency mapping changed and user-defined range added to match Windows 2000 SP 4.
ms874		Microsoft code page 874	2.1	Thai, TIS 620 plus Microsoft extensions
ms932	Yes	Microsoft code page 932	2.1	Japanese Shift-JIS plus Microsoft extensions Currency mapping changed and user-defined range added to match Windows 2000 SP 4.
ms936	Yes	Microsoft code page 936	3.0	Simplified Chinese, GB 2312-80 plus Microsoft extensions User-defined character range added to match Windows 2000 SP 4.
ms949	Yes	Microsoft code page 949	3.0	Korean KS C 5601-1992 plus Microsoft extensions. Currency mapping changed and user-defined range added to match Windows 2000 SP 4.

Data Integrator supported code pages (Continued)

Code page Name	Multi-byte?	Vendor	Mapped from Unicode Version	Notes
ms950	Yes	Microsoft code page 950	3.0	Traditional Chinese Big 5 plus Microsoft extensions User-defined range added to match Windows 2000 SP 4.
shift_jis	Yes	Shift-JIS (JIS X 0201-1976, JIS X 0208-1990)	2.1	"Standard" Shift-JIS without Microsoft extensions
us-ascii		ISO/IEC 646		7-bit ASCII
utf8	Yes		2.1	

XML encodings

Data Integrator supports the following encodings. Enter codes in lower case.

Data Integrator's Encoding	XML Encoding	Multi-byte?	Description
iso88591	ISO-8859-1		ISO 8859-1 Western European
iso88592	ISO-8859-2		ISO 8859-2 Eastern European
iso88593	ISO-8859-3		ISO 8859-3 Southeast European
iso88594	ISO-8859-4		ISO 8859-4 Baltic
iso88595	ISO-8859-5		ISO 8859-5 Cyrillic
iso88596	ISO-8859-6		ISO 8859-6 Arabic
iso88597	ISO-8859-7		ISO 8859-7 Greek
iso88598	ISO-8859-8		ISO 8859-8 Hebrew
iso88599	ISO-8859-9		ISO 8859-9 Latin 5 (Turkish)
JapanEUC	eucjis	Yes	Japanese Extended UNIX Code (incl. JIS X 0212)
latin1	Latin1		ISO 8859-1 Western European
ms1250	WINDOWS-1250		MS Latin 2 (Central Europe) (Latin 2)
ms1251	WINDOWS-1251		MS Cyrillic (Slavic)
ms1252	cp1252		MS Latin 1 (ANSI), superset of Latin1
ms1252	WINDOWS-1252		MS Latin 1 (ANSI), superset of Latin1
ms1253	WINDOWS-1253		MS Greek
ms1254	WINDOWS-1254		MS Latin 5 (Turkish), superset of ISO 8859-9
ms1255	WINDOWS-1255		MS Hebrew
ms1256	WINDOWS-1256		MS Arabic
ms1257	WINDOWS-1257		MS Baltic Rim
ms1258	WINDOWS-1258		MS Vietnamese
ms874	cp874	Yes	MS Thai, superset of TIS 620
ms932	Shift_JIS	Yes	MS Japanese, superset of Shift-JIS
ms949	iso2022kr	Yes	MS Korean, superset of KS C 5601-1992
ms949	KOREAN	Yes	MS Korean, superset of KS C 5601-1992
ms950	big5	Yes	MS Traditional Chinese (Taiwan)
utf8	UTF 8	Yes	UTF-8 encoding of Unicode

Limitations

There are several limitations to Data Integrator support for multi-byte characters:

- Data Integrator supports a variety of single- and multi-byte code pages, but it does not support UCS-4, EBCDIC, or SAP R/3 blended code pages.

  Data Integrator does not support EBCDIC code pages as they are not ASCII- or UTF-8 -compatible. This is not a problem for users targeting or sourcing data from IBM systems if the Data Integrator engine is running on a non-IBM platform. The IBM system will transparently convert its data to/from EBCDIC when communicating with a foreign architecture.

- The Data Integrator engine is constrained to operating on only one code page per installed Job Server. If you want more than one code page for data integration applications you must install additional Job Servers on separate computers to satisfy this requirement.

- In the Designer, multi-byte characters are not supported. Data Integrator does not yet fully address all formatting issues. For instance, Data Integrator supports the “dot” and “comma” currency formats used for most European currencies, but does not support the “tick” and “space” currency format used in Switzerland.

- The only way to set Data Integrator’s Job Server locale is during installation of Data Integrator components. The settings are saved in a local configuration file and referenced when Data Integrator processes data.

- Data Integrator does not support surrogates for the UTF-16 code page.

10 Reserved Words

The following words have special meanings in Data Integrator and therefore should not be used as names for work flows, data flows, transforms, or other design elements that you create. They are reserved with any combination of upper- and lower-case letters.

If you use reserved words you must put double quotation marks around them. For example:

```
"PRIMARY"
```

Reserved words appear in editor text areas in blue.

_AL_DEFINE
_AL_ELSE
_AL_IFDEF
_AL_MESSAGE
_AL_METADATA_ELEMENT
_AL_STORED_PROCEDURE
_AL_TRAN_FUNCTION
_FUNC_TABLE
_MEMORY
_RFC_FUNCTION
ABAP_PROGRAM
ACTA
ACTAGUICOMMENT
ALGUICOMMENT
ALL
AL_NEST
AL_NESTED_TABLE
AL_RFC_SCHEMA_GROUP
AL_UNNEST
AL_UNNEST_SCHEMA_GROUP
AL_UNSPECIFIED_PARAMAND
AS
ASC
BEGIN
BEGIN_SCRIPT
BY
CALL
CASE

CATCH
CHAR
CHARACTER
CONVERT
CREATE
CUSTOM
DATABASE
DATAFLOW
DATASTORE
DATE
DATETIME
DECIMAL
DECLARE
DEFAULT
DESC
DISTINCT
DISTINCT_KEY
DOMAIN
DOUBLE
ELSE
EMBEDDED_DATAFLOW
EMBEDDED_DATAFLOW_RT
END
END_TRY
FILE
FLOWOUTPUT
FOREIGN
FROM
FUNCTION
GENERATED
GLOBAL
GROUP
HAVING
IF
IN
INPUT
INT
INTEGER
INTERVAL
IS
KEY
LEFTOUTERJOIN
LIKE
LOAD
LONG
LOOKUP
NOT
NULL
NUMERIC
ON
OR
ORDER
OUT
OUTPUT
PARALLEL
PIPE
PLAN
PRIMARY
READ
REAL
REFERENCES
RETURN
RETURNS
SELECT
SESSION
SET
SYSTEM
SYSTEM_PROFILE
TABLE
TIME
TRANSFORM
TRANSFORM_SCHEMA_MAPPING

TRY
VARCHAR
VIEW
VOID
WHERE
WHILE

A Glossary

This glossary defines terms used in this documentation set including common terms that have a special meaning with respect to Data Integrator. Within a definition, other terms that are defined in the glossary appear in **bold type**.

ABAP

Advanced Business Application Programming.
A fourth-generation programming language developed by SAP in which R/3 applications are written.

ABAP program

A program that executes database operations on an **SAP R/3** server. Data Integrator **R/3 data flows** generate ABAP programs.

Access Server

A process that starts, stops, and monitors real-time services (**real-time jobs**). The Access Server dispatches requests to real-time services, ensuring optimal load balancing and complete life cycle management. The Access Server also starts, stops, and monitors **adapters**.

adapter

An external Data Integrator **interface**. There are two types of adapters:

- Custom adapters — Adapters developed using the Data Integrator Adapter Development Kit
- Prepackaged adapters — Adapters prebuilt and purchased from Business Objects, such as the Data Integrator MQSeries Adapter

Administrator

A browser-based system administration application that supports Data Integrator. Use the Administrator to configure and manage:

- Services (**real-time jobs**)
- Service providers (instances of real-time services)
- Clients (Web application clients)
- Batch jobs
- Adapters
- Job logs

after-image

The values in an UPDATE row after the row changes. You might want to pair before- and after-images of UPDATE rows when designing log-based changed-data capture jobs which Data Integrator supports.

aggregate data

Data that results when a process combines elements. This data can be presented collectively or in summary form.

ALE

Application Link Enabling. An SAP R/3 programming-related interface designed to allow reliable communication across a distributed environment.

Analytic Rapid Mart

An **Data Integrator application** that produces a predefined **data warehouse** for a specific area of business analysis. Analytic Rapid Marts are typically based on dimensional modelling. With Analytic Rapid Marts, users can manage large volumes of data.

attribute

A **property** created for a type of **object**.

BAPI

Business Application Programming Interface. A standardized SAP R/3 programming interface that allows non-SAP R/3 applications to access specific business processes and data.

back-office application

A software system that an enterprise uses to run business operations, such as human resources, general ledger, and finance. Back-office applications store data in a relational **database** optimized for operational use.

There are several types of back-office applications:

- Enterprise resource planning (ERP) — For example, SAP R/3, BAAN, J.D. Edwards, PeopleSoft
- Supply chain management (SCM) — For example, i2
- Customer relationship management (CRM) — For example, Siebel, Clarify
- Legacy

Basis

The SAP infrastructure. Basis is the foundation for all SAP products.

batch job

A group of objects (data flows, work flows, conditionals, scripts, and so forth) that executes on a schedule. This type of job is recommended for operations on large amounts of data.

before-image

The values in an UPDATE row before the row changes. You might want to pair before- and after-images of UPDATE rows when designing log-based changed-data capture jobs which Data Integrator supports.

bulk loading

A software-based mechanism that moves large data files. This mechanism supports compression, blocking, and buffering to optimize transfer times.

business component

A set of tables Siebel applications use to create a logical object called a business object.

Business Objects Data Platform

A set of applications from Business Objects, Americas, that enterprises can use to manage all types of data—from data for supply chain partners to data for customers. The Business Objects Data Platform consists of **Data Integrator** and one or more **Rapid Marts**.

CDC datastore

A CDC datastore allows you to limit extracted data to changed data only. A CDC datastore connects a **changed-data capture** table on a source database to Data Integrator.

CDC checkpoint

A CDC checkpoint enables Data Integrator to restrict CDC subscription reads. After you enable a checkpoint, the next time the CDC job runs, it reads only the rows inserted into the CDC table since the last checkpoint.

CDC Subscription

A CDC subscription name allows different data flows to extract data from the same table without corrupting data extracted by other data flows. The datastore and table name can be the same; however, the table owner name must be different in each source CDC table.

changed-data capture

The process of retrieving changes made to a production data source. This process consolidates units of work, ensures data is synchronized with the original source, and reduces data volume in a warehouse environment.

Citrix MetaFrame XP

Citrix MetaFrame XP software provides an access infrastructure for enterprise applications. You can use this software to run Data Integrator on a server which publishes instances of the Designer and other Data Integrator interfaces to users on client computers.

client/server

A distributed technology approach where the processing is divided by function. The server performs shared functions (such as managing communications and providing database services), while the client performs individual user functions.

COM

Component Object Model. Provides a set of interfaces that allow clients and servers to communicate within the same computer (running on Windows 95 or later versions).

Commerce Rapid Mart

An **Data Integrator application** that produces a predefined **data warehouse** and that includes logic to support real-time requests for data from a **back-office application**. A Commerce Rapid Mart is usually based on different data model than an **Analytic Rapid Mart**. With a Commerce Rapid Mart, users can integrate a company's back-office system with an **e-commerce** application.

Communication Structure

In **SAP BW**, a data structure that defines a set of **InfoObject**s available from an **InfoSource** to put into **InfoCube**s.

conditional

A **single-use object**, available in **work flow**s**,** that allows you to branch the execution logic based on the results of an expression. The conditional takes the form of an if/then/else statement.

custom ABAP program

A custom ABAP program runs an **ABAP program** and generates a **data set**. With a custom ABAP program, you can run an existing ABAP program as part of a job. Use a custom ABAP program as a **source** in a **data flow** or an **R/3 data flow**.

data extraction

The process of moving data from a **database** or application source to a database target (either from a legacy database to a **data mart**, or from one data mart to another).

data flow

A **reusable object** containing **steps** to define the transformation of data from **source** to **target**. Data flows are called from inside a **work flow** or **job**. You can pass information into or out of data flows using **parameters**.

Data Integrator

A software system that allows users to build and execute applications with which they can create and maintain **data warehouses**.

Data Integrator consists of several components:

- **Access Server**
- **Data Integrator engine**
- **Data Integrator repository**
- **Data Integrator Service**
- **Administrator**
- **Designer**
- **Job Server**

Data Integrator application

A set of **jobs** that process batch data extractions and real-time requests using data from a variety of data sources.

Data Integrator engine

The core process that reads job information from the **Data Integrator repository** and sets up run-time processes that execute the job. The run-time processes extract, transform, and load relational and hierarchical data. The **Job Server** starts the Data Integrator engine to execute batch or real-time jobs.

Data Integrator interface

A program that Data Integrator uses to access data sources. Specific interfaces vary by installation. There are internal interfaces—those native to the installation—and external interfaces—those that you install separately. Internal interfaces allow Data Integrator to access applications like **SAP R/3** and **SAP BW**, **messages**, relational database systems, and **legacy systems**. An external interface is also known as an **adapter**. It allows Data Integrator to access applications using information exchange technologies such as Web Services and MQ Series.

Data Integrator repository

The database that contains information about an Data Integrator application. The repository contains information about defined **reusable objects**, the **metadata** for tables, and built-in **transforms** and **functions**. When you invoke Data Integrator, you log in to the repository containing the **objects** you want to use.

Data Integrator Service

The process that ensures that the **Access Server** and the **Job Server** are running. You can configure the Data Integrator Service to restart the Access Server and Job Server whenever the computer where they are located restarts.

data loading

The process of populating a **data warehouse**. Data loading is provided by **DBMS**-specific load processes, DBMS insert processes, and independent fast-load processes.

data mapping

The process of assigning a source data element to a target data element.

data mart

A highly-focused version of a **data warehouse**. Typically, created by a department or division of a company, data marts contain data for a specific subject area, such as finance or sales. Data Integrator can populate a data mart.

data transformation

The process of filtering, merging, decoding, and translating source data to create validated data for a **data warehouse**.

data set

Rows of data with a defined schema. A step in a **data flow**—such as reading data from a **source**, joining data in a **Query transform**, or transforming data though another **transform**—yields a data set. You can view individual data sets by placing a **target** table or file at that point in the **data flow**.

data transport

A step in an **R/3 data flow** that defines a **target** to store the **data set** extracted during the flow. You can locate the target file on the **SAP R/3** server or in a location accessible to both the SAP R/3 server and to Data Integrator across a network.

data type

The format used to store a value. Data types can imply a default format for displaying and entering the value. Data read from a **source** is converted to the appropriate Data Integrator data types; data loaded to a **target** is converted from its Data Integrator data type to the type appropriate for the target.

data warehouse

A relational **database** containing subject-oriented information that supports decision analysis. Typically, information in a data warehouse is integrated from multiple sources, including operational sources. Data warehouses provide a single, non-volatile point of reference for a particular time period. Data warehouses are composed of **data marts.** Data Integrator can populate a data warehouse.

database

A collection of **table**s managed by a **DBMS** such as Microsoft SQL Server or Oracle.

DataConnector

DataConnector operator instances are used to read data files generated by Data Integrator when performing bulk loading using the Teradata Warehouse Builder.

datastore

A logical channel connecting Data Integrator to a **database**. The datastore definition includes the name and location of the database as well as user authentication information. Data Integrator uses a datastore definition to qualify a table name wherever a table is indicated in a diagram or expression. You can access the datastore definition through the **object library**.

DBMS

Database management system. A software system that builds and maintains database tables.

debug mode

Allows you to debug a job using the **interactive debugger** features in the Designer.

declarative specification

A way of indicating the desired results of a **data transformation** without placing constraints on the method of reaching the results. When you create a **data flow** in Data Integrator, you specify what data you want to read from which **source**, what you want done to that data, and where you want it loaded. What you specify constitutes the declarative specification. Data Integrator then determines the most efficient way to reach the requirements you set, including pushing operations to the source **DBMS** for execution.

degree of parallelism

You can set a value at the data flow level that allows you to parallel-process a **transform** within a data flow.

For example, if from the Properties window of a data flow you set the **Degree of parallelism** to 5, then when the job executes, Data Integrator replicates each transform in the data flow five times. Each of these replicated transforms executes in parallel using a separate process thread. Using a value for degree of parallelism can improve job performance if a **Job Server**'s computer memory and number of CPUs supports a job's parallel-processing configuration settings.

dependency analysis

A category of metadata reports that produces reports based on search criteria you specify to find specific objects in the repository. For each object returned, dependency analysis reports provide object lineage as well as object impact information.

derived data

Data that results from a computational step applied to reference or event data. Derived data is the result of either relating two or more elements of a single transaction (such as aggregation), or relating one or more elements of a transaction to an external algorithm or rule.

Designer

A graphical user interface that allows you to design and test an **Data Integrator application**.

diagram

The icons and connections between the icons that make up the definition of a **job**, **work flow**, or **data flow**.

DMZ

Demilitarized zone. A perimeter network that is more accessible than your most protected network, but located inside a security layer protected from the internet.

drill down

A method of exploring detailed data that was used in creating a summary level of data. Drill down levels depend on the granularity of the data in the **data warehouse**.

DTD

Document type definition. A text file that describes the elements (tags) in an **XML** document and the relationship among them. When an XML document is used to describe a transaction, the DTD describes the data schema used in the transaction.

e-commerce

Buy and sell transactions conducted over the internet.

EDI

Electronic Data Interchange. Electronic exchange of structured data between businesses. This exchange is not dependent on hardware, software, or communication protocols.

element

A component found within XML Schemas and DTDs.

embedded data flow

A data flow created within one or more data flows.

enterprise data

Data that is defined for use across a corporate environment.

ERP system

Enterprise resource planning system. A **back-office application** from which Data Integrator can extract data.

exception

An error that occurs while executing a **job**. You can catch individual or groups of exceptions using a **try/catch block** inside a **work flow**. Catching an exception allows you to automatically execute a solution for the error.

Executive Information Systems (EIS)

Tools programmed to provide canned reports or briefing books to executives. They offer strong reporting and drill-down capabilities. These tools allow ad hoc querying against a multi-dimensional **database**, and most offer analytical applications along functional lines such as sales or financial analysis.

extract date

The date data was extracted.

extract frequency

The interval at which data is extracted, such as daily, weekly, monthly, or quarterly. The frequency that data extracts are needed in the **data warehouse** is determined by the shortest frequency requested through an order, or by the frequency required to maintain consistency of the other associated data types in the source data warehouse.

file

A **source** or **target** for data. To read or load data to a file within Data Integrator, define a **file format** that instructs Data Integrator about the structure of the data in the file.

file format

A description of how data is or should be organized in a **file** Data Integrator reads from or loads to. A file format can be specific to a single file or generic for many files.

front-office application

Software that companies use to support reporting, **e-commerce**, and other tasks. Types of front-office applications include:

- Supply chain collaboration
- Analytical
- Sell-side e-commerce
- Customer relationship management

function

A program that operates on values that are passed to it. Data Integrator functions are available through a function wizard in a **script**, **conditional**, or **Query transform**. Data Integrator also gives you access to functions provided by the **DBMS** you are using. In addition, you can define your own functions using the Data Integrator scripting language or by writing C-language programs.

IDoc

Intermediate Document. An SAP-specific format. Used for **EDI** and **ALE**.

IDoc type

Indicates the SAP format that is used to interpret the data of a business transaction. Consists of the following components:

- A control record

  Identical for each IDoc type.

- Several data records

  A single data record consists of a fixed key part and a variable data part. The data part is interpreted using segments, which differ depending on the IDoc type selected.

- Several status records

  Identical for each IDoc type. Describe the status states an IDoc has already passed through or the status an IDoc has attained.

import

The process of acquiring information for the **Data Integrator repository**. Import the following kinds of information into Data Integrator:

- The **metadata** for **source** and **target** databases
- Descriptions and code for user-defined and DBMS **functions** and **transforms**

InfoArea

In **SAP BW**, an arbitrary grouping of **InfoCube**s.

InfoCube

In **SAP BW**, a multidimensional star schema that is limited to one fact table and 16 dimensions.

InfoCubes store only numeric data at the fact level.

SAP BW reserves three of the dimensions for currencies, time, and units of measure. Dimensions have to be linked back to master data and hierarchy tables to gain access to the information stored there.

When a user query is initiated, BW creates a temporary **OLAP** cube that stores a constrained subset of the permanent OLAP cube. The temporary InfoCube persists until the user disconnects from BW or changes the query.

InfoObject

In **SAP BW**, an InfoObject is equivalent to a data element and its associated metadata (for example, description or business rules). Can be a **Key Figure** or a Characteristic.

InfoPackage

In **SAP BW**, a schedule for running BW updates.

Information resource (IR)

A producer or consumer of enterprise information. An information resource can be a flat file, a **back-office application**, an EAI system, or a Web application.

InfoSource

In **SAP BW**, a logical construct that defines a set of **InfoObject**s and their mappings to source data elements. InfoSources do not store data, but rather serve as conduits via **Transfer Structure**s.

Integration Rapid Mart

An **Data Integrator application** that provides point-to-point content mapping and data movement mechanisms between different **back-office applications**.

interactive debugger

A Designer feature that allows you to step through the data of a job one row at a time using filters and breakpoints on a line. Like executing a job, you can start the interactive debugger from the Debug menu when a job is active in the workspace. While in debug mode, all other Designer features are set to read-only.

interface

Data Integrator offers two types of interfaces:

An internal **Data Integrator interface**, allows you to create datastore connections to natively supported applications.

An external **Data Integrator interface** (or **adapter**), allows Data Integrator to communicate with information exchange technologies such as Web Services and MQ Series.

IR

See **Information resource (IR)**.

job

The unit of work that can be scheduled independently for execution by the **Administrator**. Jobs are special **work flows** that can be called by other work flows or jobs.

Job Server

A process that receives requests from the **Designer** and the **Administrator** to start and stop **jobs**. To start batch or real-time jobs, the Job Server triggers the **Data Integrator engine**. Engine processes run on the same computer as the Job Server process that triggers them.

join rank

A value given to or calculated for all data sets in a **data flow**. Data Integrator uses the join rank to determine which **source** to read first when assembling the **data set** in a join. Data Integrator uses the source with the lower join rank as the inner source of the join and uses the source with the higher join rank as the outer source of the join. If more than one of the sources is located in the same **DBMS**, the highest rank is used for all of the sources from that DBMS. When a source is processed through a **Query transform** or another **transform**, and then used in a join, the maximum join rank defined for the source is used as the join rank in the query.

Key Figure

In **SAP BW**, an **InfoObject** that represents a numeric fact.

legacy system

An information or transaction processing system used to store data such as bank balances, inventories, payroll, and manufacturing parts.

license-controlled feature

A Data Integrator feature that is enabled or disabled based on the product license. The product license controls which icons and settings are available in Data Integrator as an internal **Data Integrator interface**.

line

The line you use to connect objects in a workspace diagram. Lines represent a left-to-right flow path for data moving between two objects during job execution.

memory datastore

A datastore connection/container for memory tables.

memory table

Internal Data Integrator table used to store a data set in memory while a job runs. Use instead of staging tables to improve performance of a real-time job built with multiple data flows. Use a memory table to move a data set between data flows.

message

Represents hierarchical data, such as a header with line items, for document-oriented transactions, such as a purchase order.

metadata

In Data Integrator, information acquired and maintained to describe tables in **source** and **target** databases. This information includes the names of tables and their columns, and the data types of the columns. All the metadata for all user tables is imported at one time for a given **datastore**.

In general, metadata typically includes a description of data models, a description of the layouts used in database design, the definition of the system of record, the mapping of data from the system of record to other places in the environment, and specific database design definitions.

multiple data flow model

One of three data flow models you can use in real-time jobs. Use this model to develop real-time jobs in separate logical object segments (data flows) rather than in a single data flow. Allows for any number of data flows in a real-time processing loop under these conditions: The first data flow must have a single message source, the last data flow must have a single message target. Other data flows in the same real-time job cannot have a message source or message target. See also **put/acknowledge data flow model** and **single data flow model**.

nested data

Data in one **table** that is related to a single row of another table. A nested table appears in Data Integrator as a column in a parent table. Columns in the nested table can themselves contain tables.

object

Any item that you create in the **Designer**. Data Integrator distinguishes two classes of objects: **reusable objects** that are complete and can be reused in your projects (such as **data flows**) and **single-use objects** that only appear as components of other objects (such as a **try/catch block**). This distinction affects how you create and retrieve each type of object.

object definition

The options that describe the operation of an **object**. To view and modify an object definition, open the object so that its definition appears in the **workspace**.

object library

A tool in the **Designer** that gives you access to reusable objects.

ODBC

Open database connectivity. A standard for **database** access initiated by the SQL Access Group consortium and now owned by Microsoft.

ODS

Operational data store. An OLAP-designed relational **database** that an enterprise has designated as the operational database of record (for example, a finance department might use an ODS to close its books).

OLAP

Online analytical processing. A relational **database** design that is optimized for random access. OLAP systems are used in a query environment, such as for a **front-office application**.

OLTP

Online transaction processing. A relational database design optimized for operational use. OLTP systems are used in an operational environment, such as for a **back-office application**.

operation code

A flag associated with a row in a **data set** that indicates the status of the data in the row. The operation codes are INSERT, UPDATE, DELETE, and NORMAL.

operational statistics

A category of reports within the metadata reporting tool that provides you with job and data flow performance information over a given period of time or number of executions.

parameter

A value passed to a **work flow** or **data flow** when that flow is called.

partition

To divide table data into sets based on a criteria such as a range or list of values in each row. You can configure Data Integrator to read and write partitioned table data in parallel threads. Designing jobs with partitioned table data can improve job performance if a Job Server's computer memory and number of CPUs supports the job's parallel-processing configuration settings.

project

The collection of **jobs** available in the **Designer** at a given time. A project provides a way to organize the **objects** you create.

property

Detailed descriptive information about **objects**.

put/acknowledge data flow model

One of three data flow models you can use in real-time jobs. Similar to the multiple data flow model, this model does not require a data flow with a target message. Data Integrator creates a target message to automatically acknowledge a request message. Designed for use with IDoc sources in SAP R/3 systems. See also **multiple data flow model** and **single data flow model**.

Query transform

A **data transformation object** that you can use to map columns from a **source** to a **target** schema, add new columns to the target schema, determine the data to extract, and perform operations on the data. Like a SQL SELECT statement, a query creates a **data set** that satisfies the conditions you specify.

Rapid Mart

A prepackaged **Data Integrator application** that delivers data for a specific business purpose. Easily deployed and customized, BusinessObjects™ Rapid Marts include the extraction, transformation, and data loading logic to support analysis or **e-commerce**. There are three types of Rapid Mart:

- **Analytic Rapid Mart**
- **Commerce Rapid Mart**
- **Integration Rapid Mart**

R/3 data flow

A **data flow** that extracts data from an **SAP R/3** source table. Data Integrator translates steps you define in an R/3 data flow into **ABAP** and then passes the ABAP program back to your SAP R/3 system for execution. The resulting **table** or **file** resides on the SAP R/3 system to be used as a **source** in the parent data flow.

real-time job

A group of objects (data flows, work flows, conditionals, scripts, and so forth) that execute on-demand as a "request-response" system. You design real-time jobs in the **Designer**, then configure them as real-time services and associate them with an **Access Server** in the **Administrator**, where they are started, managed and monitored. When a real-time service receives a request from a caller, it processes the request and returns a reply. This type of job is recommended for operations on small amounts of data.

relational data

A **data set** in which data in each column contains a scalar value. Data Integrator can process relational data; it can also process **nested data**.

repository

See **Data Integrator repository**.

reusable object

An **object** that can be defined, stored, and reused independent of other objects. Create reusable objects from the **object library**.

RFC server

Remote Function Call server. The Data Integrator RFC server allows third-party programs, including SAP R/3, to schedule and initiate Data Integrator jobs and return the results to Data Integrator.

SAP BW

Business Information Warehouse, an SAP product.

SAP R/3

An **ERP system**.

script

A step in a **work flow** that allows you to calculate values to pass to other parts of the work flow. The script can call **function**s, execute if-then-else statements, and assign values to **variable**s. Write a script in the Data Integrator scripting language.

segment

Format with which the data records of **IDocs** are interpreted.

server group

A defined collection of Job Servers on different computers. A server group automatically measures resource availability on each Job Server in the group and distributes scheduled batch jobs to the Job Server with the lightest load at run-time. Use the Server Groups node in the Administrator's navigation tree to group Job Servers that are associated with the same repository into a server group.

service request

Any message sent from a Web client that requires processing by a **real-time job**.

session

Associated with the **Adapter Properties** tab in an adapter datastore, a session is created by a stream-oriented adapter to group adapter components and maintain state for interactivity between Data Integrator and the adapter.

single data flow model

One of three data flow models you can use in real-time jobs. Allows for one data flow in a real-time processing loop. This data flow must have a single message source and a single message target. See also **multiple data flow model** and **put/acknowledge data flow model**.

single-use object

A step in a **work flow** or **data flow** that cannot be saved independently of the flow. Create single-use objects from the tool palette.

smart editor

A flexible editing tool in Data Integrator used for creating scripts, expressions, and custom functions without having to type the names of existing elements like column, function, and variable names.

SNMP

System Network Management Protocol helps network administrators manage network routing hardware. The protocol can manage a variety of hardware and software devices.

source

An **object** from which Data Integrator reads data.

SQL

Structured Query Language. A query language for accessing relational, **ODBC**, DRDA, or non-relational **database** systems.

SQL query tool

An end-user tool that accepts **SQL** to be processed against one or more relational databases.

star schema

A **database** design you can use to format data in a **data mart**. This design is based on a single fact table to which any number of dimensional tables may be joined. This type of database design supports multi-dimensional database analysis.

step

An **object** that is part of the definition of a **work flow** or **data flow**. Each step is represented by an icon in the diagram of the flow and is connected to other steps to indicate the flow of data through the data flow or the order of execution in the work flow.

table

A **database** table that Data Integrator reads data from or loads data into. The path and mechanisms for reading and loading data and apportioning the data among rows and columns are defined in the **datastore** that the table is associated with. Writing a **data set** to a database table means sending a combination of rows with appropriate **operation code** to the database table.

target

An object in which Data Integrator loads extracted and transformed data in a **data flow**. Data Integrator only loads rows flagged as INSERT or UPDATE.

TCP/IP

Transmission Control Protocol/Internet Protocol. The basic communication protocol of the internet, and often intranets and extranets. A computer having direct access to the internet contains a copy of the TCP/IP program. TCP/IP makes it possible for computers to communicate with each other.

Tdpid

(Teradata Director Program ID) The server name Data Integrator uses when loading with the bulk loader option. Data Integrator uses tdpid as a Teradata Warehouse Builder operator attribute.

Transfer Structure

In **SAP BW**, a data structure that defines a set of **InfoObjects** to be moved into BW. To an ETL tool like Data Integrator, a Transfer Structure looks like a table.

transform

A step in a **data flow** that acts on a **data set**. Data Integrator transforms are available through the **object library**. You can also import user-defined transforms and transforms from your **DBMS**.

Transformation Rule (Transfer Rule)

In **SAP BW**, a data manipulation rule between a **Transfer Structure** and a **Communication Structure**. The results of these transformation are globally available to all **InfoCubes**.

try/catch block

A combination of a try object and one or more catch objects that define alternate execution paths in case an error occurs during the execution of a **job**. You can tune try/catch blocks to trap specific errors and to provide general alerts or messages if an error occurs.

Update Rule

In **SAP BW**, a data manipulation rule between a **Communication Structure** and an **InfoCube**. The result of the data manipulation is specific to one InfoCube.

variable

A symbolic placeholder for a value. Data Integrator lets you define local variables and global variables.

Local variables pertain to the **work flow** in which they are defined. You can pass the value into another **work flow** or **data flow** using a **parameter**.

Global variables pertain to the **job** in which they are defined. With global variables, there is no need to define parameters between **objects** in the job. Global variables can also be selected at execution time. This eliminates the need to open the Designer to set global variable values.

Web Services

A standard platform for integrating applications. Web Services allow different programs, constructed in different languages, on different platforms to communicate with each other. DATA INTEGRATOR supports three XML web service technologies: SOAP, WSDL, and **XML Schema**.

work flow

A **reusable object** containing **steps** to define the order of **job** execution. Work flows call **data flow**s, but cannot manipulate data themselves.

You call work flows from inside other work flows or **job**s. You pass information into or out of work flows using **parameter**s. You reuse work flows by dragging existing work flows from the **object library**.

workspace

The window inside the **Designer** in which you define, display, and modify **objects**. The workspace for a **data flow** contains an area to build a **diagram** representing the data flow definition. The workspace for a **transform** contains an editor for modifying the transform options.

WSDL

Web Services Definition Language. Web services are self-contained, modular business process applications based on open Internet standards.

XML

Extensible Markup Language. This markup language is like HTML (Hypertext Markup Language) in that it specifies a standard with which you can define your own markup languages with their own sets of tags. XML allows you to define various tags with various rules, such as tags that represent business rules, tags that represent data description, or tags that represent data relationships.

XML Schema

The XML format used by Data Integrator to support message processing that includes **Web Services**. XML Schemas describe the data structure of an **XML** file or message. Data flows can read and write data to messages or files based on a specified XML Schema format. You can use the same XML Schema to describe multiple XML sources or targets. XML Schema properties include: Name, Description, Imported from, Root element name, and Namespace.

Index

S

Business Objects, Americas
3030 Orchard Parkway
San Jose, CA 95134
USA

info@businessobjects.com
www.businessobjects.com

Data Integrator

Performance Optimization Guide

Version 6.5

Document Number DI-65-0115-001

January 30, 2004

Contents

1 Welcome

Welcome to the *Data Integrator Performance Optimization Guide*.

Data Integrator allows you to create jobs that extract, transform, and load data from database management systems into a data warehouse. After successfully designing and testing these jobs, you might wonder:

- With my current hardware and software environment, how can I configure the Data Integrator system to achieve optimal performance? What rules do I need to follow?
- How many CPUs and how much physical and virtual memory do I need? What might cause the system to bottleneck?
- How will the performance of Data Integrator be affected by options such as join ordering, caching tables, bulk loading, and parallel execution?

These questions expose potential configuration and tuning issues with database servers and operating systems, as well as with Data Integrator. This book provides guidelines for these issues. Chapters include:

- Environment Test Strategy
- Measuring and Tuning Data Integrator Performance
- Using Bulk Loading
- Using Parallel Execution

Audience and assumptions

This and other Data Integrator product documentation assumes the following:

- You are an application developer, consultant or database administrator working on data extraction, data warehousing, or data integration.
- You understand your source and target data systems, DBMS, legacy systems, business intelligence, and messaging concepts.
- You understand your organization's data needs.
- You are familiar with SQL (Structured Query Language).
- If you are interested in using this product to design real-time processing you are familiar with:
  - DTD and XML Schema formats for XML files
  - Publishing Web Services (WSDL, HTTP/S and SOAP protocols, etc.)
- You are familiar with Data Integrator installation environments: Microsoft Windows or UNIX.

More Data Integrator product documentation

Consult the *Data Integrator Getting Started Guide* for:

- An overview of Data Integrator products and architecture
- Data Integrator installation and configuration information
- A list of product documentation and a suggested reading path

After you install Data Integrator, you can view technical documentation from many locations. To view documentation in PDF format, you can:

- Select **Start** > **Programs** > **Data Integrator** ***version*** > **Data Integrator Documentation** and choose:
  - Release Notes
  - Release Summary
  - Technical Manuals
- Select one of the following from the Designer's **Help** menu:
  - Release Notes
  - Release Summary
  - Technical Manuals
- Select **Help** from the Data Integrator Administrator

You can also view and download PDF documentation by visiting Business Objects Customer Support online. To access this Web site, you must have a valid user name and password. To obtain your user name and password, go to http://www.techsupport.businessobjects.com and click **Register**.

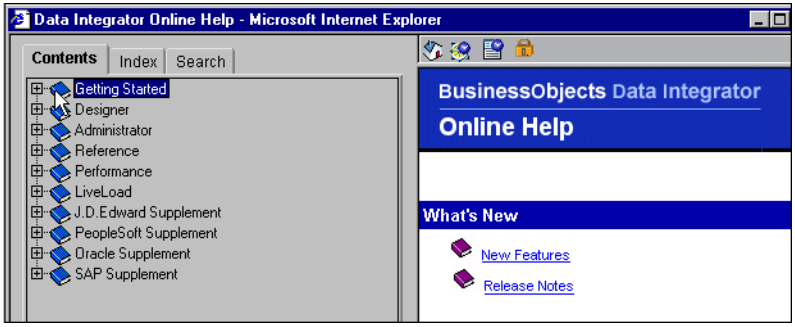
After installing Data Integrator Designer you can also view documentation in HTML format. **Online Help** opens automatically the first time you open the Designer.

You can also open **Help,** using one of the following methods:

- Choose **Contents** from the Designer's Help menu.
- Click objects in the object library or workspace and press **F1**.

  Online Help opens to the subject you selected.

Use **Online Help's** links and tool bar to navigate:



Online Help tool bar

Icon	Icon Name	Description
	Home	Returns to the Online Help home page.
	Show/Hide Navigation	Shows and hides the Table of Contents for the core suite of Data Integrator Technical Manuals. Includes a master **Index** and **Search** tabs.
	Show/Hide Descriptions	The Online Help window automatically updates its information when you click an object in the Designer window. In addition, you can click any object in the Designer window then click this button to show or hide an object's local and global variables.
	Start/Stop Help Updates	When the lock appears open, each time you click an object in the Designer window, the Online Help information automatically updates.
		When the lock icon appears closed, the Online Help window does not automatically update when you click objects in the Designer window; you must press F1.

2 Environment Test Strategy

This chapter covers suggested methods of tuning source and target database applications, their operating systems, and the network used by your Data Integrator environment. It also introduces two key Data Integrator connection options.

This chapter contains the following sections:

- The source OS and database server
- The target OS and database server
- The network
- Data Integrator Job Server OS and job options

To test and tune Data Integrator jobs, work with all four of these components in the order shown above.

In addition to the information in this chapter, you can use your UNIX or Windows operating system and database server documentation for specific techniques, commands, and utilities that can help you measure and tune the Data Integrator environment.

The source OS and database server

Tune the source operating system and database to quickly read data from disks.

Operating system

Make the input and output (I/O) operations as fast as possible. The read-ahead protocol, offered by most operating systems, can greatly improve performance. This protocol allows you to set the size of each I/O operation. Usually its default value is 4 to 8 kilobytes which is too small. Set it to at least 64K on most platforms.

Database

Tune your database on the source side to perform SELECTs as quickly as possible.

In the database layer, there are several ways to improve the performance of SELECTs:

- Increase the size of each I/O from the database server to match the OS read-ahead I/O size.
- Increase the size of the shared buffer to allow more data to be cached in the database server.
- Cache tables that are small enough to fit in the shared buffer. For example, if jobs access the same piece of data on a database server, then cache that data. Caching data on database servers will reduce the number of I/O operations and speed up access to database tables.

  A table is considered small if its size is less than 1% of physical memory. For information about how to calculate table size, see "Caching tables" on page 20.

The target OS and database server

Tune the target operating system and database to quickly write data to disks.

Operating system

Make the input and output operations as fast as possible. For example, the asynchronous I/O, offered by most operating systems, can greatly improve performance. Turn on the asynchronous I/O.

Database

Tune your database on the target side to perform INSERTs and UPDATES as quickly as possible.

In the database layer, there are several ways to improve the performance of these operations.

Here are some examples from Oracle:

- Turn off archive logging
- Turn off redo logging for all tables
- Tune rollback segments for better performance
- Place redo log files and data files on a raw device if possible
- Increase the size of the shared buffer

The network

When reading and writing data involves going through your network, its ability to efficiently move large amounts of data with minimal overhead is very important. Do not underestimate the importance of network tuning (even if you have a very fast network with lots of bandwidth).

Set network buffers to reduce the number of round trips to the database servers across the network. For example, adjust the size of the network buffer in the database client so that each client request completely fills a small number of network packets.

Data Integrator Job Server OS and job options

Tune the Job Server operating system and increase **Array fetch size** and **Rows per commit** to take advantage of source, target, and network performance.

Operating system

Data Integrator jobs are multi-threaded applications. Typically a single data flow in a job initiates one `al_engine` process that in turn initiates at least 4 threads.

For maximum performance benefits:

- Consider a design that will run one `al_engine` process per CPU at a time.
- Tune the Job Server OS so that Data Integrator threads spread to all available CPUs.

For more information, see "Checking system utilization" on page 12.

Data Integrator jobs

Set the parameters for **Array fetch size** and **Rows per commit** after:

- Tuning the database and operating system on the source and the target computers
- Adjusting the size of the network buffer
- Your data flow design seems optimal

In Data Integrator jobs, slowly increase the **Array fetch size** for sources and **Rows per commit** value for targets until they reach the CPU use capacity you want.

- Generally, you should set **Rows per commit** to at least 1000 (or higher if you have a very fast network). Do not force the database server to commit too often because this degrades performance. For more information, see "Loading method and rows per commit" on page 26.
- To determine **Array fetch size,** consider the speed of the source database server and the network speed.
- Network speed plays a very critical role in tuning the array fetch size. For more information, see "Using array fetch size" on page 18.

3 Measuring and Tuning Data Integrator Performance

This chapter contains the following sections:

Measuring performance

You can use several techniques to measure performance:

- Analyzing trace log files for task duration
- Checking system utilization
- Reading performance statistics
- Using Metadata Reports

Analyzing trace log files for task duration

The trace log shows the progress of an execution through each component of a job.

For information about accessing, viewing, and managing Data Integrator logs, see "Log" on page 75 of the *Data Integrator Reference Guide*.

Checking system utilization

To monitor process, thread, and memory information, use the following tools:

For UNIX	TOP or a third party utility (such as Glance for HPUX)
For Windows	Task Manager

Data Integrator processes

The processes Data Integrator uses to run jobs are:

- al_jobserver.ex

  The `al_jobserver.ex` initiates one process for each Job Server configured on a computer. This process does not use much CPU power because it is only responsible for launching each job and monitoring the job's execution.

- al_engine

  For batch jobs, an `al_engine` process runs when a job starts and for each of its data flows. Real-time jobs run as a single process.

  The number of processes a batch job initiates also depends upon the number of:

  - parallel work flows
  - parallel data flows

Because most computer systems do not have the CPUs and memory proportionate to CPU count to take advantage of more than eight simultaneous processes, Business Objects has set the default maximum number of processes that Data Integrator can run in parallel to `8`. However, in the Designer you can adjust this value by going to **Tools > Options > Job Server > Environment** and entering a new number in the **Maximum number of engine processes** text box.

Note that if you have more than four CPUs on your Job Server computer, you can increase this number to improve performance.

Data Integrator threads

A simple data flow typically initiates one `al_engine` process, which initiates at least four concurrent threads. For example, two sources, a query, and a target could initiate four threads.

If you are using parallel objects in data flows, the thread count will increase to approximately one thread for each source or target table partition and it will equal the number used to set the **Degree of parallelism** (DOP) option for your data flow. For example, a DOP of `5` allows five concurrent threads. To run objects within data flows in parallel, use the following Data Integrator features:

- Table partitioning
- File multithreading

- Degree of parallelism for data flows

For more information see, "Using Parallel Execution" on page 59.

Data Integrator is designed to maximize the use of CPUs and memory available to run the job.

The total number of concurrent threads a job can run depends upon job design and environment. Test your job while watching multi-threaded Data Integrator processes to see how much disk space and memory the job requires. Make needed adjustments to your job design and environment and test again to confirm improvements.

For example, if you run a job and see that only half a CPU is being used, or if you run eight jobs on an eight-way computer and CPU usage is only 50%, you can be interpret this as good or bad:

- The good news might be that Data Integrator is able to push most of the processing down to source and/or target databases.
- The bad news might be that there are bottlenecks in the database server or the network connection. Bottlenecks on database servers do not allow readers or loaders in jobs to use Job Server CPUs efficiently.

  To determine bottlenecks, examine:

  - Disk service time on database server computers

    Disk service time typically should be below 15 milliseconds. Consult your server documentation for methods of improving performance. For example, having a fast disk controller, moving database server log files to a raw device, and increasing log size could improve disk service time.

  - Number of threads per process allowed on each database server operating system. For example:

- On HPUX, the number of kernel threads per process is configurable. The CPU to thread ratio defaults to one-to-one. Business Objects recommends setting the number of kernel threads per CPU to between 512 and 1024.
- On Solaris and AIX, the number of threads per process is not configurable. The number of threads per process depends on system resources. If a process terminates with a message like "Cannot create threads," you should consider tuning the job.

  For example, split a data flow and design two data flows to execute sequentially. Since each data flow is executed by a different Data Integrator `al_engine` process, the number of threads needed for each will be 50% less than in your previous job design.

  If you are using the **Degree of parallelism** option in your data flow, reduce the number for this option in the data flow Properties window.

◆ Network connection speed

See "Using array fetch size" on page 18.

Data Integrator memory

Occasionally you might notice that memory allocated for Data Integrator a process continues to grow without shrinking until exit time. As Data Integrator allocates memory for different operations, this memory is not necessarily released to the operating system when it is freed.

Whether or not the memory is released depends on the allocation order. For example, assume memory is allocated at address 12345, then at 12370. If 12345 is freed first, it is not released until 12370 is freed. However, if Data Integrator needs to allocate memory again, it will first try to use one of the previously allocated memory spaces (like 12345) that has been freed but not released.

Virtual memory usage should not be growing to peak. If there is very little virtual memory left, paging is occurring. To improve performance, increase the system's physical memory, or redesign your job to take advantage of your overall environment.

For example, if you are reading data from a table, joining it to a file, and then grouping it to calculate an average, the grouping operation might be occurring in memory. If you stage the data after the join and before the grouping into a database on a different computer, then when a separate data flow reads the staged data and continues with the group processing, it can utilize memory from the database server on a different computer. This situation optimizes your system as a whole.

Reading performance statistics

The monitor log file indicates how many rows Data Integrator produces or loads for a job.

By viewing this log, during job execution you can observe the progress of row-counts to determine the location of bottlenecks. Since Data Integrator buffers at most 600 rows between stages, the stage where counts are behind by approximately 600 rows is the bottleneck.

For example, if a data flow has two transforms serially connected (`transform1` and `transform2`), and the log indicates that `transform1` has produced 5,000 rows and `transform2` lags behind with 4,400 rows, `transform2` is the bottleneck.

Since the buffer between these transforms fits only 600 rows, `transform1` must to stop producing rows until `transform2` catches up and closes the row-count lag to under 600. Data Integrator passes subsequent rows one at a time which is detrimental to job performance.

For information about accessing, viewing, and managing Data Integrator logs, see, "Log" on page 75 of the *Data Integrator Reference Guide*.

Using Metadata Reports

The Metadata Reports feature, accessible from the **Tools** menu in Designer, provides several operational statistics reports that allow you examine performance-related execution information for jobs and data flows.

You can view text and graphical reports for:

- All executions of a specified job
- The last execution of the most currently run jobs
- All executions of a specified data flow
- The last execution of the most currently run data flows

Operational statistics reports contain execution information for the last 15 days or the last 15 executions. You can use operational statistics reports to answer some of the following questions:

- Which jobs have consistent performance characteristics?
- Which jobs have degraded performance over time?
- Which data flows are bottlenecks?
- What is my data throughput?

For more information, see Chapter 17, "Metadata Reporting," in the *Data Integrator Designer Guide*..

Tuning techniques

The following sections describe ways you can adjust Data Integrator performance:

- Source-based performance options
  - Using array fetch size
  - Caching data
  - Join ordering
  - Minimizing extracted data
- Target-based performance options
  - Loading method and rows per commit
  - Staging tables to speed up auto-correct loads
- Job design performance options
  - Improving throughput
  - Maximizing the number of pushed-down operations
  - Minimizing data type conversion
  - Minimizing locale conversion

These techniques require that you monitor the change in performance carefully as you make small adjustments.

Using array fetch size

Data Integrator provides an easy way to request and obtain multiple data rows from source databases. The array fetch size feature allows you to retrieve data using fewer requests, thereby significantly reducing traffic on your network. Tuning array fetch size can also reduce the amount of CPU use on the Job Server computer.

The Data Integrator array fetch feature lowers the number of database requests by "fetching" multiple rows (an array) of data with each request. Enter the number of rows to fetch per request in the **Array fetch size** option on any source table editor. The default setting is 1000, meaning that with each database request, Data Integrator will automatically fetch 1000 rows of data from your source database. Business Objects recommends that you set the array fetch size based on network speed. For example:

- For a 10M-base network, set **Array fetch size** to `1000`.
- For a faster networks, set it higher and test. For example, with 1GB fiber-channel network connections, and the array fetch size set to 10,000 we saw up to a 300% performance improvement for some jobs in internal tests.

Regardless of the array fetch setting, sources reading columns with an Oracle LONG data type cannot take advantage of this feature. If a selected data column is of type LONG, the array fetch size internally defaults to `1` row per request.

NOTE: Although Data Integrator does not specify an upper array fetch limit, there could be an upper limit enforced by the source database. Therefore, check your source database documentation for more information. Also, note that higher array fetch settings will consume more processing memory proportionally to the length of the data in each row and the number of rows in each fetch.

➢ To set the Array fetch size parameter

1. In the Designer's workspace, open a source table to view its schema.
2. In the **Performance** section of the **Source** tab, enter a number in the **Array fetch size** text box.

   Array Fetch Size indicates the number of rows returned in a single fetch call to a source table. The default value is `1000`. This value reduces the number of round-trips to the database and can improve performance for table reads.

The **Array Fetch Size** option does not support long column data types. If the `SELECT` list contains a long column, Data Integrator sets the **Array Fetch Size** to `1` and reads one row of data at a time from the database.

3. Click **OK**.

Tip

The optimal number for **Array fetch size** depends on the size of your table rows (the number and type of columns involved) as well as the network round-trip time involved in the database requests and responses. If your computing environment is very powerful, (meaning that the computers running the Data Integrator Job Server, related databases, and connections are extremely fast), then try higher values for **Array fetch size** and test the performance of your jobs to find the best setting.

Caching data

You can improve the performance of data transformations that occur in memory by caching as much data as possible. By caching data, you limit the number of times the system must access the database. Cached data must fit into available memory.

Select **Yes** in a source table or file editor to specify that data from the source is cached using memory on the Job Server computer.

Caching tables

Business Objects recommends that you apply table caching for small tables (typically with a table size of less than 1% of total physical memory). Calculate the approximate size of a table with the following formula:

```
table size =  # of rows *
(in bytes)    # of columns *
              20 bytes (average column size) *
              1.3 (30% overhead)
```

Compute row count and table size on a regular basis, especially when:

- You are aware that a table has significantly changed in size.
- You experience decreased system performance.

You can specify the number of rows in a table to override the default value Data Integrator uses during execution. To specify a table size, set the **Estimated row count** attribute for the table. The default row count value is 50,000.

➢ To enter table size information

1. In the object library, select the table name.
2. Right-click and choose **Properties**.
3. Click the **Attributes** tab.
4. Enter the approximate row count.

   Approximate row count cannot be zero or negative.

Caching joins

Cache a source only if it is being used as an inner join. (Inner joins have a lower join rank than outer joins). Caching does not affect the order in which tables are joined. If optimization conditions are such that Data Integrator is pushing down operations to the underlaying database, it ignores your cache setting.

If a table becomes too large to fit in the cache, disable caching for the table.

Caching lookups

You can also improve performance by caching data when looking up individual values from tables and files.

There are two methods of looking up data:

- Using a Lookup function in a query
- Using a source table and setting it as the outer join

Using a Lookup function in a query

Data Integrator has three Lookup functions: `lookup`, `lookup_seq`, and `lookup_ext`. The `lookup` and `lookup_ext` functions have cache options. Caching lookup tables improves performance because Data Integrator avoids the expensive task of creating a database query on each row.

You can set cache options when you specify a lookup function. There are three caching options:

- NO_CACHE — Does not cache any values.
- PRE_LOAD_CACHE — Preloads the result column and compare column into memory (it loads the values before executing the lookup).

  Use this option if the table can fit in memory.
- DEMAND_LOAD_CACHE — Loads the result column and compare column into memory as the function executes.

  Use this option when looking up highly repetitive values that are a small subset of the data and when missing values are unlikely.

  Demand-load caching of lookup values is helpful when the lookup result is the same value multiple times. Each time Data Integrator cannot find the value in the cache, it must make a new request to the database for that value. Even if the value is invalid, Data Integrator has no way of knowing if it is missing or just has not been cached yet.

  When there are many values and some values might be missing, demand-load caching is significantly less efficient than caching the entire table.

Using a source table and setting it as the outer join

Although you can use lookup functions inside Data Integrator queries, an alternative is to expose the translate (lookup) table as a source table in the data flow diagram, and use an outer join (if necessary) in the query to look up the required data. This technique has some advantages:

- You can graphically see the table the job will search on the diagram, making the data flow easier to maintain
- Data Integrator can push the execution of the join down to the underlying RDBMS (even if you need an outer join)

This technique also has some disadvantages:

- Workspace can become cluttered if there are too many objects in the data flow
- There is no option to use DEMAND_LOAD caching, which is useful when looking up only a few repetitive values in a very large table.

Tip

If their size can fit in memory, cache lookup tables. Data Integrator caches the keys used for the lookup and the lookup values, not the entire table.

Caching table comparisons

You can improve the performance of a Table_Comparison transform by caching the comparison table.

There are three modes of comparisons:

- Row-by-row select
- Cached comparison table
- Sorted input

Of the three, **Row-by-row select** will likely be the slowest and **Sorted input** the fastest.

Tip

If you want to sort the input to the table comparison transform, then choose the **Sorted input** option for comparison.

If the input is not sorted and the compare table is small enough to fit in memory, then choose the **Cached comparison table** option.

Join ordering

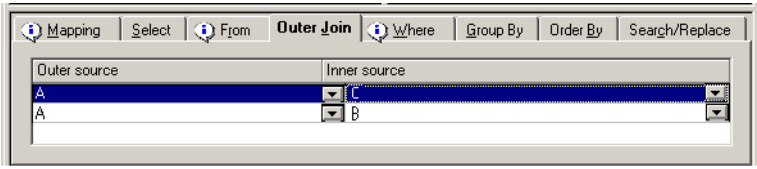
Join ordering is determined by the type of join you choose to use in Data Integrator. There are two types of joins:

- Normal — Created by drawing connection lines from multiple sources in a data flow to the same query and then entering an integer as a **Join rank** value for each source using source editors. Data Integrator orders join operations using these values.

  For a normal join between three or more tables, Data Integrator internally implements a series of two-way joins. First, it joins the two highest ranked tables together using the join ranking rules outlined above. Next, Data Integrator joins the result of the first join with the third highest join ranked table. This process continues until all tables are included in the join.

- Outer Join — Created by taking a normal join and then using the **Outer Join** tab of the query to specify the outer and inner sources. Data Integrator processes the outer source as if it was assigned the highest join rank.

  For an Outer join, Data Integrator ignores the **Join rank** values unless you are using three or more tables and a pair of joins share the same outer join table.



For example, if table A is an outer join to tables B and C, the join rank is calculated by taking the maximum join rank value of the two tables in each outer join pair (A and B and A and C). The pair with the higher join rank becomes the outer loop.

Data Integrator implements joins as nested loop joins. The source with the higher join rank or the one specified as an outer source becomes an outer loop. If all the ranks are equal or not set (defaults to `0`), then Data Integrator picks an inner and outer source at random. During job execution, Data Integrator reads the source in the outer loop one time; and reads the source in the inner loop for each row in the outer loop. Performance improves when fewer rows are read.

The number of output rows for a join depends on the type of join you use in a data flow.

- For a Normal join, the output is limited to the number of rows that meet join conditions.
- For an Outer join, the output is equal to or greater than the number of rows in the source specified as the **Outer Source**.

  Using an Outer join is often preferable with real-time jobs because often you want the whole message passed on whether or not conditions a join looks for in the inner source exist.

  In fact, if you do not use an Outer Join to order joins for a message, Data Integrator will still process the message as if it has the highest join rank. The message source editor lacks a **Join rank** option because Data Integrator automatically gives it the highest join rank.

For more information about join ranks and the other options available in a query transform, see, "Query" on page 287 of the *Data Integrator Reference Guide*.

Tips

For a join between two tables, assign a higher join rank value to the larger table and, if possible, cache the smaller table.

For a join between a table and file:

- If the file is small and can be cached, then assign it a lower join rank value and cache it.
- If you cannot cache the file, then assign it a higher join rank value so that it becomes the "outer table" in the join.

For a join between two files, assign a higher rank value to the larger file and, if possible, cache the smaller file.

Minimizing extracted data

The best way to minimize the amount of data extracted from the source systems is to retrieve only the data that has changed since the last time you performed the extraction. This technique is called *changed-data capture*, and it is described in detail in Chapter 19, "Techniques for Capturing Changed Data".

Loading method and rows per commit

You can choose to use regular loading or bulk loading with Data Integrator. However, you must use regular loading to use the following options:

- Auto-correct load
- Enable Partitioning
- Input-supplied keys
- Number of Loaders

- Operations pushed down to a database

  Data Integrator automatically selects this optimizer process when the following conditions are met:

  - The source and target in a data flow are on the same database
  - The database supports the operations in the data flow

  If the optimizer pushes down source or target operations, then it ignores the performance options set for sources (**Array fetch size**, **Caching**, and **Join rank**) because Data Integrator is not solely processing the data flow. For more information, see "Maximizing the number of pushed-down operations" on page 31.

- Overflow file
- Parameterized SQL

  Automatically selected by Data Integrator when it must generate, parse, and compile the statement. By using parameterized SQL, Data Integrator can minimize these efforts by using one handle for a set of values instead of one handle per value.

- Transactional loading

To improve performance, you can select the following options from the target table editor. Note that if you use one, you cannot use the others for the same target.

- Enable Partitioning

  Parallel loading option. The number of parallel loads is determined by the number of partitions in the target table. For more information, see "Table partitioning" on page 60.

- Number of Loaders

  Parallel loading option. The number of parallel loads is determined by the number you enter for this option.

- Transactional loading

This option allows you to commit data to multiple tables as part of the same transaction. For more information about Transactional loading, see "Include in transaction" on page 109 of the *Data Integrator Reference Guide*

For more information about bulk loading options, see "Using Bulk Loading" on page 37.

Rows per commit for regular loading defaults to 1000 rows. Setting the **Rows per commit** value significantly affects job performance. Adjust the rows per commit value in the target table editor's **Options** tab, noting the following rules:

- Do not use negative number signs and other non-numeric characters.
- If you enter nothing or `0`, the text box will automatically display `1000`.
- If you enter a number larger than `5000`, the text box automatically displays `5000`, which is the maximum value allowed.

Business Objects recommends setting rows per commit between `500` and `2000` for best performance. You might also want to calculate a value. To do this, use the following formula:

`max_IO_size/row size` (in bytes)

For most platforms, `max_IO_size` is 64K. For Solaris, `max_IO_size` is 1024K. For a formula to calculate row size, see "Caching tables" on page 20.

Note that even with a value greater than one set for **Rows per commit**, Data Integrator will submit data one row at a time if the following conditions exist:

- You set **Rows per commit** too high for your environment
- You are using an overflow file
- The initial transaction fails

In this case, performance becomes worse than setting **Rows per commit** to `1`.

Staging tables to speed up auto-correct loads

For large loads where auto-correct is required:

- For Oracle targets, create two data flows:
  - In the first data flow, extract data and load it to a staging table using a temporary table object that resides in the same datastore as your target table.
  - In the second data flow, use the staging table as the source and an Oracle database table as the target. In the target table editor, select **Auto-correct load**. This entire statement will be pushed down to the database, maximizing performance.
- For other targets, load data into a staging table and use post-load SQL, a script, or a data flow to:
  - Delete rows from the target where the key is in the staging table.
  - Insert rows from the staging table into the target.

Improving throughput

Use the following Data Integrator features to improve throughput:

- Pushed-down operations - Data Integrator automatically distributes the processing workload by pushing down as much as possible to the source database engine. For more information, see "Maximizing the number of pushed-down operations" on page 31.
- Parallel pipelining - Source, transform, and target processing is done in different threads, which run in parallel. Individual work flows and data flows will execute in parallel unless you create dependencies between them.

- Bulk loading - Data Integrator supports database bulk loading engines including the Oracle bulk load API. You can have multiple bulk load processes running in parallel. For more information, see "Using Bulk Loading" on page 37.
- Server groups - You can group Job Servers on different computers into a logical Data Integrator component called a server group. A server group automatically measures resource availability on each Job Server in the group and distributes scheduled batch jobs to the computer with the lightest load at runtime. This functionality also provides a hot backup method. If one Job Server in a server group is down, another Job Server in the group processes the job. For more information, see "Using Server Groups" on page 29 of the *Data Integrator Administrator Guide*.
- Parallel execution - Data Integrator supports partitioned source and target tables and setting the degree of parallelism for data flows. These options allow you to control the number of instances for a source, transform, and target that can run in parallel within a data flow. Each instance runs as a separate thread and can run on a separate CPU. For more information, see "Using Parallel Execution" on page 59.
- Real-time services - Data Integrator automatically launches new instances of real-time service providers (processes) to increase capacity. The maximum and minimum number of instances to launch is configurable. For more information, see "Real-Time Performance" on page 85 of the *Data Integrator Administrator Guide*.

Maximizing the number of pushed-down operations

To optimize performance, Data Integrator pushes down as many transform operations as possible to the source or target database and combines as many operations as possible into one request to the database. Data flow design influences the number of operations that Data Integrator can push to the database. Before running a job, you can view the SQL that Data Integrator generates and adjust your design to maximize the SQL that is pushed down to improve performance.

This section discusses:

- Push-down logic
- Push-down example
- Viewing SQL

Push-down logic

By pushing down operations to the source or target database, Data Integrator reduces the number of rows and operations that the engine must retrieve and process, which improves performance. When determining which operations to push to the database, Data Integrator examines the database and its environment.

Operations that Data Integrator pushes to the database include:

- Aggregations — Aggregate functions, typically used with a **Group by** statement, always produce a data set smaller than or the same size as the original data set.
- Distinct rows — When you select **Distinct rows** from the **Select** tab in the query editor, Data Integrator will only output unique rows.
- Filtering — Filtering can produce a data set smaller than or equal to the original data set.
- Joins — Joins typically produce a data set smaller than or similar in size to the original tables.

- Ordering — Ordering does not affect data-set size. Data Integrator can efficiently sort data sets that fit in memory. Because Data Integrator does not perform paging (writing out intermediate results to disk), Business Objects recommends that you use dedicated disk-sorting programs such as SyncSort or the DBMS itself to order very large data sets.
- Projection — Projection normally produces a smaller data set because it only returns columns referenced by a data flow.
- Functions — Most Data Integrator functions that have equivalents in the underlaying database are appropriately translated.

Data Integrator cannot push some transform operations to the database. For example:

- Expressions that include Data Integrator functions that do not have database correspondents
- Load operations that contain triggers

Similarly, Data Integrator cannot always combine operations into single requests. For example, when a stored procedure contains a COMMIT statement or does not return a value, Data Integrator cannot combine the stored procedure SQL with the SQL for other operations in a query.

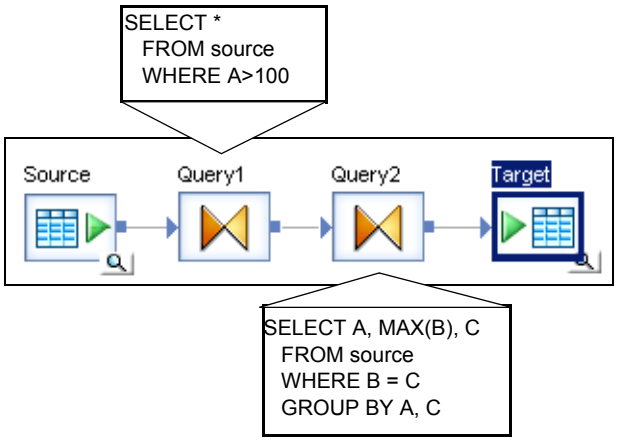
Data Integrator can only push operations supported by the DBMS down to that DBMS. Data Integrator cannot push a built-in function or transform to the source database. Therefore, for best performance, try not to intersperse Data Integrator transforms among operations that can be pushed down to the database.

NOTE: Database-specific functions can only be used in situations where they will be pushed down to the database for execution.

Push-down example

When determining how to push operations to the database, Data Integrator first collapses all the transforms into the minimum set of transformations expressed in terms of the source table columns. Next, Data Integrator pushes all possible operations on tables of the same database down to that DBMS.

For example, the following data flow extracts rows from a single source table.



The first query selects only the rows in the source where column `A` contains a value greater than `100`. The second query refines the extraction further, reducing the number of columns returned and further reducing the qualifying rows.

Data Integrator collapses the two queries into a single command for the DBMS to execute:

```
SELECT A, MAX(B), C
   FROM source
   WHERE A > 100 AND B = C
   GROUP BY A, C
```

All the operations in this data flow could be passed to the DBMS. However, if the data flow contains operations that cannot be passed to the DBMS, Data Integrator optimizes the transformation differently. For example, if Query1 called `func(A) > 100`, where `func` is a Data Integrator custom function, then Data Integrator would generate two commands. The source DBMS executes the first command. The first query becomes:

```
SELECT A, B, C
   FROM source
   WHERE B = C
```

Data Integrator executes the second command. The second query becomes:

```
SELECT A, MAX(B), C
   FROM Query1
   WHERE func(A) > 100
   GROUP BY A, C
```

Viewing SQL

Before running a job, you can view the SQL code that Data Integrator generates for table sources in data flows. By examining the SQL code, you can verify that Data Integrator generates the commands you expect. If necessary, you can alter your design to improve the data flow.

NOTE: This option is not available for R/3 data flows.

➢ To view SQL

1. Validate and save data flows.
2. Open a data flow in the workspace.
3. Select **Display Optimized SQL** from the **Debug** menu.

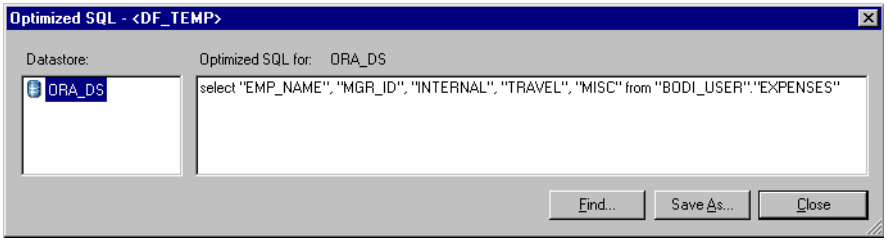
   Alternately, you can right-click a data flow in the object library and select **Display Optimized SQL**.

The Optimize SQL window opens, showing a list of datastores and the optimized SQL code for the selected datastore.

Data Integrator only shows the SQL generated for table sources. Data Integrator does not show the SQL generated for SQL sources that are not table sources, such as:

- Lookup function
- Key_generation function
- Key_Generation transform
- Table_Comparison transform
- Target tables

4. Select a datastore to view the SQL that this data flow applies against the corresponding database or application.



In the Optimize SQL window you can:

- Use the **Find** button to perform a search on the SQL displayed.
- Use the **Save As** button to save the text as a .sql file.

If you try to use the **Display Optimized SQL** command when there are no SQL sources in your data flow, Data Integrator alerts you. Examples of non-SQL sources include:

- R/3 data flows
- Message sources
- File sources
- IDoc sources

If a data flow is not valid or saved, Data Integrator alerts you.

Minimizing data type conversion

Data Integrator offers very robust and easy-to-use data type conversions via column mappings of different data types. Business Objects recommends that you:

- Avoid unnecessary data conversions.
- Verify that Data Integrator is performing the implicit conversions (selected when you drag and drop columns from input to output schemas in the query transform) as expected. This can be done by looking at the warnings generated during job validation.

Minimizing locale conversion

If your jobs do not require the use of different or multi-byte locales, you can increase performance by ensuring that locales are single-byte and not mismatched. For more information, see "Maintaining Job Server performance" on page 351 of the *Data Integrator Designer Guide*.

4 Using Bulk Loading

Data Integrator supports capabilities present in Oracle, Microsoft SQL Server, Informix, DB2, Sybase, and Teradata that enable you to load data in bulk rather than using SQL statements.

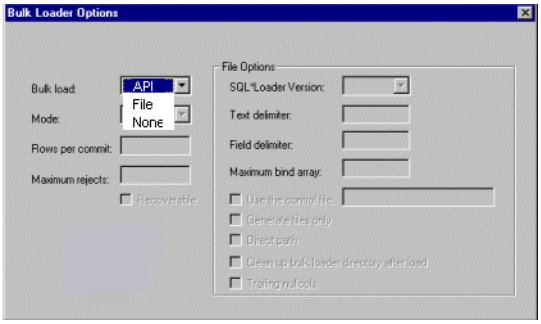
For information about the bulk loading options supported in Data Integrator, see "Target" on page 103 of the *Data Integrator Reference Guide*. Some general considerations when using bulk loading are:

- You specify bulk loading options on the Data Integrator target table editor in the **Bulk Loader Options** tab.
- If you are using bulk loading, parallel loading is available only in the Oracle and Informix environments.
- When you use a template table, bulk loading is disabled.

This chapter contains the following sections:

Bulk loading in Oracle

Data Integrator supports Oracle bulk loading.



Bulk loading methods

You can bulk load to Oracle using an API or a staging file:

- If you select the **API** method, Data Integrator accesses the direct path engine of Oracle's database server associated with the target table and connected to the target database. Using Oracle's Direct-Path Load API, input data is fed directly into database files. To use this option, you must have Oracle version 8.1 or later.
- If you select the **File** method, Data Integrator writes an intermediate staging file, control file, and log files to the local disk and invokes the Oracle SQL*Loader. This method requires more processing time than the API method.

  For detailed information about the Oracle SQL*Loader options, see the relevant Oracle product documentation.

Bulk loading modes

Bulk loading in Oracle supports two modes of data loading, *conventional-path* and *direct-path*. Conventional-path loading is implicit for the **File** option if you do not select **Direct-path** on the **Bulk Loader Options** tab in the target table editor. Data Integrator always uses direct-path loading for the **API** option.

- Conventional-path loading

  Conventional-path loads use the SQL INSERT statements to load data to tables.

- Direct-path loading

  Direct-path loads use multiple buffers for a number of formatted blocks that load data directly to database files associated with tables.

Bulk loading parallel execution options

Parallel execution options for bulk loading are on the **Options** tab.

For the API method, you can choose to select the **Enable partitioning** check box. If selected, Data Integrator generates the number of target parallel instances based on the number of partitions in the target table. If not selected or if your table target is not partitioned, Data Integrator uses one loader by default.

For the File method, enter a value in the **Number of loaders** box or select the **Enable partitioning** check box.

NOTE: The **Enable partitioning** check box does not appear on the **Options** tab if the target table is not partitioned.

Bulk loading scenarios

With two bulk loading methods, two load modes, and two parallel load options, there are several scenarios you can configure:

Example scenarios used to bulk load data to Oracle

Scenario	Method	Load Mode	Parallel Load Options
1	API	Direct-path	Enable partitions is not selected (One loader is used by default)
2	API	Direct-path	Enable partitions is selected
3	File	Direct-path	Number of loaders = 1
4	File	Direct-path	Number of loaders > 1
5	File	Direct-path	Enable partitions is selected
6	File	Conventional	Number of loaders = 1
7	File	Conventional	Number of loaders > 1
8	File	Conventional	Enable partitions is selected

Here are some tips for using these scenarios:

- The API method always uses the direct-path load type, and when it is used with a partitioned target table, Data Integrator processes loads in parallel. Data Integrator instantiates multiple loaders based on the number of partitions in a table. Each loader receives rows that meet the conditions specified by the partition.
- With the File method, direct-path is faster than conventional load, but the File method is slower than using an API because of the need to generate a staging file, logs, and invoke Oracle's SQL*Loader.
- With the File method, when you use a value of greater than one for either the **Number of Loaders** or the **Enable partitioning** option, loads cannot truly run in parallel. The creation of a staging file and log for each loader is serialized.

Using bulk loading options

As seen in the table on the previous page, there are many ways to set up bulk loading for an Oracle database. The following sections describe two scenarios in detail.

Direct-path loads using Number of Loaders and File method

In the **Options** tab of the target table editor, when you enter a value for **Number of loaders**, Data Integrator instantiates multiple loaders. Each loader receives rows equal to the amount specified in the **Rows per commit** box on the **Bulk Loader Options** tab. The loaders pipe rows to a staging file, then call the SQL*Loader to load the staging file contents into the table.

This process occurs in "round-robin" fashion. For example, if you set **Rows per commit** to `5000` and **Number of loaders** to `2`, then the first loader receives 5000 rows, writes them to the staging file, and then invokes the SQL*Loader to load the data into the table.

Meanwhile, the second loader receives the second batch of 5000 rows, writes them to a staging file, and then waits for the first loader to complete the loading. When the first loader completes the bulk load, the second loader starts, and while the second loader is loading, the first loader receives the third batch of 5000 rows. This process continues until all the data loads.

The SQL*Loader uses a control file to read staging files and load data. Data Integrator either creates this control file at runtime or uses one that is specified on the **Bulk Loader Options** tab at design time.

For parallel loading, the generated control files, data files, and log files are named as follows:

`TableNameTIDPID_LDNUM_BATCHNUM`

Where:

`TableName` The name of the table into which data loads.

TID	The thread ID.
PID	The process ID.
LDNUM	The loader number, which ranges from 0 to number of loaders minus 1. For single loaders, LDNUM is always 0.
BATCHNUM	The batch number the loader is processing. For single loaders the *BATCHNUM* is always 0.

NOTE: Product performance during this type of parallel loading depends on a number of factors such as distribution of incoming data and underlying DBMS capabilities. Under some circumstances it is possible that specifying parallel loaders can be detrimental to performance. Always test the parallel loading process before moving to production.

Direct-path loads using partitioned tables and API method

You can import partitioned tables as Data Integrator metadata. For more information see, "Table partitioning" on page 3.

In the **Options** tab of the target table editor, when you select **Enable partitioning**, Data Integrator instantiates multiple loaders based on the number of partitions in a table. Each loader receives rows that meet the conditions specified by the partition. In addition, commits occur based on the number specified in the **Rows per commit** box on the **Bulk Loader Options** tab.

For example:

- If you set **Rows per commit** to 5000, the number of partitions is 2, and your first partition includes 2500 rows, then the first loader commits after receiving all possible rows (2500) while concurrently processing the second loader.

- If you set **Rows per commit** to `5000`, the number of partitions is `2`, and your first partition includes 10,000 rows, then the first loader commits twice. Once after receiving 5000 rows and again after receiving the second batch of 5000 rows. Meanwhile, the second loader is processing its rows.

The loaders pipe rows directly to Oracle database files by using Oracle direct-path load APIs (installed with the Oracle client) that are associated with the target database.

The API method allows Data Integrator to bypass the use of the SQL* Loader (and the control and staging files it needs). In addition, by using table partitioning, bulk loaders can pass data to different partitions in the same target table at the same time. Using the API method with partitioned tables fully optimizes performance.

NOTE: If you plan to use a partitioned table as a target, the physical table partitions in the database must match the metadata table partitions in Data Integrator. If there is a mismatch, Data Integrator will not use the partition name to load partitions, which impacts processing time.

For the API method, Data Integrator records and displays error and trace logs as it does for any job. A monitor log records connection activity between components; however, it does not record activity while the API is handling the data.

Bulk loading in Microsoft SQL Server

Data Integrator supports Microsoft SQL Server bulk loading. For detailed information about the Microsoft SQL Server bulk loader options and their behavior in the Microsoft SQL Server DBMS environment, see the relevant Microsoft SQL Server product documentation.

Data Integrator supports the Microsoft SQL Server bulk copy utility BCP for rapidly loading SQL Server-based data warehouses. This Data Integrator feature includes the use of Microsoft's Bulk Copy Library, which makes it possible to invoke the BCP utility without creating intermediate data and control files.

Bulk loading in Informix

Data Integrator supports Informix bulk loading. For detailed information about Informix bulk loading utility options and their behavior in the Informix DBMS environment, see the relevant Informix product documentation.

Setting up Informix for the bulk loading option requires that you set the environment variable `INFORMIXDIR` at the system level even though the job service is configured to log on through an account with that variable defined at a user level.

➢ To set Informix server variables

1. Right-click **My Computer** and choose **Properties**.
2. From the **Properties** dialog, click the **Environment** tab.
3. From the **Environment** tab, add the following to both the System variable and User variable (where `c:\Informix` is the installation directory):

   ```
   INFORMIXDIR=c:\Informix
   ```

   ```
   INFORMIXSERVER=c:\Informix
   ```

Data Integrator provides Informix bulk loading support only for single-byte character ASCII delimited files (not for fixed-width files).

For Data Integrator to initiate Informix bulk loading directly, the Data Integrator job server and the target database must be located on the same system. If they are not, use the following procedure to initiate bulk loading:

1. Generate a command file and data file. Check **Generate files only** in the target table editor, Bulk Loader Options tab.
2. Manually move the command file and data file to the system where the target database is located.
3. Start the execution of the bulk loader manually.

Bulk loading in DB2 Universal Database

Data Integrator supports bulk loading to the DB2 Universal Database in two ways:

- Using the DB2 bulk load utility
- Using the sqluimpt API

Using the DB2 bulk load utility

The DB2 load utility improves performance by writing data directly into the data file.

➢ To configure your system to use the load utility

- Set the DB2_CLIENT_VERSION parameter in the configuration file on the Job Server computer (**Tools > Options** in the Designer). Valid numbers are 5.2, 6.1, or 7.1.
- Connect to every target database that uses the following:
  - If you use DB2 version 5.x or 6.x, run the command:

    ```
    bind <LINK_DIR>/bin/db2bulkload6.bnd
    blocking all grant public
    ```

  - If you use DB2 version 7.x, run the command:

    ```
    bind <LINK_DIR>/bin/db2bulkload.bnd
    blocking all grant public
    ```

  - (For HPUX11i only) If you use DB2 version 8.x, run:

    ```
    bind <LINK_DIR>/bin/db2bulkload8.bnd
    blocking all grant public
    ```

- Determine how Data Integrator will transmit data files to DB2. Depending on your configuration and version of DB2, there are different ways to transmit data files.

For example, if your version of DB2 is 5.x or 6.x, Data Integrator requires FTP to transmit data files between the Data Integrator Job Server and the DB2 server residing on another computer. To use the FTP option, you must define the FTP host name, user name, and password in the DB2 datastore you create in the Data Integrator Designer. The Data Integrator engine will generate the data file and use FTP to send the file to the DB2 server.

The following matrix outlines supported DB2 versions and data file transmission methods.

Configuration	DB2 version 5.x	DB2 version 6.x	DB2 version 7.x
Data Integrator and DB2 on same server	Automatic data transmission.	Automatic data transmission.	Automatic data transmission.
Data Integrator and DB2 on different servers	FTP required.	FTP required.	FTP optional. If you choose not to FTP, you must provide a working directory for Data Integrator on the DB2 server.

- Enter appropriate information in the datastore editor, on the **DB2 Properties** tab. Fields include:
  - **Bulk loader user name** — The user name Data Integrator uses when loading data with the bulk loader option. For bulk loading, you might specify a different user name—one who has import and load permissions, for example.
  - **Bulk loader password** — The password Data Integrator uses when loading with the bulk loader option.
  - **FTP host name** — If this field is left blank or contains the name of the computer where the Data Integrator Job Server resides, Data Integrator assumes that DB2 and Data Integrator share the same computer and that FTP is unnecessary. When FTP is unnecessary, all other FTP-related fields can remain blank.
  - **FTP user name** — Must be defined to use FTP.

- **FTP password** — Must be defined to use FTP.
- **Server working directory** — The working directory for the load utility on the computer that runs the DB2 server. You must complete this field whenever the DB2 server and the Data Integrator Job Server run on separate computers.

- If Data Integrator will use FTP and the DB2 server runs on Windows NT, verify connectivity with the FTP server.

  If your Job Server runs on Windows NT:

  a. Connect to the FTP server using the command:

  ```
  ftp <ServerName>
  ```

  b. Type the command:

  ```
  put <LINK_DIR>\DSConfig.txt
  <Server working directory>\DSConfig.txt
  ```

  where <*LINK_DIR*> is the Data Integrator installation directory and <*Server working directory*> is the working directory entered on the datastore's **DB2 Properties** tab.

  You can only use the load utility if this command succeeds.

  If your Job Server runs on UNIX:

  a. Connect to the FTP server.

  b. Change directories to the **Server working directory** entered on the **DB2 Properties** tab on the datastore editor.

  For example, if the directory is c:\temp, type:

  ```
  cd c:\temp
  ```

  You can only use the load utility if this command succeeds.

➢ To use the load utility in an Data Integrator job

1. Drill into your DB2 target table in the Data Integrator Designer workspace.
2. Select the **Bulk Loader Options** tab below the table schema area.
3. In the **Bulk loader** list, select **load**.

   The window updates to show all load options. Load options include these existing import bulk loader options:

   - **Generate files only**
   - **Clean up bulk loader directory**
   - **Text delimiter**
   - **Column delimiter**

   Additional load options include:

   - **Mode** — Determines load mode. Valid values are:
     - **Insert** — Appends the new records into the target table
     - **Replace** — Deletes the existing records, then inserts the loaded data
   - **Save count** — Determines the consistency point while loading data into tables.
   - **Warning row count** — Defines the number of warnings allowed for each load operation.
   - **Exception table name** — Defines the table into which the DB2 server loads rows that violate a table defined with constraints. Rows that violate those constraints are deleted from the target table and inserted into the exception table.

- **Recoverable** — Allows you to enable or disable data recovery. When this option is not selected (disabled), the load operation is not recoverable. The loaded data cannot be recovered by a roll-forward during recovery. When selected (enabled), you must also define the copy files path in the **Copy target directory** for the loaded portion of the loaded data to be copied. This copy is used in the roll-forward phase of database recovery.
- **Copy target directory** — Defines the directory of copy files when you enable both the database forward log recovery and the **Recoverable** option. Data Integrator supports local media only.
- **Data file on client computer** — When you select this option, the load utility processes the local file directly rather than using FTP to send the data file to the DB2 server. To use this option:
  - You must use DB2 version 7.x or later.
  - The target DB2 cannot be a DB2 enterprise (extended edition environment).
  - The target table and database must not be partitioned.

  This option is only applicable if Data Integrator and DB2 are on different servers.

When you use the load utility, Data Integrator writes output messages in a local message file. Data Integrator automatically generates this file when executing a job. When Data Integrator rejects input records, it writes the rejected records in a "dump" file, automatically generated in the server working directory. Check the trace log to find either of these files.

Using the sqluimpt API

Data Integrator also supports bulk loading in the DB2 Universal Database 5.2 environment using `sqluimpt` API. For Data Integrator to initiate DB2 bulk loading by this method directly, the Data Integrator job server and DB2 must be located on the same system. If they are not, use the following procedure to initiate bulk loading:

1. Generate a command file and data file. Check **Generate files only** in the target table editor, Bulk Loader Options tab.
2. Manually move the command file and data file to the system where the target database is located.
3. Start the execution of the bulk loader manually.

Bulk loading in Sybase

Data Integrator supports bulk loading of Sybase databases through the Sybase bulk copy utility. For detailed information about the Sybase bulk loader options and their behavior in the Sybase DBMS environment, see the relevant Sybase product documentation.

Bulk loading in Teradata

Data Integrator supports bulk loading with the Teradata Warehouse Builder application as well as with Teradata load utilities.

For detailed information about Teradata bulk loader options and their behavior in the Teradata DBMS environment, see the relevant Teradata product documentation.

Data Integrator supports multiple bulk loading methods for the Teradata database. From the **Bulk Loader Options** tab of your Teradata target table editor, select one of these methods depending on your Teradata environment:

- Warehouse Builder method
- Load Utilities method
- None (use ODBC to load Teradata)

Note that Data Integrator uses staging files for both Warehouse Builder and Load Utilities methods.

Warehouse Builder method

Data Integrator supports Teradata's Warehouse Builder, an ETL tool that consolidates bulk loading utilities into a single interface.

When you use the Warehouse Builder method, you can leverage Data Integrator's powerful parallel processing capabilities to specify a maximum number of files for Data Integrator to use in processing large quantities of data.

Data Integrator runs bulk loading jobs using Warehouse Builder as follows:

1. Data Integrator generates staging data file(s) containing data to be loaded into a Teradata table.

2. Data Integrator generates a loading script to be used by Warehouse Builder. The script defines read and load operators.
3. The read operator reads the staging data file, then passes the data to the load operator, which loads data into the Teradata table.

With the Teradata Warehouse Builder, you can choose from four types of load operators. Each operator processes your data in a slightly different way.

Operator	Description
Load	Loads a large amount of data at high speed into an empty table on the Teradata RDBMS. Use this operator when initially loading tables in the data warehouse.
SQL Inserter	Inserts data into a specified existing table on the Teradata RDBMS. A single SQL session can insert data, while other operators require multiple sessions. Because it uses the fewest RDBMS resources, this is the least intrusive and slowest method of loading data.
Stream	Allows parallel inserts, updates, and deletes to new or pre-existing Teradata tables. Uses multiple sessions to load data into one or more new or existing tables. Use this operator to maintain tables in the Teradata RDBMS when the system is busy and can provide minimal resources. Unlike Load and Update, which assembles large volumes of rows into blocks then moves those blocks to Teradata RDBMS, Stream loads data one row at a time.
Update	Allows highly scalable parallel inserts, updates, and deletes of vast amounts of data to new or pre-existing Teradata tables. Use this operator to maintain tables in the data warehouse

Attributes are different for each operator selection. Teradata Warehouse Builder predefines attribute default values. Note that:

- You can select and modify attribute values.

- Some attribute values are optional and some are required. You must specify a value for each attribute name shown in **bold**.
- Only the **Stream** and **Update** operators provide the "Ignore duplicate rows" option.

Parallel processing with Teradata Warehouse Builder

Data Integrator provides the option for parallel processing when you bulk load data using the Warehouse Builder method.

Using a combination of choices from the Options tab and Bulk Loader Options tab, you specify the number of Data Integrator-generated data files as well as the number of Read and Load Operator Instances. The **Number of Loaders** option distributes the workload while Read/Load Operators perform parallel processing.

In the target table Options tab, specify the **Number Of Loaders** to control the number of data files Data Integrator generates. Data Integrator writes data to these files in batches of 999 rows. For example, if you set **Number Of Loaders** to 2, Data Integrator would generate two data files, writing 999 rows to the first file, then writing the next 999 rows to the second file. If there are more rows to process, Data Integrator continues, writing to the first file again, then the second, and so forth.

In the **Bulk Loader Options** tab, specify the number of Read and Load Operator Instances in the loading scripts. If you set **Read Operator Instances** to 2 and **Load Operator Instances** to 2, Warehouse Builder will assign the first read operator instance to read one data file and the other instance to read another data file in parallel. The read operator instances then pass the data to the load operator instances for parallel loading into Teradata.

The Warehouse Builder uses a control file to read staging files and load data. Data Integrator either creates this control file at run time or uses one that is specified on the **Bulk Loader Options** tab at design time.

NOTE: Product performance during this type of parallel loading depends on a number of factors such as distribution of incoming data and underlying DBMS capabilities. Under some circumstances, it is possible that specifying parallel loaders can be detrimental to performance. Always test the parallel loading process before moving to production.

➢ To use Warehouse Builder bulk loading with Data Integrator

1. In your target table Options tab, specify the **Number of loaders** to control the number of data files Data Integrator generates. Data Integrator will write data to these files in batches of 999 rows.
2. In the Bulk Loader Options tab, choose **Warehouse Builder** as your bulk loader.
3. Specify the number of read and load operator instances in the loading scripts.

   If you set **Number of instances** to 2 (load operators) and **Number of DataConnector instances** to 2 (read operators), Warehouse Builder will assign the first read operator instance to read one data file and the other instance to read another data file in parallel. The read operator instances then pass the data to the load operator instances for parallel loading into Teradata.

Load Utilities method

In addition to the Warehouse Builder interface, Data Integrator supports several Teradata utilities that load and extract to the Teradata database. Each load utility is a separate executable designed to move large volumes of data into a Teradata database. Choose from the following bulk loader utilities:

Utility	Description
MultiLoad	Loads large quantities of data into unpopulated tables. MultiLoad also supports bulk inserts, updates, and deletions against populated tables.
FastLoad	Loads unpopulated tables only. Both the client and server environments support FastLoad. Provides a high-performance load (inserts only) to one empty table each session.
TPump	Uses standard SQL/DML to maintain data in tables. It also contains a method that you can use to specify the percentage of system resources necessary for operations on tables. Allows background maintenance for insert, delete, and update operations to take place at any time you specify. Used with small data volumes.

To use these utilities, you must write your own scripts for loading the data; however, Data Integrator generates your data file.

➢ To load using a Teradata non-Warehouse Builder load utility

1. In the **Bulk Loader Options** tab of your target table editor, choose a **Load Utility** method.
2. Enter a command to be invoked by Data Integrator in the Command line text box. For example,
   `fastload<C:\tera_script\float.ctl`
3. Enter (or browse to) the directory path where you want Data Integrator to place your data file.

5 Using Parallel Execution

You can set Data Integrator to perform data extraction, transformation, and loads in parallel by setting parallel options for sources, transforms, and targets. In addition, you can set individual data flows and work flows to run in parallel by simply not connecting them in the workspace. If the Data Integrator Job Server is running on a multi-processor computer, it takes full advantage of available CPUs.

This chapter contains the following sections:

Parallel execution in data flows

For batch jobs, Data Integrator allows you to execute parallel threads in data flows.

This section contains the following:

- Table partitioning
- Degree of parallelism
- Combining table partitioning and a degree of parallelism
- File multi-threading

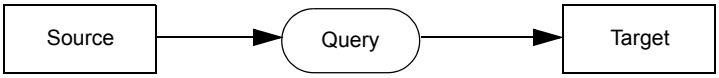
Table partitioning

Data Integrator processes data flows with partitioned tables based on the amount of partitioning defined. There are three basic scenarios:

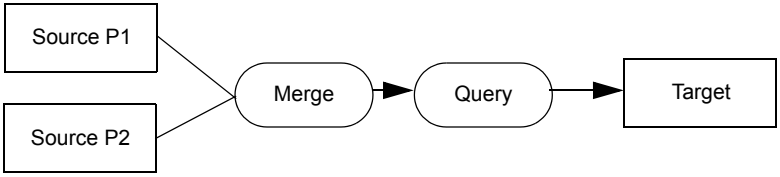
- Data flow with source partitions only
- Data flow with target partitions only
- Dataflow with source and target partitions

Data flow with source partitions only

If you have a data flow with a source that has two partitions connected to a query and a target, it appears in the workspace as shown in the following diagram:



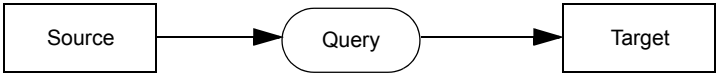
At runtime, Data Integrator translates this data flow to:



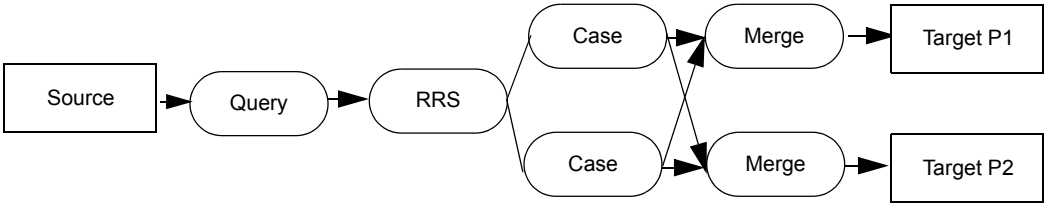
Data Integrator instantiates a source thread for each partition, and these threads run in parallel. The data from these threads later merges into a single stream by an internal merge transform before processing the query.

Data flow with target partitions only

If you have a data flow with a target that has two partitions connected to a query and a source, it appears in the workspace as shown in the following diagram:



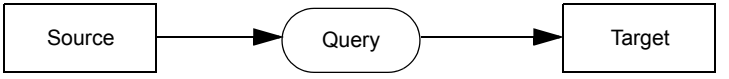
At runtime, Data Integrator translates this data flow to:



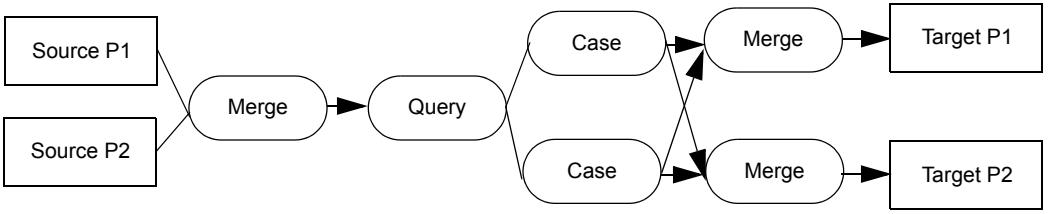
Data Integrator inserts an internal Round Robin Splitter (RRS) transform after the Query transform, which routes incoming rows in a round-robin fashion to internal Case transforms. The Case transforms evaluate the rows to determine the partition ranges. Finally, an internal Merge transform collects the incoming rows from different Case transforms and outputs a single stream of rows to the target threads. The Case, Merge, and the target threads execute in parallel.

Dataflow with source and target partitions

If you have a data flow with a source that has two partitions connected to a query and a target that has two partitions, it appears in the workspace as shown in the following diagram:



At runtime, Data Integrator translates this data flow to:



The source threads execute in parallel and the Case, Merge, and targets execute in parallel.

Viewing, creating, and enabling table partitions

Oracle databases support range, list, and hash partitioning. You can import this information as table metadata and use it to extract data in parallel. You can use range and list partitions to load data to Oracle targets. You can also specify logical range and list partitions using Data Integrator metadata for Oracle tables.

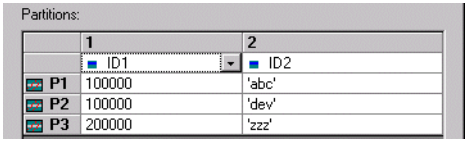
In addition, Data Integrator provides the ability to specify logical range partitions for DB2, Informix, Microsoft SQL Server, and Sybase tables by modifying imported table metadata.

Data Integrator uses partition information by instantiating a thread at runtime for each partition. These threads execute in parallel. To maximize performance benefits, use a multi-processor environment.

➢ To view partition information

1. Import a table into Data Integrator.
2. In the **Datastores** tab of the object library, right-click the table name and select **Properties**.
3. Click the **Partitions** tab.

   When you import partitioned tables from your database, you will find these partitions displayed on the **Partitions** tab of the table's Properties window. The partition name appears in the first column. The columns that are used for partitioning appear as column headings in the second row.



   If you import a table that does not have partitions, you can create logical partitions using the Table Metadata dialog.

➢ To create or edit table partition information

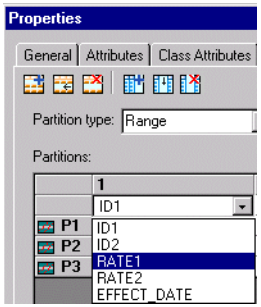
1. In the **Datastores** tab of the object library, right-click the table name and select **Properties**.
2. In the Properties window, click the **Partitions** tab.
3. Select a partition type.

Partition Type	Description
None	This table is not partitioned.
Range	Each partition contains a set of rows with column values less than those specified. For example, if the value of column one is `100,000`, then the data set for partition one will include rows with values less than `100,000` in column one.
List	Each partition contains a set of rows that contain the specified column values.
Hash	(Read-only) Each partition contains the partition name and ID. You cannot edit hash settings in Data Integrator. However, if you want to edit its partitions, you can create logical range or list partitions for an Oracle table imported with hash partitions.

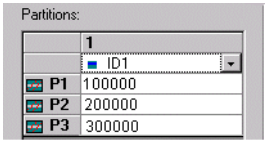
4. Add, insert, or remove partitions and columns using the tool bar.

Icon	Description
	Add Partition
	Insert Partition
	Remove Partition
	Add Column
	Insert Column
	Remove Column

5. Select the name of a column from each column list box.



6. Enter column values for each partition.



Data Integrator validates the column values entered for each partition according to the following rules:

- Values can be literal numbers and strings or datetime types.
- Column values must match column data types.
- Literal strings must include single quotes: `'Director'`.
- For range partitions, the values for a partition must be greater than the values for the previous partition.

7. Click **OK**.

   If the validation rules described in the previous step are not met, you will see an error message.

The number of partitions in a table equals the maximum number of parallel instances that Data Integrator can process for a source or target created from this table.

In addition to importing partitions or creating and editing partition metadata, enable the partition settings when you configure sources and targets.

➢ To enable partition settings in a source or target table

1. Drop a table into a data flow and select **Make Source** or **Make Target**.
2. Click the name of the table to open the source or target table editor.
3. Enable partitioning for the source or target:
   a. For a source table, click the **Enable Partitioning** check box.
   b. For a target table, click the **Options** tab, then click the **Enable Partitioning** check box.
4. Click **OK**.

When the job executes, Data Integrator generates parallel instances based on the partition information.

NOTE: If you are loading to partitioned tables, a job will execute the load in parallel according to the number of partitions in the table. If you select both **Enable Partitioning** and **Include in transaction**, the **Include in transaction** setting overrides the **Enable Partitioning** option. For example, if your job is designed to load to a partitioned table but you select **Include in transaction** and enter a value for **Transaction order**, when the job executes, Data Integrator will include the table in a transaction load and does not parallel load to the partitioned table.

Tip

If the underlying database does not support range partitioning and if you are aware of a natural distribution of ranges, for example using an Employee Key column in an Employee table, then you can edit the imported table metadata and define table ranges. Data Integrator would then instantiate multiple reader threads, one for each defined range, and execute them in parallel to extract the data.

NOTE: Table metadata editing for partitioning is designed for source tables. If you use a partitioned table as a target, the physical table partitions in the database must match the metadata table partitions in Data Integrator. If there is a mismatch, Data Integrator will not use the partition name to load partitions. Consequently, the whole table updates.

Degree of parallelism

You can run transforms in parallel by entering a number in the **Degree of Parallelism** box on a data flow's Properties window. The number is used to replicate transforms in the data flow which run as separate threads when the Job Server processes the data flow.

Degree of parallelism and transforms

Degree Of Parallelism (DOP) is a property of a data flow that defines how many times each transform defined in the data flow replicates for use on a parallel subset of data. If there are multiple transforms in a data flow, Data Integrator chains them together until it reaches a merge point.

The Query, Sort, and Table Comparison transforms always replicate when you enter a number for DOP. If an upstream transform replicates, the following transforms also replicate:

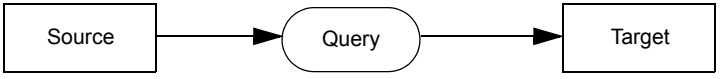
- Map_Operation
- History_Preserving
- Pivot

There are two basic scenarios:

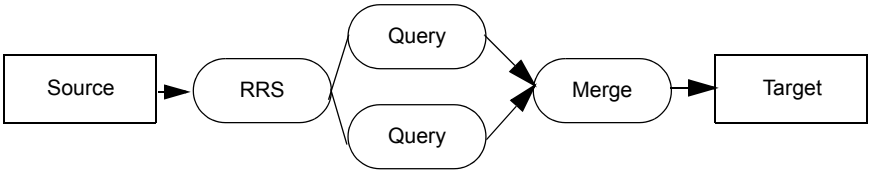
- DOP and a data flow with a single transform
- DOP and a data flow with multiple transforms

DOP and a data flow with a single transform

The following figures show runtime instances of a data flow with a DOP of `1`, and the same data flow with a DOP of `2`:



Runtime instance of a data flow where DOP =1

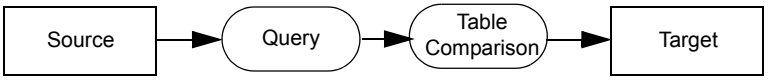


Runtime instance of a data flow where DOP = 2

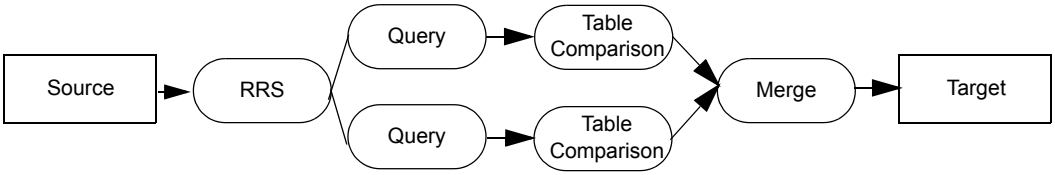
With a DOP greater than one, Data Integrator inserts an internal Round Robin Splitter (RRS) that transfers data to each of the replicated queries. The replicated queries execute in parallel, and the results merge into a single stream by an internal Merge transform.

DOP and a data flow with multiple transforms

The following figures show runtime instances of a data flow with a DOP of `1`, and the same data flow with a DOP of `2`. Notice multiple transforms in a data flow replicate and chain when the DOP is greater than `1`.:



Runtime instance of a data flow where DOP =1



Runtime instance of a data flow where DOP = 2

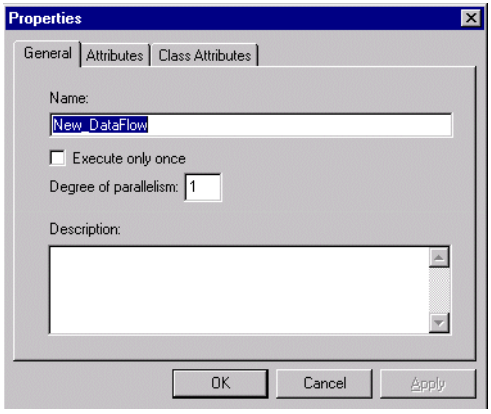
When there are multiple transforms in a data flow and the DOP is greater than 1, Data Integrator carries the replicated stream as far as possible, then merges the data into a single stream.

Setting Degree of parallelism

The Degree of Parallelism (DOP) is a data flow property that acts on transforms added to the data flow.

➢ To set the Degree of Parallelism for a data flow

1. Drop a data flow into a job.
2. Right-click the data flow icon and select **Properties**.
3. Enter a number for **Degree of parallelism**.



The default value for **Degree of parallelism** is `0`. If you set an individual data flow's **Degree of parallelism** to this default value, then you can control it using a Global_DOP value which affects all data flows run by a given Job Server. If you use any other value for a data flow's **Degree of parallelism**, it overrides the Global_DOP value.

You can use the local and global DOP options in different ways. For example:

- If you want to globally set all data flow DOP values to `4`, but one data flow is too complex and you do not want it to run in parallel, you can set the **Degree of parallelism** for this data flow locally. From the data flow's Properties window, set this data flow's **Degree of parallelism** to `1`. All other data flows will replicate and run transforms in parallel after you set the Global_DOP value to `4`. The default for the Global_DOP value is `1`.
- If you want to set the DOP on a case-by-case basis for each data flow, set the value for each data flow's **Degree of parallelism** to any value except zero.

For information about how to set the Global_DOP value, see "Changing Job Server options" on page 330 of the *Data Integrator Designer Guide*.

4. Click **OK**.

Degree of parallelism and functions

In Data Integrator, you can set stored procedures and custom functions to replicate with the transforms in which they are used. To specify this option, select the **Enable parallel execution** check box on the function's Properties window. If this option is not selected and you add the function to a transform, the transform will not replicate and run in parallel even if its parent data flow has a value greater than `1` set for **Degree of parallelism**.

When enabling functions to run in parallel, verify that:

- Your database will allow a stored procedure to run in parallel

- A custom function set to run in parallel will improve performance

All built-in functions, except the following, replicate if the transform they are used in replicates due to the DOP value:

- `exec ()`
- `key_generation()`
- `mail_to()`
- `print()`
- `set_env()`
- `sql()`
- `total_rows()`
- `ll_switch()`
- `ll_error()`
- `sleep()`
- `gen_row_num()`
- `raise_exception()`
- `raise_exception_ext()`

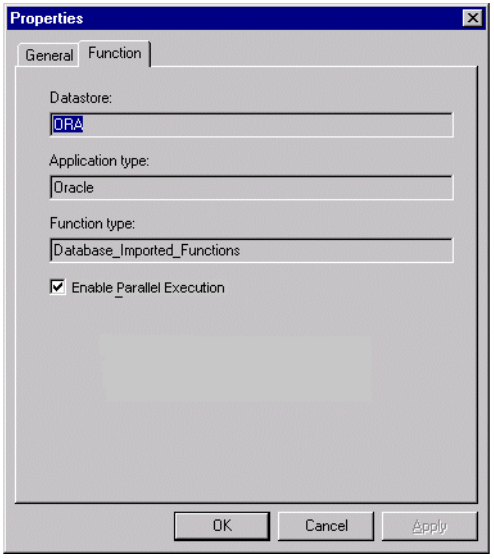
Setting functions to run in parallel

Use the **Enable parallel execution** option to set functions to run in parallel when the transforms in which they are used execute in parallel.

➢ To enable stored procedures to run in parallel

1. In the **Datastores** tab of the object library, expand a **Datastore** node.
2. Expand its **Function** node.
3. Right-click a function and select **Properties**.

4. In the Properties window, click the **Function** tab.
5. Click the **Enable Parallel Execution** check box.



6. Click **OK**.

➢ To enable custom functions to run in parallel

1. In the **Custom Functions** tab of the object library, right-click a function name and select **Properties**.
2. In the Properties window, click the **Function** tab.
3. Click the **Enable Parallel Execution** check box.
4. Click **OK**.

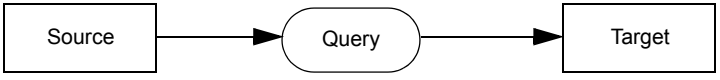
Tips

DOP can degrade performance if you do not use it judiciously. The best value to choose depends on the complexity of the flow and number of CPUs available. For example, on a computer with four CPUs, setting a DOP greater than two for the flow shown for "DOP and a data flow with multiple transforms" on page 69 will not improve performance but can potentially degrade it due to thread contention.

Perform an Order By or a Group By at the end of a data flow. A sort node (Order By, Group By) is always a merge point, after which the engine proceeds as if the DOP value is 1.

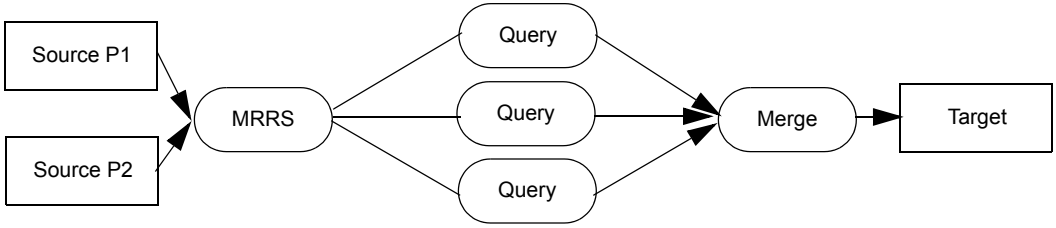
Combining table partitioning and a degree of parallelism

Different settings for source and target partitions and the degree of parallelism result in different behaviors in the Data Integrator engine. The sections that follow show some examples. For all the following scenarios, the data flow appears as follows:



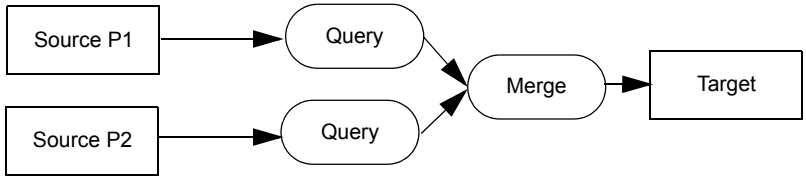
Two source partitions and a DOP of three

When a source has two partitions, it replicates twice. The input feeds into a merge-round-robin splitter (MRRS) that merges the input streams and splits them into a number equal to the value for DOP (in this case, three outputs to the query transform). The stream then merges and feeds into the target.



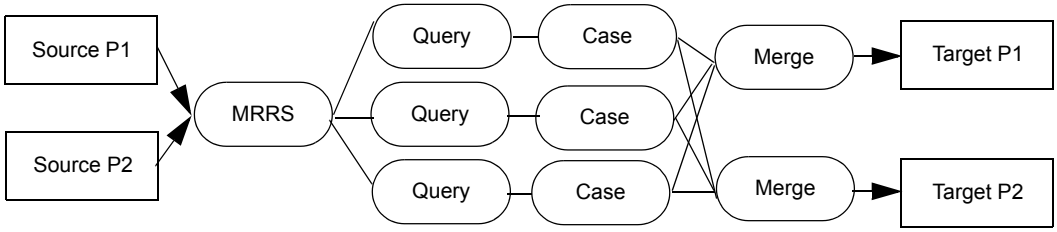
Two source partitions and a DOP of two

When the number of source partitions is the same as the value for DOP, the engine merges before the target (or before any operation that requires a merge, such as aggregation operations) and proceeds in a single stream to complete the flow.



Two source partitions, DOP of three, two target partitions

When the number of source partitions is less then the value for DOP, the input feeds into a merge-round-robin splitter (MRRS) that merges the input streams and splits them into a number equal to the value for DOP. The engine then merges the data before the target to equal the number of target partitions, then proceeds to complete the flow.



Tip

Depending on the number of CPUs available, set the DOP value equal to the number of source partitions as a general rule. This produces a data flow without the Merge Round Robin Splitter and each partition pipes the data directly into the consuming transform.

File multi-threading

You can set the number of threads Data Integrator uses to process flat file sources and targets. The **Parallel process threads** option is available on the:

- File format editor
- Source file editor
- Target file editor
- Properties window of an SAP R/3 data flow.

For SAP R/3 specific information, see the *Data Integrator Supplement for SAP*.

Without multi-threading:

- With delimited file reading, Data Integrator reads a block of data from the file system and then scans each character to determine if it is a column delimiter, a row delimiter, or a text delimiter. Then it builds a row using an internal format.
- For positional file reading, Data Integrator does not scan character by character, but it still builds a row using an internal format.
- For file loading, processing involves building a character-based row from the Data Integrator internal row format.

You can set these time-consuming operations to run in parallel. You can use the **Parallel process threads** option to specify how many threads to execute in parallel to process the I/O blocks.

This section contains:

Flat file sources

To use the **Parallel process threads** option, the following conditions must be met:

- In the file format editor:
  - There are no text delimiters defined, or if one is defined, for example in a column-delimited file, indicate that a row delimiter cannot occur within text delimiters by not quoting a row delimiter within a text delimiter.

You can set Data Integrator to read flat file data in parallel in most cases because the majority of Data Integrator jobs use fixed-width or column-delimited source files that do not have text delimiters specified.

- An end-of-file (EOF) marker for the file's input/output style is not specified.
- The language locale has a fixed character size. Only single-byte characters have a fixed character size in Data Integrator, so only single-byte characters are supported with this feature.

- In the Source File Editor, no number has been entered for **Rows to read**.

  The **Rows to read** option indicates the maximum number of rows that Data Integrator reads. It is normally used for debugging. Its default value is `none`.
- The file does not contain LONG column(s).
- The maximum row size does not exceed 128KB.

If a file source needs to read more than one file, for example, *.txt is specified for the **File(s)** option in the file format editor, Data Integrator processes the data in the first file before the data in the next file. It performs file multi-threading one file at a time.

Flat file targets

If you enter a positive value for **Parallel process threads**, Data Integrator parallel processes flat file targets when the following conditions are met:

- The file does not contain LONG column(s).
- The maximum row size does not exceed 128KB.

Tuning performance

The **Parallel process threads** option is a performance enhancement for flat file sources and targets. Performance is defined as the total elapsed time used to read a file source.

A multi-threaded file source or target achieves high performance for reads and loads by maximizing the utilization of the CPUs on your Job Server computer. You will notice higher CPU usage when you use this feature. You might also notice higher memory usage because the number of process threads you set (each consisting of blocks of rows that use 128 kilobytes) reside in memory at the same time.

To tune performance, adjust the value for **Parallel process threads**. Ideally, have at least as many CPUs as process threads. For example, if you enter the value 4 for **Parallel process threads**, have at least four CPUs on your Job Server computer.

However, increasing the value for process threads does not necessarily improve performance. The file reads and loads achieve their best performance when the work load is distributed evenly among all the CPUs and the speed of the file's input/output (I/O) thread is comparable with the speed of the process threads.

The I/O thread for a file source reads data from a file and feeds it to process threads. The I/O thread for a file target takes data from process threads and loads it to a file. Therefore, if a source file's I/O thread is too slow to keep the process threads busy, there is no need to increase the number of process threads.

If there is more than one process thread on one CPU, that CPU will need to switch between the threads. There is an overhead incurred in creating these threads and switching the CPU between them.

Tips

The best value for **Parallel process threads** depends on the complexity of your data flow and the number of available processes. If your Job Server is on a computer with multiple CPUs, the values for file sources and targets should be set to at least two.

After that, experiment with different values to determine the best value for your environment.

Here are some additional guidelines:

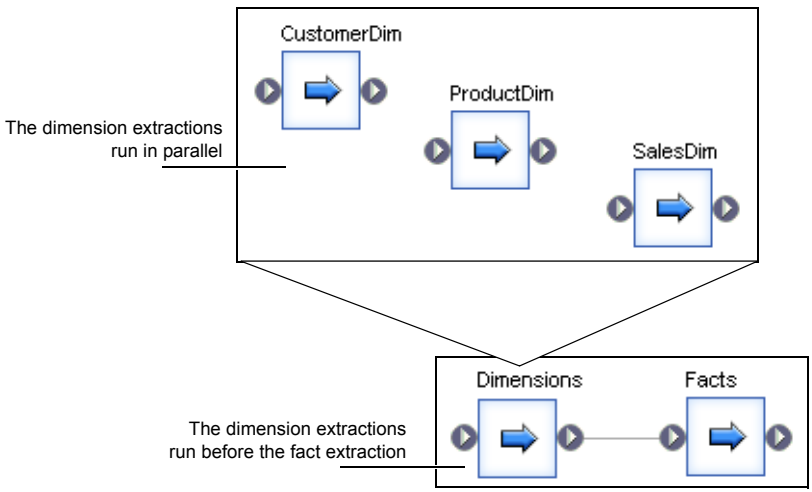
- If **Parallel process threads** is set to `none`, then flat file reads and loads are not processed in parallel.
- If **Parallel process threads** is set to `1`, (meaning that one process thread will spawn) and your Job Server computer has one CPU, then reads and loads can occur faster than single-threaded file reads and loads because Data Integrator reads the I/O thread separately and concurrently with the process thread.
- If **Parallel process threads** is set to `4`, four process threads will spawn. You can run these threads on a single CPU. However, using four CPUs would more likely maximize the performance of flat file reads and loads.

Parallel data flows and work flows

In addition to operations Data Integrator performs in parallel in a single data flow, you can explicitly execute different data flows and work flows in parallel when you specify a work flow or job.

Data Integrator coordinates the parallel steps, then waits for all steps to complete before starting the next sequential step.

For example, use parallel processing to load dimension tables by calling work flows in parallel. Then specify that your job creates dimension tables before the fact table by moving it to the left of a second (parent) work flow and connecting the flows.



You can control the maximum number of parallel Data Integrator engine processes using the Job Server options (**Tools** > **Options**> **Job Server** > **Environment**).

A Glossary

This glossary defines terms used in this documentation set including common terms that have a special meaning with respect to Data Integrator. Within a definition, other terms that are defined in the glossary appear in **bold type**.

ABAP

Advanced Business Application Programming.
A fourth-generation programming language developed by SAP in which R/3 applications are written.

ABAP program

A program that executes database operations on an **SAP R/3** server. Data Integrator **R/3 data flows** generate ABAP programs.

Access Server

A process that starts, stops, and monitors real-time services (**real-time jobs**). The Access Server dispatches requests to real-time services, ensuring optimal load balancing and complete life cycle management. The Access Server also starts, stops, and monitors **adapters**.

adapter

An external Data Integrator **interface**. There are two types of adapters:

- Custom adapters — Adapters developed using the Data Integrator Adapter Development Kit
- Prepackaged adapters — Adapters prebuilt and purchased from Business Objects, such as the Data Integrator MQSeries Adapter

Administrator

A browser-based system administration application that supports Data Integrator. Use the Administrator to configure and manage:

- Services (**real-time jobs**)
- Service providers (instances of real-time services)
- Clients (Web application clients)
- Batch jobs
- Adapters
- Job logs

after-image

The values in an UPDATE row after the row changes. You might want to pair before- and after-images of UPDATE rows when designing log-based changed-data capture jobs which Data Integrator supports.

aggregate data

Data that results when a process combines elements. This data can be presented collectively or in summary form.

ALE

Application Link Enabling. An SAP R/3 programming-related interface designed to allow reliable communication across a distributed environment.

Analytic Rapid Mart

An **Data Integrator application** that produces a predefined **data warehouse** for a specific area of business analysis. Analytic Rapid Marts are typically based on dimensional modelling. With Analytic Rapid Marts, users can manage large volumes of data.

attribute

A **property** created for a type of **object**.

BAPI

Business Application Programming Interface. A standardized SAP R/3 programming interface that allows non-SAP R/3 applications to access specific business processes and data.

back-office application

A software system that an enterprise uses to run business operations, such as human resources, general ledger, and finance. Back-office applications store data in a relational **database** optimized for operational use.

There are several types of back-office applications:

- Enterprise resource planning (ERP) — For example, SAP R/3, BAAN, J.D. Edwards, PeopleSoft
- Supply chain management (SCM) — For example, i2
- Customer relationship management (CRM) — For example, Siebel, Clarify
- Legacy

Basis

The SAP infrastructure. Basis is the foundation for all SAP products.

batch job

A group of objects (data flows, work flows, conditionals, scripts, and so forth) that executes on a schedule. This type of job is recommended for operations on large amounts of data.

before-image

The values in an UPDATE row before the row changes. You might want to pair before- and after-images of UPDATE rows when designing log-based changed-data capture jobs which Data Integrator supports.

bulk loading

A software-based mechanism that moves large data files. This mechanism supports compression, blocking, and buffering to optimize transfer times.

business component

A set of tables Siebel applications use to create a logical object called a business object.

Business Objects Data Platform

A set of applications from Business Objects, Americas, that enterprises can use to manage all types of data—from data for supply chain partners to data for customers. The Business Objects Data Platform consists of **Data Integrator** and one or more **Rapid Marts**.

CDC datastore

A CDC datastore allows you to limit extracted data to changed data only. A CDC datastore connects a **changed-data capture** table on a source database to Data Integrator.

CDC checkpoint

A CDC checkpoint enables Data Integrator to restrict CDC subscription reads. After you enable a checkpoint, the next time the CDC job runs, it reads only the rows inserted into the CDC table since the last checkpoint.

CDC Subscription

A CDC subscription name allows different data flows to extract data from the same table without corrupting data extracted by other data flows. The datastore and table name can be the same; however, the table owner name must be different in each source CDC table.

changed-data capture

The process of retrieving changes made to a production data source. This process consolidates units of work, ensures data is synchronized with the original source, and reduces data volume in a warehouse environment.

Citrix MetaFrame XP

Citrix MetaFrame XP software provides an access infrastructure for enterprise applications. You can use this software to run Data Integrator on a server which publishes instances of the Designer and other Data Integrator interfaces to users on client computers.

client/server

A distributed technology approach where the processing is divided by function. The server performs shared functions (such as managing communications and providing database services), while the client performs individual user functions.

COM

Component Object Model. Provides a set of interfaces that allow clients and servers to communicate within the same computer (running on Windows 95 or later versions).

Commerce Rapid Mart

An **Data Integrator application** that produces a predefined **data warehouse** and that includes logic to support real-time requests for data from a **back-office application**. A Commerce Rapid Mart is usually based on different data model than an **Analytic Rapid Mart**. With a Commerce Rapid Mart, users can integrate a company's back-office system with an **e-commerce** application.

Communication Structure

In **SAP BW**, a data structure that defines a set of **InfoObject**s available from an **InfoSource** to put into **InfoCube**s.

conditional

A **single-use object**, available in **work flow**s**,** that allows you to branch the execution logic based on the results of an expression. The conditional takes the form of an if/then/else statement.

custom ABAP program

A custom ABAP program runs an **ABAP program** and generates a **data set**. With a custom ABAP program, you can run an existing ABAP program as part of a job. Use a custom ABAP program as a **source** in a **data flow** or an **R/3 data flow**.

data extraction

The process of moving data from a **database** or application source to a database target (either from a legacy database to a **data mart**, or from one data mart to another).

data flow

A **reusable object** containing **steps** to define the transformation of data from **source** to **target**. Data flows are called from inside a **work flow** or **job**. You can pass information into or out of data flows using **parameters**.

Data Integrator

A software system that allows users to build and execute applications with which they can create and maintain **data warehouses**.

Data Integrator consists of several components:

- **Access Server**
- **Data Integrator engine**
- **Data Integrator repository**
- **Data Integrator Service**
- **Administrator**
- **Designer**
- **Job Server**

Data Integrator application

A set of **jobs** that process batch data extractions and real-time requests using data from a variety of data sources.

Data Integrator engine

The core process that reads job information from the **Data Integrator repository** and sets up run-time processes that execute the job. The run-time processes extract, transform, and load relational and hierarchical data. The **Job Server** starts the Data Integrator engine to execute batch or real-time jobs.

Data Integrator interface

A program that Data Integrator uses to access data sources. Specific interfaces vary by installation. There are internal interfaces—those native to the installation—and external interfaces—those that you install separately. Internal interfaces allow Data Integrator to access applications like **SAP R/3** and **SAP BW**, **messages**, relational database systems, and **legacy systems**. An external interface is also known as an **adapter**. It allows Data Integrator to access applications using information exchange technologies such as Web Services and MQ Series.

Data Integrator repository

The database that contains information about an Data Integrator application. The repository contains information about defined **reusable objects**, the **metadata** for tables, and built-in **transforms** and **functions**. When you invoke Data Integrator, you log in to the repository containing the **objects** you want to use.

Data Integrator Service

The process that ensures that the **Access Server** and the **Job Server** are running. You can configure the Data Integrator Service to restart the Access Server and Job Server whenever the computer where they are located restarts.

data loading

The process of populating a **data warehouse**. Data loading is provided by **DBMS**-specific load processes, DBMS insert processes, and independent fast-load processes.

data mapping

The process of assigning a source data element to a target data element.

data mart

A highly-focused version of a **data warehouse**. Typically, created by a department or division of a company, data marts contain data for a specific subject area, such as finance or sales. Data Integrator can populate a data mart.

data transformation

The process of filtering, merging, decoding, and translating source data to create validated data for a **data warehouse**.

data set

Rows of data with a defined schema. A step in a **data flow**—such as reading data from a **source**, joining data in a **Query transform**, or transforming data though another **transform**—yields a data set. You can view individual data sets by placing a **target** table or file at that point in the **data flow**.

data transport

A step in an **R/3 data flow** that defines a **target** to store the **data set** extracted during the flow. You can locate the target file on the **SAP R/3** server or in a location accessible to both the SAP R/3 server and to Data Integrator across a network.

data type

The format used to store a value. Data types can imply a default format for displaying and entering the value. Data read from a **source** is converted to the appropriate Data Integrator data types; data loaded to a **target** is converted from its Data Integrator data type to the type appropriate for the target.

data warehouse

A relational **database** containing subject-oriented information that supports decision analysis. Typically, information in a data warehouse is integrated from multiple sources, including operational sources. Data warehouses provide a single, non-volatile point of reference for a particular time period. Data warehouses are composed of **data marts.** Data Integrator can populate a data warehouse.

database

A collection of **table**s managed by a **DBMS** such as Microsoft SQL Server or Oracle.

DataConnector

DataConnector operator instances are used to read data files generated by Data Integrator when performing bulk loading using the Teradata Warehouse Builder.

datastore

A logical channel connecting Data Integrator to a **database**. The datastore definition includes the name and location of the database as well as user authentication information. Data Integrator uses a datastore definition to qualify a table name wherever a table is indicated in a diagram or expression. You can access the datastore definition through the **object library**.

DBMS

Database management system. A software system that builds and maintains database tables.

debug mode

Allows you to debug a job using the **interactive debugger** features in the Designer.

declarative specification

A way of indicating the desired results of a **data transformation** without placing constraints on the method of reaching the results. When you create a **data flow** in Data Integrator, you specify what data you want to read from which **source**, what you want done to that data, and where you want it loaded. What you specify constitutes the declarative specification. Data Integrator then determines the most efficient way to reach the requirements you set, including pushing operations to the source **DBMS** for execution.

degree of parallelism

You can set a value at the data flow level that allows you to parallel-process a **transform** within a data flow.

For example, if from the Properties window of a data flow you set the **Degree of parallelism** to 5, then when the job executes, Data Integrator replicates each transform in the data flow five times. Each of these replicated transforms executes in parallel using a separate process thread. Using a value for degree of parallelism can improve job performance if a **Job Server**'s computer memory and number of CPUs supports a job's parallel-processing configuration settings.

dependency analysis

A category of metadata reports that produces reports based on search criteria you specify to find specific objects in the repository. For each object returned, dependency analysis reports provide object lineage as well as object impact information.

derived data

Data that results from a computational step applied to reference or event data. Derived data is the result of either relating two or more elements of a single transaction (such as aggregation), or relating one or more elements of a transaction to an external algorithm or rule.

Designer

A graphical user interface that allows you to design and test an **Data Integrator application**.

diagram

The icons and connections between the icons that make up the definition of a **job**, **work flow**, or **data flow**.

DMZ

Demilitarized zone. A perimeter network that is more accessible than your most protected network, but located inside a security layer protected from the internet.

drill down

A method of exploring detailed data that was used in creating a summary level of data. Drill down levels depend on the granularity of the data in the **data warehouse**.

DTD

Document type definition. A text file that describes the elements (tags) in an **XML** document and the relationship among them. When an XML document is used to describe a transaction, the DTD describes the data schema used in the transaction.

e-commerce

Buy and sell transactions conducted over the internet.

EDI

Electronic Data Interchange. Electronic exchange of structured data between businesses. This exchange is not dependent on hardware, software, or communication protocols.

element

A component found within XML Schemas and DTDs.

embedded data flow

A data flow created within one or more data flows.

enterprise data

Data that is defined for use across a corporate environment.

ERP system

Enterprise resource planning system. A **back-office application** from which Data Integrator can extract data.

exception

An error that occurs while executing a **job**. You can catch individual or groups of exceptions using a **try/catch block** inside a **work flow**. Catching an exception allows you to automatically execute a solution for the error.

Executive Information Systems (EIS)

Tools programmed to provide canned reports or briefing books to executives. They offer strong reporting and drill-down capabilities. These tools allow ad hoc querying against a multi-dimensional **database**, and most offer analytical applications along functional lines such as sales or financial analysis.

extract date

The date data was extracted.

extract frequency

The interval at which data is extracted, such as daily, weekly, monthly, or quarterly. The frequency that data extracts are needed in the **data warehouse** is determined by the shortest frequency requested through an order, or by the frequency required to maintain consistency of the other associated data types in the source data warehouse.

file

A **source** or **target** for data. To read or load data to a file within Data Integrator, define a **file format** that instructs Data Integrator about the structure of the data in the file.

file format

A description of how data is or should be organized in a **file** Data Integrator reads from or loads to. A file format can be specific to a single file or generic for many files.

front-office application

Software that companies use to support reporting, **e-commerce**, and other tasks. Types of front-office applications include:

- Supply chain collaboration
- Analytical
- Sell-side e-commerce
- Customer relationship management

function

A program that operates on values that are passed to it. Data Integrator functions are available through a function wizard in a **script**, **conditional**, or **Query transform**. Data Integrator also gives you access to functions provided by the **DBMS** you are using. In addition, you can define your own functions using the Data Integrator scripting language or by writing C-language programs.

IDoc

Intermediate Document. An SAP-specific format. Used for **EDI** and **ALE**.

IDoc type

Indicates the SAP format that is used to interpret the data of a business transaction. Consists of the following components:

- A control record

  Identical for each IDoc type.

- Several data records

  A single data record consists of a fixed key part and a variable data part. The data part is interpreted using segments, which differ depending on the IDoc type selected.

- Several status records

  Identical for each IDoc type. Describe the status states an IDoc has already passed through or the status an IDoc has attained.

import

The process of acquiring information for the **Data Integrator repository**. Import the following kinds of information into Data Integrator:

- The **metadata** for **source** and **target** databases
- Descriptions and code for user-defined and DBMS **functions** and **transforms**

InfoArea

In **SAP BW**, an arbitrary grouping of **InfoCube**s.

InfoCube

In **SAP BW**, a multidimensional star schema that is limited to one fact table and 16 dimensions.

InfoCubes store only numeric data at the fact level.

SAP BW reserves three of the dimensions for currencies, time, and units of measure. Dimensions have to be linked back to master data and hierarchy tables to gain access to the information stored there.

When a user query is initiated, BW creates a temporary **OLAP** cube that stores a constrained subset of the permanent OLAP cube. The temporary InfoCube persists until the user disconnects from BW or changes the query.

InfoObject

In **SAP BW**, an InfoObject is equivalent to a data element and its associated metadata (for example, description or business rules). Can be a **Key Figure** or a Characteristic.

InfoPackage

In **SAP BW**, a schedule for running BW updates.

Information resource (IR)

A producer or consumer of enterprise information. An information resource can be a flat file, a **back-office application**, an EAI system, or a Web application.

InfoSource

In **SAP BW**, a logical construct that defines a set of **InfoObject**s and their mappings to source data elements. InfoSources do not store data, but rather serve as conduits via **Transfer Structure**s.

Integration Rapid Mart

An **Data Integrator application** that provides point-to-point content mapping and data movement mechanisms between different **back-office applications**.

interactive debugger

A Designer feature that allows you to step through the data of a job one row at a time using filters and breakpoints on a line. Like executing a job, you can start the interactive debugger from the Debug menu when a job is active in the workspace. While in debug mode, all other Designer features are set to read-only.

interface

Data Integrator offers two types of interfaces:

An internal **Data Integrator interface**, allows you to create datastore connections to natively supported applications.

An external **Data Integrator interface** (or **adapter**), allows Data Integrator to communicate with information exchange technologies such as Web Services and MQ Series.

IR

See **Information resource (IR)**.

job

The unit of work that can be scheduled independently for execution by the **Administrator**. Jobs are special **work flows** that can be called by other work flows or jobs.

Job Server

A process that receives requests from the **Designer** and the **Administrator** to start and stop **jobs**. To start batch or real-time jobs, the Job Server triggers the **Data Integrator engine**. Engine processes run on the same computer as the Job Server process that triggers them.

join rank

A value given to or calculated for all data sets in a **data flow**. Data Integrator uses the join rank to determine which **source** to read first when assembling the **data set** in a join. Data Integrator uses the source with the lower join rank as the inner source of the join and uses the source with the higher join rank as the outer source of the join. If more than one of the sources is located in the same **DBMS**, the highest rank is used for all of the sources from that DBMS. When a source is processed through a **Query transform** or another **transform**, and then used in a join, the maximum join rank defined for the source is used as the join rank in the query.

Key Figure

In **SAP BW**, an **InfoObject** that represents a numeric fact.

legacy system

An information or transaction processing system used to store data such as bank balances, inventories, payroll, and manufacturing parts.

license-controlled feature

A Data Integrator feature that is enabled or disabled based on the product license. The product license controls which icons and settings are available in Data Integrator as an internal **Data Integrator interface**.

line

The line you use to connect objects in a workspace diagram. Lines represent a left-to-right flow path for data moving between two objects during job execution.

memory datastore

A datastore connection/container for memory tables.

memory table

Internal Data Integrator table used to store a data set in memory while a job runs. Use instead of staging tables to improve performance of a real-time job built with multiple data flows. Use a memory table to move a data set between data flows.

message

Represents hierarchical data, such as a header with line items, for document-oriented transactions, such as a purchase order.

metadata

In Data Integrator, information acquired and maintained to describe tables in **source** and **target** databases. This information includes the names of tables and their columns, and the data types of the columns. All the metadata for all user tables is imported at one time for a given **datastore**.

In general, metadata typically includes a description of data models, a description of the layouts used in database design, the definition of the system of record, the mapping of data from the system of record to other places in the environment, and specific database design definitions.

multiple data flow model

One of three data flow models you can use in real-time jobs. Use this model to develop real-time jobs in separate logical object segments (data flows) rather than in a single data flow. Allows for any number of data flows in a real-time processing loop under these conditions: The first data flow must have a single message source, the last data flow must have a single message target. Other data flows in the same real-time job cannot have a message source or message target. See also **put/acknowledge data flow model** and **single data flow model**.

nested data

Data in one **table** that is related to a single row of another table. A nested table appears in Data Integrator as a column in a parent table. Columns in the nested table can themselves contain tables.

object

Any item that you create in the **Designer**. Data Integrator distinguishes two classes of objects: **reusable objects** that are complete and can be reused in your projects (such as **data flows**) and **single-use objects** that only appear as components of other objects (such as a **try/catch block**). This distinction affects how you create and retrieve each type of object.

object definition

The options that describe the operation of an **object**. To view and modify an object definition, open the object so that its definition appears in the **workspace**.

object library

A tool in the **Designer** that gives you access to reusable objects.

ODBC

Open database connectivity. A standard for **database** access initiated by the SQL Access Group consortium and now owned by Microsoft.

ODS

Operational data store. An OLAP-designed relational **database** that an enterprise has designated as the operational database of record (for example, a finance department might use an ODS to close its books).

OLAP

Online analytical processing. A relational **database** design that is optimized for random access. OLAP systems are used in a query environment, such as for a **front-office application**.

OLTP

Online transaction processing. A relational database design optimized for operational use. OLTP systems are used in an operational environment, such as for a **back-office application**.

operation code

A flag associated with a row in a **data set** that indicates the status of the data in the row. The operation codes are INSERT, UPDATE, DELETE, and NORMAL.

operational statistics

A category of reports within the metadata reporting tool that provides you with job and data flow performance information over a given period of time or number of executions.

parameter

A value passed to a **work flow** or **data flow** when that flow is called.

partition

To divide table data into sets based on a criteria such as a range or list of values in each row. You can configure Data Integrator to read and write partitioned table data in parallel threads. Designing jobs with partitioned table data can improve job performance if a Job Server's computer memory and number of CPUs supports the job's parallel-processing configuration settings.

project

The collection of **jobs** available in the **Designer** at a given time. A project provides a way to organize the **objects** you create.

property

Detailed descriptive information about **objects**.

put/acknowledge data flow model

One of three data flow models you can use in real-time jobs. Similar to the multiple data flow model, this model does not require a data flow with a target message. Data Integrator creates a target message to automatically acknowledge a request message. Designed for use with IDoc sources in SAP R/3 systems. See also **multiple data flow model** and **single data flow model**.

Query transform

A **data transformation object** that you can use to map columns from a **source** to a **target** schema, add new columns to the target schema, determine the data to extract, and perform operations on the data. Like a SQL SELECT statement, a query creates a **data set** that satisfies the conditions you specify.

Rapid Mart

A prepackaged **Data Integrator application** that delivers data for a specific business purpose. Easily deployed and customized, BusinessObjects™ Rapid Marts include the extraction, transformation, and data loading logic to support analysis or **e-commerce**. There are three types of Rapid Mart:

- **Analytic Rapid Mart**
- **Commerce Rapid Mart**
- **Integration Rapid Mart**

R/3 data flow

A **data flow** that extracts data from an **SAP R/3** source table. Data Integrator translates steps you define in an R/3 data flow into **ABAP** and then passes the ABAP program back to your SAP R/3 system for execution. The resulting **table** or **file** resides on the SAP R/3 system to be used as a **source** in the parent data flow.

real-time job

A group of objects (data flows, work flows, conditionals, scripts, and so forth) that execute on-demand as a "request-response" system. You design real-time jobs in the **Designer**, then configure them as real-time services and associate them with an **Access Server** in the **Administrator**, where they are started, managed and monitored. When a real-time service receives a request from a caller, it processes the request and returns a reply. This type of job is recommended for operations on small amounts of data.

relational data

A **data set** in which data in each column contains a scalar value. Data Integrator can process relational data; it can also process **nested data**.

repository

See **Data Integrator repository**.

reusable object

An **object** that can be defined, stored, and reused independent of other objects. Create reusable objects from the **object library**.

RFC server

Remote Function Call server. The Data Integrator RFC server allows third-party programs, including SAP R/3, to schedule and initiate Data Integrator jobs and return the results to Data Integrator.

SAP BW

Business Information Warehouse, an SAP product.

SAP R/3

An **ERP system**.

script

A step in a **work flow** that allows you to calculate values to pass to other parts of the work flow. The script can call **function**s, execute if-then-else statements, and assign values to **variable**s. Write a script in the Data Integrator scripting language.

segment

Format with which the data records of **IDocs** are interpreted.

server group

A defined collection of Job Servers on different computers. A server group automatically measures resource availability on each Job Server in the group and distributes scheduled batch jobs to the Job Server with the lightest load at run-time. Use the Server Groups node in the Administrator's navigation tree to group Job Servers that are associated with the same repository into a server group.

service request

Any message sent from a Web client that requires processing by a **real-time job**.

session

Associated with the **Adapter Properties** tab in an adapter datastore, a session is created by a stream-oriented adapter to group adapter components and maintain state for interactivity between Data Integrator and the adapter.

single data flow model

One of three data flow models you can use in real-time jobs. Allows for one data flow in a real-time processing loop. This data flow must have a single message source and a single message target. See also **multiple data flow model** and **put/acknowledge data flow model**.

single-use object

A step in a **work flow** or **data flow** that cannot be saved independently of the flow. Create single-use objects from the tool palette.

smart editor

A flexible editing tool in Data Integrator used for creating scripts, expressions, and custom functions without having to type the names of existing elements like column, function, and variable names.

SNMP

System Network Management Protocol helps network administrators manage network routing hardware. The protocol can manage a variety of hardware and software devices.

source

An **object** from which Data Integrator reads data.

SQL

Structured Query Language. A query language for accessing relational, **ODBC**, DRDA, or non-relational **database** systems.

SQL query tool

An end-user tool that accepts **SQL** to be processed against one or more relational databases.

star schema

A **database** design you can use to format data in a **data mart**. This design is based on a single fact table to which any number of dimensional tables may be joined. This type of database design supports multi-dimensional database analysis.

step

An **object** that is part of the definition of a **work flow** or **data flow**. Each step is represented by an icon in the diagram of the flow and is connected to other steps to indicate the flow of data through the data flow or the order of execution in the work flow.

table

A **database** table that Data Integrator reads data from or loads data into. The path and mechanisms for reading and loading data and apportioning the data among rows and columns are defined in the **datastore** that the table is associated with. Writing a **data set** to a database table means sending a combination of rows with appropriate **operation code** to the database table.

target

An object in which Data Integrator loads extracted and transformed data in a **data flow**. Data Integrator only loads rows flagged as INSERT or UPDATE.

TCP/IP

Transmission Control Protocol/Internet Protocol. The basic communication protocol of the internet, and often intranets and extranets. A computer having direct access to the internet contains a copy of the TCP/IP program. TCP/IP makes it possible for computers to communicate with each other.

Tdpid

(Teradata Director Program ID) The server name Data Integrator uses when loading with the bulk loader option. Data Integrator uses tdpid as a Teradata Warehouse Builder operator attribute.

Transfer Structure

In **SAP BW**, a data structure that defines a set of **InfoObject**s to be moved into BW. To an ETL tool like Data Integrator, a Transfer Structure looks like a table.

transform

A step in a **data flow** that acts on a **data set**. Data Integrator transforms are available through the **object library**. You can also import user-defined transforms and transforms from your **DBMS**.

Transformation Rule (Transfer Rule)

In **SAP BW**, a data manipulation rule between a **Transfer Structure** and a **Communication Structure**. The results of these transformation are globally available to all **InfoCube**s.

try/catch block

A combination of a try object and one or more catch objects that define alternate execution paths in case an error occurs during the execution of a **job**. You can tune try/catch blocks to trap specific errors and to provide general alerts or messages if an error occurs.

Update Rule

In **SAP BW**, a data manipulation rule between a **Communication Structure** and an **InfoCube**. The result of the data manipulation is specific to one InfoCube.

variable

A symbolic placeholder for a value. Data Integrator lets you define local variables and global variables.

Local variables pertain to the **work flow** in which they are defined. You can pass the value into another **work flow** or **data flow** using a **parameter**.

Global variables pertain to the **job** in which they are defined. With global variables, there is no need to define parameters between **objects** in the job. Global variables can also be selected at execution time. This eliminates the need to open the Designer to set global variable values.

Web Services

A standard platform for integrating applications. Web Services allow different programs, constructed in different languages, on different platforms to communicate with each other. DATA INTEGRATOR supports three XML web service technologies: SOAP, WSDL, and **XML Schema**.

work flow

A **reusable object** containing **steps** to define the order of **job** execution. Work flows call **data flow**s, but cannot manipulate data themselves.

You call work flows from inside other work flows or **job**s. You pass information into or out of work flows using **parameter**s. You reuse work flows by dragging existing work flows from the **object library**.

workspace

The window inside the **Designer** in which you define, display, and modify **objects**. The workspace for a **data flow** contains an area to build a **diagram** representing the data flow definition. The workspace for a **transform** contains an editor for modifying the transform options.

WSDL

Web Services Definition Language. Web services are self-contained, modular business process applications based on open Internet standards.

XML

Extensible Markup Language. This markup language is like HTML (Hypertext Markup Language) in that it specifies a standard with which you can define your own markup languages with their own sets of tags. XML allows you to define various tags with various rules, such as tags that represent business rules, tags that represent data description, or tags that represent data relationships.

XML Schema

The XML format used by Data Integrator to support message processing that includes **Web Services**. XML Schemas describe the data structure of an **XML** file or message. Data flows can read and write data to messages or files based on a specified XML Schema format. You can use the same XML Schema to describe multiple XML sources or targets. XML Schema properties include: Name, Description, Imported from, Root element name, and Namespace.

Index

Business Objects, Americas
3030 Orchard Parkway
San Jose, CA 95134
USA

info@businessobjects.com
www.businessobjects.com

Data Integrator

Advanced Development and Migration Guide

Version 6.5

Document Number DI-65-0118-001

January 30, 2004

Contents

1 Overview

One of the most powerful aspects of Data Integrator is its architectural flexibility through development, test, and production environments. Data Integrator is designed to support a number of environments, including large enterprises with many developers working on multiple projects. Data Integrator supports multi-site architectures whether centralized or not.

This book discusses architectural options for implementing Data Integrator in development, test, and production environments and covers the following topics:

- Migration basics
- Preparing for migration
- Export/import
- Multi-user development
- Multi-user environment setup
- Working in a multi-user environment
- Migrating multi-user jobs

This chapter also includes the following sections:

- About this document
- Audience and assumptions
- More Data Integrator product documentation

About this document

The book contains advanced development information. Topics include:

- *Migration*
  How to move your projects to different environments
- *Multi-user development*
  How to manage a project developed by multiple users

You will find this book most useful:

- After you have learned product basics
- While planning the design, test, and production phases of your data-movement projects
- As an advanced source of information during any phase of your projects

Audience and assumptions

This and other Data Integrator product documentation assumes the following:

- You are an application developer, consultant, or database administrator working on data extraction, data warehousing, or data integration.
- You understand your source data systems, RDBMS, business intelligence, and messaging concepts.
- You understand your organization's data needs.
- You are familiar with SQL (Structured Query Language).
- If you are interested in using this product to design real-time processing, you are familiar with:
  - DTD and XML Schema formats for XML files
  - Publishing Web Services (WSDL, HTTP, and SOAP protocols, etc.)
- You are familiar Data Integrator installation environments—Microsoft Windows or UNIX.

Further, Business Objects recommends that you review both the *Data Integrator Getting Started Guide* and the *Data Integrator Designer Guide* and before using advanced concepts in this document. Find both documents in *Data Integrator Technical Manuals*.

More Data Integrator product documentation

Consult the *Data Integrator Getting Started Guide* for:

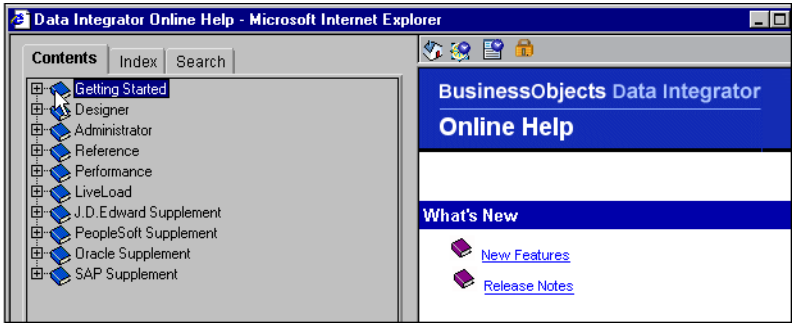
- An overview of Data Integrator products and architecture
- Data Integrator installation and configuration information
- A list of product documentation and a suggested reading path

After you install Data Integrator, you can view technical documentation from many locations. To view documentation in PDF format, you can:

- Select **Start** > **Programs** > **Data Integrator** ***version*** > **Data Integrator Documentation** and choose:
  - Release Notes
  - Release Summary
  - Technical Manuals
- Select one of the following from the Designer's **Help** menu:
  - Release Notes
  - Release Summary
  - Technical Manuals
- Select **Help** from the Data Integrator Administrator

You can also view and download PDF documentation by visiting Business Objects Customer Support online. To access this Web site, you must have a valid user name and password. To obtain your user name and password, go to http://www.techsupport.businessobjects.com and click **Register**.

After installing Data Integrator Designer you can also view documentation in HTML format. **Online Help** opens automatically the first time you open the Designer.



Use **Online Help** links and tool bar to navigate.

Online help tool bar

Icon	Icon Name	Description
	Home	Returns to the Online Help home page.
	Show/Hide Navigation	Shows and hides the Table of Contents for the core suite of Data Integrator Technical Manuals. Includes a master **Index** and **Search** tabs.
	Show/Hide Descriptions	The Online Help window automatically updates its information when you click an object in the Designer window. In addition, you can click any object in the Designer window then click this button to show or hide an object's local and global variables.
	Start/Stop Help Updates	When the lock appears open, each time you click an object in the Designer window, the Online Help information automatically updates.
		When the lock icon appears closed, the Online Help window does not automatically update when you click objects in the Designer window; you must press F1.

If you close the **Help** window, to reopen it:

- Choose **Contents** from the Help menu, or
- Click an object in the object library or workspace and press **F1**

Online Help opens to the subject you selected.

2 Migration basics

Migration as it relates to Data Integrator is the process of moving applications through multiple development phases into production. Data Integrator supports simple and complex application migration through all phases into production.

This chapter includes the following sections:

- Development phases
- Migration mechanisms and tools

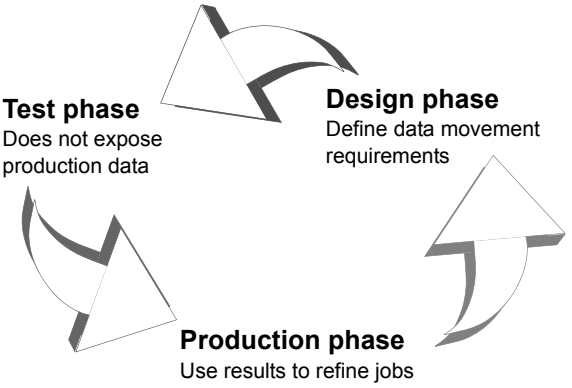
Development phases

The ETL application development process typically involves three distinct phases:

- Design phase
- Test phase
- Production phase

You can use Data Integrator in all three phases. Because each phase might require a different repository to control environment differences, Data Integrator provides controlled mechanisms for moving objects from phase to phase.

Each phase could involve a different computer in a different environment with different security settings. For example, design and initial test may only require limited sample data and low security, while final testing may require a full emulation of the production environment including strict security.



Design phase

In this phase, you define objects and build diagrams that instruct Data Integrator in your data movement requirements. Data Integrator stores these specifications so you can reuse them or modify them as your system evolves.

Design your project with migration to testing and final production in mind. Consider these basic guidelines as you design your project:

- Construct design steps as independent, testable modules.
- Use meaningful names for each step you construct.
- Make independent modules that can be used repeatedly to handle common operations.
- Use test data that reflects all the variations in your production data.

Test phase

In this phase, you use Data Integrator to test the execution of your application. At this point, you can test for errors and trace the flow of execution without exposing production data to any risk. If you discover errors during this phase, return the application to the design phase for correction, then test the corrected application.

Testing has two parts:

- The first part includes designing the data movement using your local repository.
- The second part includes fully emulating your production environment, including data volume.

Data Integrator provides feedback through trace, error, and statistics logs during both parts of this phase.

The testing repository should emulate your production environment as closely as possible, including scheduling jobs rather than manually starting them.

Production phase

In this phase, you set up a schedule in Data Integrator to run your application as a job. Evaluate results from production runs and when necessary, return to the design phase to optimize performance and refine your target requirements.

After moving a Data Integrator application into production, monitor it in the Administrator for performance and results. During production:

- Monitor your jobs and the time it takes for them to complete.

  The trace and monitoring logs provide information about each job as well as the work flows and data flows contained within the job.

  You can customize the log details. However, the more information you request in the logs, the longer the job runs. Balance job run-time against the information necessary to analyze job performance. For more information see the *Data Integrator Performance Optimization Guide*.

- Check the accuracy of your data.

To enhance or correct your jobs:

- Make changes in your design environment.
- Repeat the object testing.
- Move changed objects back into production.

Migration mechanisms and tools

Data Integrator provides two migration mechanisms:

- Export/import migration works best with small to medium-sized projects where a small number of developers work on somewhat independent Data Integrator applications through all phases of development.
- Multi-user development works best in larger projects where two or more developers or multiple teams are working on interdependent parts of Data Integrator applications through all phases of development.

Regardless of which migration mechanism you choose, Business Objects recommends you prepare for migration using one or more tools that best fit your development environment (see Chapter 3, "Preparing for migration" for more information). The mechanism and tools you use will depend on the needs of your development environment.

If your source data will come from multiple, homogeneous systems, Business Objects recommends you use Datastore and system profiles tools. When migrating applications in a multi-user environment, Business Objects strongly recommends you use Naming conventions for migration.

Which mechanism is best?

Although Data Integrator supports a multi-user environment, you may not need to implement this architecture on all projects. If your project is small to medium in size and only consists of one or two developers, then a Central Repository may not be a necessary solution to integrating the work of those developers.

For example, only two Data Integration Consultants worked on the PeopleSoft HR Rapid Mart. The Development system was architected so that while Consultant 1 managed the Master Repository, Consultant 2 worked on a new section within a complete copy of the Master Repository.

Consultant 2 then exported this new section back into the Master Repository using the export utility that allows objects to be 'Created', 'Replaced', and 'Ignored'. After updating the Master Repository, Consultant 2 took a new complete copy of the Master Repository overwriting the previous copy.

Use the following matrix to help you determine which mechanism and tools would work best in your environment.

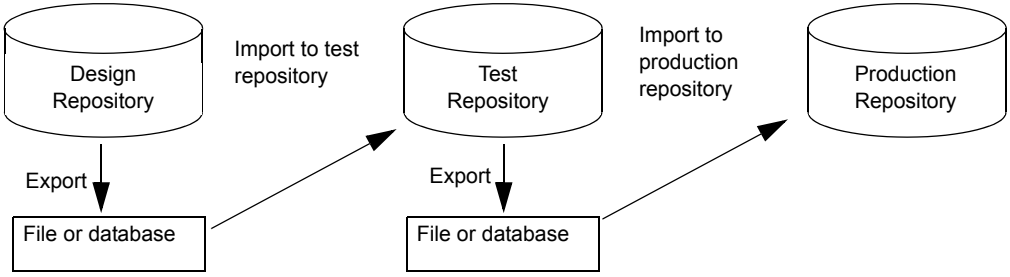
Migration guideline matrix

	Migration Mechanisms		Tools	
Situation/requirements	**Export/import**	**Multi-user**	**Naming conventions**	**Profiles**
Small to medium-sized project	☆		✓	✓
Multiple-team project		☆	☆	✓
Source data from multiple, homogeneous systems			☆	☆
Need a "fast and easy" migration solution	✓		☆	

Optimal solution: ☆ Compatible solution: ✓

Export/import migration

Export/import is the basic mechanism for migrating Data Integrator applications between phases. First, you *export* jobs from the local repository to either a file or a database, then you can import them into another local repository. For example, when moving from design repository to test repository, you export from design repository and import the file or database to your test repository.



If you find application errors and need to send back to the design repository, simply export from the test repository and import back into the design repository. After you correct any application errors, export again, import back into the test repository, and retest. For more information on exporting, see Chapter 4, "Export/import".

Multi-user migration

You can also migrate Data Integrator applications between phases in more complex development environments. Instead of exporting and importing applications, multi-user development provides a more secure *check-in*, *check-out*, and *get* mechanism, using a *central repository* to store the master copies of your application elements. Multi-user development includes other advanced features like labeling and filtering to provide you more flexibility and control in managing application objects. To learn more about multi-user development migration, see Chapter 8, "Migrating multi-user jobs".

3 Preparing for migration

Before you develop Data Integrator applications, Business Objects recommends that you first set up a comprehensive structure to facilitate the migration process between development phases.

This chapter discusses two tools that can help you build your migration structure:

- Naming conventions for migration
- Datastore and system profiles

Business Objects recommends that you implement standardized naming conventions for connectivity between computer systems. Add datastore and system profiles to more easily work with multiple homogeneous systems.

Naming conventions for migration

The best way to ensure fast and seamless migration is to use common naming conventions across all systems and phases of all your development environments.

Just as Business Objects recommends you standardize object prefixes, suffixes, and path name identifiers to simplify your projects internally, we also recommend the use of naming conventions externally for migration purposes.

NOTE: To learn more about object identification in jobs, see "Naming conventions for objects in jobs" on page 75 of the *Data Integrator Designer Guide*.

To ease migration, use common naming conventions for:

- Connections to external datastores
- Directory locations
- Schema structures and owners

You want to make it as quick and easy as possible to migrate applications between users and between phases. This translates to significantly reducing or eliminating time spent reconfiguring your jobs to work in each specific environment.

While the actual data you are extracting, transforming, and loading usually differs by database, the essential *structure* of the data should be the same on every database with which you want the same applications to work. Therefore, it makes the most sense to standardize your database naming and structuring before starting the development process.

Connections to external datastores

In Data Integrator, migration is the process of moving objects between local repositories, whether directly using the Export/import method or indirectly using the Multi-user development method. Regardless of method, you must consider how the migration will impact connection configurations associated with your jobs.

Using generic naming for similar external datastore connections reduces the time you spend on reconfiguring the connections to the same database type. For example, you should choose the same logical name for all your Oracle datastore connections to the same type of database structure regardless of migration phase environment.

When you make connection names meaningful to a certain phase and specific computer system names (Test_DW, Dev_DW, Prod_DW), you make it difficult to migrate between phases.

Development phase	Test phase
User name: Dev_DW	User name: Test_DW
Password: Dev_DW	Password: Test_DW
Host String: Dev_DW	Host String: Test_DW

For the same job to run against Test and Development, the job would have to use Test_DW and Dev_DW and this would require you to reconfigure the connection depending on whether the job was running against the Test or the Dev instance.

Instead, you could simply call the connection string DW and regardless of what instance you ran the job against, it would run without users having to edit the datastore properties.

Development Phase		Test Phase	
Database A	**Datastore Connection**	**Database B**	**Datastore Connection**
User name: DW	User name: DW	User name: DW	User name: DW
Password: DW	Password: DW	Password: DW	Password: DW
Host string: DW	Owner name: DW	Host String: DW	Owner name: DW

Examples:

- There is one Oracle source system in your company that processes order entry data. Multiple instances of this system exist for development, test, and production purposes. Therefore, you name the connection string to your Oracle source system "ORDER_SYSTEM" then in all phases configure that name to point to the correct (phase-specific) instance of the system.
- Name the connection string to your target data warehouse "DW" then point it to different databases depending on whether you are in the development, test, or production environment.

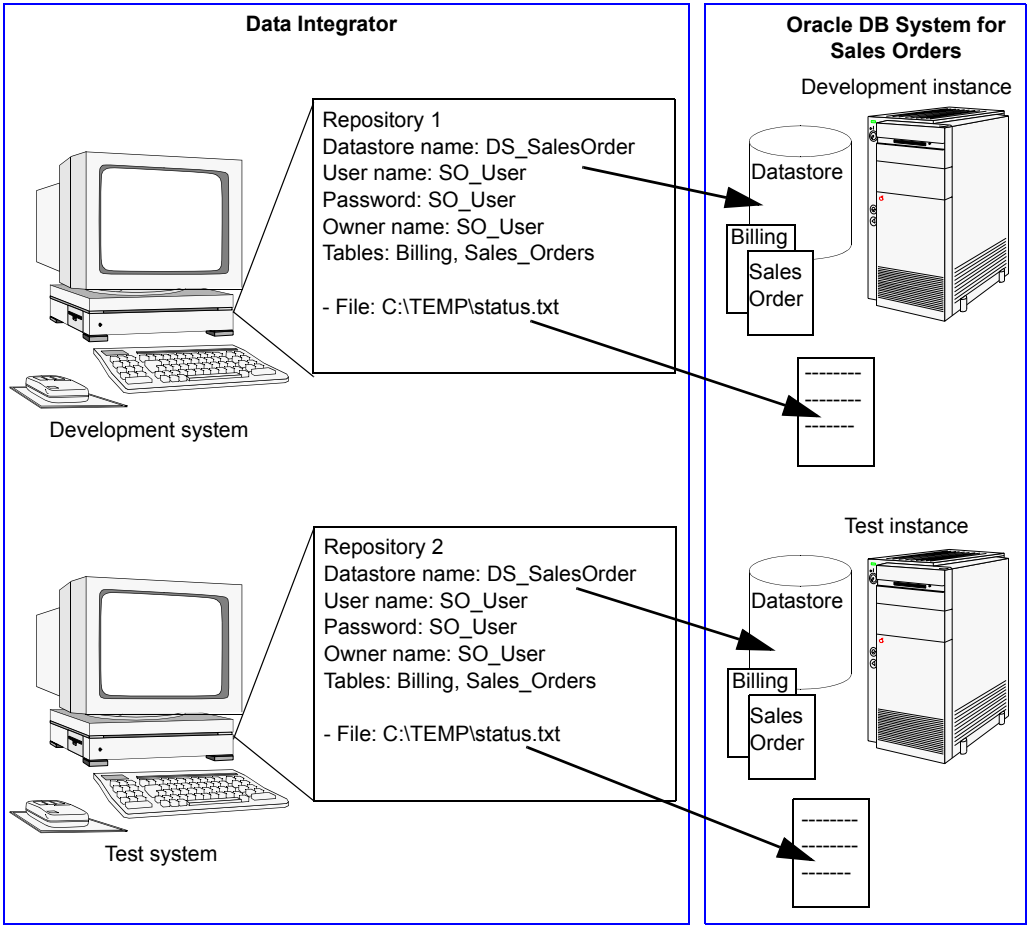
When you use this generic, cross-phase naming method, you cannot access *both* dev and test from the same computer (since the connection string maps only to one instance).

Directory locations

Business Objects recommends you use logical directory names (for example, X:\)or point to common local drives to standardize directory location. For example, since every computer has a C drive, pointing to the directory location, C:\TEMP would be a safe, reproducible standard.

Schema structures and owners

To further facilitate a seamless structure between development phases, give all your database instances the same owner name for the same schema structures from which you are reading and to which you are loading. Regardless of name, the owner of each schema structure can vary and Data Integrator will reconcile them.



Datastore and system profiles

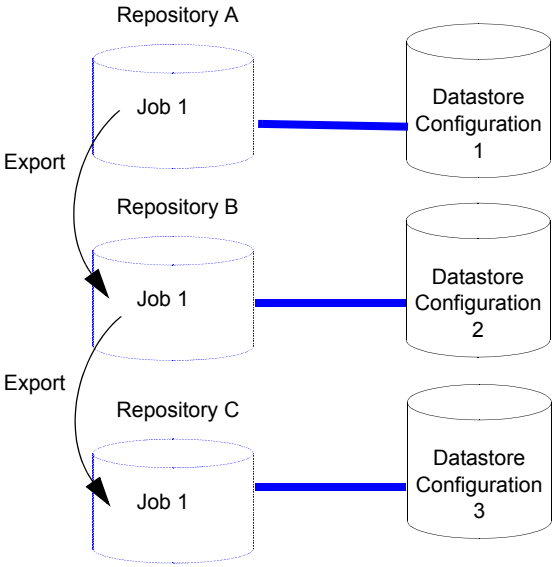
Datastore and system profiles are powerful tools for reducing the configurations required to execute the same logic against different datastore environments. With profiles, migration between development phases becomes faster and more simplified.

This chapter includes the following sections:

Datastore profiles and migration

Without multiple profile datastores, each time you export/import from one repository to another, you must spend time reconfiguring datastore connections to work with the new repository (and sometimes new host computer).

Without multiple profiles, each job in a repository can only run against one datastore configuration.



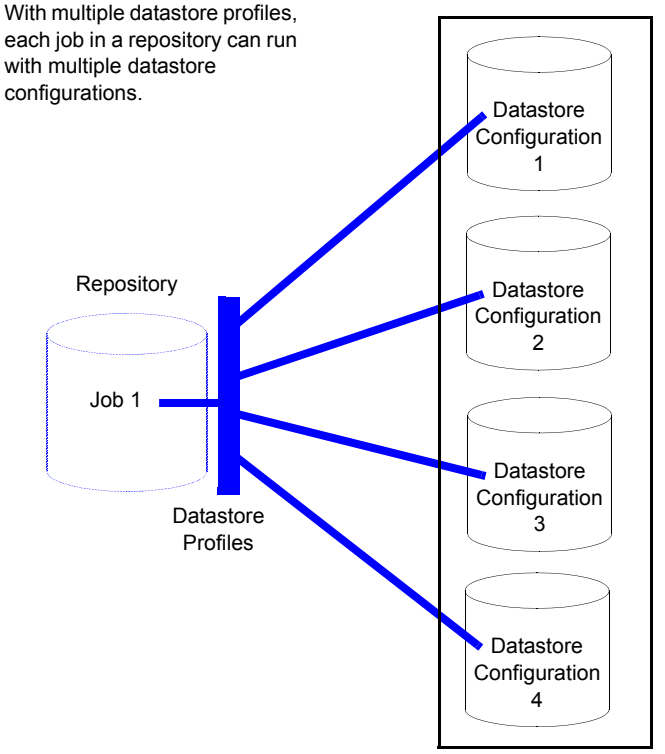
With multiple profiles, instead of a separate datastore (and datastore configuration) for each database instance, you can associate multiple *datastore profiles* with a single datastore connection.

Each datastore profile defines a connection for a particular datastore. When working with multiple datastore profiles, Data Integrator can only import objects from a database instance if those objects share the same owner name.

After importing objects, Data Integrator temporarily replaces the specific owner name with a default “dummy” name (DSOWNER). Then during run-time, Data Integrator substitutes a real owner name for DSOWNER based on your datastore profile and *system profile* selections.

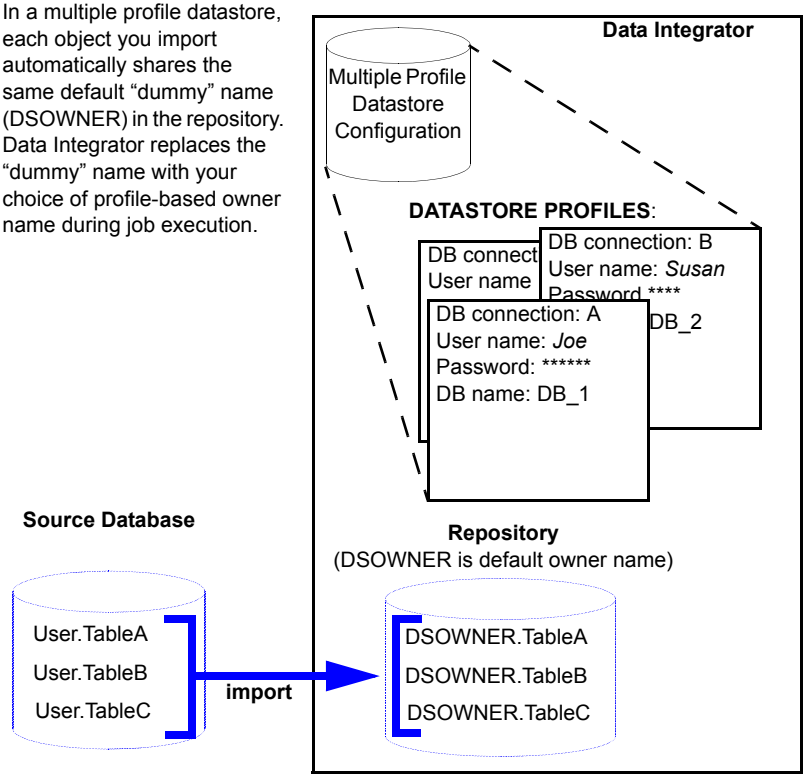
Therefore, to use datastore profiles, you must use DSOWNER as the owner of your datastore objects.

Each system profile defines a set of datastore profiles that you want to use together when running a job. You must create datastore profiles for the datastores in your repository before you can create system profiles.



All objects you want to import into a multiple profiles datastore must share the same owner.

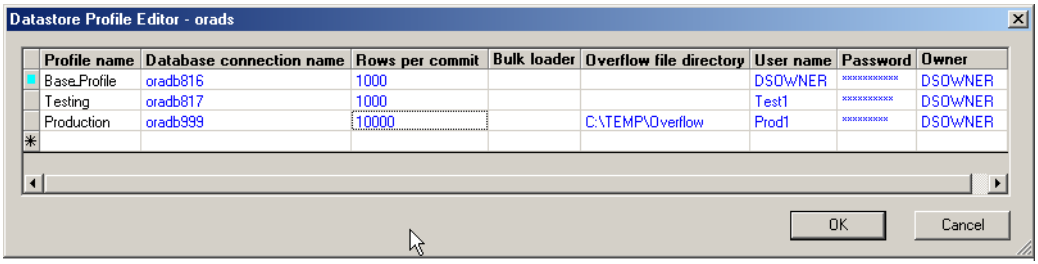
In a multiple profile datastore, each object you import automatically shares the same default “dummy” name (DSOWNER) in the repository. Data Integrator replaces the “dummy” name with your choice of profile-based owner name during job execution.



How datastore profiles work

To associate a datastore with multiple configuration profiles, when creating or editing most types of datastores, you must first select the **Multiple profiles** option. Datastores with multiple profiles support development in homogeneous environments where systems use different connections and properties but the same naming structures.

In the Datastore Profile editor, configure multiple profiles with different databases, commit sizes, bulkload and overflow directories, owners, user IDs, and passwords. You can also use multiple profiles to define Development, Test, and Production, or site-specific profiles for a datastore such as ORDERS_EAST, ORDERS_SOUTH, and so forth.



You could have several people designing jobs that can use the same datastore. Each person would have their own profile, configured for their computer and database connections. The datastore name and dummy owner (always DSOWNER) would be the same for each user. However, profile name and associated connection information would specify different designers, computers, and database connections.

Another use example: You have identical databases for different departments in your firm and want to run the same job to work with data from all of the databases, perhaps a job that tracks billable time for several departments.

Your source information exists on different databases and each has a unique database connection name. You could create a datastore for each source. However, a more elegant solution would be to create a single datastore with multiple datastore profiles. Then, create system profiles that contain each datastore profile. With the datastore profiling solution, at run-time you would just select the appropriate system profile.

Each datastore profile you define contains connection information. Your datastore profiles can support multiple development environments and cross-development phases using the same structural rules.

For specific instructions to define datastore and system profiles, see "Defining profiles" on page 111 of the *Data Integrator Designer Guide*.

Multiple profiles in multi-user environments

Data Integrator also supports a multi-user development environment. A team can work together on an application during development, testing, and production phases. Further, different teams can work on the different phases simultaneously.

Individual users work on an application in their unique local repositories. The team uses a central repository to store, check in, and check out objects that belong to the application master copy. The central repository preserves all versions of an application's objects, allowing you to revert to a previous version if needed.

The easiest way to set up your environment to work with multi-user functionality is by establishing the exact same environment naming standards among your developers. In each developer's environment, the configuration would be different. For example a database connection string would point to their local database. However, if implementing these naming standards is not possible, you can still save time and streamline your multi-user environment by using multiple-profile datastores.

For example, if your database owners use the same metadata structure but different database instances and owners, you can define a datastore profile for each owner on your design team. This way, they can share and contribute to the same projects without having to set up their datastore connection information each time they check out a project from the central repository.

To find out more about multi-user environments, see Chapter 5, "Multi-user development", Chapter 6, "Multi-user environment setup", and Chapter 7, "Working in a multi-user environment". For more details on migrating within multi-user environments, see Chapter 8, "Migrating multi-user jobs".

4 Export/import

The simplest type of migration in Data Integrator is called export/import.

This chapter discusses the export/import method and includes the following sections:

- Exporting/importing objects in Data Integrator
- Removing obsolete repository contents
- Backing up repositories
- Maintaining Job Server performance

Exporting/importing objects in Data Integrator

The export feature gives you the flexibility to manage and migrate projects involving multiple developers and different execution environments. When you export a job from a development repository to a production repository, you can change the properties of objects being exported to match your production environment.

In particular, you can change datastore definitions—application and database locations and login information—to reflect production sources and targets.

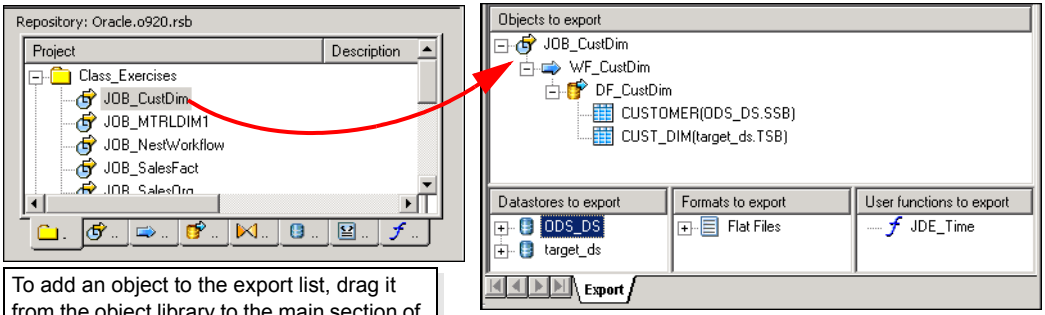
You can export objects to another repository or a flat file (.atl or .xml). If the destination is another repository, you must be able to connect to and have write permission for that repository, and your repository versions must match.

This section discusses:

The Export editor

In the Export editor, specify the objects you want to export and an export location. Choose **Tools > Export** or select an object and right-click **Export** to open the export editor.

To specify an object to export, drag the object from the object library into the **Objects to Export** window.



To add an object to the export list, drag it from the object library to the main section of the export editor.

The Object to Export window shows the final list of objects to be exported. When you drag any object from the object library, the datastores, file formats, and custom functions included in the object definition are automatically added to the other export sections. Each object in an export window opens to show objects called by this object.

You can control which associated objects to exclude or include. For example, you can export a work flow and all tables contained in the work flow without exporting an associated data flow.

To control which objects to export, either select an object, right-click, and choose a shortcut menu option, or select the white space in the Export editor, right-click, and choose a shortcut menu option:

- **Exclude**

  Removes only the selected object from the list of objects to be exported. The object remains in the list, but its exclusion is indicated by a red "x" on the object icon.

  All occurrences of the object are excluded.

When you export the list, excluded objects are not copied to the destination. Objects called by this object are not removed from the list of objects to be exported, unless they are specifically excluded.

- **Include**

  Adds an excluded object to the export plan. The red "**X**" on the icon disappears. All occurrences of the object are included.

  When you export, the included objects are copied to the destination.

- **Exclude Tree**

  Removes the selected object and all objects called by this object from the export. The objects remain in the list, but their exclusion is indicated by a red "x" on the icons—the selected object and any objects it calls are excluded.

  When you export the list, the excluded objects are not copied to the destination.

- **Include Tree**

  Add the selected excluded object and the objects it calls to the export list. The red x on the selected object and dependents disappears. When you export the list, the included objects are copied to the destination.

- **Exclude environmental information**

  Removes all connections (datastores and formats) and their dependent content (tables, files, functions) from the objects in the Export editor.

  From the white space in the Export editor, you can right-click and select this option. Using this option you can export jobs without connections so that you avoid connection errors. Business Objects recommends you configure datastores and formats for the new environment separately.

When you export, excluded objects are not copied to the destination.

- **Include environmental information**

  Adds all connections (datastores and formats) and their dependent content (tables, files, functions) to the objects you want to export.

- **Clear All**

  Removes all objects from all sections of the editor.

- **Delete**

  Removes the selected object and objects it calls from the Export editor. Only the selected occurrence is deleted; if any of the effected objects appear in another place in the export plan, the objects are still exported.

  This option is available only at the top level. You cannot delete other objects; you can only exclude them.

- **Export**

  Starts the export process.

Exporting objects to a database

You can export objects from the current repository to another repository. However, the other repository must be the same version as the current one. The export process allows you to change environment-specific information defined in datastores and file formats to match the new environment.

➢ To export a repository object to a database

1. In the object library, choose an object to export.

   Right-click and choose **Export**.

   The Export editor opens in the workspace. To add more objects to the list of objects to export, drag the objects from the object library into the Objects to Export section of the editor.

2. Refine the list of objects to export.

   You can use the options available in the right-click menu for each object to include or exclude the object from the export list.

3. When your list is complete, right-click and choose **Export**.

4. In the Export Destination window, add the destination database connection information.

5. In Export Confirmation window, verify the components to export.

   The Destination status column shows the status of the component in the target database and the proposed action.

Destination Status	Action
Does not exist	Create/Exclude
Exists	Replace/Exclude

   To edit an action, select any number of objects (using SHIFT and CTRL keys) and select either **Create**, **Exclude**, or **Replace** from the **Target Status** list box.

6. Click **Next**.

7. In the Datastore Export Options window, select the datastore.

   You can change the owner of a table or the connection properties of the datastore.

Click **Advanced**.

8. Change the database connection information as required by the target database.

   Click **Next**.

9. In the File Format Mapping dialog, select a file and change the Destination Root Path if necessary.

   You can change the **Destination Root Path** for any file formats to match the new destination.

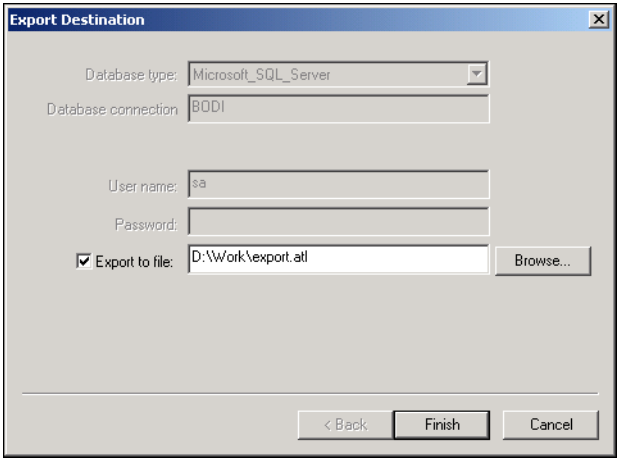
10. Click **Finish**.

Data Integrator copies objects in the Export editor to the target destination. When copying is complete, the objects display in the Output window. The Output window shows the number of objects exported as well as a list of any errors.

Exporting objects to a file

You can also export objects to a file. If you choose a file as the export destination, Data Integrator does not provide options to change environment-specific information.

NOTE: Objects in a repository are exported in the .atl format, while whole repositories can be exported in either the .atl or .xml format. ATL is Data Integrator's scripting language format. Using the .xml file format might make repository content easier for you to read. It also allows you to export Data Integrator to other products.



Exporting a repository to a file

You can also export an entire repository to a file. When you export or import a repository, jobs and their schedules (created in Data Integrator) are automatically exported or imported as well. Schedules cannot be exported or imported without an associated job and its repository.

If you choose a file as the export destination, Data Integrator does not provide options to change environment-specific information.

➢ To export a repository to a file

1. From the object library, right-click and choose **Repository > Export To File**.

   A window opens to prompt you for the destination of the export file. You can browse the directory to change the location, set the file type (.xml or .atl), and enter a name for the file.

2. Click **Open**.

   The repository is exported to the file.

Importing from a file

Importing objects or an entire repository from a file overwrites existing objects with the same names in the destination repository. You must restart Data Integrator after the import process completes.

➢ To import a repository from a file

1. There are two ways to import Data Integrator repository files into another repository. Use **Tools > Import from file**, or in the object library, right-click and choose **Repository > Import from File**.

   A window opens for you to specify the file to import. You can import individual files or the whole repository using either an XML or ATL file type. (ATL is Data Integrator's internal scripting language.)

2. Select the file to import and click **Open**.

Data Integrator displays this warning:

```
Warning: You are about to import using the
path filename file. This will create new
versions of existing objects and requires
shutting down the application. This may take
a long time. Are you sure?
```

3. Click **OK**.

   Data Integrator imports the repository file and shuts down.

4. Choose **Programs** > **BUSINESSOBJECTS DATA INTEGRATOR 6.1** > **Data Integrator Designer** from the **Start** menu.
5. Log in to the repository where the file was imported.
6. Verify that the file or repository was imported.

Removing obsolete repository contents

Data Integrator saves a version of each object every time you save the object. Repeatedly modified object definitions can consume a substantial amount of space. If you notice your repository performance degrading, consider compacting the repository.

To access the **Compact Repository** command, select **Project > Compact Repository** from the menu bar. This command removes previous object versions maintained by Data Integrator.

You can also compact your repository manually. If you have never compacted the repository, the majority of space in the repository could be occupied by old versions of Data Integrator objects. In this case, the **Compact Repository** command might be too slow and tedious. Instead, you can export the latest versions of the repository object definitions to a file, clear the repository database by creating a new repository, then reimport the object definitions.

➢ To compact your repository by creating a new repository

1. Export the repository to a file.

   The file type can be either XML or ATL. The latest version of each object is exported.

2. Choose **Repository Manager** from the **Start > Programs > Data Integrator** menu.

3. From the Repository Manager, add the database connection information for the repository.

4. Click **Create**.

   Data Integrator warns that a valid repository already exists.

5. Click **Yes** to overwrite the old repository.

   The Repository Manager creates a new repository, removing all of the old objects.

6. Import the previously exported repository.

Backing up repositories

Use your DBMS utilities to back up your repositories regularly. For information, refer to your DBMS documentation.

Maintaining Job Server performance

If you are designing jobs, typically you might use the same computer for your Designer, repository, and Job Server. In addition, you might use the same datastore for both your repository and your target database.

However, when you migrate your jobs into a test environment, the Job Server could move to a separate computer (typically from a Windows to a UNIX platform). The Data Integrator Job Server computer uses source, target, and repository database client libraries to extract, transform, and load data according to a job's design. Therefore, the Job Server computer must have a database client installed for each database you are using to run a Data Integrator job. In addition, Data Integrator allows you to localize source and target databases using locale and codepage settings. For more information about locales, see "Locales and Multi-Byte Functionality" on page 549 of the *Data Integrator Reference Guide*.

When migrating jobs between different Job Servers verify that the codepage used by each source and target database is the same as the codepage set for the corresponding database client on the Job Server's computer.

The database client codepage used by a Job Server on a Windows might be different from the one used on UNIX. For example, the Oracle client codepage MS1252 on Windows should be changed to the ISO88591 codepage on UNIX.

Data Integrator allows different codepages to be used in sources and targets. Mismatched locale settings do not cause errors. However, mismatches typically result in performance degradation from transcoding done by Data Integrator during job execution.

If your jobs do not require the use of different locales, you can increase performance by ensuring that default locales are not mismatched. After migration, if you notice a significant difference between the speed of design and test environments, check locale settings. In the Designer, check to see that datastore codepages for sources and targets match client codepages on the Job Server computer.

5 Multi-user development

Data Integrator supports a multi-user development environment. A team can work together on an application during the development, testing, or production phase. And different teams can work on the different phases at the same time.

Each individual user works on an application in a unique local repository. The team uses a central repository to store the master copy of its application. The central repository preserves all versions of an application's objects, allowing you to revert to a previous version if needed.

This chapter discusses:

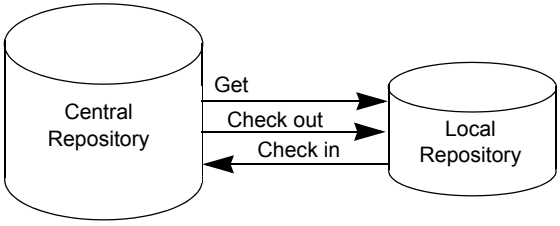
- Central versus local repository
- Data Integrator and multiple users

Central versus local repository

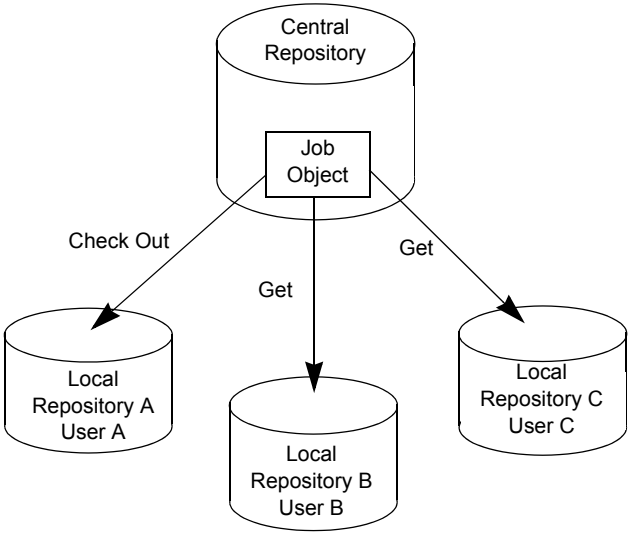
Data Integrator allows you to create a *central repository* for storing the master copy of a Data Integrator application. The central repository contains all information normally found in a repository such as definitions for each object in an application. However, the central repository is merely a storage location for this information. To change the information, you must work in a local repository.

A local repository provides a view of the central repository. You can "get" or copy objects from the central repository into your local repository. To make changes, "check out" an object from the central repository into your local repository. While you have an object checked out from the central repository, other users cannot change the information.

When done, you "check in" the changed object. When you check in objects, Data Integrator saves the new, modified objects in the central repository.



Multiple users working from unique local repositories can connect to the same central repository. These users can work on the same application and share their work. However, at any given time only one user can check out and change a particular object. While an object is checked out to one user, other users can "get" the object but cannot make changes that will update the central repository.



The central repository retains each object's history. Therefore, if you find you made a change that did not work as planned, you can revert to a previous version of the object.

The local repository and the central repository must use the same repository version. For example, you can run Data Integrator Designer X.2 with a central and local repository version X.1. However, you cannot run Data Integrator Designer X.2 with a central repository X.1 and a local repository X.2

Data Integrator and multiple users

A multi-user environment affects how you use Data Integrator and how you manage different phases of an application. For success in a multi-user environment, you must maintain consistency between your local repository and the central repository.

The following terms apply when discussing multi-user environments and Data Integrator:

- Highest level object

  The highest level object is the object that is not a dependent of any object in the object hierarchy. For example, if Job 1 is comprised of Work Flow 1 and Data Flow 1, then Job 1 is the highest level object.

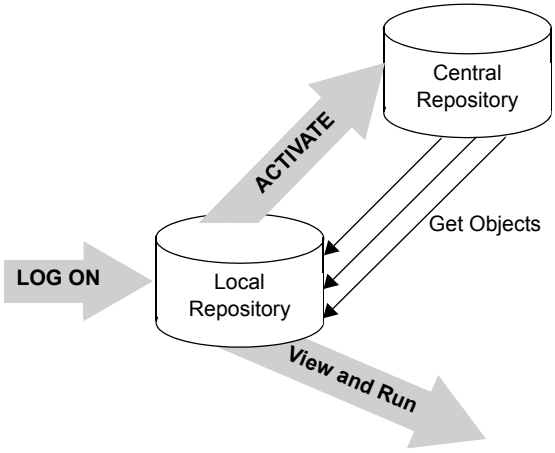
- Object dependents

  Object dependents are objects associated beneath the highest level object in the hierarchy. For example, if Job 1 is comprised of Work Flow 1 which contains Data Flow 1, then both Work Flow 1 and Data Flow 1 are dependents of Job 1. Further, Data Flow 1 is a dependent of Work Flow 1.

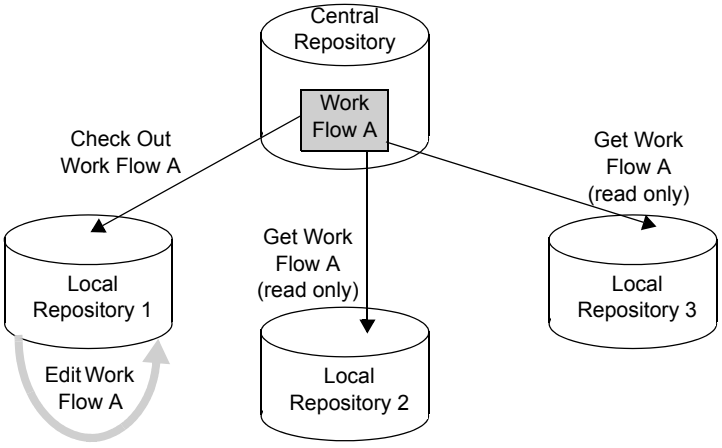
- Object version

  An object version is an instance of an object. Each time a you add or check in an object into the central repository, Data Integrator creates a version of the object. The latest version of an object is the last or most recent version created.

When working in a multi-user environment, you activate the link between your local repository and the corresponding central repository each time you log in. To ensure that your repository is current, you can get, or copy, the latest version of each object in the central repository. Once you get an application in your local repository, you can view and run it from the Designer.



However, if you plan to make changes to objects in the application, you should check out the objects. When you check out an object, no other user can make changes. Essentially, you lock the version in the central repository; only you can change that version. Other users can only get and view the object.



When you are done making changes to an object, save those changes in the local repository and check the object back into the central repository. Data Integrator saves the changed object in the central repository and makes the object available for check-out by others. Data Integrator maintains all versions of saved objects in the central repository. Thus later, you can copy an old version of a saved object, even after replacing it in your local repository with a new version.

At any time, you can label an object or a group of objects. An object label provides a convenient mechanism for identifying objects later. For example, you may find it helpful to label objects by feature. Later, if you decide you want to eliminate a recently-added feature, you can get all objects that have the label without that feature.

You can also compare two objects—such as two different object versions in the central repository, or an object in your local repository—to an object in the central repository. By comparing two objects, you can determine what parts of an object changed and decide whether you want to revert to an older version of an object.

6 Multi-user environment setup

To support multiple users, you must configure a multi-user environment and set up several repositories. Specifically, you must:

- Create a local repository for each user
- Create a central repository—see "Creating a central repository".
- Define a connection to central repository from each local repository—see "Defining connection to central repository" on page 51.
- Activate the connection to a central repository—see "Activating a central repository" on page 52.

Creating a central repository

To support multiple users in a single development environment, you must define a central repository. The central repository stores master information for the development environment.

➢ To create a central repository

1. Create a database to be used for the central repository using your database management system.
2. From the **Start** menu, choose **Programs** > **Data Integrator** > **Repository Manager** (assuming you installed Data Integrator in the Data Integrator directory).
3. In the Repository Manager window, click the **Central** button in the **Repository Type** field, and enter the database connection information for the central repository.
4. Click **Create**.

   Data Integrator creates repository tables in the database you identified.

Defining connection to central repository

A team working on an application only needs one central repository. However, each team member requires a local repository. Furthermore, each local repository requires connection information to any central repository it must access.

NOTE: The version of the central repository must match the version of the local repository.

➢ To define a connection to a central repository

1. Start the Data Integrator Designer and log in to your local repository.
2. Choose **Tools** > **Central Repositories** to open the Options window.

   The **Central Repository Connections** option is selected in the **Designer Options** list.
3. Right-click in the **Central Repository Connections** box and select **Add**.

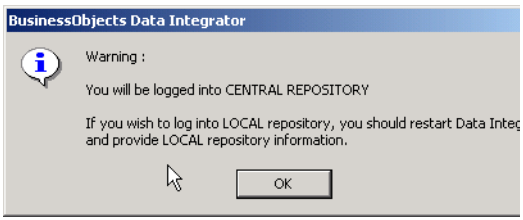
   The Datastore Administrator window opens.
4. In the **Name** box, enter a name to identify your central repository.
5. In the **Database Type** list, select the appropriate database type for your central repository.
6. Complete the appropriate login information for your database type.
7. Click **OK**.

   The list of central repository datastores now includes the newly connected central repository. You can continue adding additional connections or you can proceed to the next step.

Activating a central repository

To connect to a central repository, you must activate the link between your local repository and a specific central repository.

NOTE: When you start the Designer, always log in to a local repository. Never log into a central repository. If you do, then the central repository acts as a local repository. Then you run the risk of corrupting version information. If you attempt to log in to the central repository, Data Integrator will present a warning message. You should log out immediately and log into a local repository.



Your local repository provides a view of the objects in the active central repository. Whenever you get or check out objects, you copy objects from the active central repository. Whenever you check in objects, you save the version from your local repository into the active central repository.

You must activate the correct central repository each time you log in. When you activate a central repository, Data Integrator opens the central object library, which shows all the objects in the central repository and the check-out status of each object.

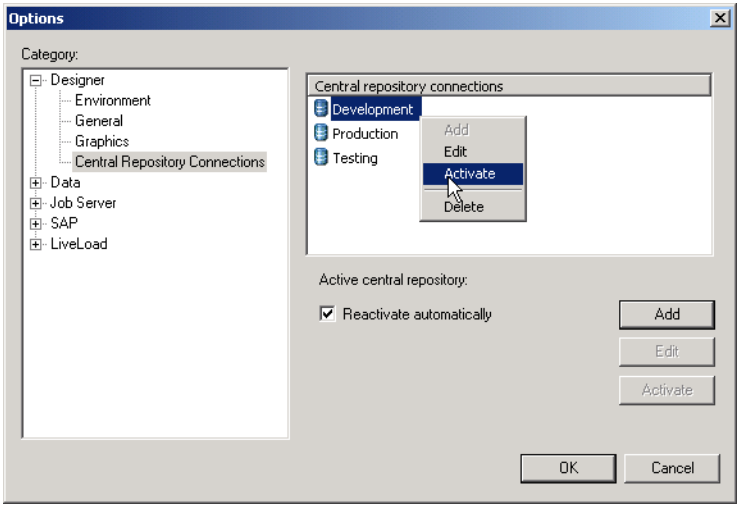
➢ To activate a central repository

1. Choose **Tools** > **Central Repositories** to open the Options window.

   The **Central Repository Connections** option is selected in the **Designer Options** list.

2. In the **Central Repository Datastore** list, select a central repository to make active.
3. Right-click and select **Activate**.

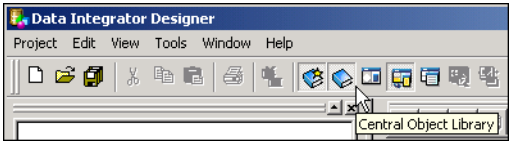
   Data Integrator activates the link between your local repository and the selected central repository and opens the central object library. The Options window indicates that the selected central repository is active.



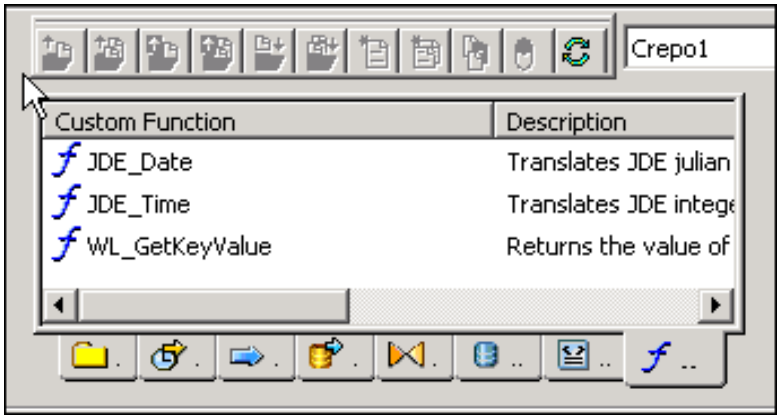
4. Check **Reactivate automatically** if you want the active central repository to be reactivated when you next log on to this local repository.
5. The Options window will be closed automatically.

➢ To open the central object library

Click the **Central Object Library** button on the toolbar.



The central object library looks like the object library—it shows all the objects in the repository, grouped on appropriate tabs.



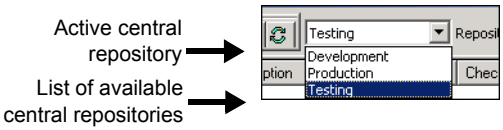
The window opens floating and can be docked by dragging. You can change the docking state by right-clicking on the window's tool bar and selecting **Allow Docking**.

You can also change central repository connection information from the central object library.

➢ To change the active central repository

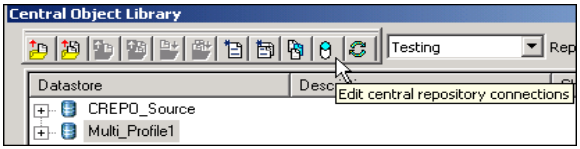
Select a central repository from the list on the top of the central object library.

Data Integrator makes the selected central repository active—objects from that repository appear in the central object library. Connection information about that repository appears in the upper right corner of the central object library.



➢ To change central repository connections

1. Click the **Edit Central Repository Connection** button on the top of the central object library.



   Data Integrator opens the Options window with the **Central Repository Connections** option selected in the **Designer Options** list.

   Alternatively, you can open the Options window by selecting **Tools** > **Central Repositories.**

2. Select a central repository in the **Central Repository Connections** box, right-click, and select **Edit.**

   The Datastore Administrator window opens.

   Follow the instructions beginning at step 4 "To define a connection to a central repository" on page 51.

3. To delete connection information for a central repository, right-click the central repository in the **Central Repository Datastores** box and select **Delete**.

   After confirming your selection, Data Integrator deletes the connection information from this local repository. You can no longer connect to that central repository from this local repository.

   NOTE: You are not deleting the central repository; you are only deleting the connection information between your local repository and this central repository.

4. To make another repository the active central repository, right-click the central repository in the **Central Repository Datastores** box and select **Activate**.

5. To disconnect from the currently active central repository, right-click the central repository in the **Central Repository Datastores** box and select **Deactivate**.

7 Working in a multi-user environment

To obtain optimal results from development in a multi-user environment, Business Objects recommends certain processes, such as checking in and checking out objects that you change, and establishing a set of conventions that your team follows, such as labeling objects.

You will complete several tasks on a regular basis:

- Filtering
- Adding objects to the central repository
- Checking out objects
- Undoing check out
- Checking in objects
- Labeling objects
- Getting objects
- Comparing objects
- Viewing object history
- Deleting objects
- Using aliases

Also in this chapter, find information on Project support as well as multi-user specific Application phase management.

Filtering

Data Integrator allows you to customize by filtering (selectively changing) environment-specific information in object definitions. Application objects can contain repository-specific information. For example, datastores and database tables might refer to a particular database connection unique to a user or a phase of development. When multiple users work on an application, they can change repository-specific information.

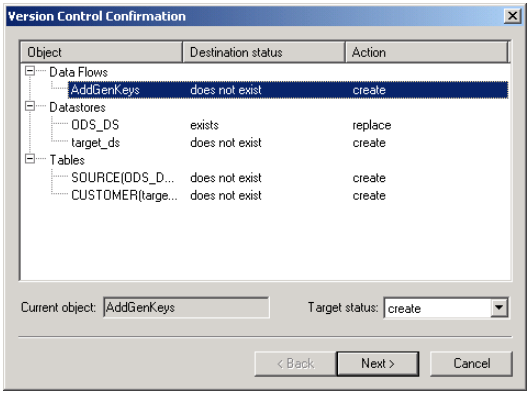
Specifically, filtering allows you to:

- Change datastore and database connection information
- Change the root directory for files associated with a particular file format
- Select or clear specific dependent objects

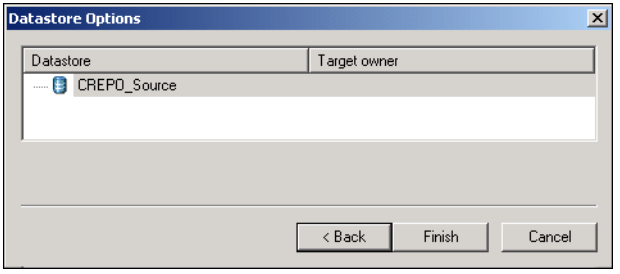
The filtering process is available when adding, checking in, checking out, or getting latest objects in a central repository.

When you select any command that uses the filtering option, Data Integrator presents the following two windows:

1. The Version Control Confirmation window shows your selected object and any dependent objects. You can exclude objects by selecting the object and changing the **Target status** (lower right) from **create** or **replace** to **exclude**.



2. The Datastore Options window shows any datastores used by the object. This window only opens if the objects that you are adding, checking in, or checking out include a datastore.



Adding objects to the central repository

After creating a central repository, connecting it to the local repository, and activating the central repository, you can add objects from the local repository to the central repository. Remember that you do all design work—the creation of jobs, work flows, and data flows—in a local repository. Therefore, you use a local repository for the initial creation of any objects in an application. After the initial creation of an object, you add it to the central repository. Once in the central repository, the object is subject to version control and can be shared among users.

You can add objects to the central repository at any point. Of course, you cannot add an object that already exists in the central repository.

You can add a single object to the central repository, or you can add an object with all of its dependents to the central repository. When you add a single object, such as a data flow, you add only that object. No dependent objects are added.

➢ To add a single object to the central repository

1. Open the local object library.
2. Right-click the object and select **Add to Central Repository > Object**.
3. The Comments window opens. Enter any comments in the **Comments** field, and click **OK**.

   Data Integrator adds the object to the active central repository.

   NOTE: The **Add to Central Repository** command is not available if the object already exists in the central repository.

➢ To add an object and its dependent objects to the central repository

1. Open the local object library.

2. Right-click the object and select either **Add to Central Repository** > **Object and dependents** or **Add to Central Repository** > **With filtering** (if filtering is required).
3. The Comments window opens. Enter any comments in the **Comments** field, and click **OK**.
4. If you selected **With filtering**, complete the filtering windows, as described in "Filtering" on page 58.
5. Click **Finish** to add the selected objects.

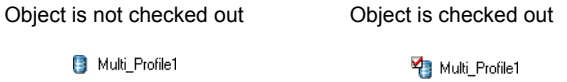
Alternatively, you can select the object and drag it to the central object library to add the object and its dependents to the central repository. The filtering windows are displayed.

NOTE: The **Add to Central Repository** command is not available if the object already exists in the central repository. However, the **Add to Central Repository** command is available if the object's dependents already exist in the central repository but the object itself does not.

Checking out objects

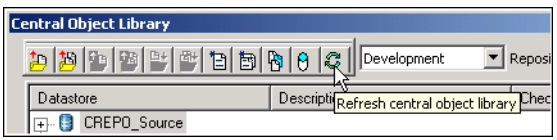
When you might change any of the objects in an application, you should check out the objects that you expect to change. When you check out an object, you make that object unavailable to other users—other users can view the object but cannot make changes to the object. Checking out an object ensures that two users do not make conflicting changes to the object simultaneously.

Data Integrator changes the object icons in both the local and central object libraries to indicate that the object is checked out.



When an object is checked out, your central object library shows you the local repository that has checked out the object. Based on the repository name, you can determine which user is working with that object.

To see periodic changes, refresh the central object library by clicking on the refresh central object library icon in the toolbar of the central object library.



Choose a check-out command based on what you will do to an object:

- Check out single objects or objects with dependents
- Check out single objects or objects with dependents without replacement
- Check out objects with filtering

Check out single objects or objects with dependents

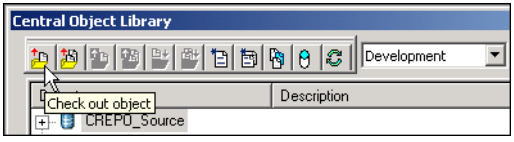
Dependents are objects used by another object—for example, data flows that are called from within a work flow. You can check out a single object or an object with all of its dependents (as calculated in the central repository). For example, you can simply check out a work flow. In that case, you can change that work flow, such as adding a new script to the work flow; however, you cannot change dependent objects in the work flow, such as data flows, and retain the changes in the central repository. Changes to dependent objects will only be retained in the local repository. Alternatively, you can check out the work flow with all of its dependents. In that case, you can make changes to the work flow or any of its dependents and retain the changes in both central and local repositories.

Generally, it is safest to check out an object with all dependents. When you do this, you prevent others from accidentally changing dependent objects.

➢ To check out a single object

1. Open the central object library.
2. Right-click the object you want to check out.
3. Choose **Check Out** > **Object**.

   Alternatively, you can select the object in the central object library, and click the **Check Out Object** button on the top of the central object library.

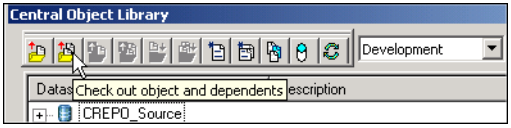


Data Integrator copies the most recent version of the selected object from the central repository to your local repository, then marks the object as checked out.

➢ To check out an object and its dependent objects

1. Open the central object library.
2. Right-click the object you want to check out.
3. Choose **Check Out > Object and dependents.**

   Alternatively, you can select the object in the central object library, and click the **Check out object and dependents** button on the top of the central object library.



   Data Integrator copies the most recent version of the selected object and all of its dependent objects from the central repository and marks these objects as checked out.

   If a dependent object is checked out by you or another user, then Data Integrator alerts you with a Check Out Alert window, asking to get the latest version of the checked out object. See "Getting objects" on page 74.

Check out single objects or objects with dependents without replacement

When you check out an object, you can replace the object in your local repository with the latest version from the central repository, or you can leave the current version in your local repository intact.

When you check out an object, Data Integrator copies the object definition from the central repository and replaces any existing definitions for that object in your local repository.

You can check out objects without replacing the objects in your local repository. For example, suppose you are working in your local repository and you make a change to an object that is not checked out. If you determine that the change improves the design or performance of your application, you will want to include that change in the central repository.

To do this, check out the object without replacing the object in your local repository—the object that you have already improved with a change. Then, check the changed object back into the central repository.

NOTE: Use caution when checking out objects without replacing the version in your local repository. When you do not replace the version in your local repository, you can lose changes that others have incorporated into those objects.

➢ To check out an object or an object and its dependent objects without replacement

1. Open the central object library.
2. Right-click the object you want to check out and choose **Check Out > Object without replacement** to check out the single object or choose **Check Out > Object and dependents without replacement** to check out the object and all of its dependent objects.

   Data Integrator marks all appropriate objects as checked out—in both the object library and in the workspace—but does not copy any objects from the central repository to the local repository.

Check out objects with filtering

When you check out an object with filtering, the object and all its dependents are checked out.

NOTE: When you check out objects with filtering, you always replace local versions with the filtered objects from the central repository.

➢ To check out and object and its dependent objects with filtering

1. Open the central object library.
2. Right-click the object you want to check out and choose **Check Out > With filtering**.
3. Complete the filtering windows, as described in "Filtering" on page 58.
4. Click **Finish** to check out the selected objects.

Undoing check out

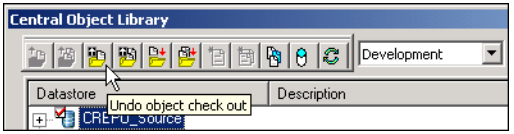
Occasionally, you may decide that you did not need to check out an object because you made no changes. Or, you may decide that the changes you made to a checked-out object are not useful and you prefer to leave the master copy of the object as is. In these cases, you can undo the check out.

When you undo a check out, you leave both the object in your local repository as well as the object in the central repository as is; no changes are made and no additional version is saved in the central repository. Only the object status changes from checked out to available.

After you undo a check out, other users can check out and make changes to the object.

➢ To undo single object check out

1. Open the central object library.
2. Select a checked-out object.
3. Click the **Undo object check out** button.

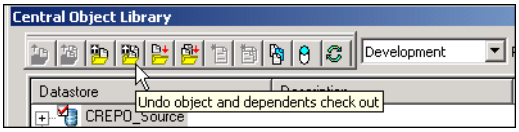


Data Integrator removes the check-out symbol and makes no changes to the object in the central repository. Any checked-out dependent objects remain checked out.

Alternatively, you can right-click the object and select **Undo Check Out** > **Object**.

➢ To undo check out of an object and its dependents

1. Open the central object library.
2. Select the checked-out object that is the highest level for which you want to undo the check out.
3. Click the **Undo object and dependents check out** button.



Data Integrator removes the check-out symbols for the object and any dependent objects that are also checked out. No changes are made to these objects in the central repository.

Alternatively, you can right-click the object in the central object library and select **Undo Check Out > Object and dependents**.

Checking in objects

After you finish making changes to checked out objects, you must check them back into the central repository. Checking in objects creates a new version in the central repository, and allows others to get the changes that you have made. Checking in objects also preserves a copy of the changes for revision control purposes. Later, you can get a particular version of a checked in object and compare it to subsequent changes or even revert to the previous version.

Check in an object when you are done making changes, when others need the object that contains your changes, or when you want to preserve a copy of the object in its present state.

Choose a check-in command based on what you will do to an object:

- Checking in single objects, objects with dependents
- Checking in an object with filtering

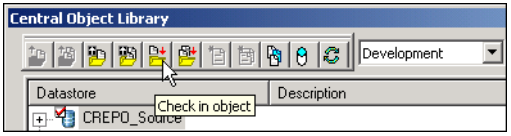
Checking in single objects, objects with dependents

Just as you can check out a single object or an object with all dependent objects, you can check in a single object or an object with all checked-out dependent objects (as calculated in the local repository).

➢ To check in a single object

1. Open the central object library.
2. Select the object you want to check in.

3. Click **Check in object** button at the top of the central object library.



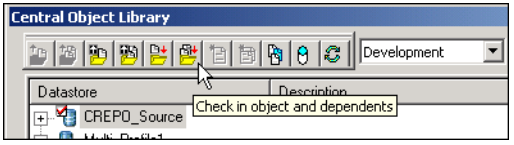
Alternatively, you can right-click the object in the central object library and select **Check In > Object**.

4. A Check In window opens with a **Comment** box, in which you can enter comments. After entering any comments, click **OK**.

Data Integrator copies the object from your local repository to the central repository, and removes the check-out mark.

➢ To check in an object and its dependent objects

1. Open the central object library.
2. Select the highest level object you want to check in.
3. Click **Check in object and dependents** button at the top of the central object library.



Alternatively, you can right-click the object in the central object library and select **Check In > Object and dependents**.

4. A Check In window opens with a **Comment** box, in which you can enter comments. After entering any comments, click **OK**.

Data Integrator copies the selected object and all of its dependent objects from your repository to the central repository and removes the check-out mark.

Checking in an object with filtering

Just as you could check out objects with filtering, you can check in objects with filtering. When you check in an object with filtering, the object and all its dependent objects are checked in.

➢ To check in an object with filtering

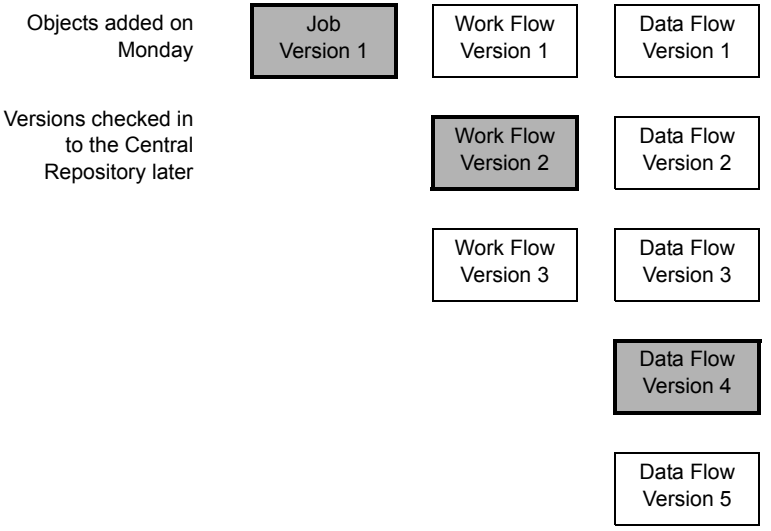
1. Open the central object library.
2. Right-click the object you want to check out and choose **Check In** > **With filtering**.
3. A Check In window opens with a **Comment** box, in which you can enter comments. After entering any comments, click **OK**.

   Data Integrator warns you that you are about to create a new version of the object in the central repository.
4. Click **Yes** to continue with the check in.
5. Complete the filtering windows, as described in "Filtering" on page 58.
6. Click **Finish** to check in the selected objects.

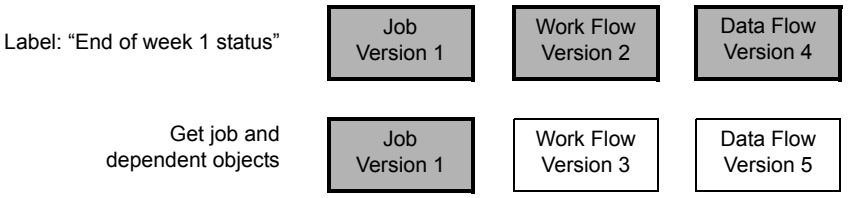
Labeling objects

To help organize and track the status of objects in your application, you can label objects. When an object is labeled, the object and all its dependent objects are labeled. A label not only describes an object, but also allows you to maintain relationships between various versions of objects.

For example, suppose a job is added to the central repository where user 1 works on a work flow in the job, and user 2 works on a data flow in the same job. At the end of the week, after user 1 checks in two versions of the work flow and user 2 checks in four versions of the data flow into the central repository, the job is labeled "End of week 1 status." This label contains version 1 of the job, version 2 of the work flow, and version 4 of the data flow. Later, user 1 and user 2 continue to change their respective work flow and data flow.



At some later point, if you want to get the job with the version of the data flow with this label, getting the job by its label (see "Getting objects" on page 74) accomplishes this, whereas checking out the job and its dependents does not.

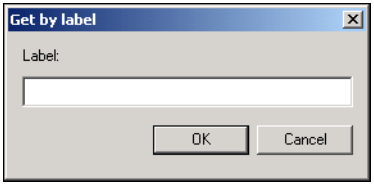


The label "End of week 1 status" serves the purpose of collecting the versions of the work flow and data flow that were checked in at the end of the week. Without this label, you would have to get a particular version of each object in order to reassemble the collection of objects labeled "End of week 1 status."

➢ To label an object and its dependents

1. Open the central object library.
2. Right-click the object you want to label and choose **Label**.

   Data Integrator opens the Get by Label window.



3. In the **Label** box, enter text that describes the current status of the object, then click **OK**.

   Data Integrator inserts this label in the history of the object and its dependents. To view the labels associated with a particular object, see "Viewing object history" on page 79.

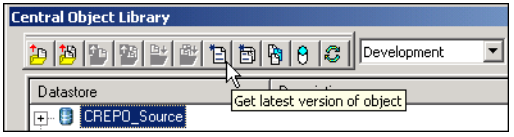
Getting objects

To make sure that your repository is up-to-date, you "get" objects. When you get an object, you copy the latest version of that object in the central object library and copy it into your local repository, replacing the version in your local repository. When you get an object, you do not check out the object. The object remains free for others to check out and change.

You can get an object with or without dependent objects and filtering. For information about getting earlier versions of objects or objects with particular labels, see "Viewing object history" on page 79.

➢ To get a single object

1. Open the central object library.
2. Select the object you want to get.
3. Click **Get latest version of object** at the top of the central object library.



Alternatively, right-click the object in the central object library and select **Get Latest Version** > **Object**.

Data Integrator copies the most recent version of the object in the central repository to your local repository.

➢ To get an object and its dependent objects

1. Open the central object library.
2. Select the highest level object you want to get.

3. Click **Get latest version of objects and dependents** at the top of the central object library.

   Alternatively, right-click the object in the central object library and select **Get Latest Version** > **Object and dependents**.

   Data Integrator copies the most recent version of the selected object and all dependent objects from the central repository to your local repository.

➢ To get an object and its dependent objects with filtering

1. Open the central object library.
2. Select the highest level object you want to get.
3. Right-click the object in the central object library and select **Get Latest Version** > **With filtering**.
4. Complete the filtering windows, as described in "Filtering" on page 58.
5. Click **Finish** to get the selected objects.

➢ To get a previous version of an object

See "To get a previous version of an object" on page 80.

➢ To get an object with a particular label

See "To get an object with a particular label" on page 80.

Comparing objects

Data Integrator allows you to compare two objects from local and central repositories to determine the differences between those objects.

NOTE: When two objects are compared, only the internal structures of the objects are compared. The names of the objects and WHERE clauses are not considered during the comparison.

➢ To compare two objects

1. Select **Tools** > **Diff View**.

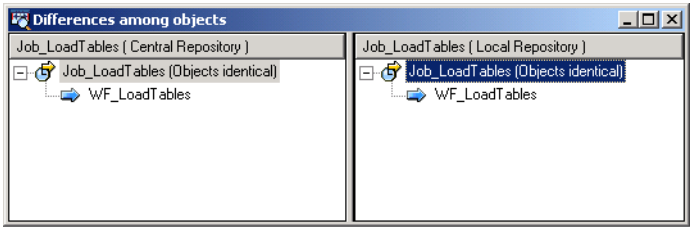
   Data Integrator opens the Differences Among Objects window.

2. Drag the first object you want to compare into one of the window panes.

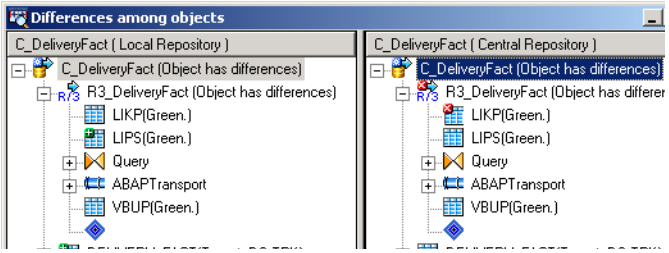
3. Drag the second object you want to compare into the other window pane.

4. Right-click in either pane and select **Show Differences**.

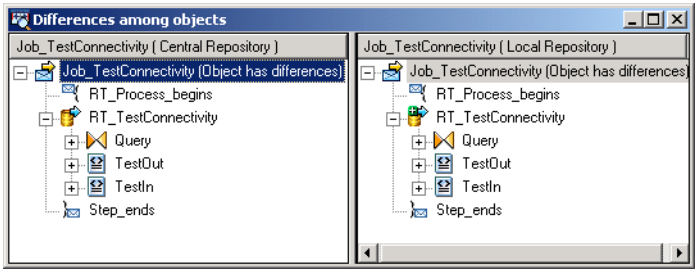
   Data Integrator shows the differences between the two objects in the window panes. If there are no differences, Data Integrator labels the highest level object "Objects identical."



If differences exist between objects, Data Integrator indicates this with the message, "Object has differences!" Expand the object tree until you find the changed object.



You can resize the window to see hidden contents.



Data Integrator uses symbols to indicate the change type.

Symbol	Indicates
	Object was changed. Open up the object tree to see the specific change.
	Object was added, but does not appear in the compared flow structure.
	Object was deleted from one flow structure, but is appearing in the compared flow structure.

5. Right-click an object and choose **Open** to view an object's definition.

   Data Integrator opens the workspace diagram that shows the selected object.

6. Right-click anywhere in either window pane and select **Clear All** to clear the current objects and select new ones.

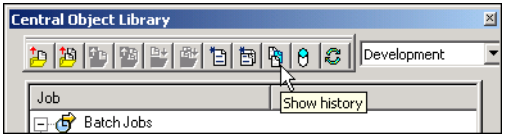
Viewing object history

The central repository retains a history of all changes made to objects in the central repository. Use this history to help manage and control development of your application. For example, you can use the central repository:

- To examine the history of an object
- To get a previous version of an object
- To get an object with a particular label

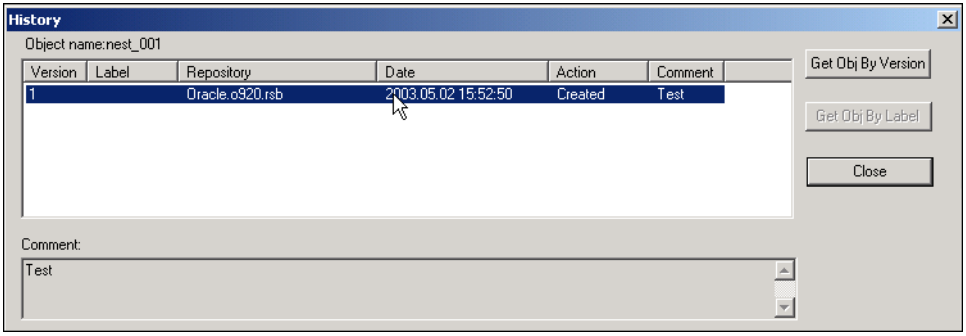
➢ To examine the history of an object

1. Open the central object library.
2. Select an object.
3. Click the **Show History** button at the top of the central object library.



Alternatively, you can right-click the object in the central object library and select **Show History**.

Data Integrator opens the History window.



This window shows several pieces of information about each revision of the object.

Column	Description
Version	The object revision number. Each time a user saves the object, Data Integrator creates a new version.
Label	Text that a user enters to describe the status of the object at a given point. See "Labeling objects" on page 72 for more information about object labeling.
Repository	Information about the local repository from which Data Integrator saved this version of the object. Other information includes: • user name • database connection name • type of database
Date	The date and time Data Integrator saved this version of the object.
Action	The type of change a user made to the object. This table records actions such as: Checked in — User checked in object
Comment	Comments a user enters when adding an object or checking it into a central repository.

➢ To get a previous version of an object

1. Select an object.
2. Click the **Show History** button at the top of the central object library.
3. Click the version of the object you want.
4. Click the **Get Obj By Version** button.

NOTE: When you get a previous version of an object, you only get the object but not its dependent objects.

➢ To get an object with a particular label

1. Select an object.

2. Click the **Show History** button at the top of the central object library.
3. Click the version of the object with the particular label you want.
4. Click the **Get By Label** button.

Deleting objects

You can delete objects from either the central repository or a local repository. To delete an object from the central repository, right-click the object in the central object library and select **Delete**. To delete an object from the local repository, right-click on the object in the object library and select **Delete**.

When you delete an object from a local repository, you do not automatically delete that object from the active central repository. In fact, you can get the object from the central repository to re-insert it.

Similarly, when you delete an object from a central repository, you do not automatically delete the object from connected local repositories. Until you delete the object from the local repository, you can add the object back to the central repository.

When you delete objects from the central repository, you only delete the selected object and all versions of the selected object; you do not delete any dependent objects.

To delete all versions except the latest version of an object from the central repository, use the compact repository. Select **Project** > **Compact Repository** from the menu bar.

Using aliases

You can specify an alias for a specified table owner and datastore combination. In your local repository, Data Integrator displays your alias as the table owner name.

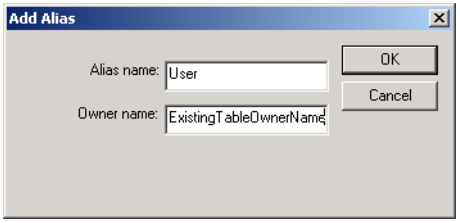
Using an alias can be beneficial when you are working in a multi-user environment. With an alias, you can move a table between different Local Repositories using the Central Repository. Further, Data Integrator can load to a table in another owner's schema without modifying the owner name.

➢ To create an alias

1. In the object library, click the Datastore tab.
2. Right-click a datastore and select **Properties**.

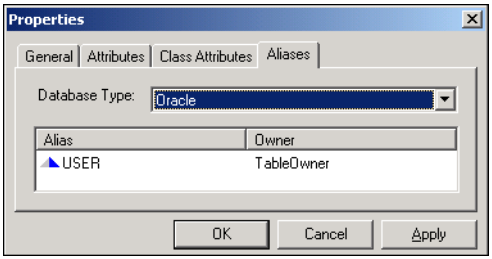
   The Properties window opens.
3. In the Properties window, select the Aliases tab, right-click, and select **Add**.

   The Add Alias window opens.



4. In the Add Alias window, enter:
   - **Alias name**: The owner name displayed when viewing objects.

- **Owner name**: The owner name used at runtime.



5. Click **OK**.

   The alias information appears in the Properties window.

6. Click **Apply** to save the alias information, keeping the Properties window open for more editing. Or, click **OK** to save the alias information and close the Properties window.

After you save alias information, all tables or functions that you import into the repository or check out from the repository will be identified in the object library by table or function name, datastore, and alias name.

Example: You enter “development” for **Alias name** and “user_1” for **Owner name**.

After establishing this alias, all tables owned by 'user_1' that you import into the datastore will show the owner name, 'user_1' in the local object library. If you use one of these objects as a target table in a data flow, the owner name displays the alias 'development'. At run time, Data Integrator replaces the alias (development) with the associated owner name (user_1). If you check this data flow with its dependent table into a Central Repository, the owner for the target table would be the alias, 'development'.

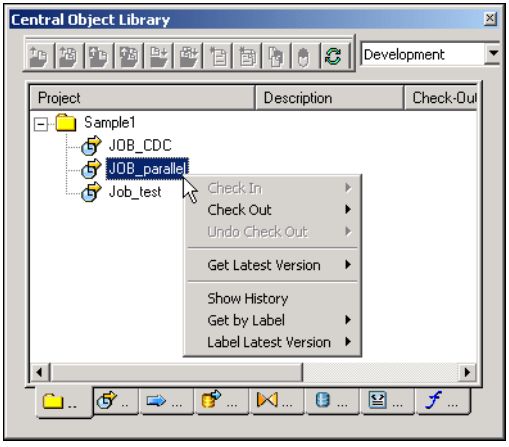
Another user (user_2) could use the same data flow and without modifying it, have it load to a table owned by another user. After user_2 checks out or gets the latest version of the data flow they should delete the existing alias for dev then recreate the alias associating it with the proper owner. If the same table existed in user_2's schema, and that is where you want to load to or pull data from, the alias of development should be associated with user_2. At load time, Data Integrator substitutes the owner alias of 'development' with 'user_2'.

NOTE: The following guidelines apply when you use the alias feature:

- The table must exist in the schema of the owner before job execution unless the target is a template table.
- Specifying an alias will not convert the owner name of existing objects to the alias name. If you want to use the alias, you must re-import the objects.
- If you export a datastore using aliases either directly to another repository or to a file, the datastore will retain the alias owner names along with the alias definition.
- For LiveLoad datastores, you must specify a table owner for both the master and the mirror databases. For more information, see "To create a LiveLoad alias for your LiveLoad datastore" on page 51 of the *Data Integrator LiveLoad User's Guide*.

Project support

Projects group jobs. The **Projects** tab in the central object library displays the projects under source control. You can expand each project to display a list of the jobs included in that project. All of the commands available for jobs in the **Jobs** tab of the central repository are available for each job in this tab except **Delete**. Also, to get versions of jobs besides the latest version, you need to identify the label of the individual job you want to get.



With project support, you can:

- Add projects to the central repository
- Get the latest version of a project
- Update a project

If you create a project in your local repository, you can add the project and its dependent objects to the central repository. When you add a project, only new objects are added. Previously existing dependent objects are not overwritten. For example, suppose you add a project that contains two jobs, JobA and JobB, and JobA is in the central repository and JobB is not. Then the project and JobB are added to the central repository and JobA is not overwritten.

NOTE: You cannot check out, check in, label a project, or get a project by version or label.

➢ To add a project to the central repository

You can get the latest version of a project in the central repository. When you get the latest version of a project, the project definition and its associated jobs are copied to your local repository. Getting the latest version of a project is equivalent to getting the latest version of all the jobs and their dependents in the project.

For example, suppose a developer, working in their local repository, creates the project Proj_Sales to group all jobs extracting data from a sales system. Proj_Sales contains the job Job_Sales_Monthly for monthly extractions, and the job Job_Sales_Weekly for weekly extractions. The developer adds these objects to the central repository so that others can get Proj_Sales. Later, suppose a manager checks out Job_Sales_Monthly from the central repository, modifies it, and checks it back in to the central repository, and the developer checks out Job_Sales_Weekly from the central repository, modifies it, and checks it back in to the central repository. Then the developer and the manager can get the latest version of Proj_Sales to keep track of any changes to the jobs made by each other as well as others.

➢ To get the latest version of a project

1. Right-click the project in the central object library, and select **Get Latest Version** > **Object and dependents**.

NOTE: The drag and drop method and the **Get latest version of object with dependents** button at the top of the central object library are not available when getting the latest version of a project. You must right-click the project and select **Get Latest Version** > **Object and dependents**.

You can update a project definition in the central repository. If you are working in your local repository and add a new job to a project, then you can update the project definition in the central repository so it includes the new job. Updating updates the project definition in the central repository, adds the new jobs to the central repository, but does not overwrite existing jobs in the central repository.

Continuing with the previous example, suppose the developer, working in their local repository, creates the job Job_Sales_Daily, for daily extractions, and adds it to Proj_Sales. Since the addition of Proj_Sales to the central repository, others have checked out Job_Sales_Monthly and Job_Sales_Weekly from the central repository, modified them, and checked them back in to the central repository. Now, the developer updates Proj_Sales. In the update process, the versions of Job_Sales_Monthly and Job_Sales_Weekly in the developer's local repository will not overwrite the versions of these jobs in the central repository. Only the definition of Proj_Sales will be updated, and Job_Sales_Daily will be added to the central repository.

➢ To update a project

Right-click the project in your local repository, and select **Update**.

Application phase management

Typically, applications pass through different phases on the way from development to production. For example, an application might pass through three phases:

- Developers creating an application
- Testers validating the application
- Administrators running the application

A single central repository can support your application through these phases. Use job labeling and projects to maintain application components independently for the phases. In addition, filtering of database owners and file locations allows you to configure the application to run in each local environment.

In some situations, you may require more than one central repository for application phase management. If you choose to support multiple central repositories, use a single local repository as a staging location for the transition.

8 Migrating multi-user jobs

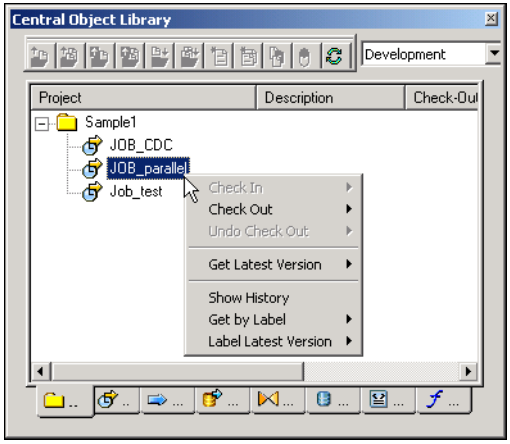
Migration is slightly more complex in a multi-user environment.

- Project support
- Application phase management
- Copying contents between central repositories

Project support

Projects group jobs. The **Projects** tab in the central object library displays projects under source control. You can expand a project to show a list of the jobs included in that project.

All commands available in the **Jobs** tab of the central repository except **Delete** are available for each job displayed on the tab. Also, to get a previous versions of a job, you must identify the label of that job.



With project support, you can:

- Add projects to the central repository
- Get the latest version of a project
- Update a project

You can add to the central repository any project (and its dependent objects) you created in your local repository. When you add a project, only new objects appear. Data Integrator does not overwrite previously existing dependent objects.

For example, suppose you are adding Project Alphabet to your central repository. It contains two jobs: JobA and JobB. However, JobA already exists in the central repository. So, when you add the project, only Project Alphabet and JobB actually get added to the central repository. Since JobA already exists, there is no need for Data Integrator to override it.

NOTE: You cannot check out, check in, or label a project. You cannot get a project by version or label.

➢ To add a project to the central repository

You can get the latest version of a project in the central repository. When you get the latest version of a project, Data Integrator copies the project definition and its associated jobs to your local repository. Getting the latest version of a project is equivalent to getting the latest version of all the jobs and their dependents in the project.

For example, suppose a developer, working in their local repository, creates the project Proj_Sales to group all jobs extracting data from a sales system. Proj_Sales contains the job Job_Sales_Monthly for monthly extractions, and the job Job_Sales_Weekly for weekly extractions. The developer adds these objects to the central repository so that others can get Proj_Sales. Later, suppose a manager checks out Job_Sales_Monthly from the central repository, modifies it, and checks it back in to the central repository, and the developer checks out Job_Sales_Weekly from the central repository, modifies it, and checks it back in to the central repository. Then the developer and the manager can get the latest version of Proj_Sales to keep track of any changes to the jobs made by each other as well as others.

➢ To get the latest version of a project

1. Right-click the project in the central object library, and select **Get Latest Version** > **Object and dependents**.

NOTE: The drag and drop method and the **Get latest version of object with dependents** button at the top of the central object library are not available when getting the latest version of a project. You must right-click the project and select **Get Latest Version** > **Object and dependents**.

You can update a project definition in the central repository. If you are working in your local repository and add a new job to a project, then you can update the project definition in the central repository so it includes the new job. Updating updates the project definition in the central repository, adds the new jobs to the central repository, but does not overwrite existing jobs in the central repository.

Continuing with the previous example, suppose the developer, working in their local repository, creates the job Job_Sales_Daily, for daily extractions, and adds it to Proj_Sales. Since the addition of Proj_Sales to the central repository, others have checked out Job_Sales_Monthly and Job_Sales_Weekly from the central repository, modified them, and checked them back in to the central repository. Now, the developer updates Proj_Sales. In the update process, the versions of Job_Sales_Monthly and Job_Sales_Weekly in the developer's local repository will not overwrite the versions of these jobs in the central repository. Only the definition of Proj_Sales will be updated, and Job_Sales_Daily will be added to the central repository.

➢ To update a project

Right-click the project in your local repository, and select **Update**.

Application phase management

Typically, applications pass through different phases on the way from development to production. For example, an application might pass through three phases:

- Developers creating an application
- Testers validating the application
- Administrators running the application

A single central repository can support your application through these phases. Use job labeling and projects to maintain application components independently for the phases. In addition, filtering of database owners and file locations allows you to configure the application to run in each local environment.

In some situations, you may require more than one central repository for application phase management. If you choose to support multiple central repositories, use a single local repository as a staging location for the transition.

Copying contents between central repositories

You cannot *directly* copy the contents of one central repository to another central repository. Rather, you must use your *local* repository as an intermediate repository.

➢ To copy the contents of one central repository to another central repository

1. Activate the central repository whose contents you will copy. See "Activating a central repository" on page 52,
2. Get the latest version of the contents of this active central repository by getting the latest version of the objects in this central repository. See "Getting objects" on page 74.
3. Activate the central repository into which you want to copy the contents. See "Activating a central repository" on page 52.
4. The first time you copy the contents, add the objects from your local repository into this central repository. See "Adding objects to the central repository" on page 60.

   However, if you must re-copy the contents of one central repository into another (for example, during your testing phase some part of a job was reassigned to the development phase for redesign), the process is slightly more complex:

   a. First check out specific objects without replacement from the second central repository. See "Checking out objects" on page 62.
   b. From your local repository, get the latest version of the objects from the first (for example, development) central repository.
   c. Then, instead of adding, check in the updated objects from your local repository to the second (for example, test) central repository. See"Checking in objects" on page 69.

Index

Business Objects, Americas
3030 Orchard Parkway
San Jose, CA 95134
USA

info@businessobjects.com
www.businessobjects.com

Data Integrator™

LiveLoad User's Guide

Version 6.1

Document Number DI-61-0104-001

June 16, 2003

Contents

1 Welcome

Welcome to the *Data Integrator Liveload User's Guide*.

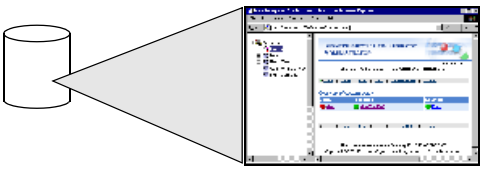
This guide contains information about Data Integrator's control system for maintaining a single data warehouse that is accessible 24 hours per day, 7 days per week.

This chapter contains the following sections:

How LiveLoad works

LiveLoad maintains two identical databases—a *master* and a *mirror*—through a single access point. One of the databases is always available for queries (live) while the other is being loaded with the most recent data (load).

Without LiveLoad

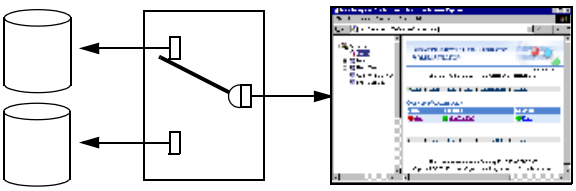


Data warehouse

Query Application

A single data warehouse database is subject to reduced performance and temporary inconsistency during data loading.

With LiveLoad



LiveLoad Database

LiveLoad Connector

Query Application

LiveLoad performs loading only on the off-line database. The second database is always ready for querying.

In addition to providing a single point of access for queries to the data warehouse, LiveLoad also provides a single point of access for loading data using a Data Integrator job.

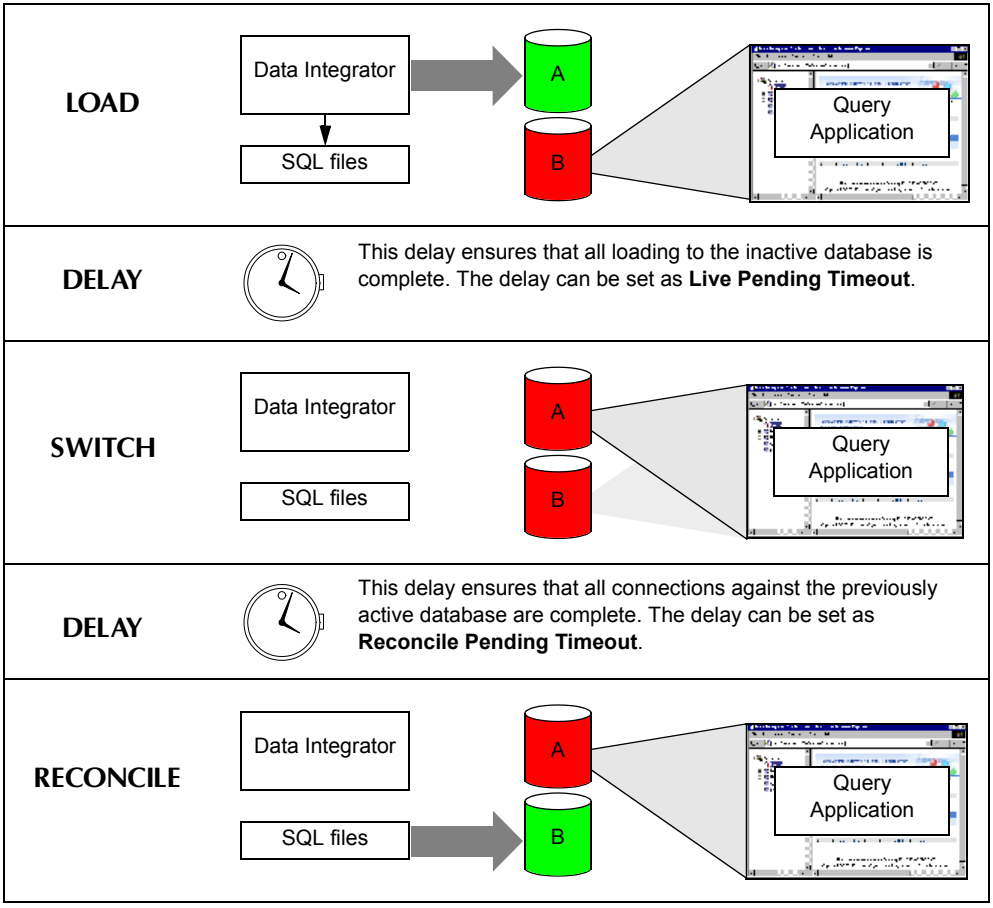
LiveLoad operation cycle

To make sure your query application accesses the latest data available, you must update the LiveLoad database from its source regularly.

The cycle of updating and switching the LiveLoad database proceeds through these steps:

1. Load — Data Integrator loads database A while it applies queries to database B. At the same time, LiveLoad produces files containing the SQL commands required to produce the same update later in database B.
2. Delay — Some amount of delay at this point ensures that all operations applied to database A complete successfully.
3. Switch — The LiveLoad Connector directs all new connections to database A. The switch can be an automatic or manual process.
4. Delay — Some amount of delay ensures that all existing connections to database B complete.
5. Reconcile — LiveLoad applies SQL commands to database B. Now the two databases have identical content.
6. Load — The cycle continues with Data Integrator loading database B.

The following diagram illustrates this cycle:


LOAD
Data Integrator
SQL files
A
B
Query Application
DELAY
This delay ensures that all loading to the inactive database is complete. The delay can be set as **Live Pending Timeout**.
SWITCH
Data Integrator
SQL files
A
B
Query Application
DELAY
This delay ensures that all connections against the previously active database are complete. The delay can be set as **Reconcile Pending Timeout**.
RECONCILE
Data Integrator
SQL files
A
B
Query Application

Querying the LiveLoad databases

To use Liveload, two installations must take place. First, when you install Data Integrator Designer, a license-controlled feature called **LiveLoad** must be included in your license. This feature allows you to create datastores enabled for target LiveLoad databases, and it also adds LiveLoad graphical user interface options to the Designer.

The second installation must occur on the computer you will use to access LiveLoad-controlled databases. This can be on a computer with other Data Integrator components or on another computer. This computer is called the *query application* computer.

- To install a COM connector: Use the Message Client setup.exe program to install the **LiveLoad COM Connector**.
- To install a JDBC connector: Manually copy and configure files from the Data Integrator CD or the Data Integrator Administrator's computer to the query application computer.

See more detailed instructions for installing LiveLoad in the next chapter.

When the query application is ready to connect to a database and apply a query, it calls the locally installed LiveLoad Connector (COM or JDBC) to determine which database to query (which database is "live") and the connection information required to connect to that database. The LiveLoad Connector returns a key value used to return the connection information.

Calls to the LiveLoad Connector that your application can make are described in "LiveLoad Connector" on page 30.

When the LiveLoad database switches, there is a point where it applies one query to one database and the next query to the other database. A delay occurs in this LiveLoad cycle to ensure that the first query is finishes before operations continue on that database. You configure the delay to correspond to the amount of time required by the queries your application applies to the LiveLoad database.

2 Installing LiveLoad Connectors

This chapter provides a high-level overview of the steps required to implement LiveLoad.

LiveLoad can process data using either a COM API or a Java API connector. Both the Java and COM connectors offer identical functionality.

The COM connector runs on a Windows platform. It can direct queries to a data warehouse on the following databases:

- Oracle
- DB2
- MS SQL Server
- Informix
- ODBC-compatible databases

See "System requirements" on page 46 of the *Data Integrator Getting Started Guide* for version information. The Java connector will run on Windows or UNIX, but it will load only to Oracle 8.0 and later.

This chapter includes the following sections:

Setting up the LiveLoad environment

For both the COM and Java API connectors, you first must create master and mirror databases using the following procedure.

➢ To set up the LiveLoad environment

1. Create master and mirror databases populated with identical tables.

   For development, you can create the database schemas using template tables from inside Data Integrator.

   NOTE: The tables in the master and mirror databases must have the same owner and the same names. However, you can set Oracle databases with different owner names using aliases. See "Configuring LiveLoad for a single Oracle instance" on page 50.

2. Install the Data Integrator Designer.

   LiveLoad is a licensed-controlled feature. If you purchase LiveLoad with Data Integrator, Data Integrator can use LiveLoad when you install the Designer. See Chapter 6, "Installing Data Integrator on Windows Systems," in the *Data Integrator Getting Started Guide* for step-by-step installation instructions.

3. Define your target datastore by selecting the **LiveLoad** option on the Datastore Editor and specifying connection information for both the master and mirror databases.

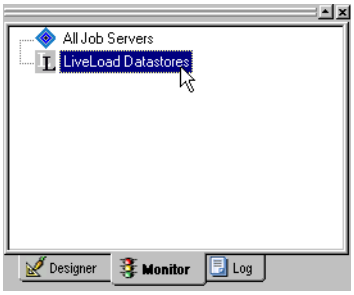
   See "Datastore object" on page 18.

4. Include target tables from the LiveLoad datastore in the data flows that fill your target.

5. Include a mechanism to switch between databases—master or mirror—depending upon which is available to accept queries.

   Common database switching methods include:

   - Add a trigger at the end of a job to switch at the completion of all of the relevant data flows.
   - Create a stand-alone job that performs the switch; then schedule the job at the best times for the switch.
   - Manually switch between **Live** and **Load** databases from the Designer using the display in the **Monitor** tab.



6. Run the initial load job to populate the data warehouse in the master database.
7. Use your database replication tools to populate the mirror database from the master.

You must install a LiveLoad connector on the computer that will be sending queries to the data warehouse. This computer is referred to as the *query application computer*. This computer may be running any application external to Data Integrator. The purpose of the LiveLoad connector is to direct queries to the *live* data warehouse database.

The remainder of this chapter covers installation and configuration of the LiveLoad connectors.

Installing and configuring the COM API

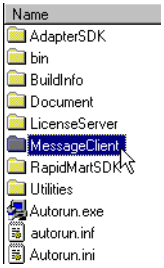
Use the following procedure to use the LiveLoad connector with a COM API. See "Installing and configuring the Java API" on page 13 for Java API procedures.

➢ To install and configure the LiveLoad COM connector

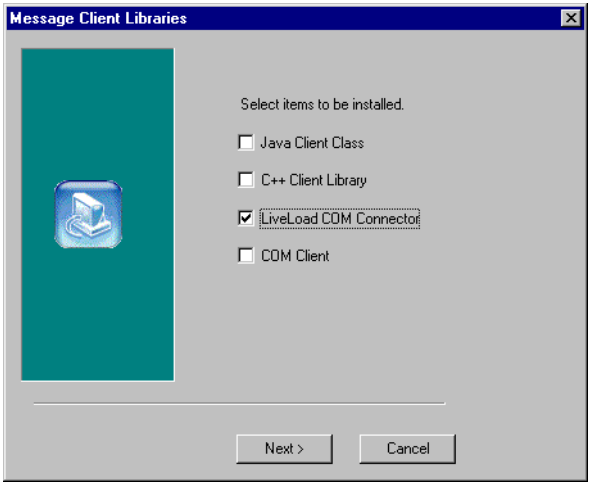
1. On the computer running the query application (a Web application for example), insert the Data Integrator installation CD.

   The main Data Integrator installation program starts automatically. When it does, click **Exit**.

2. From the MessageClient subdirectory on the CD, double-click Setup.exe.



3. Select **LiveLoad COM Connector** in the Message Client Libraries installer.



4. Click **Next**.
5. Select a Destination.

   If you are installing LiveLoad COM Connector on a computer with other Data Integrator components, choose the same installation location (the location specified by the *LINK_DIR* environment variable). For example, `C:\Data Integrator 6.5`.

   If you are installing LiveLoad COM Connector on a computer without other Data Integrator components, use the same destination path as on a Data Integrator computer. For example, `C:\Data Integrator 6.5`.
6. Click **Next**.

   `Crypto.dll` and `LLConAtl.dll` install into the *LINK_DIR*`/Bin` directory.
7. Click **Finish**.

At this point the .dlls should be registered. See "LiveLoad Connector" on page 30 to learn how to use these .dlls to create a script to act as your LiveLoad COM connector.

8. Move the LiveLoad configuration file from the Job Server's computer (created when you define a LiveLoad datastore) to your query application computer.

   Move the file from this location on the Job Server's computer:

   `LINK_DIR\LLRoot\datastore_name\datastore.cfg`

   To this location on the query application computer:

   `LINK_DIR\LLRoot\datastore_name`

   where `LINK_DIR` is the location where Data Integrator is installed and `datastore_name` is a directory named the same as the LiveLoad datastore.

   NOTE: Data Integrator creates the configuration file when you define a LiveLoad datastore. If you make changes to the datastore definition, Data Integrator updates the configuration file; you need to move the updated file to the query application computer.

9. Include calls to the LiveLoad COM Connector from your query application to determine which database is available to respond to the query.

   See "LiveLoad Connector" on page 30.

10. Create ODBC connections for the master and mirror databases on the computer on which the LiveLoad COM Connector runs.

    The ODBC data source names must be the same as the connection names defined for the datastore in Data Integrator.

Installing and configuring the Java API

For the Java API connector to run, the computer requires:

- Java Runtime Environment (JRE) version 1.3 or compatible
- Oracle JDBC library `classes12.zip`.

When you install Data Integrator Administrator, both of these are installed in the *`LINK_DIR`*`\ext` directory.

The LiveLoad Java connector files are:

- `ActaLLClient.property`
- `LLJConnector.jar`

When you install Data Integrator Administrator, these two files are installed in the *`LINK_DIR`*`\ext\lib` directory.

➢ To install LiveLoad Java (JDBC) Connector on a Data Integrator computer

Add the `LLJConnector.jar` and the Oracle JDBC library `classes12.zip` install paths into the environment variable `CLASSPATH`. (If `CLASSPATH` doesn't exist, create it.)

For example, to add the path for `LLJConnector.jar`:

- For ksh on UNIX, use:

```
export CLASSPATH =
$CLASSPATH:$LINK_DIR/ext/lib/LLJConnector.jar
```

- For NT, edit the `CLASSPATH` variable from the Environment tab in the System Properties control panel to include: `$LINK_DIR/ext/lib/LLJConnector.jar`.

➢ To install LiveLoad Java (JDBC) Connector on any computer

1. Create the installation structure.

   For Windows NT: *`LINK_DIR`*`\ext\lib`

For UNIX: *LINK_DIR*/ext/lib

where *LINK_DIR* is the location of your Data Integrator installation.

2. Install the Java Runtime Environment (JRE) version 1.3 or compatible and the Oracle JDBC library classes12.zip file in the *LINK_DIR*\ext directory.

   This is the software required to run the connector.

3. Copy both the LLJConnector.jar and the ActaLLClient.property from the Utilities directory on the Data Integrator CD to the *LINK_DIR*\ext\lib directory.

4. Define an environment variable LINK_DIR that points to the top-level of the directory structure.

   For example, if you created the structure "c:\ActaWorks\ext\lib", create *LINK_DIR* as "c:\ActaWorks".

5. Add the LLJConnector.jar and the Oracle JDBC library classes12.zip install paths into the environment variable CLASSPATH. (If CLASSPATH doesn't exist, create it.)

   For example to add the path for LLJConnector.jar:

   - For ksh on Unix, use:

     ```
     export CLASSPATH =
     $CLASSPATH:$LINK_DIR/ext/lib/LLJConnector.
     jar
     ```

   - For NT, edit the CLASSPATH variable from the Environment tab in the System Properties control panel to include:
     $*LINK_DIR*/ext/lib/LLJConnector.jar.

➢ To configure the LiveLoad Connector for JDBC

In the ActaLLClient.property file, change:

- The connector installation location
- The Oracle server host, port, and SID

Here is the content of an `ActaLLClient.property` file:

```
LINK_DIR: C:\ActaWorks
#Oracle connection string to JDBC connection
string mapping
#Master server database name short cut:
<host>:<port>:<sid>
ConnectionString:
OracleServerName:OraclePort:OracleSID
```

3 LiveLoad Components

LiveLoad consists of the following components:

- Datastore object in Data Integrator — Defines logical connections to the master and mirror databases
- Target table options — Produces SQL control files and statements to reconcile the master and mirror databases
- Control functions — (available in Data Integrator scripts) Controls switching between the master and mirror databases
- Delay configuration settings — Ensures database switching and reconciliation are timed appropriately relative to other operations
- Monitor and trace views — Displays the state of the LiveLoad databases
- LiveLoad Connector — (used by the application) Determines the database connection for data retrieval

Datastore object

A datastore in Data Integrator describes a connection to a database or application. Any datastore for a database connection can be "LiveLoad enabled."

If LiveLoad is enabled for a datastore, when Data Integrator loads the database it applies all SQL statements produced against the datastore (in data flows and SQL function calls) and also captures these statements in binary files (*.SQL). Data Integrator holds this buffered data until it applies the data to the second database during the reconcile phase of the LiveLoad cycle.

Data Integrator collects binary data representing SQL statements for each work flow or data flow for:

- Targets associated with the datastore without bulk loading enabled
- Scripts with SQL function calls against tables in the datastore
- Template tables associated with the datastore

If the targets indicate that bulk loading is enabled, Data Integrator does not produce any `.SQL` files. Instead it creates a `.BLK` file that contains a reference to the bulk loading command file. However, if the target has options set that are not bulk load options, such as instructions to drop the table before loading, Data Integrator generates `.SQL` files and a `.BLK` file. The `.SQL` files include only the operations not related to the bulk loading.

In addition, Data Integrator creates a control file that describes the logical progression of the steps of the job including database connection information, the job name, the data flows executed, the transforms executed, and the sources and targets involved.

Data Integrator stores these files in a directory on the computer where the Job Server is running:

LINK_DIR/LLRoot/*DatastoreName*/*Connection*

Where *LINK_DIR* is the location where Data Integrator is installed; *DatastoreName* is the name of the LiveLoad datastore; and *Connection* is the database user name, connection string, and database name concatenated together.

The format for the names of these files is as follows:

StartTime_1_Priority_Sequence_FinishTime_ParentFlow

where the components have these formats:

StartTime	A timestamp for the beginning of the processing of this data flow. The timestamp is in the form of MM_DD_YYYY_HH_MI_SS, where MM_DD_YYYY is the current date, HH_MI_SS is the current time.
Priority	An integer indicating the priority of this data flow relative to other currently running data flows. Zero (0) indicates the highest priority.
Sequence	An integer indicating the processor thread performing the data flow operations.
FinishTime	A timestamp for the end of the processing of this data flow. If no timestamp appears in the file name, the data flow did not complete successfully.
ParentFlow	The name of the data flow that generates the SQL statements. If a SQL function in a script, generates the statements, then LiveLoad includes the datastore name instead of the data flow name.

The performance of the reconciliation phase in LiveLoad has improved tenfold due to the use of these binary files. For best results, run Informix, ODBC, Oracle, and DB2 database updates in non-auto correct mode. This allows the updates to use parameterized SQL. Likewise, select bulk loading for an MS SQL Server database.

During the reconciliation phase, LiveLoad applies the binary files to the appropriate database, then deletes them.

The datastore definition now requires not one, but two database connections: one for the master database and one for the mirror. In the datastore definition, the two databases:

- Must have the same database owner
- Cannot specify the same physical database

NOTE: LiveLoad (on Oracle) does not require that the tables in the master and mirror databases share the same owner. By aliasing owner names, Liveload represents the behavior of separate owners. Furthermore, you can specify one physical Oracle database for LiveLoad. See Chapter 5, "Using LiveLoad with Oracle" for these datastore configuration procedures.

Use the same connection (or database server) names in the datastore definition and when you create ODBC connections on the computer where the LiveLoad Connector is running.

NOTE: If you must change the databases controlled by the LiveLoad datastore, Business Objects recommends that you create a new datastore.

➢ To define a LiveLoad datastore

1. In your database management system, create two servers, databases, or table spaces to be the master and mirror databases.

   The databases can be on the same database server or on separate servers. They must have the same user or owner name.

2. In Data Integrator, go to the **Datastore** tab in the object library.
3. Right-click and choose **New**.

4. Enter a name for the datastore.
5. Select **custom** from the **Application Type** list.
6. Select a database type.
7. Enter the database connection information for the master database.

   Use this connection string (for master and for mirror) when you define ODBC data sources on the computer where the LiveLoad connector is running.
8. Activate the **LiveLoad** option at the bottom of the window.

   Data Integrator assigns your connection information to the master database.
9. Select the **Mirror** option, then enter connection information for the mirror database.
10. If you are using DB2, enter separate bulk loader options for the master and mirror databases.

    Go to the **DB2 Properties** tab to specify these options.

    Bulk loader options for other database types are either not required or are specified for each target table added to a data flow.
11. Click **OK**.

The datastore appears in the object library.

Target table options

Each table from a LiveLoad-enabled datastore used as a target in a batch data flow is LiveLoad-enabled by default. To use a template table with LiveLoad, you must activate the LiveLoad option in the template table editor.

➢ To activate LiveLoad for a template table

1. Create the LiveLoad datastore as described in "Datastore object" on page 18.
2. Add the template table to a data flow as a target.

   Select the template table from the LiveLoad datastore in the object library.
3. Open the target editor.
4. Go to the **Options** tab.
5. Activate the **LiveLoad Cache** option.
6. Configure the target options as desired.

   For Informix datastores, bulk loader options require separate specifications for the master and mirror databases.

LiveLoad tables in real-time jobs

When LiveLoad tables appear as targets in real-time jobs, the real-time job always loads data into the **Live** database. Data Integrator does not produce reconciliation files for the data loaded to the tables.

Control functions

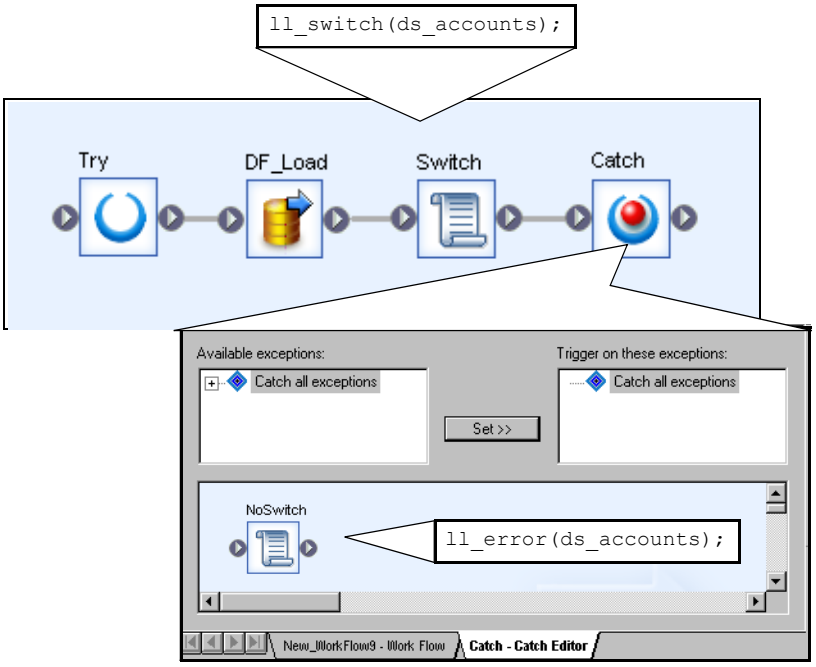
When the Data Integrator job is finished loading the inactive database and buffering control files for an application to the currently active database, a trigger must indicate that it is time to switch the active and inactive databases.

The trigger is the Data Integrator function `ll_switch`. Calling the function at the end of a work flow or job signals the LiveLoad manager to initiate the switching process. For details, see "ll_switch" on page 403 of the *Data Integrator Reference Guide*.

A second function, `ll_error`, allows you to continue using the active database if problems occur in the update job. If you use `ll_error` in your job when an error occurs, the monitor shows that the load failed and the database is in an error state. If you do not call `ll_error`, the error will not appear in the monitor until Data Integrator attempts to apply the reconciliation files to the database and fails. For details, see "ll_error" on page 402 of the *Data Integrator Reference Guide*.

The `ll_error` and `ll_switch` functions are listed under **Miscellaneous Functions** in the Data Integrator Function wizard. From the Query Editor's **Mapping** tab, select **Functions** to open the Function wizard.

Typical use of the two control functions might occur in a work flow containing a try/catch block and a data flow that loads a LiveLoad-enabled table. If an error occurs in the data flow, LiveLoad does not switch the databases: connections continue to be made to the database in the **Live** state until you correct the error and start the switching process manually. If no error occurs, LiveLoad switches the databases after the data flow completes.



Refreshing caches in real-time jobs

You can refresh caches used in real-time jobs when you switch the load database to live. To do so, include a batch file as a parameter to the `ll_switch` function call that triggers the live and load databases to switch.

The batch file includes a command to the Access Server to have it restart any or all configured service providers. For more information, contact Business Objects Technical Support.

Delay configuration settings

Successful switching of databases between **Live** and **Load** states depends upon several predictable factors. For example:

- Do not switch a **Live** database to **Load** or try to reconcile databases while queries are in process against the database.

  Estimate the maximum processing time of any given query.

- A Data Integrator data flow cannot load a database in the reconciliation state.

  Estimate the duration of the reconciliation process by correlating it to the number of changes made to the data in a given period.

- Do not switch the **Load** database to **Live** until it is successfully loaded.

  Though load time might vary, you can use the `ll_switch` function to explicitly initiate the switch at the end of the load. This function is described in "Control functions" on page 23.

Switching databases during reconciliation or loading states produces unpredictable results.

Data Integrator provides some system-wide configuration controls that allow you to estimate and set the delays required to set up the load-switch-reconcile cycle. Set controls by choosing **Tools > Options > LiveLoad Time-outs** in the Designer.

Option	Description
Reconcile pending	The delay between switching a database to **Load** state and when LiveLoad runs reconciliation statements against the database. This delay ensures that all queries against the database are complete.
Live pending	The delay between completing a job and switching the loaded database to a **Live** state. This delay ensures that all load jobs in progress at switch time have time to complete.

Monitor and trace views

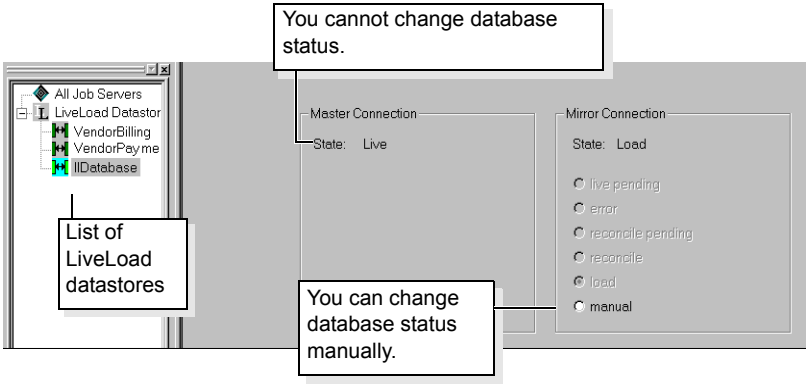
After you configure a LiveLoad datastore, it becomes "active" immediately. You can see a log of the events and errors associated with LiveLoad datastores from the Data Integrator **Log** tab. You can view database status defined by the datastore from the Data Integrator **Monitor** tab.

➢ To view the status of a LiveLoad datastore

1. In the project area, click the **Monitor** tab.

2. In the LiveLoad datastores list, select the datastore.

   The datastore monitor appears in the workspace.



Live Database State

The **Live** database is available only for queries. After a database becomes live, Data Integrator does not monitor its status.

Load Database States

The **Load** database can exist in any of the following states:

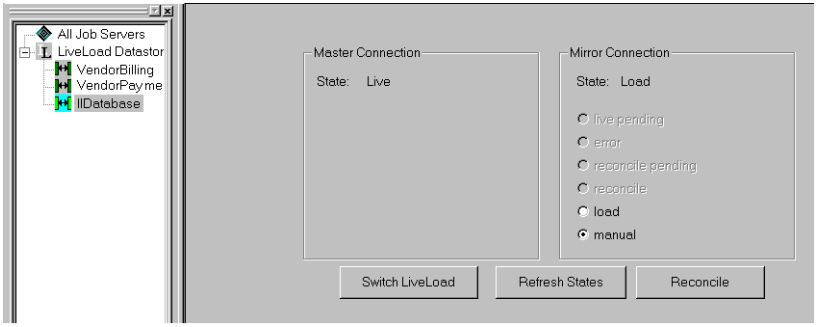
- Live pending — Data Integrator is waiting for any queries against the live database to complete before switching the **Load** database to **Live**.
- Error — The database is in an error state because reconciliation failed or because the `ll_error` function returned an error during the load. The switch between databases did not occur, and Data Integrator delays any scheduled jobs applied to the load database.
- Reconcile pending — Data Integrator is waiting for remaining queries against this database to complete before allowing the reconciliation to start.
- Reconcile — Data Integrator is ready to or in the process of reconciling this database by applying the content of the binary files.
- Load — Data Integrator is loading or has successfully loaded the database. The database is available to be switched to the **Live** state.
- Manual — You can control the state of the database from the monitor screen. No automatic operations can change the state. You can change the state of the database to manual to keep existing scheduled jobs from changing the state inadvertently. For example, change the database to **Manual** when you need to debug load or reconciliation problems.

From the monitor, you can change the state of the database not currently available for queries. For example, you might need to move the database manually through the stages of the LiveLoad process to recover from an error or to test your LiveLoad application before setting an automatic schedule. In addition, you might want to set the state to **Manual** to prevent an automated switch from occurring.

➢ To manually control the state of LiveLoad databases

1. Open the monitor to show the state of both databases.
2. Select **Manual** in the state list for the database currently in the **Load** or **Error** state. (You cannot change the state of a database in the **Live** state.)

   Data Integrator displays database state control options.



3. Click **Switch LiveLoad**.
4. Wait long enough to allow the Reconcile Pending delay to complete.
5. Verify that the LiveLoad database switched successfully.

➢ To view the LiveLoad trace and error logs

1. In the project area, click the **Log** tab.

2. In the **LiveLoad Datastores** list, select the datastore.

   The datastore trace log appears in the workspace.

   The trace log contains an entry for each LiveLoad operation, including:

   - Database state changes
   - Requests to switch live and load databases
   - Requests to identify the live database
   - Reconciliation progress
   - Creation of load control files

3. In the trace window, click the error log button.

   The datastore error log appears in the workspace.

   The error log contains an entry for each LiveLoad error including:

   - Creation and update of metadata tables for LiveLoad databases
   - Unresolved state of one or the other database
   - Creation of reconciliation files
   - Application of reconciliation files
   - Success of database connection
   - Database switching

LiveLoad Connector

The LiveLoad connector (COM or JDBC) is an object that controls the connection between a custom query application and databases managed by LiveLoad.

These methods control the connection:

- LiveConnection — Given a datastore name, this method opens a connection to the LiveLoad manager for the datastore. It returns an integer handle to indicate which database is active. Use this handle to get information about the LiveLoad database such as server name, connection string, or user/owner name. If the LiveLoad Connector cannot determine which database is live, the method returns `-1`.
- User — Given the database number returned by LiveConnection, this method returns the user name associated with the database.
- Password — Given the database number returned by LiveConnection, this method returns the password associated with the database.

  NOTE: The password is returned unencrypted.
- Server — (Microsoft SQL Server only) Given the database number returned by LiveConnection, this method returns the server name associated with the database.
- Database — (Microsoft SQL Server only) Given the database number returned by LiveConnection, this method returns the database name.
- ConnectionString — Given the database number returned by LiveConnection, this method returns the connection string required to access the database.
- DBType — This method returns the database type of the master and mirror databases (for example, Oracle or DB2).

- **ErrorInfo** — This function returns an error message to indicate the state of the LiveLoad databases.

In addition, the computer running the query application must have two ODBC data source names (DSN) defined, one for the master database and one for the mirror database.

The DSNs must be named as follows:

Database type	DSN name
Oracle	Connection string
ODBC	Host name
Informix	Connection string
DB2	Connection string
MS SQL Server	Database name

NOTE: The LiveLoad Java connector will load only to Oracle 8.0 and later.

LiveConnection

`LiveConnection(`*`datastore_name`*`)`

This method opens a connection to the LiveLoad manager. It returns a handle that identifies the database from the master and mirror pair ready for querying given a string containing the LiveLoad datastore name (logical connection name).

Parameter	Description
datatore_name	The datastore name created in Data Integrator to describe the logical connection to the LiveLoad databases.

You can pass the handle returned from LiveConnection to other methods to return the connection information required to access the **Live** database of the master and mirror pair.

This method returns `-1` if an error occurs.

See "Connector examples" on page 40.

User

`User(`*`connection`*`)`

This method returns the logon name for the **Live** database. You provide a database handle that specifies which of the LiveLoad databases is currently live.

Parameter	Description
connection	A one or a zero that indicates which of the two LiveLoad databases is currently live.

The return value corresponds to the currently open LiveLoad connection (call LiveConnection). If the open LiveLoad connection passes an illegal handle, this method returns a blank value.

Call this method to determine the proper user name with which to access the LiveLoad database.

See "Connector examples" on page 40.

Password

`Password(`*`connection`*`)`

This method returns the password for the "live" database. You provide a database handle that specifies which of the LiveLoad databases is currently live.

Parameter	Description
connection	A one or a zero that indicates which of the two LiveLoad databases is currently live.

The return value corresponds to the currently open LiveLoad connection (call LiveConnection). If the open LiveLoad connection passes an illegal handle, this method returns a blank value.

Call this method to determine the proper password with which

See "Connector examples" on page 40.

Server

`Server(`*`connection`*`)`

This method returns the server name for the "live" database. You provide a database handle that specifies which of the LiveLoad databases is currently live.

Parameter	Description
connection	A one or a zero that indicates which of the two LiveLoad databases is currently live.

The return value corresponds to the currently open LiveLoad connection (call LiveConnection). If the open LiveLoad connection passes an illegal handle, this method returns a blank value.

Call this method to determine the server name with which to access the LiveLoad database.

See "Connector examples" on page 40.

Database

`Database(`*`connection`*`)`

This method returns the database name for the "live" database. You provide a database handle that specifies which of the LiveLoad databases is currently live.

Parameter	Description
connection	A one or a zero that indicates which of the two LiveLoad databases is currently live.

The return value corresponds to the currently open LiveLoad connection (call LiveConnection). If the open LiveLoad connection passes an illegal handle, this method returns a blank value.

Call this method to determine the database name with which to access the LiveLoad database.

See "Connector examples" on page 40.

ConnectionString

`ConnectionString(`*`connection`*`)`

This method returns the connection string for the "live" database. You provide a database handle that specifies which of the LiveLoad databases is currently live.

Parameter	Description
connection	A one or a zero that indicates which of the two LiveLoad databases is currently live.

The return value corresponds to the currently open LiveLoad connection (call LiveConnection). If the open LiveLoad connection passes an illegal handle, this method returns a blank value.

Call this method to determine the connection string with which to access the LiveLoad database.

See "Connector examples" on page 40.

DBType

```
integer DBType ()
```

This method returns an integer value that corresponds to the database manager name for the LiveLoad-controlled database.

The return value corresponds to the currently open LiveLoad connection (called LiveConnection). Values can include:

Database type	Value
Oracle	0
ODBC	2
DB2	3
Informix	5
MS SQL Server	4

If the open LiveLoad connection passes an illegal handle, this method returns a blank value.

NOTE: The LiveLoad Java connector will load only to Oracle 8.0 and later.

ErrorInfo

```
string ErrorInfo ()
```

This method returns an error message string that corresponds to the current error state of the LiveLoad databases. If the LiveLoad connection is not in an error state, the method returns a blank value.

The return value corresponds to the currently open LiveLoad connection (call LiveConnection). If the open LiveLoad connection passes an illegal handle, this method returns a blank value.

Call this method to determine an error in case LiveConnection returns a -1.

See "Connector examples" on page 40.

Connector examples

This section contains two examples of scripts you can use for a LiveLoad connector:

- JDBC script example
- VB script example

JDBC script example

Use the following procedure to create a Java connector (API) script.

➢ To create a Java connector script

1. Create a file called `test.java`.

   See the example script on the next page.

2. Compile it using `javac test.java`.
3. Execute it using `java test`.

The connector returns:

- UserName
- Password
- Connection string for the live database

The following script is a sample Java connector:

```
import com.acta.LiveloadConnector.LLJConnector;
public class test
{
public static void main (String[] args)
{
LLJConnector llJTest = new LLJConnector();
int handle;
handle = llJTest.LiveConnection("ll_ora") ;
if ( handle != -1 )
{
String usr = llJTest.User(handle) ;
String pwd = llJTest.Password(handle);
String dsn = llJTest.ConnectionString(handle);
String dbs = llJTest.Server(handle);
String dbn = llJTest.Database(handle);
if ( usr != null )
{
System.out.println("User: " + usr);
System.out.println("1 is master in load status; 0 is
master in live status => " + handle);
}
else
System.out.println("No user: " + llJTest.ErrorInfo());
System.out.println("Password: " + pwd);
```

VB script example

The following script is a sample COM connector (API):

```
' Instantiate the LLConnector object
set obj= Server.CreateObject("LLConnect.LLConnect.1")
dim i
' Pass the LiveLoad datastore name collected
' earlier to return which database is live.
' i contains the handle.
i = obj.LiveConnection (ll_name)
' Check for connection failure.
   if i = (-1) then
      ErrorHandlingFunction(obj.ErrorInfo())
      ' Collect connection info to query database.
   else
      username = obj.User(i)
      clear_pwd = obj.Password(i)
      dsn_name = obj.ConnectionString(i)
      db_server = obj.Server(i)
      db_name = obj.Database(i)
   end if
' Build a login string to connect to the database
' through ODBC.
login_s = "DSN=" & dsn_name & "; UID=" & username & ";
   PWD=" & clear_pwd & ";"
' Instantiate the Microsoft COM object to make the
' database connection.
set RMConn = Server.CreateObject("ADODB.Connection")
' Open the connection using the login string.
MConn.Open login_s
```

4 Administering LiveLoad Databases

Data Integrator provides utilities and troubleshooting information to determine detailed information about the current state of your master and mirror databases. These utilities help you to:

- Determine differences between tables in the two databases
- Inspect LiveLoad error and trace logs if the job did not complete
- Manually apply database operations to resynchronize the databases

Determine differences between tables

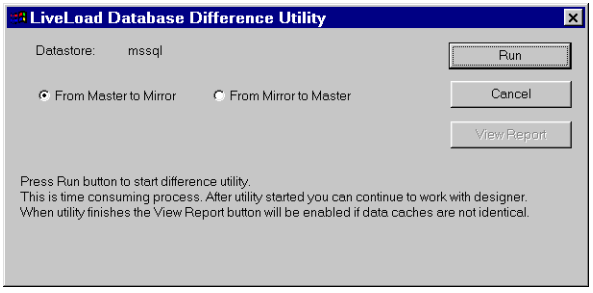
To verify that the master and mirror databases contain the same data, Data Integrator provides a Difference utility. This utility can produce a high-level report specifying differences between the tables in the two databases or detailed reports of the differences for any given table. In addition, it generates SQL statements and applies them against one database or the other to reconcile the differences.

Run the Difference utility after Data Integrator has applied the reconciliation files to one database and before loading begins. At that time, the master and mirror databases should be identical.

➢ To report differences between master and mirror databases

1. In the object library, locate the LiveLoad datastore.
2. Right-click the datastore name or any table included in it and choose **LL Database Difference Utility**.

   The LiveLoad Database Difference Utility window opens.



3. Choose a calculation direction.

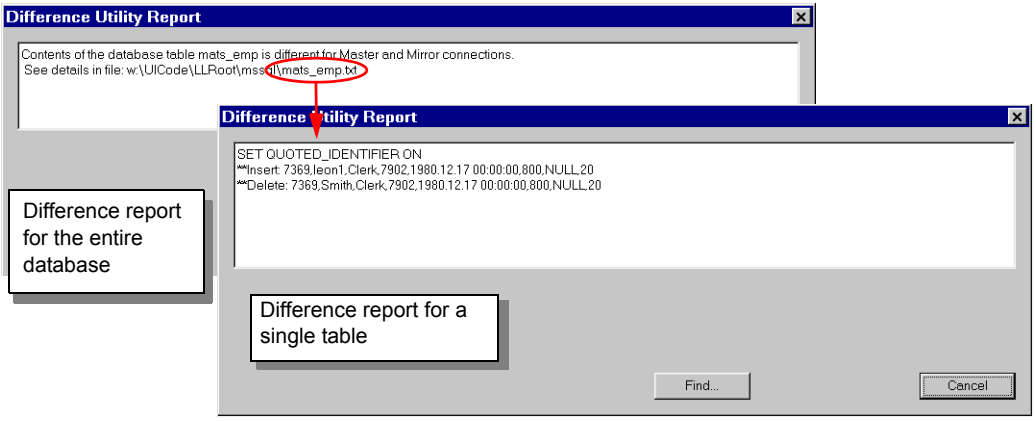
   For example, if you think the master database is the most complete and the mirror might need to be updated to match it, select **From Master to Mirror**.

4. Click **Run**.

When the calculation completes, a results message appears in the window.

5. If there are differences, click **View Report**.

   The report lists tables in which there are differences and the name of the file containing specific details about the differences.



   The Difference Utility creates text files containing the report in the following directory on the computer running the Job Server:

   LINK_DIR/LLRoot/*DatastoreName*/*Connection*

   Where *LINK_DIR* is the location of Data Integrator; and *DatastoreName* is the name of the LiveLoad datastore. The Difference utility names the files for the table or datastore being compared. These files have a `.TXT` extension.

   The Difference utility also produces a `.SQL` file that contains the SQL statements required to reconcile the second table in comparison to the first. For example, if you choose **From Master to Mirror**, this file contains the SQL required to update the mirror table to the content of the master table.

6. Click **Cancel** to close the report window.

Inspect LiveLoad error and trace logs

As described in "Datastore object" on page 18, LiveLoad captures all of the SQL generated by a Data Integrator job in files (one for each table being loaded) and applies it to the newly inactive database after a switch occurs. LiveLoad deletes each file after it is applied successfully to the database.

If a job fails, the reconciliation fails, or reconciliation has not occurred yet, you can find the reconciliation file in the following directory on the computer running the Job Server: `LINK_DIR/LLRoot/DatastoreName/Connection`. It can also be identified in the LiveLoad error log file.

Both the error and trace log files are located in the `LINK_DIR/LLRoot/DatastoreName` directory. See "To view the LiveLoad trace and error logs" on page 29 for information about how to find these files from within Data Integrator Designer.

In the case where a reconciliation job fails, use the error and trace logs to determine the cause of the error (most likely a data or connectivity problem) and which reconciliation files are affected. Remove the file that includes the error (from the `LINK_DIR/LLRoot/DatastoreName/Connection` directory). Then from the Designer, manually re-run the load job; this applies any files remaining in the directory against the database.

See "Manually apply database operations" on page 47 for more information.

Manually apply database operations

If you find a database error and then resolve the issue, you might need to manually step through stages in the LiveLoad cycle. For example, if the reconciliation stage fails because of an error such as a lost connection, you can use the LiveLoad monitor to reapply the reconciliation files to the database.

Failure	Result	Response
Database experiences an error during reconciliation	• Reconciliation not fully applied to the database • State of the database in the LiveLoad monitor is **Reconcile** or **Error**	Locate and fix the source of the failure. If your job is designed to be re-executed without adding duplicate data to the target, restart the reconciliation process manually through the monitor (change state to **Manual** and click **Reconcile**). If you cannot re-execute your job without compromising the target: • Find the last reconciliation files that LiveLoad applied using timestamps on the files. • Restart the reconciliation process manually through the monitor (change state to **Manual** and click **Reconcile**)
Data Integrator experiences an error during job execution	• Potentially the load was only partially applied to the database • State of the database is **Load**	If your job is designed to be re-executed without adding duplicate data to the target, restart the job. If you cannot re-execute your job without compromising the target: • Remove any rows applied to the target database (the Difference utility can produce SQL commands to make this database the same as the live database) • Delete all reconciliation files. • Restart the job. Consider changing the LiveLoad state to **Manual** to prevent automatic switching.

➢ To manually apply database operations

1. On the Designer's **Monitor** tab, select the name of the datastore connection under **LiveLoad**.

   See "Monitor and trace views" on page 26 for more information about the options on the LiveLoad monitor window.

2. Select the **Manual** option.

3. Debug the error.

4. Reconcile the database.

   To apply the existing reconciliation files, click **Reconcile** in the monitor window.

   The state of the database changes to **Reconcile Pending**. When the reconciliation is complete, the state changes to **Load**.

   If an error occurs, the state of the database changes to **Error**. On the Designer's **Log** tab, select the name of the datastore connection under **LiveLoad** to check the error log for specific messages.

5. Verify the reconciliation completed successfully.

   To further verify that the reconciliation produced the appropriate results, you can run the Difference utility described in "Determine differences between tables" on page 44.

   Business Objects recommends that you set the database to the manual state until you are satisfied that the LiveLoad database is in a consistent state. The manual state prevents load jobs from trying to load the database at the same time you are working with it. The new load jobs will be blocked until you reset the database state to **Load.**

5 Using LiveLoad with Oracle

If the database you are using for a data warehouse (target) is an Oracle database, you can use different owners on the master and mirror databases. By aliasing owner names, LiveLoad effectively represents the behavior of separate owners. Furthermore, you can specify one physical Oracle database for LiveLoad.

To configure LiveLoad on a single Oracle instance, first you must define a LiveLoad datastore. A datastore in Data Integrator describes a connection to a database or application. Any datastore for a database connection can be "LiveLoad activated." The datastore must be "LiveLoad activated" by specifying mirror and master datastore information.

If your master and mirror databases do not share the same owner name, then you must assign an alias to the datastore. Any tables or functions imported through that datastore will adopt this alias.

Configuring LiveLoad for a single Oracle instance

When configuring and activating LiveLoad for a single Oracle instance, complete the following steps:

- Define a LiveLoad datastore
- Create a LiveLoad alias for your LiveLoad datastore

➢ To define a LiveLoad datastore

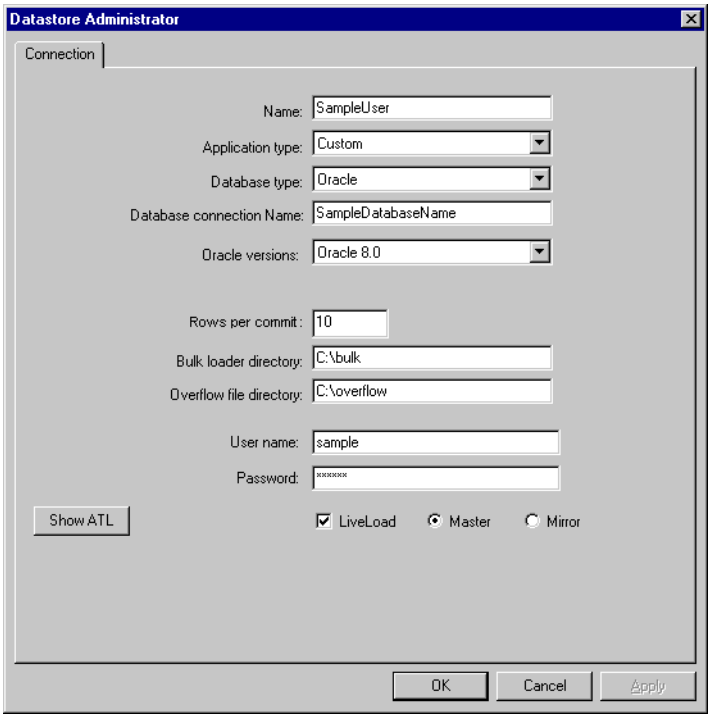
1. In your database management system, create two databases: one assigned the master and the other, the mirror.
2. In Data Integrator, select the **Datastore** tab in the object library.
3. Right-click and choose **New**.

   The Datastore Administrator appears.
4. Enter a unique name for the datastore. Set the application type to **Custom** and choose **Oracle** as your database type.
5. Enter the database connection information for the master database.
6. Click the **LiveLoad** check box. This activates LiveLoad.

   Your connection information is assigned to the master database.

7. Select the **Mirror** option, then enter connection information for the mirror database. Except for the information in **User Name** and **Password**, this information must match the **Master** connection information.

   NOTE: For this example, the mirror and the master databases are the same. LiveLoad will be activated on a single instance.



8. Click **OK**.

   The datastore appears in the object library.

➢ To create a LiveLoad alias for your LiveLoad datastore

1. In the Data Integrator object library, select the **Datastore** tab.
2. Right-click the **LiveLoad** datastore.

3. From the shortcut menu, select **Properties**.
4. In the Properties window, select the **Aliases** tab and right-click to view a shortcut menu. Select **Add**.

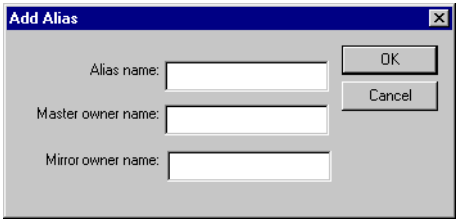
   The Add Alias window appears.
5. In the Add Alias window, enter:
   - Alias name

     A name you set that will appear in the Properties window, the Designer, and the Datastore.
   - Master user (owner) name

     The existing user name for the database you defined as the master database in the datastore.
   - Mirror user (owner) name

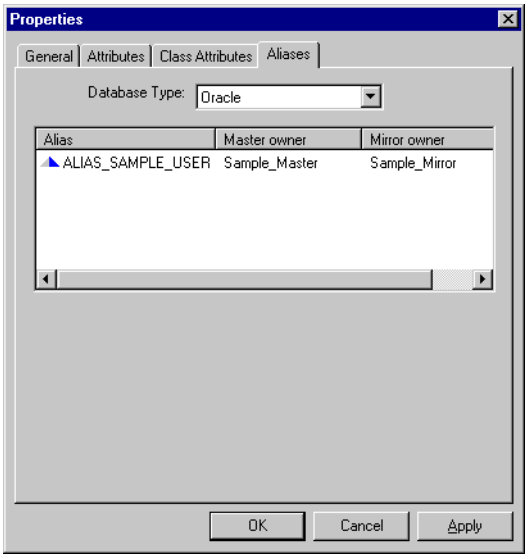
     The existing user name for the database you defined as the mirror database in the datastore.

   The master and mirror names must correspond to those defined in the LiveLoad datastore editor.



6. Click **OK**.

The alias information appears in the Properties window.



7. From the **Datastore** tab in the object library, you can import tables or functions from the repository to include in your LiveLoad configuration.

   Once imported, a function or table is identified in the object library by table or function name, datastore, and alias name.

A Glossary

This glossary defines terms used in this documentation set including common terms that have a special meaning with respect to Data Integrator. Within a definition, other terms that are defined in the glossary appear in **bold type**.

ABAP

Advanced Business Application Programming.
A fourth-generation programming language developed by SAP in which R/3 applications are written.

ABAP program

A program that executes database operations on an **SAP R/3** server. Data Integrator **R/3 data flows** generate ABAP programs.

Access Server

A process that starts, stops, and monitors real-time services (**real-time jobs**). The Access Server dispatches requests to real-time services, ensuring optimal load balancing and complete life cycle management. The Access Server also starts, stops, and monitors **adapters**.

adapter

An external Data Integrator **interface**. There are two types of adapters:

- Custom adapters — Adapters developed using the Data Integrator Adapter Development Kit
- Prepackaged adapters — Adapters prebuilt and purchased from Business Objects, such as the Data Integrator MQSeries Adapter

Administrator

A browser-based system administration application that supports Data Integrator. Use the Administrator to configure and manage:

- Services (**real-time jobs**)
- Service providers (instances of real-time services)
- Clients (Web application clients)
- Batch jobs
- Adapters
- Job logs

after-image

The values in an UPDATE row after the row changes. You might want to pair before- and after-images of UPDATE rows when designing log-based changed-data capture jobs which Data Integrator supports.

aggregate data

Data that results when a process combines elements. This data can be presented collectively or in summary form.

ALE

Application Link Enabling. An SAP R/3 programming-related interface designed to allow reliable communication across a distributed environment.

Analytic Rapid Mart

An **Data Integrator application** that produces a predefined **data warehouse** for a specific area of business analysis. Analytic Rapid Marts are typically based on dimensional modelling. With Analytic Rapid Marts, users can manage large volumes of data.

attribute

A **property** created for a type of **object**.

BAPI

Business Application Programming Interface. A standardized SAP R/3 programming interface that allows non-SAP R/3 applications to access specific business processes and data.

back-office application

A software system that an enterprise uses to run business operations, such as human resources, general ledger, and finance. Back-office applications store data in a relational **database** optimized for operational use.

There are several types of back-office applications:

- Enterprise resource planning (ERP) — For example, SAP R/3, BAAN, J.D. Edwards, PeopleSoft
- Supply chain management (SCM) — For example, i2
- Customer relationship management (CRM) — For example, Siebel, Clarify
- Legacy

Basis

The SAP infrastructure. Basis is the foundation for all SAP products.

batch job

A group of objects (data flows, work flows, conditionals, scripts, and so forth) that executes on a schedule. This type of job is recommended for operations on large amounts of data.

before-image

The values in an UPDATE row before the row changes. You might want to pair before- and after-images of UPDATE rows when designing log-based changed-data capture jobs which Data Integrator supports.

bulk loading

A software-based mechanism that moves large data files. This mechanism supports compression, blocking, and buffering to optimize transfer times.

business component

A set of tables Siebel applications use to create a logical object called a business object.

Business Objects Data Platform

A set of applications from Business Objects, Americas, that enterprises can use to manage all types of data—from data for supply chain partners to data for customers. The Business Objects Data Platform consists of **Data Integrator** and one or more **Rapid Marts**.

CDC datastore

A CDC datastore allows you to limit extracted data to changed data only. A CDC datastore connects a **changed-data capture** table on a source database to Data Integrator.

CDC checkpoint

A CDC checkpoint enables Data Integrator to restrict CDC subscription reads. After you enable a checkpoint, the next time the CDC job runs, it reads only the rows inserted into the CDC table since the last checkpoint.

CDC Subscription

A CDC subscription name allows different data flows to extract data from the same table without corrupting data extracted by other data flows. The datastore and table name can be the same; however, the table owner name must be different in each source CDC table.

changed-data capture

The process of retrieving changes made to a production data source. This process consolidates units of work, ensures data is synchronized with the original source, and reduces data volume in a warehouse environment.

Citrix MetaFrame XP

Citrix MetaFrame XP software provides an access infrastructure for enterprise applications. You can use this software to run Data Integrator on a server which publishes instances of the Designer and other Data Integrator interfaces to users on client computers.

client/server

A distributed technology approach where the processing is divided by function. The server performs shared functions (such as managing communications and providing database services), while the client performs individual user functions.

COM

Component Object Model. Provides a set of interfaces that allow clients and servers to communicate within the same computer (running on Windows 95 or later versions).

Commerce Rapid Mart

An **Data Integrator application** that produces a predefined **data warehouse** and that includes logic to support real-time requests for data from a **back-office application**. A Commerce Rapid Mart is usually based on different data model than an **Analytic Rapid Mart**. With a Commerce Rapid Mart, users can integrate a company's back-office system with an **e-commerce** application.

Communication Structure

In **SAP BW**, a data structure that defines a set of **InfoObject**s available from an **InfoSource** to put into **InfoCube**s.

conditional

A **single-use object**, available in **work flow**s**,** that allows you to branch the execution logic based on the results of an expression. The conditional takes the form of an if/then/else statement.

custom ABAP program

A custom ABAP program runs an **ABAP program** and generates a **data set**. With a custom ABAP program, you can run an existing ABAP program as part of a job. Use a custom ABAP program as a **source** in a **data flow** or an **R/3 data flow**.

data extraction

The process of moving data from a **database** or application source to a database target (either from a legacy database to a **data mart**, or from one data mart to another).

data flow

A **reusable object** containing **steps** to define the transformation of data from **source** to **target**. Data flows are called from inside a **work flow** or **job**. You can pass information into or out of data flows using **parameters**.

Data Integrator

A software system that allows users to build and execute applications with which they can create and maintain **data warehouses**.

Data Integrator consists of several components:

- **Access Server**
- **Data Integrator engine**
- **Data Integrator repository**
- **Data Integrator Service**
- **Administrator**
- **Designer**
- **Job Server**

Data Integrator application

A set of **jobs** that process batch data extractions and real-time requests using data from a variety of data sources.

Data Integrator engine

The core process that reads job information from the **Data Integrator repository** and sets up run-time processes that execute the job. The run-time processes extract, transform, and load relational and hierarchical data. The **Job Server** starts the Data Integrator engine to execute batch or real-time jobs.

Data Integrator interface

A program that Data Integrator uses to access data sources. Specific interfaces vary by installation. There are internal interfaces—those native to the installation—and external interfaces—those that you install separately. Internal interfaces allow Data Integrator to access applications like **SAP R/3** and **SAP BW**, **messages**, relational database systems, and **legacy systems**. An external interface is also known as an **adapter**. It allows Data Integrator to access applications using information exchange technologies such as Web Services and MQ Series.

Data Integrator repository

The database that contains information about an Data Integrator application. The repository contains information about defined **reusable objects**, the **metadata** for tables, and built-in **transforms** and **functions**. When you invoke Data Integrator, you log in to the repository containing the **objects** you want to use.

Data Integrator Service

The process that ensures that the **Access Server** and the **Job Server** are running. You can configure the Data Integrator Service to restart the Access Server and Job Server whenever the computer where they are located restarts.

data loading

The process of populating a **data warehouse**. Data loading is provided by **DBMS**-specific load processes, DBMS insert processes, and independent fast-load processes.

data mapping

The process of assigning a source data element to a target data element.

data mart

A highly-focused version of a **data warehouse**. Typically, created by a department or division of a company, data marts contain data for a specific subject area, such as finance or sales. Data Integrator can populate a data mart.

data transformation

The process of filtering, merging, decoding, and translating source data to create validated data for a **data warehouse**.

data set

Rows of data with a defined schema. A step in a **data flow**—such as reading data from a **source**, joining data in a **Query transform**, or transforming data though another **transform**—yields a data set. You can view individual data sets by placing a **target** table or file at that point in the **data flow**.

data transport

A step in an **R/3 data flow** that defines a **target** to store the **data set** extracted during the flow. You can locate the target file on the **SAP R/3** server or in a location accessible to both the SAP R/3 server and to Data Integrator across a network.

data type

The format used to store a value. Data types can imply a default format for displaying and entering the value. Data read from a **source** is converted to the appropriate Data Integrator data types; data loaded to a **target** is converted from its Data Integrator data type to the type appropriate for the target.

data warehouse

A relational **database** containing subject-oriented information that supports decision analysis. Typically, information in a data warehouse is integrated from multiple sources, including operational sources. Data warehouses provide a single, non-volatile point of reference for a particular time period. Data warehouses are composed of **data marts.** Data Integrator can populate a data warehouse.

database

A collection of **table**s managed by a **DBMS** such as Microsoft SQL Server or Oracle.

DataConnector

DataConnector operator instances are used to read data files generated by Data Integrator when performing bulk loading using the Teradata Warehouse Builder.

datastore

A logical channel connecting Data Integrator to a **database**. The datastore definition includes the name and location of the database as well as user authentication information. Data Integrator uses a datastore definition to qualify a table name wherever a table is indicated in a diagram or expression. You can access the datastore definition through the **object library**.

DBMS

Database management system. A software system that builds and maintains database tables.

debug mode

Allows you to debug a job using the **interactive debugger** features in the Designer.

declarative specification

A way of indicating the desired results of a **data transformation** without placing constraints on the method of reaching the results. When you create a **data flow** in Data Integrator, you specify what data you want to read from which **source**, what you want done to that data, and where you want it loaded. What you specify constitutes the declarative specification. Data Integrator then determines the most efficient way to reach the requirements you set, including pushing operations to the source **DBMS** for execution.

degree of parallelism

You can set a value at the data flow level that allows you to parallel-process a **transform** within a data flow.

For example, if from the Properties window of a data flow you set the **Degree of parallelism** to 5, then when the job executes, Data Integrator replicates each transform in the data flow five times. Each of these replicated transforms executes in parallel using a separate process thread. Using a value for degree of parallelism can improve job performance if a **Job Server**'s computer memory and number of CPUs supports a job's parallel-processing configuration settings.

dependency analysis

A category of metadata reports that produces reports based on search criteria you specify to find specific objects in the repository. For each object returned, dependency analysis reports provide object lineage as well as object impact information.

derived data

Data that results from a computational step applied to reference or event data. Derived data is the result of either relating two or more elements of a single transaction (such as aggregation), or relating one or more elements of a transaction to an external algorithm or rule.

Designer

A graphical user interface that allows you to design and test an **Data Integrator application**.

diagram

The icons and connections between the icons that make up the definition of a **job**, **work flow**, or **data flow**.

DMZ

Demilitarized zone. A perimeter network that is more accessible than your most protected network, but located inside a security layer protected from the internet.

drill down

A method of exploring detailed data that was used in creating a summary level of data. Drill down levels depend on the granularity of the data in the **data warehouse**.

DTD

Document type definition. A text file that describes the elements (tags) in an **XML** document and the relationship among them. When an XML document is used to describe a transaction, the DTD describes the data schema used in the transaction.

e-commerce

Buy and sell transactions conducted over the internet.

EDI

Electronic Data Interchange. Electronic exchange of structured data between businesses. This exchange is not dependent on hardware, software, or communication protocols.

element

A component found within XML Schemas and DTDs.

embedded data flow

A data flow created within one or more data flows.

enterprise data

Data that is defined for use across a corporate environment.

ERP system

Enterprise resource planning system. A **back-office application** from which Data Integrator can extract data.

exception

An error that occurs while executing a **job**. You can catch individual or groups of exceptions using a **try/catch block** inside a **work flow**. Catching an exception allows you to automatically execute a solution for the error.

Executive Information Systems (EIS)

Tools programmed to provide canned reports or briefing books to executives. They offer strong reporting and drill-down capabilities. These tools allow ad hoc querying against a multi-dimensional **database**, and most offer analytical applications along functional lines such as sales or financial analysis.

extract date

The date data was extracted.

extract frequency

The interval at which data is extracted, such as daily, weekly, monthly, or quarterly. The frequency that data extracts are needed in the **data warehouse** is determined by the shortest frequency requested through an order, or by the frequency required to maintain consistency of the other associated data types in the source data warehouse.

file

A **source** or **target** for data. To read or load data to a file within Data Integrator, define a **file format** that instructs Data Integrator about the structure of the data in the file.

file format

A description of how data is or should be organized in a **file** Data Integrator reads from or loads to. A file format can be specific to a single file or generic for many files.

front-office application

Software that companies use to support reporting, **e-commerce**, and other tasks. Types of front-office applications include:

- Supply chain collaboration
- Analytical
- Sell-side e-commerce
- Customer relationship management

function

A program that operates on values that are passed to it. Data Integrator functions are available through a function wizard in a **script**, **conditional**, or **Query transform**. Data Integrator also gives you access to functions provided by the **DBMS** you are using. In addition, you can define your own functions using the Data Integrator scripting language or by writing C-language programs.

IDoc

Intermediate Document. An SAP-specific format. Used for **EDI** and **ALE**.

IDoc type

Indicates the SAP format that is used to interpret the data of a business transaction. Consists of the following components:

- A control record

  Identical for each IDoc type.

- Several data records

  A single data record consists of a fixed key part and a variable data part. The data part is interpreted using segments, which differ depending on the IDoc type selected.

- Several status records

  Identical for each IDoc type. Describe the status states an IDoc has already passed through or the status an IDoc has attained.

import

The process of acquiring information for the **Data Integrator repository**. Import the following kinds of information into Data Integrator:

- The **metadata** for **source** and **target** databases
- Descriptions and code for user-defined and DBMS **functions** and **transforms**

InfoArea

In **SAP BW**, an arbitrary grouping of **InfoCube**s.

InfoCube

In **SAP BW**, a multidimensional star schema that is limited to one fact table and 16 dimensions.

InfoCubes store only numeric data at the fact level.

SAP BW reserves three of the dimensions for currencies, time, and units of measure. Dimensions have to be linked back to master data and hierarchy tables to gain access to the information stored there.

When a user query is initiated, BW creates a temporary **OLAP** cube that stores a constrained subset of the permanent OLAP cube. The temporary InfoCube persists until the user disconnects from BW or changes the query.

InfoObject

In **SAP BW**, an InfoObject is equivalent to a data element and its associated metadata (for example, description or business rules). Can be a **Key Figure** or a Characteristic.

InfoPackage

In **SAP BW**, a schedule for running BW updates.

Information resource (IR)

A producer or consumer of enterprise information. An information resource can be a flat file, a **back-office application**, an EAI system, or a Web application.

InfoSource

In **SAP BW**, a logical construct that defines a set of **InfoObject**s and their mappings to source data elements. InfoSources do not store data, but rather serve as conduits via **Transfer Structure**s.

Integration Rapid Mart

An **Data Integrator application** that provides point-to-point content mapping and data movement mechanisms between different **back-office applications**.

interactive debugger

A Designer feature that allows you to step through the data of a job one row at a time using filters and breakpoints on a line. Like executing a job, you can start the interactive debugger from the Debug menu when a job is active in the workspace. While in debug mode, all other Designer features are set to read-only.

interface

Data Integrator offers two types of interfaces:

An internal **Data Integrator interface**, allows you to create datastore connections to natively supported applications.

An external **Data Integrator interface** (or **adapter**), allows Data Integrator to communicate with information exchange technologies such as Web Services and MQ Series.

IR

See **Information resource (IR)**.

job

The unit of work that can be scheduled independently for execution by the **Administrator**. Jobs are special **work flows** that can be called by other work flows or jobs.

Job Server

A process that receives requests from the **Designer** and the **Administrator** to start and stop **jobs**. To start batch or real-time jobs, the Job Server triggers the **Data Integrator engine**. Engine processes run on the same computer as the Job Server process that triggers them.

join rank

A value given to or calculated for all data sets in a **data flow**. Data Integrator uses the join rank to determine which **source** to read first when assembling the **data set** in a join. Data Integrator uses the source with the lower join rank as the inner source of the join and uses the source with the higher join rank as the outer source of the join. If more than one of the sources is located in the same **DBMS**, the highest rank is used for all of the sources from that DBMS. When a source is processed through a **Query transform** or another **transform**, and then used in a join, the maximum join rank defined for the source is used as the join rank in the query.

Key Figure

In **SAP BW**, an **InfoObject** that represents a numeric fact.

legacy system

An information or transaction processing system used to store data such as bank balances, inventories, payroll, and manufacturing parts.

license-controlled feature

A Data Integrator feature that is enabled or disabled based on the product license. The product license controls which icons and settings are available in Data Integrator as an internal **Data Integrator interface**.

line

The line you use to connect objects in a workspace diagram. Lines represent a left-to-right flow path for data moving between two objects during job execution.

memory datastore

A datastore connection/container for memory tables.

memory table

Internal Data Integrator table used to store a data set in memory while a job runs. Use instead of staging tables to improve performance of a real-time job built with multiple data flows. Use a memory table to move a data set between data flows.

message

Represents hierarchical data, such as a header with line items, for document-oriented transactions, such as a purchase order.

metadata

In Data Integrator, information acquired and maintained to describe tables in **source** and **target** databases. This information includes the names of tables and their columns, and the data types of the columns. All the metadata for all user tables is imported at one time for a given **datastore**.

In general, metadata typically includes a description of data models, a description of the layouts used in database design, the definition of the system of record, the mapping of data from the system of record to other places in the environment, and specific database design definitions.

multiple data flow model

One of three data flow models you can use in real-time jobs. Use this model to develop real-time jobs in separate logical object segments (data flows) rather than in a single data flow. Allows for any number of data flows in a real-time processing loop under these conditions: The first data flow must have a single message source, the last data flow must have a single message target. Other data flows in the same real-time job cannot have a message source or message target. See also **put/acknowledge data flow model** and **single data flow model**.

nested data

Data in one **table** that is related to a single row of another table. A nested table appears in Data Integrator as a column in a parent table. Columns in the nested table can themselves contain tables.

object

Any item that you create in the **Designer**. Data Integrator distinguishes two classes of objects: **reusable objects** that are complete and can be reused in your projects (such as **data flows**) and **single-use objects** that only appear as components of other objects (such as a **try/catch block**). This distinction affects how you create and retrieve each type of object.

object definition

The options that describe the operation of an **object**. To view and modify an object definition, open the object so that its definition appears in the **workspace**.

object library

A tool in the **Designer** that gives you access to reusable objects.

ODBC

Open database connectivity. A standard for **database** access initiated by the SQL Access Group consortium and now owned by Microsoft.

ODS

Operational data store. An OLAP-designed relational **database** that an enterprise has designated as the operational database of record (for example, a finance department might use an ODS to close its books).

OLAP

Online analytical processing. A relational **database** design that is optimized for random access. OLAP systems are used in a query environment, such as for a **front-office application**.

OLTP

Online transaction processing. A relational database design optimized for operational use. OLTP systems are used in an operational environment, such as for a **back-office application**.

operation code

A flag associated with a row in a **data set** that indicates the status of the data in the row. The operation codes are INSERT, UPDATE, DELETE, and NORMAL.

operational statistics

A category of reports within the metadata reporting tool that provides you with job and data flow performance information over a given period of time or number of executions.

parameter

A value passed to a **work flow** or **data flow** when that flow is called.

partition

To divide table data into sets based on a criteria such as a range or list of values in each row. You can configure Data Integrator to read and write partitioned table data in parallel threads. Designing jobs with partitioned table data can improve job performance if a Job Server's computer memory and number of CPUs supports the job's parallel-processing configuration settings.

project

The collection of **jobs** available in the **Designer** at a given time. A project provides a way to organize the **objects** you create.

property

Detailed descriptive information about **objects**.

put/acknowledge data flow model

One of three data flow models you can use in real-time jobs. Similar to the multiple data flow model, this model does not require a data flow with a target message. Data Integrator creates a target message to automatically acknowledge a request message. Designed for use with IDoc sources in SAP R/3 systems. See also **multiple data flow model** and **single data flow model**.

Query transform

A **data transformation object** that you can use to map columns from a **source** to a **target** schema, add new columns to the target schema, determine the data to extract, and perform operations on the data. Like a SQL SELECT statement, a query creates a **data set** that satisfies the conditions you specify.

Rapid Mart

A prepackaged **Data Integrator application** that delivers data for a specific business purpose. Easily deployed and customized, BusinessObjects™ Rapid Marts include the extraction, transformation, and data loading logic to support analysis or **e-commerce**. There are three types of Rapid Mart:

- **Analytic Rapid Mart**
- **Commerce Rapid Mart**
- **Integration Rapid Mart**

R/3 data flow

A **data flow** that extracts data from an **SAP R/3** source table. Data Integrator translates steps you define in an R/3 data flow into **ABAP** and then passes the ABAP program back to your SAP R/3 system for execution. The resulting **table** or **file** resides on the SAP R/3 system to be used as a **source** in the parent data flow.

real-time job

A group of objects (data flows, work flows, conditionals, scripts, and so forth) that execute on-demand as a "request-response" system. You design real-time jobs in the **Designer**, then configure them as real-time services and associate them with an **Access Server** in the **Administrator**, where they are started, managed and monitored. When a real-time service receives a request from a caller, it processes the request and returns a reply. This type of job is recommended for operations on small amounts of data.

relational data

A **data set** in which data in each column contains a scalar value. Data Integrator can process relational data; it can also process **nested data**.

repository

See **Data Integrator repository**.

reusable object

An **object** that can be defined, stored, and reused independent of other objects. Create reusable objects from the **object library**.

RFC server

Remote Function Call server. The Data Integrator RFC server allows third-party programs, including SAP R/3, to schedule and initiate Data Integrator jobs and return the results to Data Integrator.

SAP BW

Business Information Warehouse, an SAP product.

SAP R/3

An **ERP system**.

script

A step in a **work flow** that allows you to calculate values to pass to other parts of the work flow. The script can call **function**s, execute if-then-else statements, and assign values to **variable**s. Write a script in the Data Integrator scripting language.

segment

Format with which the data records of **IDocs** are interpreted.

server group

A defined collection of Job Servers on different computers. A server group automatically measures resource availability on each Job Server in the group and distributes scheduled batch jobs to the Job Server with the lightest load at run-time. Use the Server Groups node in the Administrator's navigation tree to group Job Servers that are associated with the same repository into a server group.

service request

Any message sent from a Web client that requires processing by a **real-time job**.

session

Associated with the **Adapter Properties** tab in an adapter datastore, a session is created by a stream-oriented adapter to group adapter components and maintain state for interactivity between Data Integrator and the adapter.

single data flow model

One of three data flow models you can use in real-time jobs. Allows for one data flow in a real-time processing loop. This data flow must have a single message source and a single message target. See also **multiple data flow model** and **put/acknowledge data flow model**.

single-use object

A step in a **work flow** or **data flow** that cannot be saved independently of the flow. Create single-use objects from the tool palette.

smart editor

A flexible editing tool in Data Integrator used for creating scripts, expressions, and custom functions without having to type the names of existing elements like column, function, and variable names.

SNMP

System Network Management Protocol helps network administrators manage network routing hardware. The protocol can manage a variety of hardware and software devices.

source

An **object** from which Data Integrator reads data.

SQL

Structured Query Language. A query language for accessing relational, **ODBC**, DRDA, or non-relational **database** systems.

SQL query tool

An end-user tool that accepts **SQL** to be processed against one or more relational databases.

star schema

A **database** design you can use to format data in a **data mart**. This design is based on a single fact table to which any number of dimensional tables may be joined. This type of database design supports multi-dimensional database analysis.

step

An **object** that is part of the definition of a **work flow** or **data flow**. Each step is represented by an icon in the diagram of the flow and is connected to other steps to indicate the flow of data through the data flow or the order of execution in the work flow.

table

A **database** table that Data Integrator reads data from or loads data into. The path and mechanisms for reading and loading data and apportioning the data among rows and columns are defined in the **datastore** that the table is associated with. Writing a **data set** to a database table means sending a combination of rows with appropriate **operation code** to the database table.

target

An object in which Data Integrator loads extracted and transformed data in a **data flow**. Data Integrator only loads rows flagged as INSERT or UPDATE.

TCP/IP

Transmission Control Protocol/Internet Protocol. The basic communication protocol of the internet, and often intranets and extranets. A computer having direct access to the internet contains a copy of the TCP/IP program. TCP/IP makes it possible for computers to communicate with each other.

Tdpid

(Teradata Director Program ID) The server name Data Integrator uses when loading with the bulk loader option. Data Integrator uses tdpid as a Teradata Warehouse Builder operator attribute.

Transfer Structure

In **SAP BW**, a data structure that defines a set of **InfoObject**s to be moved into BW. To an ETL tool like Data Integrator, a Transfer Structure looks like a table.

transform

A step in a **data flow** that acts on a **data set**. Data Integrator transforms are available through the **object library**. You can also import user-defined transforms and transforms from your **DBMS**.

Transformation Rule (Transfer Rule)

In **SAP BW**, a data manipulation rule between a **Transfer Structure** and a **Communication Structure**. The results of these transformation are globally available to all **InfoCube**s.

try/catch block

A combination of a try object and one or more catch objects that define alternate execution paths in case an error occurs during the execution of a **job**. You can tune try/catch blocks to trap specific errors and to provide general alerts or messages if an error occurs.

Update Rule

In **SAP BW**, a data manipulation rule between a **Communication Structure** and an **InfoCube**. The result of the data manipulation is specific to one InfoCube.

variable

A symbolic placeholder for a value. Data Integrator lets you define local variables and global variables.

Local variables pertain to the **work flow** in which they are defined. You can pass the value into another **work flow** or **data flow** using a **parameter**.

Global variables pertain to the **job** in which they are defined. With global variables, there is no need to define parameters between **objects** in the job. Global variables can also be selected at execution time. This eliminates the need to open the Designer to set global variable values.

Web Services

A standard platform for integrating applications. Web Services allow different programs, constructed in different languages, on different platforms to communicate with each other. DATA INTEGRATOR supports three XML web service technologies: SOAP, WSDL, and **XML Schema**.

work flow

A **reusable object** containing **steps** to define the order of **job** execution. Work flows call **data flow**s, but cannot manipulate data themselves.

You call work flows from inside other work flows or **job**s. You pass information into or out of work flows using **parameter**s. You reuse work flows by dragging existing work flows from the **object library**.

workspace

The window inside the **Designer** in which you define, display, and modify **objects**. The workspace for a **data flow** contains an area to build a **diagram** representing the data flow definition. The workspace for a **transform** contains an editor for modifying the transform options.

WSDL

Web Services Definition Language. Web services are self-contained, modular business process applications based on open Internet standards.

XML

Extensible Markup Language. This markup language is like HTML (Hypertext Markup Language) in that it specifies a standard with which you can define your own markup languages with their own sets of tags. XML allows you to define various tags with various rules, such as tags that represent business rules, tags that represent data description, or tags that represent data relationships.

XML Schema

The XML format used by Data Integrator to support message processing that includes **Web Services**. XML Schemas describe the data structure of an **XML** file or message. Data flows can read and write data to messages or files based on a specified XML Schema format. You can use the same XML Schema to describe multiple XML sources or targets. XML Schema properties include: Name, Description, Imported from, Root element name, and Namespace.

Index

Business Objects, Americas
3030 Orchard Parkway
San Jose, CA 95134
USA

info@businessobjects.com
www.businessobjects.com

Data Integrator™

Supplement for J.D. Edwards

Version 6.5

Document Number DI-65-0112-001

January 30, 2004

Contents

J.D. Edwards Interface

The Data Integrator J.D. Edwards (JDE) interface is a license-controlled feature. With this interface, you can use the Data Integrator Designer to:

- View modules and tables in a database instance under a specified J.D. Edwards environment
- Import metadata for tables into Data Integrator
- Create batch data flows that use J.D. Edwards tables as sources

The *Data Integrator Supplement for J.D. Edwards* contains these sections:

- More Data Integrator product documentation
- System requirements
- Datastores
- Reference information

More Data Integrator product documentation

Consult the *Data Integrator Getting Started Guide* for:

- An overview of Data Integrator products and architecture
- Data Integrator installation and configuration information
- A list of product documentation and a suggested reading path

After you install Data Integrator, you can view technical documentation from many locations. To view documentation in PDF format, you can:

- Select **Start** > **Programs** > **Data Integrator** ***version*** > **Data Integrator Documentation** and choose:
  - Release Notes
  - Release Summary
  - Technical Manuals
- Select one of the following from the Designer's **Help** menu:
  - Release Notes
  - Release Summary
  - Technical Manuals
- Select **Help** from the Data Integrator Administrator

You can also view and download PDF documentation by visiting Business Objects Customer Support online. To access this Web site, you must have a valid user name and password. To obtain your user name and password, go to http://www.techsupport.businessobjects.com and click **Register**.

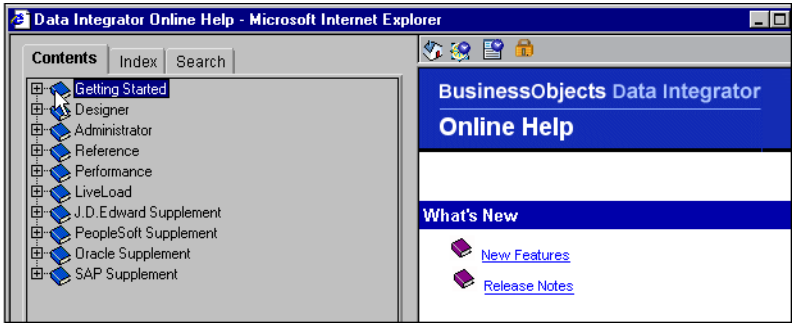
After installing Data Integrator Designer you can also view documentation in HTML format. **Online Help** opens automatically the first time you open the Designer.

You can also open **Help,** using one of the following methods:

- Choose **Contents** from the Designer's Help menu.
- Click objects in the object library or workspace and press **F1**.

  Online Help opens to the subject you selected.

Use **Online Help's** links and tool bar to navigate:



Online Help tool bar

Icon	Icon Name	Description
	Home	Returns to the Online Help home page.
	Show/Hide Navigation	Shows and hides the Table of Contents for the core suite of Data Integrator Technical Manuals. Includes a master **Index** and **Search** tabs.
	Show/Hide Descriptions	The Online Help window automatically updates its information when you click an object in the Designer window. In addition, you can click any object in the Designer window then click this button to show or hide an object's local and global variables.
	Start/Stop Help Updates	When the lock appears open, each time you click an object in the Designer window, the Online Help information automatically updates.
		When the lock icon appears closed, the Online Help window does not automatically update when you click objects in the Designer window; you must press F1.

System requirements

The J.D. Edwards interface allows you to connect Data Integrator with two J.D. Edwards applications:

- World
- OneWorld

Both of these applications operate on an underlying database. Install the drivers you need to connect J.D. Edwards with Data Integrator on the same computers on which you install the Data Integrator Designer and Job Server components. See "System requirements" on page 46 of the *Data Integrator Getting Started Guide* for general information about Windows and UNIX software and hardware requirements.

World

The J.D. Edwards interface supports World version A7.3 and later. Data Integrator only supports this application when World uses an underlying DB2 database on an AS/400 computer. Use either an ODBC datastore connection (using IBM's iSeries Access driver) or a Detail_DB2 datastore connection.

See "Defining a J.D. Edwards World datastore" on page 6.

OneWorld

The J.D. Edwards interface supports OneWorld version B7.3 and compatible versions, including Xe. Data Integrator supports this application if OneWorld uses one of three underlying databases:

- DB2 for AS/400 (use DETAIL_DB2 or an ODBC datastore connection using IBM's iSeries Access)
- Microsoft SQL Server
- Oracle

See "Defining a J.D. Edwards OneWorld datastore" on page 8.

J.D. Edwards OneWorld environment data sources include system-control data sources and business data sources. Your J.D. Edwards system-control data sources and your business data sources must reside on the same database instance. If a system control table is not located in the same database instance, you cannot create a Data Integrator datastore for that J.D. Edwards application.

Datastores

Data Integrator uses datastore connections to link with other applications or databases. In a design environment, you use datastores to browse, search, or import metadata that represents external tables, files, messages, and other database objects. When running jobs, Data Integrator uses datastore information to move data between source and target databases and applications.

After defining a J.D. Edwards datastore in Data Integrator, you can browse application modules, tables, table descriptions, and column descriptions.

You must define J.D. Edwards datastore connections with accurate J.D. Edwards information to ensure the accessibility of the tables.

This section discusses:

- Defining a J.D. Edwards World datastore
- Defining a J.D. Edwards OneWorld datastore
- Browsing and importing metadata
- Extracting data from J.D. Edwards systems

Defining a J.D. Edwards World datastore

You can use two methods to access the J.D. Edwards World application:

- ODBC driver
- Mainframe interfaces (DETAIL)

Both options access a DB2 for AS/400 database.

ODBC driver

You can access the J.D. Edwards World application using an ODBC driver called iSeries Access. Before creating a datastore using this ODBC driver, you must install and configure the driver on the same computers on which you installed the Data Integrator Designer and Job Server components. Use the ODBC Administration utility to install and configure the driver. In the driver's library list, be sure to list all the libraries needed in your particular J.D. Edwards environment.

Mainframe interfaces (DETAIL)

You can access the J.D. Edwards World application using Mainframe interfaces (DETAIL). Refer to the pertinent documentation for installation and setup instructions. The DETAIL application connects to AS/400 and MVS mainframe systems.

Before using the DETAIL_DB2 option with your J.D. Edwards datastore, you must:

- Define a node in DETAIL. A node identifies the computer where the database and the DETAIL Listener are located.
- Make sure that the DETAIL Listener is running. The DETAIL Listener is a background process that facilitates communication between Data Integrator and the AS/400 computer.

➢ To define a J.D. Edwards World datastore

1. In the object library of the Data Integrator Designer, go to the **Datastores** tab.
2. Right-click inside the object library window and select **New**.
3. In the Datastore Editor window, enter a name for this datastore (`DS_JDEWorld`, for example).
4. In the **Application type** list, select **JDE_World**.

5. In the **Database Type** list, select either **DETAIL_DB2** or **ODBC**.
6. Enter the connection information.

   See the "Custom application datastores" on page 35 of the *Data Integrator Reference Guide* for detailed information about the options for both DETAIL_DB2 and ODBC connections.
7. Click the **JDE World Properties** tab and enter the required information. Data Integrator handles J.D. Edwards data in a logical grouping of libraries.

Option	Description
Environment	Enter the J.D. Edwards application environment name.
Local library	Enter the name of the library where Data Integrator can find the J.D. Edwards system control table F0005.
Data dictionary library	Enter the name of the library where Data Integrator can find the J.D. Edwards data dictionary table F9201.
Security library	Enter the name of the library where Data Integrator can find J.D. Edwards security tables, such as F0094.

8. Click **OK**.

   The J.D. Edwards datastore appears in the object library.

Defining a J.D. Edwards OneWorld datastore

A J.D. Edwards environment uses a logical construct called a *data source* to associate the J.D. Edwards application layer data with the database layer data (for example, Microsoft SQL Server). J.D. Edwards uses some data sources to associate system-control data and uses other data sources to associate business data.

The distributed nature of J.D. Edwards architecture allows an instance of J.D Edwards to have data sources on multiple database servers. For example, a finance data source can be on a Microsoft SQL Server while the system-control data source is on Oracle. Because a Data Integrator datastore can only be associated with one database server, you must follow these rules when defining a J.D. Edwards OneWorld datastore connection in Data Integrator:

- The data sources within a J.D. Edwards environment that you want to access must be in the same Microsoft SQL Server, DB2 for AS/400, or Oracle instance.
- You must specify required system control data sources, and in some cases, you must enter the owner ID of the data source. For Microsoft SQL Server, you must also enter the database name for each, and for DB2 for AS/400, you must enter the library for each.
- If using Microsoft SQL Server, you must configure a new datastore for each business data source on a unique database even if all databases are on the same Microsoft SQL Server. For example, if you want to extract Human Resource and Finance information from separate databases on the same Microsoft SQL Server instance, you must create two different Data Integrator datastores.

➢ To define a J.D. Edwards OneWorld datastore

1. In the object library of the Data Integrator Designer, go to the **Datastores** tab.
2. Right-click inside the object library window and select **New**.
3. In the Datastore Editor window, enter a name for this datastore (`DS_JDEOneWorld`, for example).
4. In the **Application Type** list, select **JDE_One_World**.

5. In the **Database Type** list, select the database on which your J.D. Edwards application runs.

If you choose:	Do this:
DETAIL_DB2	Enter the DETAIL node and DB2 database name. See the "Custom application datastores" on page 35 of the *Data Integrator Reference Guide* for a complete description of DETAIL_DB2 datastore options. **NOTE**: With **DETAIL_DB2** as your database connection choice, you must configure DETAIL to point to a DB2 for AS/400 database.
Microsoft_SQL_Server	Enter the database-related information including the user name and password for the database. In the **Database server name** box, enter the name of the SQL Server instance. In the **Database name** box, enter the name of the database containing your business data, such as a finance database. If you have more than one database of business data create a separate J.D. Edwards datastore like the first except enter a different database name here. See "Custom application datastores" on page 35 of the *Data Integrator Reference Guide* for a complete description of Microsoft SQL Server datastore options. **NOTE**: With Microsoft SQL Server as your database, the Data Integrator Job Server must be installed on Windows NT or 2000.
Oracle	Enter the information required to connect to the Oracle database. See the "Custom application datastores" on page 35 of the *Data Integrator Reference Guide* for a complete description of Oracle datastore options.

6. Click the **JDE OneWorld Properties** tab and enter the required information: the environment name and the databases to which the data sources point.

Some J.D. Edwards data sources contain system-control information. Data Integrator needs this system-control information to properly translate J.D. Edwards data. Like all data sources, these system-control data sources can be anywhere on the Microsoft SQL Server, DB2 for AS/400, or Oracle instance. Enter the database names for each data source.

Option	Description
Environment	Enter the J.D. Edwards application environment name.
System data source	Enter the name of the database where the tables F986101, F98611, and F00941 are located. This option is available for DETAIL, DB2, and Microsoft SQL Server databases.
System data source owner	Enter the owner ID for the system data source. This option is available for Microsoft SQL Server and Oracle databases.
Object librarian data source	Enter the name of the database where the tables F9860 and F9861 are located. This option is available for DETAIL, DB2, and Microsoft SQL Server databases.
Local data source	Enter the name of the database where the table F0005 is located. This option is available for DETAIL, DB2, and Microsoft SQL Server databases.
Data dictionary data source	Enter the name of the database where the table F9203 is located. This option is available for DETAIL, DB2, and Microsoft SQL Server databases.

7. Click **OK**.

   The J.D. Edwards OneWorld datastore appears in the object library.

Browsing and importing metadata

After creating a J.D. Edwards datastore, you can browse and import the metadata from the connected application.

➢ To browse and import the metadata

1. View the modules in the J.D. Edwards application. You can:
   - Right-click the datastore name and select **Open**
   - Double-click the datastore name

   The workspace shows the list of application modules.
2. View the tables in a particular module. You can:
   - Expand the module tree
   - Double-click a module (folder)
3. To import table data, right-click a table name and select **Import**.

   NOTE: When you import a table into Data Integrator via the J.D. Edwards interface, Data Integrator does not preserve the table hierarchy. All tables are listed at the same level.

You can also import tables using the Designer **Import by Name** and **Search** features.

Extracting data from J.D. Edwards systems

When extracting data from either J.D. Edwards application, Data Integrator processes and converts data types appropriately. For example, Data Integrator recognizes null values for date columns. J.D. Edwards, on the other hand, does not support null values. Instead, J.D. Edwards stores dates as numeric values. When date data is not present, J.D. Edwards stores the number 0. Therefore, Data Integrator automatically translates a value of zero for a date to NULL. If you expect a column to have zeros in it, do not use the date type in the primary key.

Data Integrator automatically translates most data types. In some cases, you must translate output explicitly. See "Data types" on page 14 for more information about data type translation.

Reference information

This section contains additional J.D. Edwards-specific information to supplement information contained in the *Data Integrator Reference Guide*. For general conceptual and procedural information, see the *Data Integrator Designer Guide*. Find both books in *Data Integrator Technical Manuals*.

This section discusses:

- Data types
- Functions

Data types

Data Integrator uses unique processing to translate some J.D. Edwards data types:

- Translating decimals
- Translating dates
- Translating time

Translating decimals

J.D. Edwards translates decimal values using an additional piece of information from the underlying database: the decimal separator shift integer. This value indicates the number of digits to move left from the end of the value number. For example, to represent a number like 1.23, J.D. Edwards stores the value as "123" with a decimal separator shift of "2" (starting at the end of the number, the decimal shifts two places to the left).

To process J.D. Edwards decimal values, Data Integrator extracts the numeric value stored in the underlying database and then applies the decimal separator shift to determine where the decimal belongs. Data Integrator applies the translation when a decimal column appears in a SQL statement against the database.

Data Integrator interprets J.D. Edwards currency data types as decimals.

Data Integrator does not automatically translate decimal data types in three cases:

- SQL transform
- sql function
- pushdown_sql function

In these cases, translate decimals explicitly. To translate decimals explicitly, you must know the decimal shift value for your selected column. Use the shift number to determine the denominator for decimal translation. For example, if column `COL26` in table `JDETAB` is a JDE decimal column and it has a shift value of `2`, you would manually modify the SQL statement to say something like `SELECT COL26/100 FROM JDETAB`. The denominator in this statement has two zeros following the `1`. If the shift value was `3`, the statement would be `SELECT COL26/1000 FROM JDETAB`.

Translating dates

J.D. Edwards stores dates as numeric values. The J.D. Edwards date format is similar to Julian date format except the year starts with 1900.

Data Integrator handles J.D. Edwards numeric values by reading the date information and translating with an internal formula based on the J.D. Edwards start year. For example, Data Integrator would handle the date February 18, 1999 as follows:

```
Year = (1999 - 1900) * 1000;
```

```
Day = 31 + 18;
```
[31 is the # of days in January]

The resulting Day value is the number of days since 1/1/1999. The Julian date value is Year + Day = `99049`.

Like decimal translation, there are three cases when Data Integrator does not automatically translate dates:

- SQL transform
- sql function
- pushdown_sql function

In these cases, you must use the `JDE_Date` function. See "JDE_Date" on page 17 for a details.

Translating time

Data Integrator does not automatically interpret J.D. Edwards time data types. If you need to translate a number to its time value, you can use the `JDE_Time` function. See "JDE_Time" on page 18.

Functions

When conversion is not automatic, you can use Data Integrator functions to convert J.D. Edwards data to internal Data Integrator data types:

- JDE_Date
- JDE_Time

You can access these functions from the query editor: go to the **Mapping** or the **Where** tabs, and click the **Functions** button.

JDE_Date

Given a Julian date stored in a J.D. Edwards database, `JDE_Date` returns the equivalent value as a Data Integrator date data type.

SYNTAX

`JDE_Date`(*jde_julian_date*)

RETURN VALUE

date	The Data Integrator date data type equivalent to the specified Julian date.

WHERE

jde_julian_date	The integer column in the input table that contains a J.D. Edwards Julian date.

EXAMPLE

Function	Results
`JDE_Date (99049)`	`2/18/1999`

You can translate J.D. Edwards Julian dates using column mapping. When you input a J.D. Edwards Julian date number (for example, `99049`), a Data Integrator date type results (for example, `2/18/1999`). Enter the function in the **Mapping** section of the Query transform editor.

Use this function when Data Integrator does not automatically translate dates. See "Translating dates" on page 15.

NOTE: Because this function takes an integer as input, map the incoming column to an integer column before applying the function.

JDE_Time

Given a number representing time in J.D. Edwards, `JDE_Time` returns a Data Integrator character value that represents the equivalent military time in *`HH`*`24:`*`MI`*`:`*`SS`* format (where *`HH`* is hours, *`MI`* is minutes, and *`SS`* is seconds).

SYNTAX

`JDE_Time`(*time_integer*)

RETURN VALUE

Char(8)	The time in *`HH`*`24:`*`MI`*`:`*`SS`* format.

WHERE

time_integer	An integer column in the input table that represents a J.D. Edwards time.

EXAMPLE

Function	Results
`JDE_Time(92513)`	`09:25:13`

You can translate times from J.D. Edwards formatting into Data Integrator formatting using column mapping. For instance, if you input a J.D. Edwards record update time of 92513, the resulting Data Integrator time would be 9:25:13. Enter the function in the **Mapping** section of the Query transform editor.

A Glossary

This glossary defines terms used in this documentation set including common terms that have a special meaning with respect to Data Integrator. Within a definition, other terms that are defined in the glossary appear in **bold type**.

ABAP

Advanced Business Application Programming.
A fourth-generation programming language developed by SAP in which R/3 applications are written.

ABAP program

A program that executes database operations on an **SAP R/3** server. Data Integrator **R/3 data flows** generate ABAP programs.

Access Server

A process that starts, stops, and monitors real-time services (**real-time jobs**). The Access Server dispatches requests to real-time services, ensuring optimal load balancing and complete life cycle management. The Access Server also starts, stops, and monitors **adapters**.

adapter

An external Data Integrator **interface**. There are two types of adapters:

- Custom adapters — Adapters developed using the Data Integrator Adapter Development Kit
- Prepackaged adapters — Adapters prebuilt and purchased from Business Objects, such as the Data Integrator MQSeries Adapter

Administrator

A browser-based system administration application that supports Data Integrator. Use the Administrator to configure and manage:

- Services (**real-time jobs**)
- Service providers (instances of real-time services)
- Clients (Web application clients)
- Batch jobs
- Adapters
- Job logs

after-image

The values in an UPDATE row after the row changes. You might want to pair before- and after-images of UPDATE rows when designing log-based changed-data capture jobs which Data Integrator supports.

aggregate data

Data that results when a process combines elements. This data can be presented collectively or in summary form.

ALE

Application Link Enabling. An SAP R/3 programming-related interface designed to allow reliable communication across a distributed environment.

Analytic Rapid Mart

An **Data Integrator application** that produces a predefined **data warehouse** for a specific area of business analysis. Analytic Rapid Marts are typically based on dimensional modelling. With Analytic Rapid Marts, users can manage large volumes of data.

attribute

A **property** created for a type of **object**.

BAPI

Business Application Programming Interface. A standardized SAP R/3 programming interface that allows non-SAP R/3 applications to access specific business processes and data.

back-office application

A software system that an enterprise uses to run business operations, such as human resources, general ledger, and finance. Back-office applications store data in a relational **database** optimized for operational use.

There are several types of back-office applications:

- Enterprise resource planning (ERP) — For example, SAP R/3, BAAN, J.D. Edwards, PeopleSoft
- Supply chain management (SCM) — For example, i2
- Customer relationship management (CRM) — For example, Siebel, Clarify
- Legacy

Basis

The SAP infrastructure. Basis is the foundation for all SAP products.

batch job

A group of objects (data flows, work flows, conditionals, scripts, and so forth) that executes on a schedule. This type of job is recommended for operations on large amounts of data.

before-image

The values in an UPDATE row before the row changes. You might want to pair before- and after-images of UPDATE rows when designing log-based changed-data capture jobs which Data Integrator supports.

bulk loading

A software-based mechanism that moves large data files. This mechanism supports compression, blocking, and buffering to optimize transfer times.

business component

A set of tables Siebel applications use to create a logical object called a business object.

Business Objects Data Platform

A set of applications from Business Objects, Americas, that enterprises can use to manage all types of data—from data for supply chain partners to data for customers. The Business Objects Data Platform consists of **Data Integrator** and one or more **Rapid Marts**.

CDC datastore

A CDC datastore allows you to limit extracted data to changed data only. A CDC datastore connects a **changed-data capture** table on a source database to Data Integrator.

CDC checkpoint

A CDC checkpoint enables Data Integrator to restrict CDC subscription reads. After you enable a checkpoint, the next time the CDC job runs, it reads only the rows inserted into the CDC table since the last checkpoint.

CDC Subscription

A CDC subscription name allows different data flows to extract data from the same table without corrupting data extracted by other data flows. The datastore and table name can be the same; however, the table owner name must be different in each source CDC table.

changed-data capture

The process of retrieving changes made to a production data source. This process consolidates units of work, ensures data is synchronized with the original source, and reduces data volume in a warehouse environment.

Citrix MetaFrame XP

Citrix MetaFrame XP software provides an access infrastructure for enterprise applications. You can use this software to run Data Integrator on a server which publishes instances of the Designer and other Data Integrator interfaces to users on client computers.

client/server

A distributed technology approach where the processing is divided by function. The server performs shared functions (such as managing communications and providing database services), while the client performs individual user functions.

COM

Component Object Model. Provides a set of interfaces that allow clients and servers to communicate within the same computer (running on Windows 95 or later versions).

Commerce Rapid Mart

An **Data Integrator application** that produces a predefined **data warehouse** and that includes logic to support real-time requests for data from a **back-office application**. A Commerce Rapid Mart is usually based on different data model than an **Analytic Rapid Mart**. With a Commerce Rapid Mart, users can integrate a company's back-office system with an **e-commerce** application.

Communication Structure

In **SAP BW**, a data structure that defines a set of **InfoObject**s available from an **InfoSource** to put into **InfoCube**s.

conditional

A **single-use object**, available in **work flow**s**,** that allows you to branch the execution logic based on the results of an expression. The conditional takes the form of an if/then/else statement.

custom ABAP program

A custom ABAP program runs an **ABAP program** and generates a **data set**. With a custom ABAP program, you can run an existing ABAP program as part of a job. Use a custom ABAP program as a **source** in a **data flow** or an **R/3 data flow**.

data extraction

The process of moving data from a **database** or application source to a database target (either from a legacy database to a **data mart**, or from one data mart to another).

data flow

A **reusable object** containing **steps** to define the transformation of data from **source** to **target**. Data flows are called from inside a **work flow** or **job**. You can pass information into or out of data flows using **parameters**.

Data Integrator

A software system that allows users to build and execute applications with which they can create and maintain **data warehouses**.

Data Integrator consists of several components:

- **Access Server**
- **Data Integrator engine**
- **Data Integrator repository**
- **Data Integrator Service**
- **Administrator**
- **Designer**
- **Job Server**

Data Integrator application

A set of **jobs** that process batch data extractions and real-time requests using data from a variety of data sources.

Data Integrator engine

The core process that reads job information from the **Data Integrator repository** and sets up run-time processes that execute the job. The run-time processes extract, transform, and load relational and hierarchical data. The **Job Server** starts the Data Integrator engine to execute batch or real-time jobs.

Data Integrator interface

A program that Data Integrator uses to access data sources. Specific interfaces vary by installation. There are internal interfaces—those native to the installation—and external interfaces—those that you install separately. Internal interfaces allow Data Integrator to access applications like **SAP R/3** and **SAP BW**, **messages**, relational database systems, and **legacy systems**. An external interface is also known as an **adapter**. It allows Data Integrator to access applications using information exchange technologies such as Web Services and MQ Series.

Data Integrator repository

The database that contains information about an Data Integrator application. The repository contains information about defined **reusable objects**, the **metadata** for tables, and built-in **transforms** and **functions**. When you invoke Data Integrator, you log in to the repository containing the **objects** you want to use.

Data Integrator Service

The process that ensures that the **Access Server** and the **Job Server** are running. You can configure the Data Integrator Service to restart the Access Server and Job Server whenever the computer where they are located restarts.

data loading

The process of populating a **data warehouse**. Data loading is provided by **DBMS**-specific load processes, DBMS insert processes, and independent fast-load processes.

data mapping

The process of assigning a source data element to a target data element.

data mart

A highly-focused version of a **data warehouse**. Typically, created by a department or division of a company, data marts contain data for a specific subject area, such as finance or sales. Data Integrator can populate a data mart.

data transformation

The process of filtering, merging, decoding, and translating source data to create validated data for a **data warehouse**.

data set

Rows of data with a defined schema. A step in a **data flow**—such as reading data from a **source**, joining data in a **Query transform**, or transforming data though another **transform**—yields a data set. You can view individual data sets by placing a **target** table or file at that point in the **data flow**.

data transport

A step in an **R/3 data flow** that defines a **target** to store the **data set** extracted during the flow. You can locate the target file on the **SAP R/3** server or in a location accessible to both the SAP R/3 server and to Data Integrator across a network.

data type

The format used to store a value. Data types can imply a default format for displaying and entering the value. Data read from a **source** is converted to the appropriate Data Integrator data types; data loaded to a **target** is converted from its Data Integrator data type to the type appropriate for the target.

data warehouse

A relational **database** containing subject-oriented information that supports decision analysis. Typically, information in a data warehouse is integrated from multiple sources, including operational sources. Data warehouses provide a single, non-volatile point of reference for a particular time period. Data warehouses are composed of **data marts.** Data Integrator can populate a data warehouse.

database

A collection of **table**s managed by a **DBMS** such as Microsoft SQL Server or Oracle.

DataConnector

DataConnector operator instances are used to read data files generated by Data Integrator when performing bulk loading using the Teradata Warehouse Builder.

datastore

A logical channel connecting Data Integrator to a **database**. The datastore definition includes the name and location of the database as well as user authentication information. Data Integrator uses a datastore definition to qualify a table name wherever a table is indicated in a diagram or expression. You can access the datastore definition through the **object library**.

DBMS

Database management system. A software system that builds and maintains database tables.

debug mode

Allows you to debug a job using the **interactive debugger** features in the Designer.

declarative specification

A way of indicating the desired results of a **data transformation** without placing constraints on the method of reaching the results. When you create a **data flow** in Data Integrator, you specify what data you want to read from which **source**, what you want done to that data, and where you want it loaded. What you specify constitutes the declarative specification. Data Integrator then determines the most efficient way to reach the requirements you set, including pushing operations to the source **DBMS** for execution.

degree of parallelism

You can set a value at the data flow level that allows you to parallel-process a **transform** within a data flow.

For example, if from the Properties window of a data flow you set the **Degree of parallelism** to 5, then when the job executes, Data Integrator replicates each transform in the data flow five times. Each of these replicated transforms executes in parallel using a separate process thread. Using a value for degree of parallelism can improve job performance if a **Job Server**'s computer memory and number of CPUs supports a job's parallel-processing configuration settings.

dependency analysis

A category of metadata reports that produces reports based on search criteria you specify to find specific objects in the repository. For each object returned, dependency analysis reports provide object lineage as well as object impact information.

derived data

Data that results from a computational step applied to reference or event data. Derived data is the result of either relating two or more elements of a single transaction (such as aggregation), or relating one or more elements of a transaction to an external algorithm or rule.

Designer

A graphical user interface that allows you to design and test an **Data Integrator application**.

diagram

The icons and connections between the icons that make up the definition of a **job**, **work flow**, or **data flow**.

DMZ

Demilitarized zone. A perimeter network that is more accessible than your most protected network, but located inside a security layer protected from the internet.

drill down

A method of exploring detailed data that was used in creating a summary level of data. Drill down levels depend on the granularity of the data in the **data warehouse**.

DTD

Document type definition. A text file that describes the elements (tags) in an **XML** document and the relationship among them. When an XML document is used to describe a transaction, the DTD describes the data schema used in the transaction.

e-commerce

Buy and sell transactions conducted over the internet.

EDI

Electronic Data Interchange. Electronic exchange of structured data between businesses. This exchange is not dependent on hardware, software, or communication protocols.

element

A component found within XML Schemas and DTDs.

embedded data flow

A data flow created within one or more data flows.

enterprise data

Data that is defined for use across a corporate environment.

ERP system

Enterprise resource planning system. A **back-office application** from which Data Integrator can extract data.

exception

An error that occurs while executing a **job**. You can catch individual or groups of exceptions using a **try/catch block** inside a **work flow**. Catching an exception allows you to automatically execute a solution for the error.

Executive Information Systems (EIS)

Tools programmed to provide canned reports or briefing books to executives. They offer strong reporting and drill-down capabilities. These tools allow ad hoc querying against a multi-dimensional **database**, and most offer analytical applications along functional lines such as sales or financial analysis.

extract date

The date data was extracted.

extract frequency

The interval at which data is extracted, such as daily, weekly, monthly, or quarterly. The frequency that data extracts are needed in the **data warehouse** is determined by the shortest frequency requested through an order, or by the frequency required to maintain consistency of the other associated data types in the source data warehouse.

file

A **source** or **target** for data. To read or load data to a file within Data Integrator, define a **file format** that instructs Data Integrator about the structure of the data in the file.

file format

A description of how data is or should be organized in a **file** Data Integrator reads from or loads to. A file format can be specific to a single file or generic for many files.

front-office application

Software that companies use to support reporting, **e-commerce**, and other tasks. Types of front-office applications include:

- Supply chain collaboration
- Analytical
- Sell-side e-commerce
- Customer relationship management

function

A program that operates on values that are passed to it. Data Integrator functions are available through a function wizard in a **script**, **conditional**, or **Query transform**. Data Integrator also gives you access to functions provided by the **DBMS** you are using. In addition, you can define your own functions using the Data Integrator scripting language or by writing C-language programs.

IDoc

Intermediate Document. An SAP-specific format. Used for **EDI** and **ALE**.

IDoc type

Indicates the SAP format that is used to interpret the data of a business transaction. Consists of the following components:

- A control record

  Identical for each IDoc type.

- Several data records

  A single data record consists of a fixed key part and a variable data part. The data part is interpreted using segments, which differ depending on the IDoc type selected.

- Several status records

  Identical for each IDoc type. Describe the status states an IDoc has already passed through or the status an IDoc has attained.

import

The process of acquiring information for the **Data Integrator repository**. Import the following kinds of information into Data Integrator:

- The **metadata** for **source** and **target** databases
- Descriptions and code for user-defined and DBMS **functions** and **transforms**

InfoArea

In **SAP BW**, an arbitrary grouping of **InfoCube**s.

InfoCube

In **SAP BW**, a multidimensional star schema that is limited to one fact table and 16 dimensions.

InfoCubes store only numeric data at the fact level.

SAP BW reserves three of the dimensions for currencies, time, and units of measure. Dimensions have to be linked back to master data and hierarchy tables to gain access to the information stored there.

When a user query is initiated, BW creates a temporary **OLAP** cube that stores a constrained subset of the permanent OLAP cube. The temporary InfoCube persists until the user disconnects from BW or changes the query.

InfoObject

In **SAP BW**, an InfoObject is equivalent to a data element and its associated metadata (for example, description or business rules). Can be a **Key Figure** or a Characteristic.

InfoPackage

In **SAP BW**, a schedule for running BW updates.

Information resource (IR)

A producer or consumer of enterprise information. An information resource can be a flat file, a **back-office application**, an EAI system, or a Web application.

InfoSource

In **SAP BW**, a logical construct that defines a set of **InfoObject**s and their mappings to source data elements. InfoSources do not store data, but rather serve as conduits via **Transfer Structure**s.

Integration Rapid Mart

An **Data Integrator application** that provides point-to-point content mapping and data movement mechanisms between different **back-office applications**.

interactive debugger

A Designer feature that allows you to step through the data of a job one row at a time using filters and breakpoints on a line. Like executing a job, you can start the interactive debugger from the Debug menu when a job is active in the workspace. While in debug mode, all other Designer features are set to read-only.

interface

Data Integrator offers two types of interfaces:

An internal **Data Integrator interface**, allows you to create datastore connections to natively supported applications.

An external **Data Integrator interface** (or **adapter**), allows Data Integrator to communicate with information exchange technologies such as Web Services and MQ Series.

IR

See **Information resource (IR)**.

job

The unit of work that can be scheduled independently for execution by the **Administrator**. Jobs are special **work flows** that can be called by other work flows or jobs.

Job Server

A process that receives requests from the **Designer** and the **Administrator** to start and stop **jobs**. To start batch or real-time jobs, the Job Server triggers the **Data Integrator engine**. Engine processes run on the same computer as the Job Server process that triggers them.

join rank

A value given to or calculated for all data sets in a **data flow**. Data Integrator uses the join rank to determine which **source** to read first when assembling the **data set** in a join. Data Integrator uses the source with the lower join rank as the inner source of the join and uses the source with the higher join rank as the outer source of the join. If more than one of the sources is located in the same **DBMS**, the highest rank is used for all of the sources from that DBMS. When a source is processed through a **Query transform** or another **transform**, and then used in a join, the maximum join rank defined for the source is used as the join rank in the query.

Key Figure

In **SAP BW**, an **InfoObject** that represents a numeric fact.

legacy system

An information or transaction processing system used to store data such as bank balances, inventories, payroll, and manufacturing parts.

license-controlled feature

A Data Integrator feature that is enabled or disabled based on the product license. The product license controls which icons and settings are available in Data Integrator as an internal **Data Integrator interface**.

line

The line you use to connect objects in a workspace diagram. Lines represent a left-to-right flow path for data moving between two objects during job execution.

memory datastore

A datastore connection/container for memory tables.

memory table

Internal Data Integrator table used to store a data set in memory while a job runs. Use instead of staging tables to improve performance of a real-time job built with multiple data flows. Use a memory table to move a data set between data flows.

message

Represents hierarchical data, such as a header with line items, for document-oriented transactions, such as a purchase order.

metadata

In Data Integrator, information acquired and maintained to describe tables in **source** and **target** databases. This information includes the names of tables and their columns, and the data types of the columns. All the metadata for all user tables is imported at one time for a given **datastore**.

In general, metadata typically includes a description of data models, a description of the layouts used in database design, the definition of the system of record, the mapping of data from the system of record to other places in the environment, and specific database design definitions.

multiple data flow model

One of three data flow models you can use in real-time jobs. Use this model to develop real-time jobs in separate logical object segments (data flows) rather than in a single data flow. Allows for any number of data flows in a real-time processing loop under these conditions: The first data flow must have a single message source, the last data flow must have a single message target. Other data flows in the same real-time job cannot have a message source or message target. See also **put/acknowledge data flow model** and **single data flow model**.

nested data

Data in one **table** that is related to a single row of another table. A nested table appears in Data Integrator as a column in a parent table. Columns in the nested table can themselves contain tables.

object

Any item that you create in the **Designer**. Data Integrator distinguishes two classes of objects: **reusable objects** that are complete and can be reused in your projects (such as **data flows**) and **single-use objects** that only appear as components of other objects (such as a **try/catch block**). This distinction affects how you create and retrieve each type of object.

object definition

The options that describe the operation of an **object**. To view and modify an object definition, open the object so that its definition appears in the **workspace**.

object library

A tool in the **Designer** that gives you access to reusable objects.

ODBC

Open database connectivity. A standard for **database** access initiated by the SQL Access Group consortium and now owned by Microsoft.

ODS

Operational data store. An OLAP-designed relational **database** that an enterprise has designated as the operational database of record (for example, a finance department might use an ODS to close its books).

OLAP

Online analytical processing. A relational **database** design that is optimized for random access. OLAP systems are used in a query environment, such as for a **front-office application**.

OLTP

Online transaction processing. A relational database design optimized for operational use. OLTP systems are used in an operational environment, such as for a **back-office application**.

operation code

A flag associated with a row in a **data set** that indicates the status of the data in the row. The operation codes are INSERT, UPDATE, DELETE, and NORMAL.

operational statistics

A category of reports within the metadata reporting tool that provides you with job and data flow performance information over a given period of time or number of executions.

parameter

A value passed to a **work flow** or **data flow** when that flow is called.

partition

To divide table data into sets based on a criteria such as a range or list of values in each row. You can configure Data Integrator to read and write partitioned table data in parallel threads. Designing jobs with partitioned table data can improve job performance if a Job Server's computer memory and number of CPUs supports the job's parallel-processing configuration settings.

project

The collection of **jobs** available in the **Designer** at a given time. A project provides a way to organize the **objects** you create.

property

Detailed descriptive information about **objects**.

put/acknowledge data flow model

One of three data flow models you can use in real-time jobs. Similar to the multiple data flow model, this model does not require a data flow with a target message. Data Integrator creates a target message to automatically acknowledge a request message. Designed for use with IDoc sources in SAP R/3 systems. See also **multiple data flow model** and **single data flow model**.

Query transform

A **data transformation object** that you can use to map columns from a **source** to a **target** schema, add new columns to the target schema, determine the data to extract, and perform operations on the data. Like a SQL SELECT statement, a query creates a **data set** that satisfies the conditions you specify.

Rapid Mart

A prepackaged **Data Integrator application** that delivers data for a specific business purpose. Easily deployed and customized, BusinessObjects™ Rapid Marts include the extraction, transformation, and data loading logic to support analysis or **e-commerce**. There are three types of Rapid Mart:

- **Analytic Rapid Mart**
- **Commerce Rapid Mart**
- **Integration Rapid Mart**

R/3 data flow

A **data flow** that extracts data from an **SAP R/3** source table. Data Integrator translates steps you define in an R/3 data flow into **ABAP** and then passes the ABAP program back to your SAP R/3 system for execution. The resulting **table** or **file** resides on the SAP R/3 system to be used as a **source** in the parent data flow.

real-time job

A group of objects (data flows, work flows, conditionals, scripts, and so forth) that execute on-demand as a "request-response" system. You design real-time jobs in the **Designer**, then configure them as real-time services and associate them with an **Access Server** in the **Administrator**, where they are started, managed and monitored. When a real-time service receives a request from a caller, it processes the request and returns a reply. This type of job is recommended for operations on small amounts of data.

relational data

A **data set** in which data in each column contains a scalar value. Data Integrator can process relational data; it can also process **nested data**.

repository

See **Data Integrator repository**.

reusable object

An **object** that can be defined, stored, and reused independent of other objects. Create reusable objects from the **object library**.

RFC server

Remote Function Call server. The Data Integrator RFC server allows third-party programs, including SAP R/3, to schedule and initiate Data Integrator jobs and return the results to Data Integrator.

SAP BW

Business Information Warehouse, an SAP product.

SAP R/3

An **ERP system**.

script

A step in a **work flow** that allows you to calculate values to pass to other parts of the work flow. The script can call **function**s, execute if-then-else statements, and assign values to **variable**s. Write a script in the Data Integrator scripting language.

segment

Format with which the data records of **IDocs** are interpreted.

server group

A defined collection of Job Servers on different computers. A server group automatically measures resource availability on each Job Server in the group and distributes scheduled batch jobs to the Job Server with the lightest load at run-time. Use the Server Groups node in the Administrator's navigation tree to group Job Servers that are associated with the same repository into a server group.

service request

Any message sent from a Web client that requires processing by a **real-time job**.

session

Associated with the **Adapter Properties** tab in an adapter datastore, a session is created by a stream-oriented adapter to group adapter components and maintain state for interactivity between Data Integrator and the adapter.

single data flow model

One of three data flow models you can use in real-time jobs. Allows for one data flow in a real-time processing loop. This data flow must have a single message source and a single message target. See also **multiple data flow model** and **put/acknowledge data flow model**.

single-use object

A step in a **work flow** or **data flow** that cannot be saved independently of the flow. Create single-use objects from the tool palette.

smart editor

A flexible editing tool in Data Integrator used for creating scripts, expressions, and custom functions without having to type the names of existing elements like column, function, and variable names.

SNMP

System Network Management Protocol helps network administrators manage network routing hardware. The protocol can manage a variety of hardware and software devices.

source

An **object** from which Data Integrator reads data.

SQL

Structured Query Language. A query language for accessing relational, **ODBC**, DRDA, or non-relational **database** systems.

SQL query tool

An end-user tool that accepts **SQL** to be processed against one or more relational databases.

star schema

A **database** design you can use to format data in a **data mart**. This design is based on a single fact table to which any number of dimensional tables may be joined. This type of database design supports multi-dimensional database analysis.

step

An **object** that is part of the definition of a **work flow** or **data flow**. Each step is represented by an icon in the diagram of the flow and is connected to other steps to indicate the flow of data through the data flow or the order of execution in the work flow.

table

A **database** table that Data Integrator reads data from or loads data into. The path and mechanisms for reading and loading data and apportioning the data among rows and columns are defined in the **datastore** that the table is associated with. Writing a **data set** to a database table means sending a combination of rows with appropriate **operation code** to the database table.

target

An object in which Data Integrator loads extracted and transformed data in a **data flow**. Data Integrator only loads rows flagged as INSERT or UPDATE.

TCP/IP

Transmission Control Protocol/Internet Protocol. The basic communication protocol of the internet, and often intranets and extranets. A computer having direct access to the internet contains a copy of the TCP/IP program. TCP/IP makes it possible for computers to communicate with each other.

Tdpid

(Teradata Director Program ID) The server name Data Integrator uses when loading with the bulk loader option. Data Integrator uses tdpid as a Teradata Warehouse Builder operator attribute.

Transfer Structure

In **SAP BW**, a data structure that defines a set of **InfoObjects** to be moved into BW. To an ETL tool like Data Integrator, a Transfer Structure looks like a table.

transform

A step in a **data flow** that acts on a **data set**. Data Integrator transforms are available through the **object library**. You can also import user-defined transforms and transforms from your **DBMS**.

Transformation Rule (Transfer Rule)

In **SAP BW**, a data manipulation rule between a **Transfer Structure** and a **Communication Structure**. The results of these transformation are globally available to all **InfoCubes**.

try/catch block

A combination of a try object and one or more catch objects that define alternate execution paths in case an error occurs during the execution of a **job**. You can tune try/catch blocks to trap specific errors and to provide general alerts or messages if an error occurs.

Update Rule

In **SAP BW**, a data manipulation rule between a **Communication Structure** and an **InfoCube**. The result of the data manipulation is specific to one InfoCube.

variable

A symbolic placeholder for a value. Data Integrator lets you define local variables and global variables.

Local variables pertain to the **work flow** in which they are defined. You can pass the value into another **work flow** or **data flow** using a **parameter**.

Global variables pertain to the **job** in which they are defined. With global variables, there is no need to define parameters between **objects** in the job. Global variables can also be selected at execution time. This eliminates the need to open the Designer to set global variable values.

Web Services

A standard platform for integrating applications. Web Services allow different programs, constructed in different languages, on different platforms to communicate with each other. DATA INTEGRATOR supports three XML web service technologies: SOAP, WSDL, and **XML Schema**.

work flow

A **reusable object** containing **steps** to define the order of **job** execution. Work flows call **data flow**s, but cannot manipulate data themselves.

You call work flows from inside other work flows or **job**s. You pass information into or out of work flows using **parameter**s. You reuse work flows by dragging existing work flows from the **object library**.

workspace

The window inside the **Designer** in which you define, display, and modify **objects**. The workspace for a **data flow** contains an area to build a **diagram** representing the data flow definition. The workspace for a **transform** contains an editor for modifying the transform options.

WSDL

Web Services Definition Language. Web services are self-contained, modular business process applications based on open Internet standards.

XML

Extensible Markup Language. This markup language is like HTML (Hypertext Markup Language) in that it specifies a standard with which you can define your own markup languages with their own sets of tags. XML allows you to define various tags with various rules, such as tags that represent business rules, tags that represent data description, or tags that represent data relationships.

XML Schema

The XML format used by Data Integrator to support message processing that includes **Web Services**. XML Schemas describe the data structure of an **XML** file or message. Data flows can read and write data to messages or files based on a specified XML Schema format. You can use the same XML Schema to describe multiple XML sources or targets. XML Schema properties include: Name, Description, Imported from, Root element name, and Namespace.

Index

Business Objects, Americas
3030 Orchard Parkway
San Jose, CA 95134
USA

info@businessobjects.com
www.businessobjects.com

Data Integrator

Supplement for Oracle Applications

Version 6.5

Document Number DI-65-0116-001

January 30, 2004

Contents

1 Oracle Applications Interface

The Data Integrator Oracle Applications interface allows you to create Oracle Application datastores and import tables for use as sources in Data Integrator jobs. With this interface, you can use the Data Integrator Designer to:

- View objects such as tables, functions, and procedures in a database instance under a specified Oracle Applications environment.
- Import metadata for Oracle Applications objects into Data Integrator.
- Create data flows that use Oracle Applications objects (such as tables) as sources and targets.

This guide contains the following sections:

More Data Integrator product documentation

Consult the *Data Integrator Getting Started Guide* for:

- An overview of Data Integrator products and architecture
- Data Integrator installation and configuration information
- A list of product documentation and a suggested reading path

After you install Data Integrator, you can view technical documentation from many locations. To view documentation in PDF format, you can:

- Select **Start** > **Programs** > **Data Integrator** ***version*** > **Data Integrator Documentation** and choose:
  - Release Notes
  - Release Summary
  - Technical Manuals
- Select one of the following from the Designer's **Help** menu:
  - Release Notes
  - Release Summary
  - Technical Manuals
- Select **Help** from the Data Integrator Administrator

You can also view and download PDF documentation by visiting Business Objects Customer Support online. To access this Web site, you must have a valid user name and password. To obtain your user name and password, go to http://www.techsupport.businessobjects.com and click **Register**.

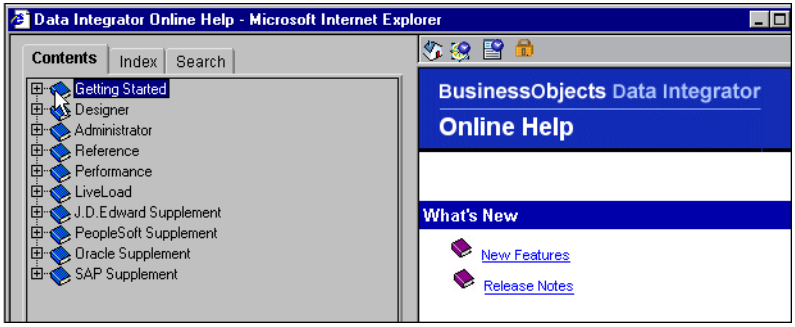
After installing Data Integrator Designer you can also view documentation in HTML format. **Online Help** opens automatically the first time you open the Designer.

You can also open **Help,** using one of the following methods:

- Choose **Contents** from the Designer's Help menu.
- Click objects in the object library or workspace and press **F1**.

  Online Help opens to the subject you selected.

Use **Online Help's** links and tool bar to navigate:



Online Help tool bar

Icon	Icon Name	Description
	Home	Returns to the Online Help home page.
	Show/Hide Navigation	Shows and hides the Table of Contents for the core suite of Data Integrator Technical Manuals. Includes a master **Index** and **Search** tabs.
	Show/Hide Descriptions	The Online Help window automatically updates its information when you click an object in the Designer window. In addition, you can click any object in the Designer window then click this button to show or hide an object's local and global variables.
	Start/Stop Help Updates	When the lock appears open, each time you click an object in the Designer window, the Online Help information automatically updates.
		When the lock icon appears closed, the Online Help window does not automatically update when you click objects in the Designer window; you must press F1.

System requirements

The Oracle Applications interface allows you to connect Data Integrator with Oracle Applications for importing metadata.

These applications operate on an underlying Oracle database. Install the drivers you need to connect Oracle with Data Integrator on the same computers on which you install the Data Integrator Designer and Job Server components. See "System requirements" on page 46 of the *Data Integrator Getting Started Guide* for general information about Windows and UNIX software and hardware requirements.

The Oracle Applications interface supports version 11.5 and later versions.

The Oracle Applications interface can be installed using the Data Integrator installer. The interface is license-controlled. For more information about licence-controlled interfaces for Data Integrator, see "Optional license-controlled features" on page 63 of the *Data Integrator Getting Started Guide*.

Because Data Integrator connects to Oracle through a database connection, Data Integrator can see, import, and use all Oracle Applications objects that the provided Oracle database user (such as APPS) can access without requiring application user login security information.

The Oracle Applications interface uses Oracle Application object library tables (which use the prefix `FND_%`) to get metadata information. For more information about Oracle Application metadata, see your Oracle Application Object Library/Workflow Technical Reference Manual.

Datastores

Data Integrator uses datastore connections to link with other applications or databases. In a design environment, you use datastores to browse, search, or import metadata that represents external tables, files, messages, and other database objects. When running jobs, Data Integrator uses datastore information to move data between source and target databases and applications.

After defining an Oracle Applications datastore in Data Integrator, you can browse application modules, tables, table descriptions, and column descriptions. While stored procedures and functions are also displayed, they are not grouped by application because they are not application-specific.

You must define Oracle Applications datastore connections with accurate Oracle Applications information to ensure the accessibility of the tables.

This section discusses:

- Defining an Oracle Applications datastore
- Browsing and importing metadata
- Extracting data from Oracle Applications systems

Defining an Oracle Applications datastore

With an Oracle Applications datastore connection, you can browse Oracle Applications metadata.

➢ To define an Oracle Applications datastore

1. In the object library of the Data Integrator Designer, go to the **Datastores** tab
2. Right-click inside the object library window and select **New**.
3. In the Datastore Editor window, enter a name for this datastore (`DS_OracleApps`, for example).

4. In the **Application type** list, select **Oracle_Applications**.
5. Enter the connection information.

   See "Custom application datastores" on page 35 of the *Data Integrator Reference Guide* for Oracle connection options and descriptions.
6. Click **OK**.

   The Oracle Applications datastore appears in the object library.

Browsing and importing metadata

After creating an Oracle Applications datastore, you can browse and import the metadata from the connected application.

NOTE: While you cannot browse packages (there are too many), you can import them by name.

➢ To browse and import the metadata

1. View the modules in the Oracle Application. You can:
   - Right-click the datastore name and select **Open**, or
   - Double-click the datastore name

   The workspace displays the list of application modules.
2. View the tables in a particular module. You can:
   - Expand the module tree, or
   - Double-click a module (folder)
3. To import table data, right-click a table name and select **Import**.

   NOTE: When you import a table into Data Integrator via the Oracle Applications interface, Data Integrator does not preserver the table hierarchy. All tables are listed at the same level.

You can also import tables using the Designer's **Import by Name** feature.

Extracting data from Oracle Applications systems

When extracting data from Oracle Applications, Data Integrator processes and converts data types appropriately. For data type conversion details, see "Data type processing" on page 207 of the *Data Integrator Reference Guide*.

Flexfields

Data Integrator processes both **descriptive flexfields** and **key flexfields** from Oracle Applications if they are in use. Furthermore, Data Integrator can distinguish between context-specific and non-context-specific flexfields. The meaning of a context-specific flexfield is based on the value of another (context or structure) field.

Columns created from flexfields have column names based on the physical database name or end-user column name for that flexfield.

For consistency, when Data Integrator bases a column name on an end-user column name, it will translate the end-user column name from lowercase letters to uppercase, replacing all non-alphanumeric characters with an underscore.

If the physical name of a flexfield is descriptive (like `TAX_CODE_FLAG`), Data Integrator displays that physical name as the column name in a schema. However, if the physical name is generic (for example, `ATTRIBUTE1` or `SEGMENT1`) and associated with a descriptive end-user column name, Data Integrator truncates the end-user column name to 50 characters, appends it with an alphanumeric suffix, and uses it as the column name.

Data Integrator appends column names created from descriptive flexfield end-user column names with a `_D#` while column names created from key flexfields are appended with a `_K#` where # is a unique number. For example, if the physical name of a descriptive flexfield is `ATTRIBUTE1` and the end-user column name is `Status code`, the logical name might display as `STATUS_CODE_D6`.

When a descriptive flexfield's end-user column name is context-specific, representing multiple names, the first end-user column name (with suffix identifier) appears as the column name and a note appears in the key flexfield's column description, structured as follows:

<first end-user column name> (multi-flex column - multiple structures based on *<content field COLUMN NAME>* = "*<content value>*" - *<field comment>*)

When a key flexfield's end-user column name is structure-specific, representing multiple names, the first end-user column name (with suffix identifier) appears as the column name and a note appears in the key flexfield's column description, structured as follows:

<*first end-user column name*> (multi-flex column - multiple structures based on ID_FLEX_CODE= "*<structure code value>*"ID_FLEX_NUM- *<column name>* = *<structure number>*-*<field comment>*)

The following table shows how flexfield column names for both descriptive and key flexfields are based on physical name and end user column name:

Descriptive Flexfields

From Oracle Applications		To Data Integrator	
Physical name:	**End-user column name:**	**Column name:**	**Business name:**
STAT_CODE	N/A	STAT_CODE	STAT_CODE
ATTRIBUTE1	Status code	STATUS_CODE_D1	Status code
ATTRIBUTE1	-(None)	ATTRIBUTE1	-(None)

Key Flexfields

From Oracle Applications		To Data Integrator	
Physical name:	**End-user column name:**	**Column name:**	**Business name:**
STAT_CODE	N/A	STAT_CODE	STAT_CODE
SEGMENT1	Status code	STATUS_CODE_K1	Status code
SEGMENT1	-(None)	SEGMENT1	-(None)

NOTE: Using flexfields with SQL function calls is not supported and causes an invalid column name error. Instead, use physical column names in functions that pass commands directly to the database such as SQL and PUSHDOWN_SQL.

Refer to the *Oracle Applications Flexfields Guide* to learn more about flexfields.

A Glossary

This glossary defines terms used in this documentation set including common terms that have a special meaning with respect to Data Integrator. Within a definition, other terms that are defined in the glossary appear in **bold type**.

ABAP

Advanced Business Application Programming.
A fourth-generation programming language developed by SAP in which R/3 applications are written.

ABAP program

A program that executes database operations on an **SAP R/3** server. Data Integrator **R/3 data flows** generate ABAP programs.

Access Server

A process that starts, stops, and monitors real-time services (**real-time jobs**). The Access Server dispatches requests to real-time services, ensuring optimal load balancing and complete life cycle management. The Access Server also starts, stops, and monitors **adapters**.

adapter

An external Data Integrator **interface**. There are two types of adapters:

- Custom adapters — Adapters developed using the Data Integrator Adapter Development Kit
- Prepackaged adapters — Adapters prebuilt and purchased from Business Objects, such as the Data Integrator MQSeries Adapter

Administrator

A browser-based system administration application that supports Data Integrator. Use the Administrator to configure and manage:

- Services (**real-time jobs**)
- Service providers (instances of real-time services)
- Clients (Web application clients)
- Batch jobs
- Adapters
- Job logs

after-image

The values in an UPDATE row after the row changes. You might want to pair before- and after-images of UPDATE rows when designing log-based changed-data capture jobs which Data Integrator supports.

aggregate data

Data that results when a process combines elements. This data can be presented collectively or in summary form.

ALE

Application Link Enabling. An SAP R/3 programming-related interface designed to allow reliable communication across a distributed environment.

Analytic Rapid Mart

An **Data Integrator application** that produces a predefined **data warehouse** for a specific area of business analysis. Analytic Rapid Marts are typically based on dimensional modelling. With Analytic Rapid Marts, users can manage large volumes of data.

attribute

A **property** created for a type of **object**.

BAPI

Business Application Programming Interface. A standardized SAP R/3 programming interface that allows non-SAP R/3 applications to access specific business processes and data.

back-office application

A software system that an enterprise uses to run business operations, such as human resources, general ledger, and finance. Back-office applications store data in a relational **database** optimized for operational use.

There are several types of back-office applications:

- Enterprise resource planning (ERP) — For example, SAP R/3, BAAN, J.D. Edwards, PeopleSoft
- Supply chain management (SCM) — For example, i2
- Customer relationship management (CRM) — For example, Siebel, Clarify
- Legacy

Basis

The SAP infrastructure. Basis is the foundation for all SAP products.

batch job

A group of objects (data flows, work flows, conditionals, scripts, and so forth) that executes on a schedule. This type of job is recommended for operations on large amounts of data.

before-image

The values in an UPDATE row before the row changes. You might want to pair before- and after-images of UPDATE rows when designing log-based changed-data capture jobs which Data Integrator supports.

bulk loading

A software-based mechanism that moves large data files. This mechanism supports compression, blocking, and buffering to optimize transfer times.

business component

A set of tables Siebel applications use to create a logical object called a business object.

Business Objects Data Platform

A set of applications from Business Objects, Americas, that enterprises can use to manage all types of data—from data for supply chain partners to data for customers. The Business Objects Data Platform consists of **Data Integrator** and one or more **Rapid Marts**.

CDC datastore

A CDC datastore allows you to limit extracted data to changed data only. A CDC datastore connects a **changed-data capture** table on a source database to Data Integrator.

CDC checkpoint

A CDC checkpoint enables Data Integrator to restrict CDC subscription reads. After you enable a checkpoint, the next time the CDC job runs, it reads only the rows inserted into the CDC table since the last checkpoint.

CDC Subscription

A CDC subscription name allows different data flows to extract data from the same table without corrupting data extracted by other data flows. The datastore and table name can be the same; however, the table owner name must be different in each source CDC table.

changed-data capture

The process of retrieving changes made to a production data source. This process consolidates units of work, ensures data is synchronized with the original source, and reduces data volume in a warehouse environment.

Citrix MetaFrame XP

Citrix MetaFrame XP software provides an access infrastructure for enterprise applications. You can use this software to run Data Integrator on a server which publishes instances of the Designer and other Data Integrator interfaces to users on client computers.

client/server

A distributed technology approach where the processing is divided by function. The server performs shared functions (such as managing communications and providing database services), while the client performs individual user functions.

COM

Component Object Model. Provides a set of interfaces that allow clients and servers to communicate within the same computer (running on Windows 95 or later versions).

Commerce Rapid Mart

An **Data Integrator application** that produces a predefined **data warehouse** and that includes logic to support real-time requests for data from a **back-office application**. A Commerce Rapid Mart is usually based on different data model than an **Analytic Rapid Mart**. With a Commerce Rapid Mart, users can integrate a company's back-office system with an **e-commerce** application.

Communication Structure

In **SAP BW**, a data structure that defines a set of **InfoObject**s available from an **InfoSource** to put into **InfoCube**s.

conditional

A **single-use object**, available in **work flow**s**,** that allows you to branch the execution logic based on the results of an expression. The conditional takes the form of an if/then/else statement.

custom ABAP program

A custom ABAP program runs an **ABAP program** and generates a **data set**. With a custom ABAP program, you can run an existing ABAP program as part of a job. Use a custom ABAP program as a **source** in a **data flow** or an **R/3 data flow**.

data extraction

The process of moving data from a **database** or application source to a database target (either from a legacy database to a **data mart**, or from one data mart to another).

data flow

A **reusable object** containing **steps** to define the transformation of data from **source** to **target**. Data flows are called from inside a **work flow** or **job**. You can pass information into or out of data flows using **parameters**.

Data Integrator

A software system that allows users to build and execute applications with which they can create and maintain **data warehouses**.

Data Integrator consists of several components:

- **Access Server**
- **Data Integrator engine**
- **Data Integrator repository**
- **Data Integrator Service**
- **Administrator**
- **Designer**
- **Job Server**

Data Integrator application

A set of **jobs** that process batch data extractions and real-time requests using data from a variety of data sources.

Data Integrator engine

The core process that reads job information from the **Data Integrator repository** and sets up run-time processes that execute the job. The run-time processes extract, transform, and load relational and hierarchical data. The **Job Server** starts the Data Integrator engine to execute batch or real-time jobs.

Data Integrator interface

A program that Data Integrator uses to access data sources. Specific interfaces vary by installation. There are internal interfaces—those native to the installation—and external interfaces—those that you install separately. Internal interfaces allow Data Integrator to access **SAP R/3**, **SAP BW**, **messages**, relational database systems, and **legacy systems**. An external interface is also known as an **adapter**.

Data Integrator repository

The database that contains information about an Data Integrator application. The repository contains information about defined **reusable objects**, the **metadata** for tables, and built-in **transforms** and **functions**. When you invoke Data Integrator, you log in to the repository containing the **objects** you want to use.

Data Integrator Service

The process that ensures that the **Access Server** and the **Job Server** are running. You can configure the Data Integrator Service to restart the Access Server and Job Server whenever the computer where they are located restarts.

data loading

The process of populating a **data warehouse**. Data loading is provided by **DBMS**-specific load processes, DBMS insert processes, and independent fast-load processes.

data mapping

The process of assigning a source data element to a target data element.

data mart

A highly-focused version of a **data warehouse**. Typically, created by a department or division of a company, data marts contain data for a specific subject area, such as finance or sales. Data Integrator can populate a data mart.

data transformation

The process of filtering, merging, decoding, and translating source data to create validated data for a **data warehouse**.

data set

Rows of data with a defined schema. A step in a **data flow**—such as reading data from a **source**, joining data in a **Query transform**, or transforming data though another **transform**—yields a data set. You can view individual data sets by placing a **target** table or file at that point in the **data flow**.

data transport

A step in an **R/3 data flow** that defines a **target** to store the **data set** extracted during the flow. You can locate the target file on the **SAP R/3** server or in a location accessible to both the SAP R/3 server and to Data Integrator across a network.

data type

The format used to store a value. Data types can imply a default format for displaying and entering the value. Data read from a **source** is converted to the appropriate Data Integrator data types; data loaded to a **target** is converted from its Data Integrator data type to the type appropriate for the target.

data warehouse

A relational **database** containing subject-oriented information that supports decision analysis. Typically, information in a data warehouse is integrated from multiple sources, including operational sources. Data warehouses provide a single, non-volatile point of reference for a particular time period. Data warehouses are composed of **data marts.** Data Integrator can populate a data warehouse.

database

A collection of **table**s managed by a **DBMS** such as Microsoft SQL Server or Oracle.

DataConnector

DataConnector operator instances are used to read data files generated by Data Integrator when performing bulk loading using the Teradata Warehouse Builder.

datastore

A logical channel connecting Data Integrator to a **database**. The datastore definition includes the name and location of the database as well as user authentication information. Data Integrator uses a datastore definition to qualify a table name wherever a table is indicated in a diagram or expression. You can access the datastore definition through the **object library**.

DBMS

Database management system. A software system that builds and maintains database tables.

debug mode

Allows you to debug a job using the **interactive debugger** features in the Designer.

declarative specification

A way of indicating the desired results of a **data transformation** without placing constraints on the method of reaching the results. When you create a **data flow** in Data Integrator, you specify what data you want to read from which **source**, what you want done to that data, and where you want it loaded. What you specify constitutes the declarative specification. Data Integrator then determines the most efficient way to reach the requirements you set, including pushing operations to the source **DBMS** for execution.

degree of parallelism

You can set a value at the data flow level that allows you to parallel-process a **transform** within a data flow.

For example, if from the Properties window of a data flow you set the **Degree of parallelism** to 5, then when the job executes, Data Integrator replicates each transform in the data flow five times. Each of these replicated transforms executes in parallel using a separate process thread. Using a value for degree of parallelism can improve job performance if a **Job Server**'s computer memory and number of CPUs supports a job's parallel-processing configuration settings.

dependency analysis

A category of metadata reports that produces reports based on search criteria you specify to find specific objects in the repository. For each object returned, dependency analysis reports provide object lineage as well as object impact information.

derived data

Data that results from a computational step applied to reference or event data. Derived data is the result of either relating two or more elements of a single transaction (such as aggregation), or relating one or more elements of a transaction to an external algorithm or rule.

Designer

A graphical user interface that allows you to design and test an **Data Integrator application**.

diagram

The icons and connections between the icons that make up the definition of a **job**, **work flow**, or **data flow**.

DMZ

Demilitarized zone. A perimeter network that is more accessible than your most protected network, but located inside a security layer protected from the internet.

drill down

A method of exploring detailed data that was used in creating a summary level of data. Drill down levels depend on the granularity of the data in the **data warehouse**.

DTD

Document type definition. A text file that describes the elements (tags) in an **XML** document and the relationship among them. When an XML document is used to describe a transaction, the DTD describes the data schema used in the transaction.

e-commerce

Buy and sell transactions conducted over the internet.

EDI

Electronic Data Interchange. Electronic exchange of structured data between businesses. This exchange is not dependent on hardware, software, or communication protocols.

element

A component found within XML Schemas and DTDs.

embedded data flow

A data flow created within one or more data flows.

enterprise data

Data that is defined for use across a corporate environment.

ERP system

Enterprise resource planning system. A **back-office application** from which Data Integrator can extract data.

exception

An error that occurs while executing a **job**. You can catch individual or groups of exceptions using a **try/catch block** inside a **work flow**. Catching an exception allows you to automatically execute a solution for the error.

Executive Information Systems (EIS)

Tools programmed to provide canned reports or briefing books to executives. They offer strong reporting and drill-down capabilities. These tools allow ad hoc querying against a multi-dimensional **database**, and most offer analytical applications along functional lines such as sales or financial analysis.

extract date

The date data was extracted.

extract frequency

The interval at which data is extracted, such as daily, weekly, monthly, or quarterly. The frequency that data extracts are needed in the **data warehouse** is determined by the shortest frequency requested through an order, or by the frequency required to maintain consistency of the other associated data types in the source data warehouse.

file

A **source** or **target** for data. To read or load data to a file within Data Integrator, define a **file format** that instructs Data Integrator about the structure of the data in the file.

file format

A description of how data is or should be organized in a **file** Data Integrator reads from or loads to. A file format can be specific to a single file or generic for many files.

front-office application

Software that companies use to support reporting, **e-commerce**, and other tasks. Types of front-office applications include:

- Supply chain collaboration
- Analytical
- Sell-side e-commerce
- Customer relationship management

function

A program that operates on values that are passed to it. Data Integrator functions are available through a function wizard in a **script**, **conditional**, or **Query transform**. Data Integrator also gives you access to functions provided by the **DBMS** you are using. In addition, you can define your own functions using the Data Integrator scripting language or by writing C-language programs.

IDoc

Intermediate Document. An SAP-specific format. Used for **EDI** and **ALE**.

IDoc type

Indicates the SAP format that is used to interpret the data of a business transaction. Consists of the following components:

- A control record

  Identical for each IDoc type.

- Several data records

  A single data record consists of a fixed key part and a variable data part. The data part is interpreted using segments, which differ depending on the IDoc type selected.

- Several status records

  Identical for each IDoc type. Describe the status states an IDoc has already passed through or the status an IDoc has attained.

import

The process of acquiring information for the **Data Integrator repository**. Import the following kinds of information into Data Integrator:

- The **metadata** for **source** and **target** databases
- Descriptions and code for user-defined and DBMS **functions** and **transforms**

InfoArea

In **SAP BW**, an arbitrary grouping of **InfoCube**s.

InfoCube

In **SAP BW**, a multidimensional star schema that is limited to one fact table and 16 dimensions.

InfoCubes store only numeric data at the fact level.

SAP BW reserves three of the dimensions for currencies, time, and units of measure. Dimensions have to be linked back to master data and hierarchy tables to gain access to the information stored there.

When a user query is initiated, BW creates a temporary **OLAP** cube that stores a constrained subset of the permanent OLAP cube. The temporary InfoCube persists until the user disconnects from BW or changes the query.

InfoObject

In **SAP BW**, an InfoObject is equivalent to a data element and its associated metadata (for example, description or business rules). Can be a **Key Figure** or a Characteristic.

InfoPackage

In **SAP BW**, a schedule for running BW updates.

Information resource (IR)

A producer or consumer of enterprise information. An information resource can be a flat file, a **back-office application**, an EAI system, or a Web application.

InfoSource

In **SAP BW**, a logical construct that defines a set of **InfoObject**s and their mappings to source data elements. InfoSources do not store data, but rather serve as conduits via **Transfer Structure**s.

Integration Rapid Mart

An **Data Integrator application** that provides point-to-point content mapping and data movement mechanisms between different **back-office applications**.

interactive debugger

A Designer feature that allows you to step through the data of a job one row at a time using filters and breakpoints on a line. Like executing a job, you can start the interactive debugger from the Debug menu when a job is active in the workspace. While in debug mode, all other Designer features are set to read-only.

interface

Data Integrator offers two types of interfaces:

An internal **Data Integrator interface**, allows you to create datastore connections to natively supported applications.

An external **Data Integrator interface** (or **adapter**), allows Data Integrator to communicate with information exchange technologies such as Web Services and MQ Series.

IR

See **Information resource (IR)**.

job

The unit of work that can be scheduled independently for execution by the **Administrator**. Jobs are special **work flows** that can be called by other work flows or jobs.

Job Server

A process that receives requests from the **Designer** and the **Administrator** to start and stop **jobs**. To start batch or real-time jobs, the Job Server triggers the **Data Integrator engine**. Engine processes run on the same computer as the Job Server process that triggers them.

join rank

A value given to or calculated for all data sets in a **data flow**. Data Integrator uses the join rank to determine which **source** to read first when assembling the **data set** in a join. Data Integrator uses the source with the lower join rank as the inner source of the join and uses the source with the higher join rank as the outer source of the join. If more than one of the sources is located in the same **DBMS**, the highest rank is used for all of the sources from that DBMS. When a source is processed through a **Query transform** or another **transform**, and then used in a join, the maximum join rank defined for the source is used as the join rank in the query.

Key Figure

In **SAP BW**, an **InfoObject** that represents a numeric fact.

legacy system

An information or transaction processing system used to store data such as bank balances, inventories, payroll, and manufacturing parts.

license-controlled feature

A Data Integrator feature that is enabled or disabled based on the product license. The product license controls which icons and settings are available in the **Designer** as an internal **Data Integrator interface**. License-controlled features include:

- J.D. Edwards interface
- Oracle applications
- PeopleSoft interface
- SAP R/3 ABAP interface
- SAP R/3 BAPI interface
- SAP R/3 IDoc interface
- SAP BW Source
- SAP BW Target
- Siebel applications
- LiveLoad Connector
- Multi-user development support

line

The line you use to connect objects in a workspace diagram. Lines represent a left-to-right flow path for data moving between two objects during job execution.

memory datastore

A datastore connection/container for memory tables.

memory table

Internal Data Integrator table used to store a data set in memory while a job runs. Use instead of staging tables to improve performance of a real-time job built with multiple data flows. Use a memory table to move a data set between data flows.

message

Represents hierarchical data, such as a header with line items, for document-oriented transactions, such as a purchase order.

metadata

In Data Integrator, information acquired and maintained to describe tables in **source** and **target** databases. This information includes the names of tables and their columns, and the data types of the columns. All the metadata for all user tables is imported at one time for a given **datastore**.

In general, metadata typically includes a description of data models, a description of the layouts used in database design, the definition of the system of record, the mapping of data from the system of record to other places in the environment, and specific database design definitions.

multiple data flow model

One of three data flow models you can use in real-time jobs. Use this model to develop real-time jobs in separate logical object segments (data flows) rather than in a single data flow. Allows for any number of data flows in a real-time processing loop under these conditions: The first data flow must have a single message source, the last data flow must have a single message target. Other data flows in the same real-time job cannot have a message source or message target. See also **put/acknowledge data flow model** and **single data flow model**.

nested data

Data in one **table** that is related to a single row of another table. A nested table appears in Data Integrator as a column in a parent table. Columns in the nested table can themselves contain tables.

object

Any item that you create in the **Designer**. Data Integrator distinguishes two classes of objects: **reusable objects** that are complete and can be reused in your projects (such as **data flows**) and **single-use objects** that only appear as components of other objects (such as a **try/catch block**). This distinction affects how you create and retrieve each type of object.

object definition

The options that describe the operation of an **object**. To view and modify an object definition, open the object so that its definition appears in the **workspace**.

object library

A tool in the **Designer** that gives you access to reusable objects.

ODBC

Open database connectivity. A standard for **database** access initiated by the SQL Access Group consortium and now owned by Microsoft.

ODS

Operational data store. An OLAP-designed relational **database** that an enterprise has designated as the operational database of record (for example, a finance department might use an ODS to close its books).

OLAP

Online analytical processing. A relational **database** design that is optimized for random access. OLAP systems are used in a query environment, such as for a **front-office application**.

OLTP

Online transaction processing. A relational database design optimized for operational use. OLTP systems are used in an operational environment, such as for a **back-office application**.

operation code

A flag associated with a row in a **data set** that indicates the status of the data in the row. The operation codes are INSERT, UPDATE, DELETE, and NORMAL.

operational statistics

A category of reports within the metadata reporting tool that provides you with job and data flow performance information over a given period of time or number of executions.

parameter

A value passed to a **work flow** or **data flow** when that flow is called.

partition

To divide table data into sets based on a criteria such as a range or list of values in each row. You can configure Data Integrator to read and write partitioned table data in parallel threads. Designing jobs with partitioned table data can improve job performance if a Job Server's computer memory and number of CPUs supports the job's parallel-processing configuration settings.

project

The collection of **job**s available in the **Designer** at a given time. A project provides a way to organize the **objects** you create.

property

Detailed descriptive information about **objects**.

put/acknowledge data flow model

One of three data flow models you can use in real-time jobs. Similar to the multiple data flow model, this model does not require a data flow with a target message. Data Integrator creates a target message to automatically acknowledge a request message. Designed for use with IDoc sources in SAP R/3 systems. See also **multiple data flow model** and **single data flow model**.

Query transform

A **data transformation object** that you can use to map columns from a **source** to a **target** schema, add new columns to the target schema, determine the data to extract, and perform operations on the data. Like a SQL SELECT statement, a query creates a **data set** that satisfies the conditions you specify.

Rapid Mart

A prepackaged **Data Integrator application** that delivers data for a specific business purpose. Easily deployed and customized, BusinessObjects™ Rapid Marts include the extraction, transformation, and data loading logic to support analysis or **e-commerce**. There are three types of Rapid Mart:

- **Analytic Rapid Mart**
- **Commerce Rapid Mart**
- **Integration Rapid Mart**

R/3 data flow

A **data flow** that extracts data from an **SAP R/3** source table. Data Integrator translates steps you define in an R/3 data flow into **ABAP** and then passes the ABAP program back to your SAP R/3 system for execution. The resulting **table** or **file** resides on the SAP R/3 system to be used as a **source** in the parent data flow.

real-time job

A group of objects (data flows, work flows, conditionals, scripts, and so forth) that execute on-demand as a "request-response" system. You design real-time jobs in the **Designer**, then configure them as real-time services and associate them with an **Access Server** in the **Administrator**, where they are started, managed and monitored. When a real-time service receives a request from a caller, it processes the request and returns a reply. This type of job is recommended for operations on small amounts of data.

relational data

A **data set** in which data in each column contains a scalar value. Data Integrator can process relational data; it can also process **nested data**.

repository

See **Data Integrator repository**.

reusable object

An **object** that can be defined, stored, and reused independent of other objects. Create reusable objects from the **object library**.

RFC server

Remote Function Call server. The Data Integrator RFC server allows third-party programs, including SAP R/3, to schedule and initiate Data Integrator jobs and return the results to Data Integrator.

SAP BW

Business Information Warehouse, an SAP product.

SAP R/3

An **ERP system**.

script

A step in a **work flow** that allows you to calculate values to pass to other parts of the work flow. The script can call **function**s, execute if-then-else statements, and assign values to **variable**s. Write a script in the Data Integrator scripting language.

segment

Format with which the data records of **IDocs** are interpreted.

server group

A defined collection of Job Servers on different computers. A server group automatically measures resource availability on each Job Server in the group and distributes scheduled batch jobs to the Job Server with the lightest load at run-time. Use the Server Groups node in the Administrator's navigation tree to group Job Servers that are associated with the same repository into a server group.

service request

Any message sent from a Web client that requires processing by a **real-time job**.

session

Associated with the **Adapter Properties** tab in an adapter datastore, a session is created by a stream-oriented adapter to group adapter components and maintain state for interactivity between Data Integrator and the adapter.

single data flow model

One of three data flow models you can use in real-time jobs. Allows for one data flow in a real-time processing loop. This data flow must have a single message source and a single message target. See also **multiple data flow model** and **put/acknowledge data flow model**.

single-use object

A step in a **work flow** or **data flow** that cannot be saved independently of the flow. Create single-use objects from the tool palette.

smart editor

A flexible editing tool in Data Integrator used for creating scripts, expressions, and custom functions without having to type the names of existing elements like column, function, and variable names.

SNMP

System Network Management Protocol helps network administrators manage network routing hardware. The protocol can manage a variety of hardware and software devices.

source

An **object** from which Data Integrator reads data.

SQL

Structured Query Language. A query language for accessing relational, **ODBC**, DRDA, or non-relational **database** systems.

SQL query tool

An end-user tool that accepts **SQL** to be processed against one or more relational databases.

star schema

A **database** design you can use to format data in a **data mart**. This design is based on a single fact table to which any number of dimensional tables may be joined. This type of database design supports multi-dimensional database analysis.

step

An **object** that is part of the definition of a **work flow** or **data flow**. Each step is represented by an icon in the diagram of the flow and is connected to other steps to indicate the flow of data through the data flow or the order of execution in the work flow.

table

A **database** table that Data Integrator reads data from or loads data into. The path and mechanisms for reading and loading data and apportioning the data among rows and columns are defined in the **datastore** that the table is associated with. Writing a **data set** to a database table means sending a combination of rows with appropriate **operation code** to the database table.

target

An object in which Data Integrator loads extracted and transformed data in a **data flow**. Data Integrator only loads rows flagged as INSERT or UPDATE.

TCP/IP

Transmission Control Protocol/Internet Protocol. The basic communication protocol of the internet, and often intranets and extranets. A computer having direct access to the internet contains a copy of the TCP/IP program. TCP/IP makes it possible for computers to communicate with each other.

Tdpid

(Teradata Director Program ID) The server name Data Integrator uses when loading with the bulk loader option. Data Integrator uses tdpid as a Teradata Warehouse Builder operator attribute.

Transfer Structure

In **SAP BW**, a data structure that defines a set of **InfoObject**s to be moved into BW. To an ETL tool like Data Integrator, a Transfer Structure looks like a table.

transform

A step in a **data flow** that acts on a **data set**. Data Integrator transforms are available through the **object library**. You can also import user-defined transforms and transforms from your **DBMS**.

Transformation Rule (Transfer Rule)

In **SAP BW**, a data manipulation rule between a **Transfer Structure** and a **Communication Structure**. The results of these transformation are globally available to all **InfoCube**s.

try/catch block

A combination of a try object and one or more catch objects that define alternate execution paths in case an error occurs during the execution of a **job**. You can tune try/catch blocks to trap specific errors and to provide general alerts or messages if an error occurs.

Update Rule

In **SAP BW**, a data manipulation rule between a **Communication Structure** and an **InfoCube**. The result of the data manipulation is specific to one InfoCube.

variable

A symbolic placeholder for a value. Data Integrator lets you define local variables and global variables.

Local variables pertain to the **work flow** in which they are defined. You can pass the value into another **work flow** or **data flow** using a **parameter**.

Global variables pertain to the **job** in which they are defined. With global variables, there is no need to define parameters between **objects** in the job. Global variables can also be selected at execution time. This eliminates the need to open the Designer to set global variable values.

Web Services

A standard platform for integrating applications. Web Services allow different programs, constructed in different languages, on different platforms to communicate with each other. DATA INTEGRATOR supports three XML web service technologies: SOAP, WSDL, and **XML Schema**.

work flow

A **reusable object** containing **steps** to define the order of **job** execution. Work flows call **data flows**, but cannot manipulate data themselves.

You call work flows from inside other work flows or **jobs**. You pass information into or out of work flows using **parameters**. You reuse work flows by dragging existing work flows from the **object library**.

workspace

The window inside the **Designer** in which you define, display, and modify **objects**. The workspace for a **data flow** contains an area to build a **diagram** representing the data flow definition. The workspace for a **transform** contains an editor for modifying the transform options.

WSDL

Web Services Definition Language. Web services are self-contained, modular business process applications based on open Internet standards.

XML

Extensible Markup Language. This markup language is like HTML (Hypertext Markup Language) in that it specifies a standard with which you can define your own markup languages with their own sets of tags. XML allows you to define various tags with various rules, such as tags that represent business rules, tags that represent data description, or tags that represent data relationships.

XML Schema

The XML format used by Data Integrator to support message processing that includes **Web Services**. XML Schemas describe the data structure of an **XML** file or message. Data flows can read and write data to messages or files based on a specified XML Schema format. You can use the same XML Schema to describe multiple XML sources or targets. XML Schema properties include: Name, Description, Imported from, Root element name, and Namespace.

Index

Business Objects, Americas
3030 Orchard Parkway
San Jose, CA 95134
USA

info@businessobjects.com
www.businessobjects.com

Data Integrator™

Supplement for PeopleSoft

Version 6.5

Document Number DI-65-0113-001

January 30, 2004

Contents

1 Welcome

Welcome to the *Data Integrator Supplement for PeopleSoft*. This guide contains information about how to use Data Integrator with PeopleSoft.

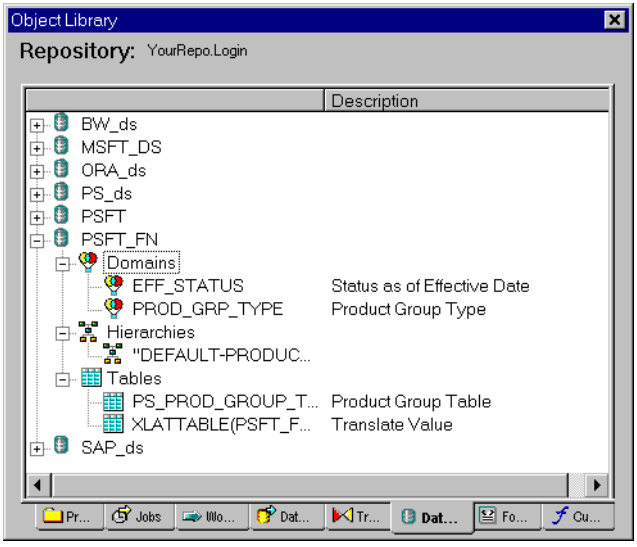
This chapter contains the following sections:

Using Data Integrator in a PeopleSoft environment

If you install the PeopleSoft interface, you can use PeopleSoft HRMS and ERP application data as a data source in Data Integrator.

Through the PeopleSoft datastore you define in Data Integrator, you can navigate PeopleSoft metadata external to Data Integrator (using standard PeopleSoft panels and menus) and import metadata for PeopleSoft source tables into the Data Integrator repository.

The following diagram shows PeopleSoft tables, hierarchies, and domains that have been imported into Data Integrator.



In addition, capabilities in Data Integrator that support PeopleSoft include the following:

- You can view valid domain values for a column from within the query transform.

  This allows you to filter on specific domain values for a column.

- Data Integrator can determine whether a given value for a column is valid within the domain of the column.

  Data Integrator flags those that are not valid.

- Data Integrator allows extraction of data based on effective dates without requiring you to write a secondary query. This functionality is particularly useful in implementing data warehouses containing HR data—it significantly reduces the complexity and increases the performance of extracting effective-dated data.

- Data Integrator allows extraction of the hierarchical data in PeopleSoft trees.

About this document

This document describes how Data Integrator interacts with PeopleSoft in the following chapters:

- Chapter 2, "PeopleSoft Datastores" — Describes how to define a PeopleSoft datastore so that you can use PeopleSoft as a data source in Data Integrator
- Chapter 3, "PeopleSoft Data Flows" — Describes information unique to data flows that extract, transform, and load PeopleSoft data
- Chapter 4, "Reference Information" — Describes Data Integrator objects specific to PeopleSoft and Data Integrator objects with supplemental information for the PeopleSoft interface

More Data Integrator product documentation

Consult the *Data Integrator Getting Started Guide* for:

- An overview of Data Integrator products and architecture
- Data Integrator installation and configuration information
- A list of product documentation and a suggested reading path

After you install Data Integrator, you can view technical documentation from many locations. To view documentation in PDF format, you can:

- Select **Start** > **Programs** > **Data Integrator** ***version*** > **Data Integrator Documentation** and choose:
  - Release Notes
  - Release Summary
  - Technical Manuals
- Select one of the following from the Designer's **Help** menu:
  - Release Notes
  - Release Summary
  - Technical Manuals
- Select **Help** from the Data Integrator Administrator

You can also view and download PDF documentation by visiting Business Objects Customer Support online. To access this Web site, you must have a valid user name and password. To obtain your user name and password, go to http://www.techsupport.businessobjects.com and click **Register**.

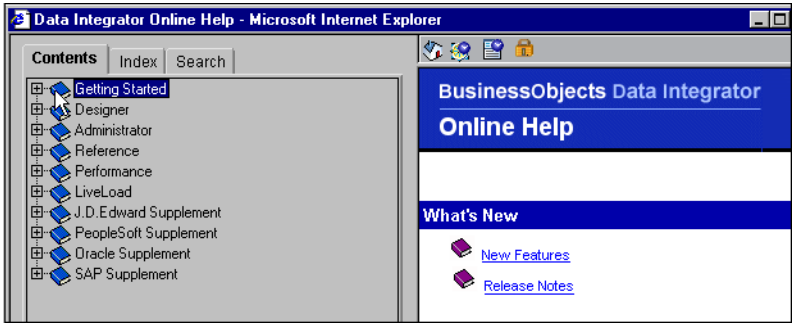
After installing Data Integrator Designer you can also view documentation in HTML format. **Online Help** opens automatically the first time you open the Designer.

You can also open **Help,** using one of the following methods:

- Choose **Contents** from the Designer's Help menu.
- Click objects in the object library or workspace and press **F1**.

  Online Help opens to the subject you selected.

Use **Online Help's** links and tool bar to navigate:



Online Help tool bar

Icon	Icon Name	Description
	Home	Returns to the Online Help home page.
	Show/Hide Navigation	Shows and hides the Table of Contents for the core suite of Data Integrator Technical Manuals. Includes a master **Index** and **Search** tabs.
	Show/Hide Descriptions	The Online Help window automatically updates its information when you click an object in the Designer window. In addition, you can click any object in the Designer window then click this button to show or hide an object's local and global variables.
	Start/Stop Help Updates	When the lock appears open, each time you click an object in the Designer window, the Online Help information automatically updates.
		When the lock icon appears closed, the Online Help window does not automatically update when you click objects in the Designer window; you must press F1.

2 PeopleSoft Datastores

With the PeopleSoft interface, you can use a PeopleSoft system as a Data Integrator source. To use a PeopleSoft data source, you must:

- Define a Data Integrator datastore that will serve as the logical link to your PeopleSoft system
- Import your PeopleSoft metadata into the Data Integrator datastore

This chapter describes the steps to complete these tasks. This chapter contains the following sections:

For generic information about Data Integrator datastores, see "Datastores" on page 77 of the *Data Integrator Designer Guide*.

Defining PeopleSoft datastores

You must define a PeopleSoft datastore in Data Integrator when you are extracting data from or loading data to PeopleSoft.

➢ To define a PeopleSoft datastore

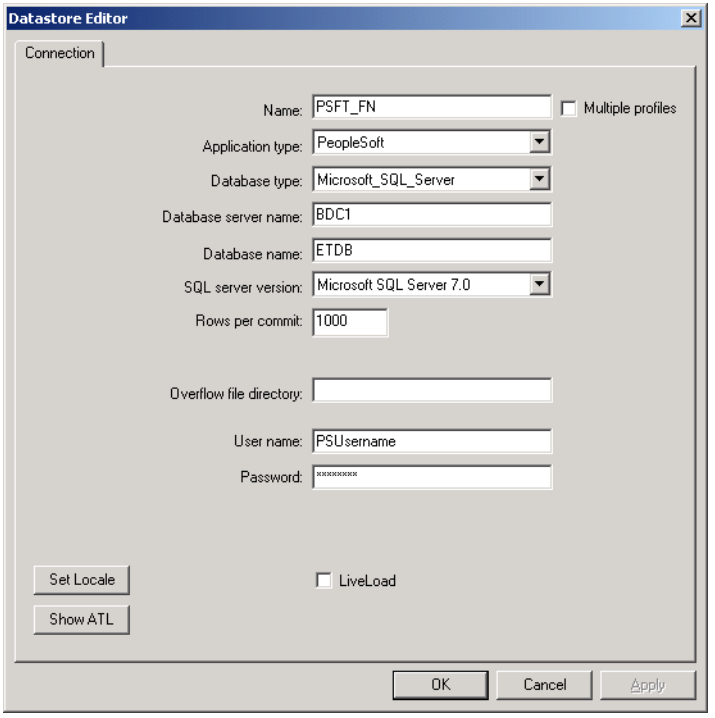
1. Go to the **Datastores** tab of the object library.
2. Right-click inside the object library window and choose **New**.

   The Datastore Administrator window appears.
3. Enter a name for the new datastore in the **Name** box.

   You can call the datastore anything you like, and you can change the name later if necessary. The name can contain any alpha or numeric characters or underscores (_). It cannot contain spaces.
4. In the **Application type** box, choose **PeopleSoft**.
5. In the **Database type** box, choose the appropriate database.

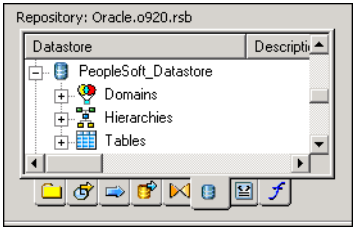
6. Enter the appropriate information for the database type you selected.

   For detailed information about the options, see "Datastore" on page 34 of the *Data Integrator Reference Guide*.



7. Click **OK**.

   Data Integrator creates the datastore and it appears in the object library window.



Browsing PeopleSoft metadata

After you create the datastore, you can access the PeopleSoft metadata via the Data Integrator metadata browser. Information is displayed in a tree format.

The information displayed consists of metadata related directly to database data—no information about calculation fields, images, buttons, or other GUI-related items is included.

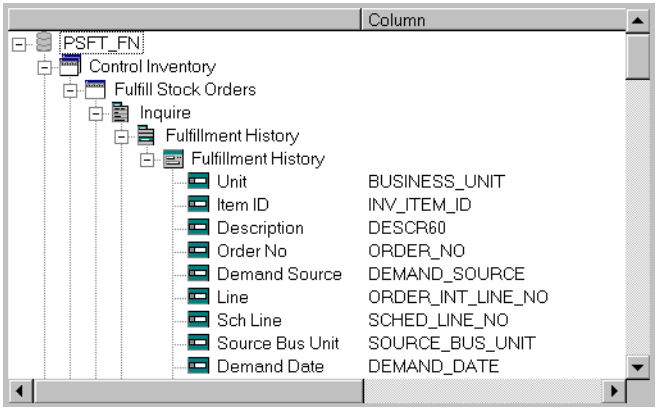
Browsing PeopleSoft data in Data Integrator is a lot like browsing in PeopleTools—icons in the display represent PeopleSoft menu groups, menus, menu bars, menu items, panels, and panel fields. Each panel field displays the caption, table column, and table name.

Icon	Metadata type	Description
	Menu Group	The entire set of commands available in PeopleSoft applications for a specific database.
	Menu	A set of commands for a specific application.
	Menu Bar	The top level of the menu.
	Menu Item	The commands that make up each menu bar.
	Panel	A vehicle for capturing and displaying data. Each panel can have multiple sub-panels, each represented by the same icon.
	Panel Fields	Represents a column on a table or in a specific view.

➢ To browse PeopleSoft metadata

1. In the object library, go to the **Datastores** tab.
2. Right-click the PeopleSoft datastore name and select **Open**.

   The metadata browser window opens.



Importing PeopleSoft metadata

To access PeopleSoft metadata from Data Integrator, you must import the metadata into the object library.

You can import tables, PeopleSoft trees (called *hierarchies* in Data Integrator), and domains.

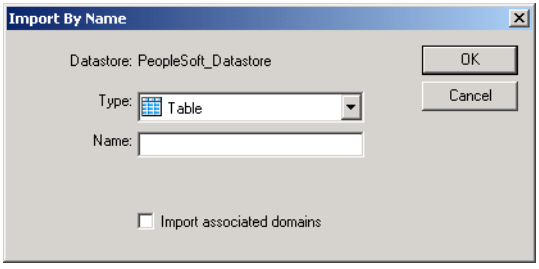
You can import PeopleSoft metadata in one of three ways:

- By name
- By browsing
- By searching

➢ To import PeopleSoft metadata by name

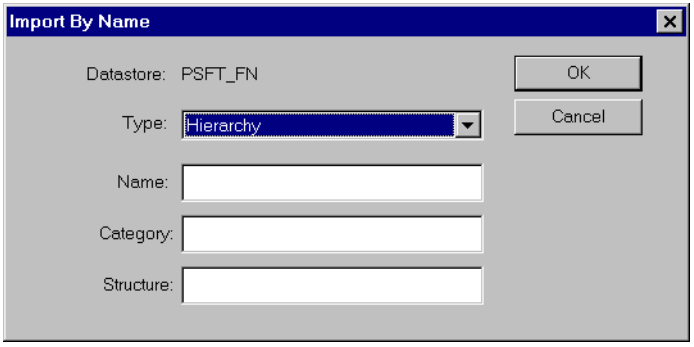
1. In the object library, go to the **Datastore** tab.
2. Right-click the datastore name and select **Import By Name**.
3. In the Import by Name dialog box, specify the **Type** of the item to import.

   To import a PeopleSoft tree, select **Hierarchy**.
4. In the **Name** box, enter the name of the item to import.



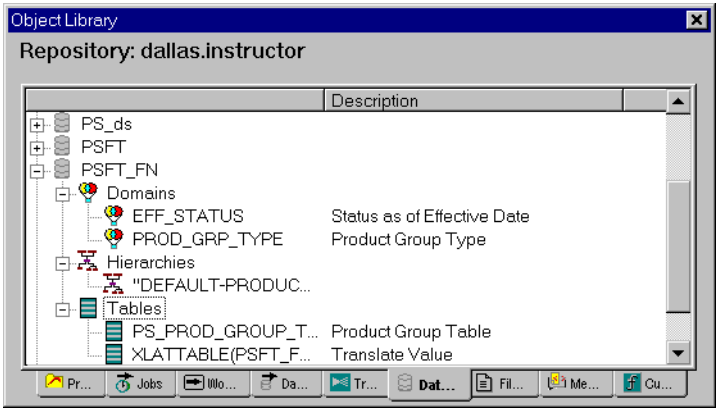
You can choose to import domain information automatically with any tables you import.

If you are importing a hierarchy, enter the category and structure associated with the hierarchy.



5. Click **OK**.

   The information appears in the object library.



➢ To import PeopleSoft metadata by searching

You can search for PeopleSoft items as you would any items to be imported into Data Integrator. That is, right-click the datastore name in the object library and choose **Search**.

For more information, see "Importing metadata through a custom datastore" on page 90 of the *Data Integrator Designer Guide*.

➢ To import PeopleSoft metadata by browsing

1. In the object library, go to the **Datastores** tab.
2. Right-click the datastore name and choose **Open**.

   A list of the available tables, domains, and hierarchies opens in the workspace. See "Browsing PeopleSoft metadata" on page 10 for information about the icons that appear in the list.
3. Right-click the names of the items you want to import and choose **Import**.

Metadata for PeopleSoft domains

This section discusses metadata for PeopleSoft domains. It includes the following topics:

What is a domain?

A *domain* is a lookup table that pairs a coded value with a textual description of the value. Domain values are typically used to specify column data where the possible range of values is constrained within a particular set.

Using a domain value instead of the text description can save space when the number of records is large. However, when working with the table to create queries, you might prefer to see a text description as opposed to looking for the appropriate code for the source data.

In PeopleSoft, domain values play a major role as the category name (or link) between a data value and its description.

Using PeopleSoft domains in Data Integrator

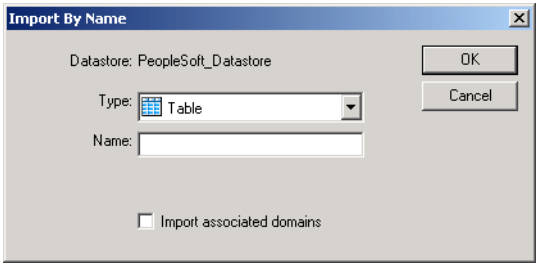
Domain values in PeopleSoft are available to end users only through PeopleSoft panels.

Data Integrator has built-in capabilities that make it much easier to deal with domain values in data movement applications.

You browse, search, and import domain metadata in the same way you browse, search, and import any PeopleSoft metadata. For specific instructions, see "Browsing PeopleSoft metadata" on page 10 and "Importing PeopleSoft metadata" on page 12.

PeopleSoft tables have columns that should be constrained to contain only values from a particular set (that is, a set within a domain). This set of possible domain values is in another table in the PeopleSoft database called XLATTABLE.

When importing table metadata from a PeopleSoft database, Data Integrator can automatically import associated domain data to make it easier to work with the values that appear in rows from the imported table. By default, Data Integrator imports associated domain data, but you can prevent this by deselecting the **Import associated domains** check box on the Import By Name window.



When importing domain values, the repository only stores the currently effective value. The currently effective value is determined using the date when the import takes place.

Imported domains appear nested under the datastore in the object library.

For more information about PeopleSoft domains, see "Using domains in PeopleSoft data flows" on page 25.

Metadata for PeopleSoft trees

You can perform the same kind of business model navigation and data browsing on PeopleSoft trees (called *hierarchies* in Data Integrator) that you can on other objects.

You browse, search, and import hierarchy metadata similarly to the way you browse, search, and import any PeopleSoft metadata. For specific instructions, see "Browsing PeopleSoft metadata" on page 10 and "Importing PeopleSoft metadata" on page 12.

Imported hierarchies appear nested under the datastore in the object library.

For more information about PeopleSoft domains, see "Extracting PeopleSoft tree data" on page 29.

3 PeopleSoft Data Flows

Data flows extract, transform, and load data. This chapter describes information unique to data flows that extract, transform, and load PeopleSoft data.

This chapter contains the following topics:

- Using effective dates from PeopleSoft data
- Using domains in PeopleSoft data flows
- Extracting PeopleSoft tree data

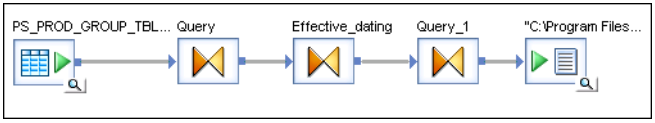
For information about Data Integrator data flows that is not PeopleSoft-specific, see "Data Flows" on page 153 of the *Data Integrator Designer Guide*.

Using effective dates from PeopleSoft data

This section provides an example that shows how you can use effective dates when populating a dimension table in the product group. This example includes several operations:

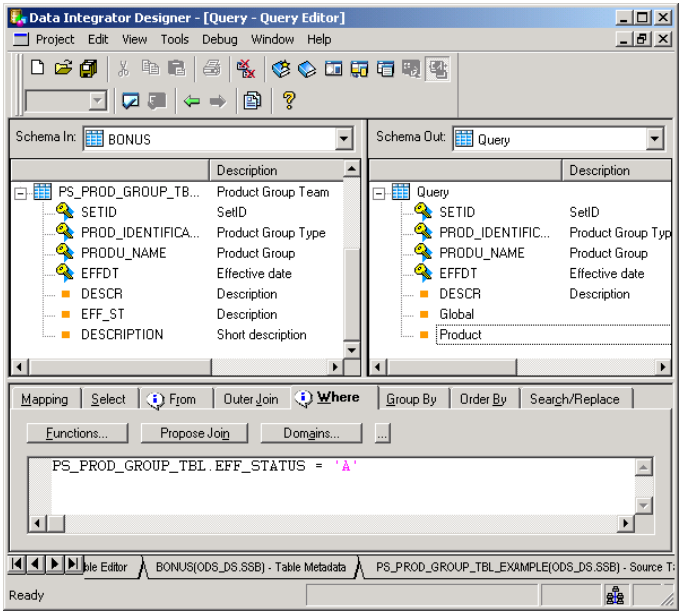
- Extracts data from table `PS_PROD_GROUP_TBL`
- Selects a subset of columns for the target
- Filters the data based on status
- Creates effective date ranges using the Effective Date transform
- Retrieves only the rows effective on a particular date
- Loads the data into the target

One data flow completes these operations.



Selecting a subset of source columns

The first query selects a subset of the columns.

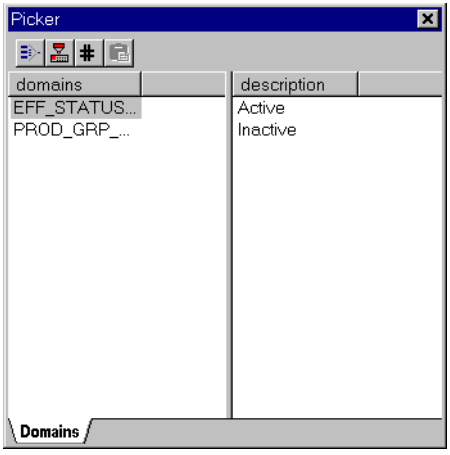


Filtering based on domain values

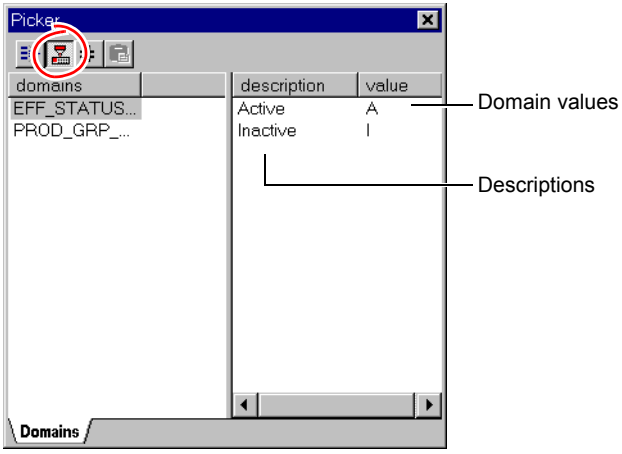
For general information about domains, see "Using domains in PeopleSoft data flows" on page 25.

The first query includes a `WHERE` clause that limits the products selected to those with a status of Active. The status information comes from the `EFF_STATUS` column in the input.

To display the relevant domain values for this table, click **Domains** in the query editor. The Picker window opens.



To see the available domain values, click the **Show Domain Values** button. The values appear to the right of the descriptions.



To put a domain value in the `WHERE` clause, place the cursor where the value needs to appear in the **Where** tab and click the description name in the Picker window.

Creating effective date ranges

This example retrieves only those product groups that are valid today. There are three steps involved:

1. Retrieve the effective-from date (`EFFDT` column) from the source.
2. Generate effective-to dates for the source rows using the Effective Date transform.
3. Filter out rows that are not effective on today's date.

The Effective Date transform is described in detail in Chapter 5, "Transforms," in the *Data Integrator Reference Guide*.

Retrieving records with current effective dates

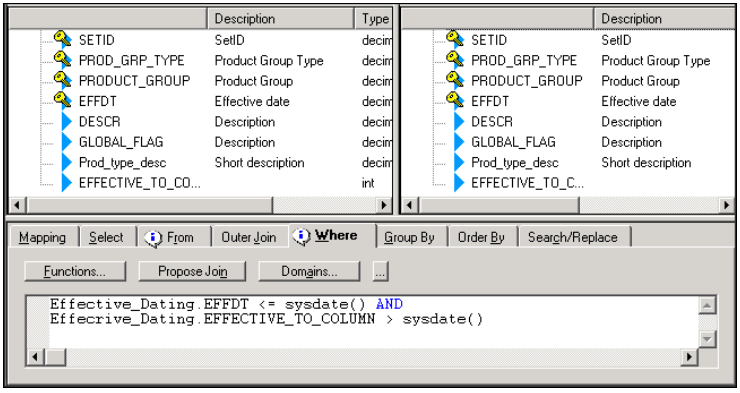
The second query in the data flow contains a WHERE clause that selects:

- Those rows where the effective-from date from the source (`EFFDT`) is less than or equal to the system date

  AND

- Those rows where the effective-to date from the results of the Effective Date transform is greater than the system date

The query editor contains the `WHERE` clause.



Using domains in PeopleSoft data flows

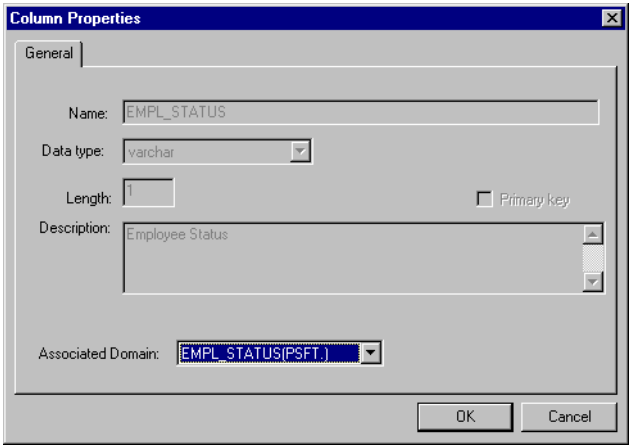
For general information about domains, see "Metadata for PeopleSoft domains" on page 15.

Data Integrator helps you:

- Build queries containing domain values
- Substitute domain descriptions for cryptic domain values
- Build queries where you need to know some domain values, for example while filtering
- Validate imported data that contains domain values

The properties for a given column enable you to explicitly associate a given column with a given domain.

To open the column properties, right-click a column name in the query editor and select **Properties**.



Any available domain can be assigned to any column regardless of data type.

Using the Picker window

To open the domain Picker window, click the **Domains** button in a query editor that has a table as a source.

The four buttons at the top of the Picker window control the content of the window and initiate actions.


Click to toggle between displaying "all" and "relevant" domains. All domains (default) are listed in alphabetical order; relevant domains  are shown in the order in which they are used in the table or tables you have selected.


Click to display the domain values in the window. Click again to hide the domain values.


Click to toggle between pasting only the domain value (default) and both the value and a commented description  in the `WHERE` clause when you select the value.

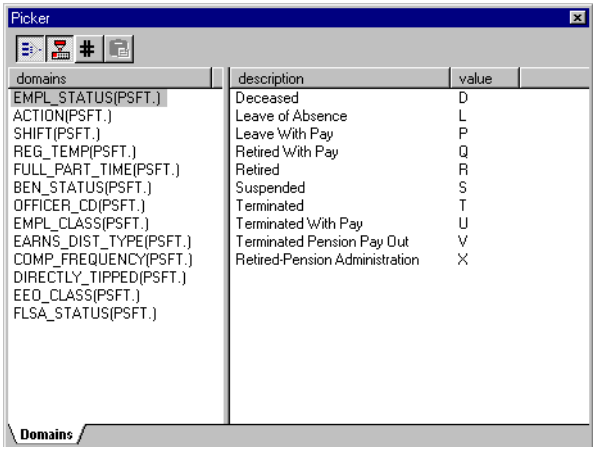

Pastes all selected values in the `WHERE` clause. Descriptions are not included. To enable this button, use Ctrl-clicks to select more than one domain description.

➢ To include a column and a related domain value in a WHERE clause

1. Drag the column from the source schema into the `WHERE` clause.
2. Enter an equal sign (=) after the column name.

3. Enter a value or paste the value from the Picker window.

   Click **Domains** to open the Picker window.



Validating domains

Capabilities built into Data Integrator allow you to validate data that contains domain values:

- *value IN domain* clause (part of a `WHERE` clause)

  This clause enables you to ensure that all rows in a table have a valid value for a domain column.

  Syntax:

  `value` `IN` *`datastore_name.owner.domain_name`*

  This clause returns `TRUE` if the value is a member of the domain specified.

  If the clause is part of a job being executed, Data Integrator searches the domain values in the `XLATTABLE`.

- `get_domain_description` function

  This function returns the description for a domain name. The description is returned as a quoted string.

Syntax:

```
get_domain_description
('datastore_name..domain_name',
  table_name.domain_name)
```

Restrictions using domains

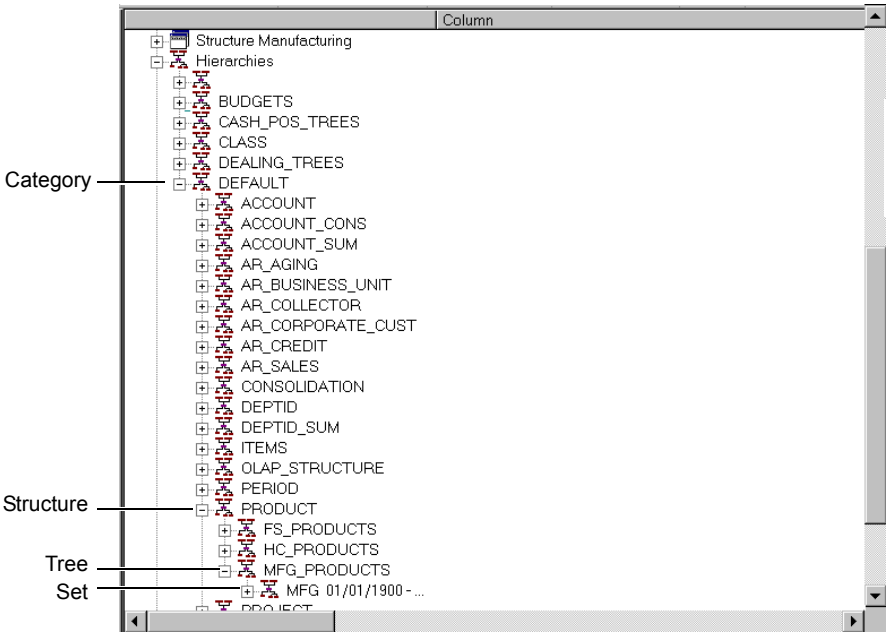
Some restrictions involving domains include:

- Prompt tables are not supported. (A prompt table is a user-defined table similar to the XLATTABLE in a database.)
- Prior to validation at execution time, there is no type checking when a domain is associated with a column.

Extracting PeopleSoft tree data

In this example, Data Integrator extracts data from a hierarchy (tree) and loads it into an Oracle table.

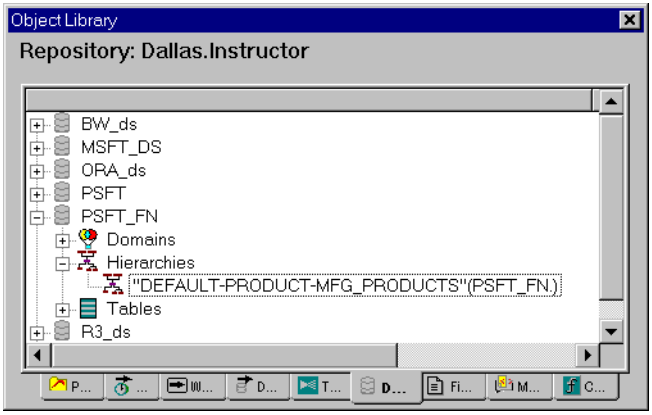
You can browse hierarchy information when you open a datastore. Opening hierarchy groups in the workspace shows the tree levels.



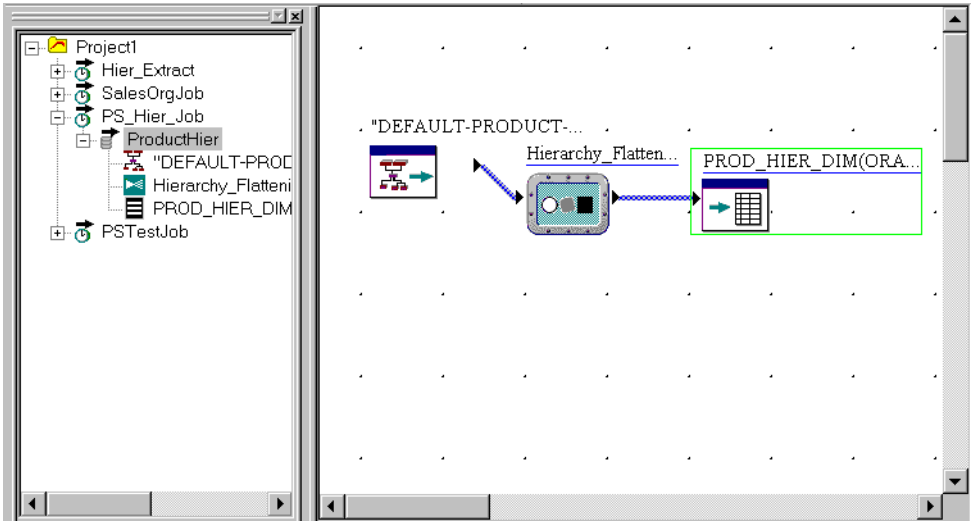
To import the tree into Data Integrator, select the tree, right-click, and choose **Import**.

For more information about importing metadata, see "PeopleSoft Datastores" on page 7.

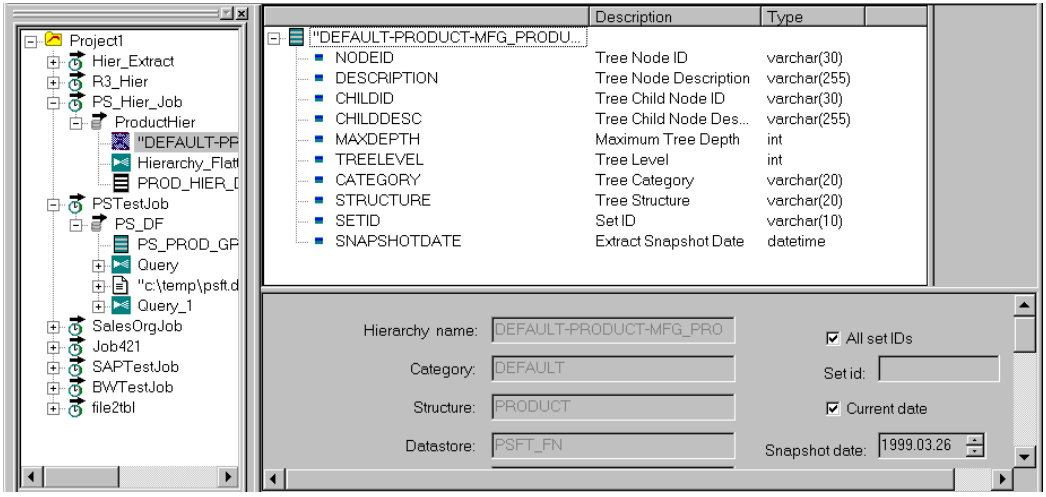
After you import the tree, it appears as a hierarchy in the object library.



The job consists of a single data flow that extracts the data and loads it into the Oracle table.



Double-clicking the hierarchy object opens the editor to show the options available for the object.



For information about the details of the hierarchy object, see "Hierarchy" on page 38.

You must specify:

- All set IDs or a specific set ID

  Selecting the **All set IDs** check box gives you all the set IDs associated with this hierarchy.

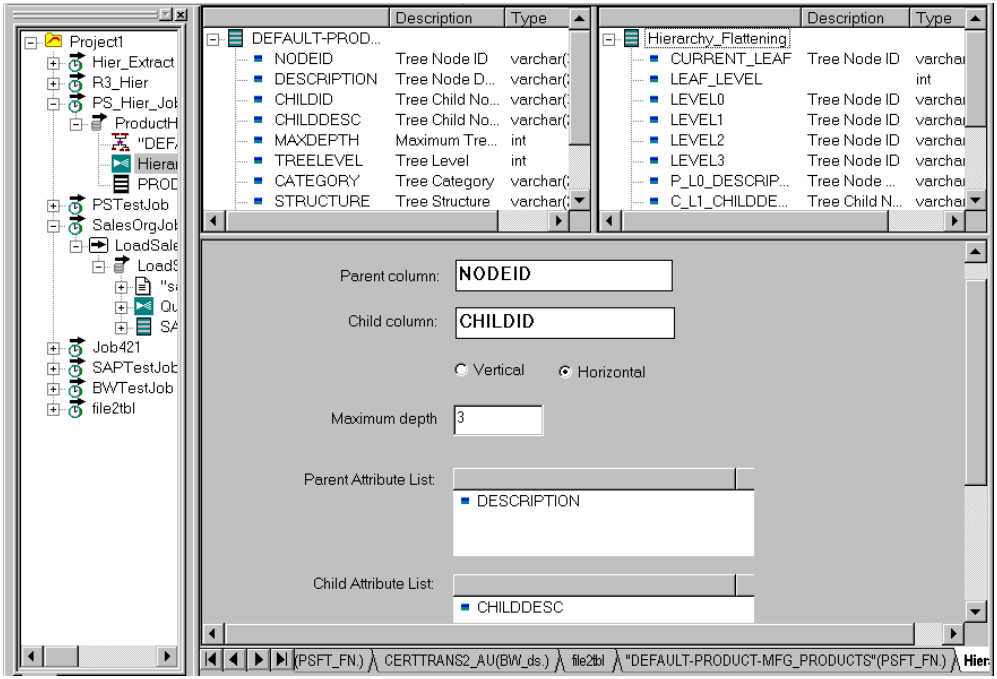
  Clearing the **All set IDs** check box opens the box below it where you can enter the name of the specific set ID for which you want to extract data. In this example, there is only one set in the tree, `MFG`.

- Current date or Snapshot date

  Selecting the **Current date** check box will give you information as of the date returned by the `sysdate` function.

Clearing the **Current date** check box gives you the opportunity to specify a snapshot date, which is any date in the past or future. Make sure to specify the date as a four-digit year, a period, a two digit month, a period, and a two-digit day (YYYY.MM.DD).

Before Data Integrator loads data into the table it flattens it horizontally. This is accomplished through the Data Integrator Hierarchy_Flattening transform. The options available in the transform appear in the transform editor.



For information about the Hierarchy_Flattening transform, see Chapter 5, "Transforms," in the *Data Integrator Reference Guide*.

4 Reference Information

This chapter contains reference information specific to the PeopleSoft interface for Data Integrator.

Specifically, this section describes Data Integrator objects specific to PeopleSoft and Data Integrator objects with supplemental information for the PeopleSoft interface.

This chapter contains information about the following objects:

Object	Class	Description
Datastore	Single-use	Specifies the connection information Data Integrator needs to access a database or other data source. Cannot be dropped.
Domain	Reusable	A lookup table that pairs a coded value with a textual description of the value. Domain values are typically used to specify column data where the possible range of values is constrained within a particular set.
Hierarchy	Reusable	Browse, search, and import hierarchy metadata.
Query	Single-use	Retrieves a data set that satisfies conditions that you specify.

Datastore

CLASS

Reusable

ACCESS

In the object library, click the **Datastores** tab.

DESCRIPTION

A datastore provides a connection to a data source such as a database. Through the datastore connection, Data Integrator is able to import descriptions of the data source such as its metadata. When you specify tables as sources or targets in a data flow, Data Integrator uses the datastore to determine how to read data from or load data to those tables. In addition, some transforms and functions require a datastore name to qualify the tables they access.

If you delete a datastore from the object library, you must remove references to the datastore from the following locations:

- Source or target tables using this datastore in your diagrams
- The `lookup` and `key_generation` functions and Key_Generation, History_Preserving, Table_Comparison, and SQL transform references

Datastores have the following attribute:

Attribute	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object.

After you create a datastore, you can import metadata about the objects, such as tables and functions, in that datastore. See Chapter 5, "Datastores," in the *Data Integrator Designer Guide*.

PEOPLESOFT DATASTORES

Set the following options to define a PeopleSoft datastore:

Option	Description
Name	Specify the datastore name. Data Integrator uses this name to reference the datastore from other object definitions.
Application type	Choose PeopleSoft to display the options for PeopleSoft datastores. You cannot edit this option after creating the datastore.
Database type	Select either **Microsoft_SQL_Server** or **Oracle** to indicate the database type used as the database layer of your PeopleSoft application server. The remaining options in the datastore definition are specific to the database type. Refer to the "Custom application datastores" on page 35 of the *Data Integrator Reference Guide* for details.

Domain

CLASS

Reusable

ACCESS

- To view the list of imported domains and domain values from the object library, click the **Datastores** tab.
- Use domain values from inside query transforms with inputs from tables associated with domains. Click **Domains** to open a window that helps you pick domain values.

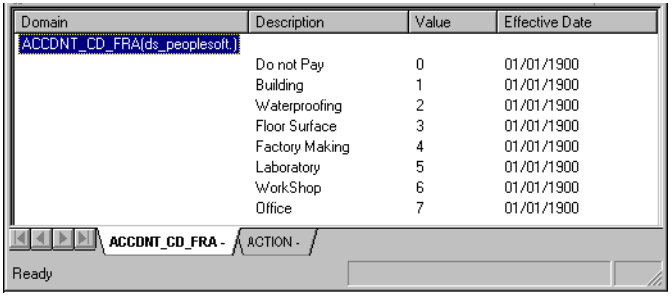
DESCRIPTION

A domain is a lookup table that pairs a coded value with a text description of the value. You will typically use domain values to specify column data where the possible range of values is constrained to a particular set.

Domains have two built-in attributes.

Attribute	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object.
Description	The description of the domain imported from the application or database.

View domain values, descriptions of the values, and the effective dates for each value by opening the domain from the object library.



Hierarchy

CLASS

Reusable

ACCESS

In the object library, click the **Datastores** tab and expand a datastore listing (click the plus sign next to the datastore name).

DESCRIPTION

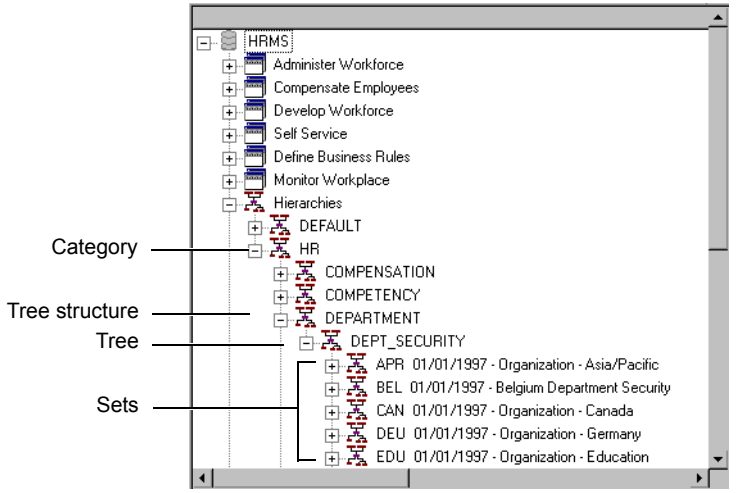
Hierarchy objects allow you to browse, select, and import hierarchical information from a source database. Hierarchy objects can import the following hierarchical information:

Source application	Hierarchies extracted
PeopleSoft	You can browse, navigate, and import metadata from PeopleSoft trees.

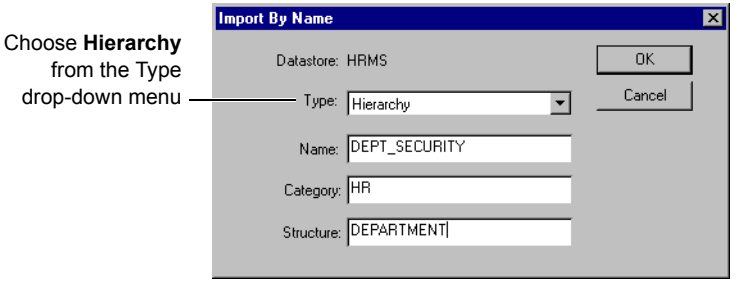
To extract hierarchical information, select a hierarchy type or tree, import its metadata into your repository, then drag the hierarchy icon from the object library into a data flow. Importing metadata is described in Chapter 5, "Datastores," in the *Data Integrator Designer Guide*.

EXTERNAL VIEW

Hierarchy types are listed in the external browsing view for a PeopleSoft datastore. From this view, you can choose to the hierarchy to import.

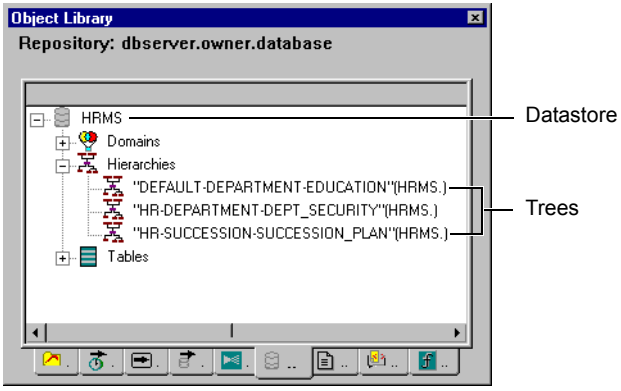


You can also import a tree by specifying the tree by name. Select the datastore in the object library, right-click, and choose **Import By Name**. The following dialog box opens.



IMPORTED TREE

After you import the tree, it appears as a hierarchy object in the object library nested under the datastore name.



The hierarchy object has the following properties:

Property	Description
Name	The name of the hierarchy object. The name is constructed as follows: *category-structure-tree(datastore)* The name (except the datastore name) is case sensitive.
Description	The description of the tree as included in the PeopleSoft table.

The hierarchy object has the following attributes, the values of which are included in the metadata imported for the hierarchy.

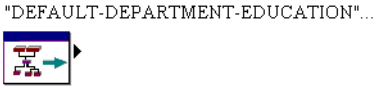
Category	Tree_Node_Description
Structure	Tree_Node_Record_Primpary_Key1 to Tree_Node_Record_Primpary_Key9
Set_ID	
Snapshot_Date	Tree_Leaf_Record_Name
Tree_Name	Tree_Leaf_Field_Name
Tree_with_Leaf?	Tree_Leaf_Description
Tree_Node_Record_Name	Tree_Leaf_Record_Primary_Key1 to Tree_Leaf_Record_Primary_Key9
Tree_Node_Field_Name	

Selecting the hierarchy in the object library and choosing **Open** displays the object properties and the source schema that results from the hierarchy extraction.

Column name	Description
NodeID	The parent value in the relationship described by this row.
Description	The parent description.
ChildID	The child value in the relationship described by this row.
ChildDesc	The child description.
MaxDepth	The maximum number of nodes between the root node and the lowest node in the hierarchy.
TreeLevel	The level in the hierarchy that this row describes.
Category	The category in which this hierarchy is included.
Structure	The tree structure in which this hierarchy is included.
SETID	The version of the hierarchy.
SnapshotDate	The effective date of the hierarchy.

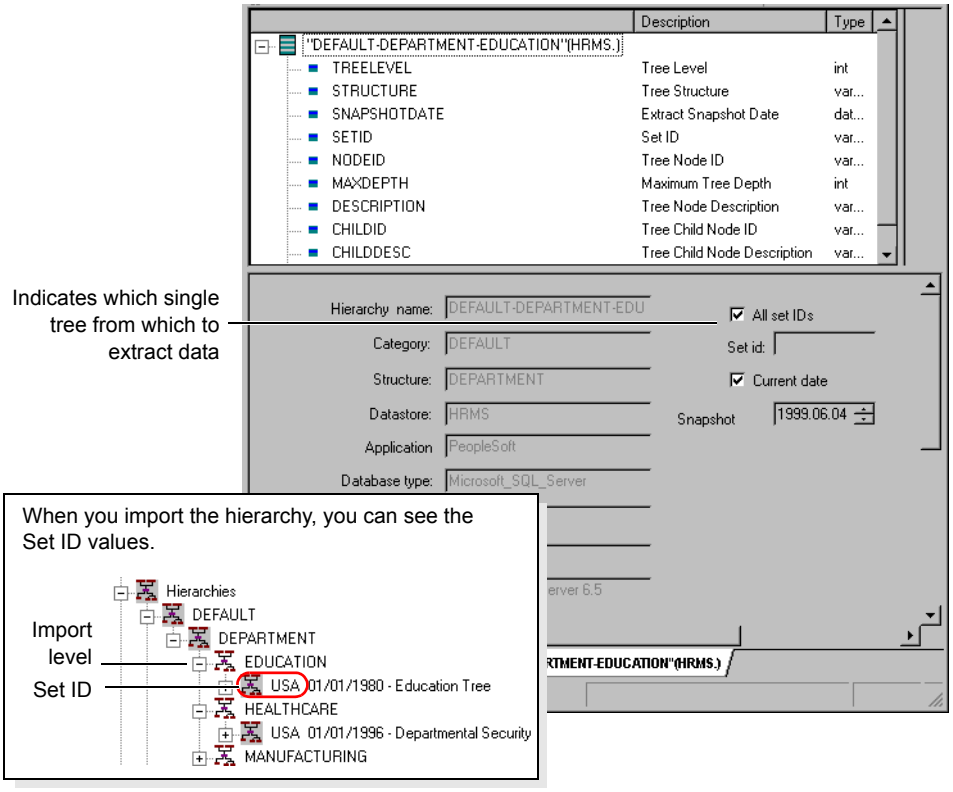
HIERARCHY INSTANCE

You can drag a hierarchy from the object library into a data flow definition. The hierarchy object appears in the data flow definition as follows:



The hierarchy editor displays the datastore information for the hierarchy and provides options for the instance of the hierarchy object:

Option	Description
All set IDs or Set ID	Allows you to select a single tree out of the structure imported into Data Integrator. The set ID appears as part of the name of the level below the level of hierarchy structure imported into Data Integrator.
Current date or Snapshot	Allows you to filter the extracted values by their effective dates.



Query

CLASS

Single-use

ACCESS

With a data flow diagram in the workspace, click the query icon in the tool palette, then click in the workspace.

DESCRIPTION

A Query transform, like a SQL SELECT statement, retrieves a data set that satisfies the conditions you specify. With a Query transform, you can:

- Map columns from input to output schema
- Add new columns, nested schemas, and functions to the output schema
- Choose the data to extract
- Perform operations on the data
- Join data from multiple sources

For a complete description of the Query transform, see "Query" on page 287 of the *Data Integrator Reference Guide*.

EDITOR

Where tab — Use the **Where** tab to set conditions that determine which rows Data Integrator outputs.

With the PeopleSoft interface, you can use the **Domains** button. Domains constrain retrieved data sets. Click the **Domains** button to open the Picker window, which helps you build an expression using a domain. For more information about domains, see "Metadata for PeopleSoft domains" on page 15.

A Glossary

This glossary defines terms used in this documentation set including common terms that have a special meaning with respect to Data Integrator. Within a definition, other terms that are defined in the glossary appear in **bold type**.

ABAP

Advanced Business Application Programming.
A fourth-generation programming language developed by SAP in which R/3 applications are written.

ABAP program

A program that executes database operations on an **SAP R/3** server. Data Integrator **R/3 data flows** generate ABAP programs.

Access Server

A process that starts, stops, and monitors real-time services (**real-time jobs**). The Access Server dispatches requests to real-time services, ensuring optimal load balancing and complete life cycle management. The Access Server also starts, stops, and monitors **adapters**.

adapter

An external Data Integrator **interface**. There are two types of adapters:

- Custom adapters — Adapters developed using the Data Integrator Adapter Development Kit
- Prepackaged adapters — Adapters prebuilt and purchased from Business Objects, such as the Data Integrator MQSeries Adapter

Administrator

A browser-based system administration application that supports Data Integrator. Use the Administrator to configure and manage:

- Services (**real-time jobs**)
- Service providers (instances of real-time services)
- Clients (Web application clients)
- Batch jobs
- Adapters
- Job logs

after-image

The values in an UPDATE row after the row changes. You might want to pair before- and after-images of UPDATE rows when designing log-based changed-data capture jobs which Data Integrator supports.

aggregate data

Data that results when a process combines elements. This data can be presented collectively or in summary form.

ALE

Application Link Enabling. An SAP R/3 programming-related interface designed to allow reliable communication across a distributed environment.

Analytic Rapid Mart

An **Data Integrator application** that produces a predefined **data warehouse** for a specific area of business analysis. Analytic Rapid Marts are typically based on dimensional modelling. With Analytic Rapid Marts, users can manage large volumes of data.

attribute

A **property** created for a type of **object**.

BAPI

Business Application Programming Interface. A standardized SAP R/3 programming interface that allows non-SAP R/3 applications to access specific business processes and data.

back-office application

A software system that an enterprise uses to run business operations, such as human resources, general ledger, and finance. Back-office applications store data in a relational **database** optimized for operational use.

There are several types of back-office applications:

- Enterprise resource planning (ERP) — For example, SAP R/3, BAAN, J.D. Edwards, PeopleSoft
- Supply chain management (SCM) — For example, i2
- Customer relationship management (CRM) — For example, Siebel, Clarify
- Legacy

Basis

The SAP infrastructure. Basis is the foundation for all SAP products.

batch job

A group of objects (data flows, work flows, conditionals, scripts, and so forth) that executes on a schedule. This type of job is recommended for operations on large amounts of data.

before-image

The values in an UPDATE row before the row changes. You might want to pair before- and after-images of UPDATE rows when designing log-based changed-data capture jobs which Data Integrator supports.

bulk loading

A software-based mechanism that moves large data files. This mechanism supports compression, blocking, and buffering to optimize transfer times.

business component

A set of tables Siebel applications use to create a logical object called a business object.

Business Objects Data Platform

A set of applications from Business Objects, Americas, that enterprises can use to manage all types of data—from data for supply chain partners to data for customers. The Business Objects Data Platform consists of **Data Integrator** and one or more **Rapid Marts**.

CDC datastore

A CDC datastore allows you to limit extracted data to changed data only. A CDC datastore connects a **changed-data capture** table on a source database to Data Integrator.

CDC checkpoint

A CDC checkpoint enables Data Integrator to restrict CDC subscription reads. After you enable a checkpoint, the next time the CDC job runs, it reads only the rows inserted into the CDC table since the last checkpoint.

CDC Subscription

A CDC subscription name allows different data flows to extract data from the same table without corrupting data extracted by other data flows. The datastore and table name can be the same; however, the table owner name must be different in each source CDC table.

changed-data capture

The process of retrieving changes made to a production data source. This process consolidates units of work, ensures data is synchronized with the original source, and reduces data volume in a warehouse environment.

Citrix MetaFrame XP

Citrix MetaFrame XP software provides an access infrastructure for enterprise applications. You can use this software to run Data Integrator on a server which publishes instances of the Designer and other Data Integrator interfaces to users on client computers.

client/server

A distributed technology approach where the processing is divided by function. The server performs shared functions (such as managing communications and providing database services), while the client performs individual user functions.

COM

Component Object Model. Provides a set of interfaces that allow clients and servers to communicate within the same computer (running on Windows 95 or later versions).

Commerce Rapid Mart

An **Data Integrator application** that produces a predefined **data warehouse** and that includes logic to support real-time requests for data from a **back-office application**. A Commerce Rapid Mart is usually based on different data model than an **Analytic Rapid Mart**. With a Commerce Rapid Mart, users can integrate a company's back-office system with an **e-commerce** application.

Communication Structure

In **SAP BW**, a data structure that defines a set of **InfoObject**s available from an **InfoSource** to put into **InfoCube**s.

conditional

A **single-use object**, available in **work flow**s**,** that allows you to branch the execution logic based on the results of an expression. The conditional takes the form of an if/then/else statement.

custom ABAP program

A custom ABAP program runs an **ABAP program** and generates a **data set**. With a custom ABAP program, you can run an existing ABAP program as part of a job. Use a custom ABAP program as a **source** in a **data flow** or an **R/3 data flow**.

data extraction

The process of moving data from a **database** or application source to a database target (either from a legacy database to a **data mart**, or from one data mart to another).

data flow

A **reusable object** containing **steps** to define the transformation of data from **source** to **target**. Data flows are called from inside a **work flow** or **job**. You can pass information into or out of data flows using **parameters**.

Data Integrator

A software system that allows users to build and execute applications with which they can create and maintain **data warehouses**.

Data Integrator consists of several components:

- **Access Server**
- **Data Integrator engine**
- **Data Integrator repository**
- **Data Integrator Service**
- **Administrator**
- **Designer**
- **Job Server**

Data Integrator application

A set of **jobs** that process batch data extractions and real-time requests using data from a variety of data sources.

Data Integrator engine

The core process that reads job information from the **Data Integrator repository** and sets up run-time processes that execute the job. The run-time processes extract, transform, and load relational and hierarchical data. The **Job Server** starts the Data Integrator engine to execute batch or real-time jobs.

Data Integrator interface

A program that Data Integrator uses to access data sources. Specific interfaces vary by installation. There are internal interfaces—those native to the installation—and external interfaces—those that you install separately. Internal interfaces allow Data Integrator to access applications like **SAP R/3** and **SAP BW**, **messages**, relational database systems, and **legacy systems**. An external interface is also known as an **adapter**. It allows Data Integrator to access applications using information exchange technologies such as Web Services and MQ Series.

Data Integrator repository

The database that contains information about an Data Integrator application. The repository contains information about defined **reusable objects**, the **metadata** for tables, and built-in **transforms** and **functions**. When you invoke Data Integrator, you log in to the repository containing the **objects** you want to use.

Data Integrator Service

The process that ensures that the **Access Server** and the **Job Server** are running. You can configure the Data Integrator Service to restart the Access Server and Job Server whenever the computer where they are located restarts.

data loading

The process of populating a **data warehouse**. Data loading is provided by **DBMS**-specific load processes, DBMS insert processes, and independent fast-load processes.

data mapping

The process of assigning a source data element to a target data element.

data mart

A highly-focused version of a **data warehouse**. Typically, created by a department or division of a company, data marts contain data for a specific subject area, such as finance or sales. Data Integrator can populate a data mart.

data transformation

The process of filtering, merging, decoding, and translating source data to create validated data for a **data warehouse**.

data set

Rows of data with a defined schema. A step in a **data flow**—such as reading data from a **source**, joining data in a **Query transform**, or transforming data though another **transform**—yields a data set. You can view individual data sets by placing a **target** table or file at that point in the **data flow**.

data transport

A step in an **R/3 data flow** that defines a **target** to store the **data set** extracted during the flow. You can locate the target file on the **SAP R/3** server or in a location accessible to both the SAP R/3 server and to Data Integrator across a network.

data type

The format used to store a value. Data types can imply a default format for displaying and entering the value. Data read from a **source** is converted to the appropriate Data Integrator data types; data loaded to a **target** is converted from its Data Integrator data type to the type appropriate for the target.

data warehouse

A relational **database** containing subject-oriented information that supports decision analysis. Typically, information in a data warehouse is integrated from multiple sources, including operational sources. Data warehouses provide a single, non-volatile point of reference for a particular time period. Data warehouses are composed of **data marts.** Data Integrator can populate a data warehouse.

database

A collection of **table**s managed by a **DBMS** such as Microsoft SQL Server or Oracle.

DataConnector

DataConnector operator instances are used to read data files generated by Data Integrator when performing bulk loading using the Teradata Warehouse Builder.

datastore

A logical channel connecting Data Integrator to a **database**. The datastore definition includes the name and location of the database as well as user authentication information. Data Integrator uses a datastore definition to qualify a table name wherever a table is indicated in a diagram or expression. You can access the datastore definition through the **object library**.

DBMS

Database management system. A software system that builds and maintains database tables.

debug mode

Allows you to debug a job using the **interactive debugger** features in the Designer.

declarative specification

A way of indicating the desired results of a **data transformation** without placing constraints on the method of reaching the results. When you create a **data flow** in Data Integrator, you specify what data you want to read from which **source**, what you want done to that data, and where you want it loaded. What you specify constitutes the declarative specification. Data Integrator then determines the most efficient way to reach the requirements you set, including pushing operations to the source **DBMS** for execution.

degree of parallelism

You can set a value at the data flow level that allows you to parallel-process a **transform** within a data flow.

For example, if from the Properties window of a data flow you set the **Degree of parallelism** to 5, then when the job executes, Data Integrator replicates each transform in the data flow five times. Each of these replicated transforms executes in parallel using a separate process thread. Using a value for degree of parallelism can improve job performance if a **Job Server**'s computer memory and number of CPUs supports a job's parallel-processing configuration settings.

dependency analysis

A category of metadata reports that produces reports based on search criteria you specify to find specific objects in the repository. For each object returned, dependency analysis reports provide object lineage as well as object impact information.

derived data

Data that results from a computational step applied to reference or event data. Derived data is the result of either relating two or more elements of a single transaction (such as aggregation), or relating one or more elements of a transaction to an external algorithm or rule.

Designer

A graphical user interface that allows you to design and test an **Data Integrator application**.

diagram

The icons and connections between the icons that make up the definition of a **job**, **work flow**, or **data flow**.

DMZ

Demilitarized zone. A perimeter network that is more accessible than your most protected network, but located inside a security layer protected from the internet.

drill down

A method of exploring detailed data that was used in creating a summary level of data. Drill down levels depend on the granularity of the data in the **data warehouse**.

DTD

Document type definition. A text file that describes the elements (tags) in an **XML** document and the relationship among them. When an XML document is used to describe a transaction, the DTD describes the data schema used in the transaction.

e-commerce

Buy and sell transactions conducted over the internet.

EDI

Electronic Data Interchange. Electronic exchange of structured data between businesses. This exchange is not dependent on hardware, software, or communication protocols.

element

A component found within XML Schemas and DTDs.

embedded data flow

A data flow created within one or more data flows.

enterprise data

Data that is defined for use across a corporate environment.

ERP system

Enterprise resource planning system. A **back-office application** from which Data Integrator can extract data.

exception

An error that occurs while executing a **job**. You can catch individual or groups of exceptions using a **try/catch block** inside a **work flow**. Catching an exception allows you to automatically execute a solution for the error.

Executive Information Systems (EIS)

Tools programmed to provide canned reports or briefing books to executives. They offer strong reporting and drill-down capabilities. These tools allow ad hoc querying against a multi-dimensional **database**, and most offer analytical applications along functional lines such as sales or financial analysis.

extract date

The date data was extracted.

extract frequency

The interval at which data is extracted, such as daily, weekly, monthly, or quarterly. The frequency that data extracts are needed in the **data warehouse** is determined by the shortest frequency requested through an order, or by the frequency required to maintain consistency of the other associated data types in the source data warehouse.

file

A **source** or **target** for data. To read or load data to a file within Data Integrator, define a **file format** that instructs Data Integrator about the structure of the data in the file.

file format

A description of how data is or should be organized in a **file** Data Integrator reads from or loads to. A file format can be specific to a single file or generic for many files.

front-office application

Software that companies use to support reporting, **e-commerce**, and other tasks. Types of front-office applications include:

- Supply chain collaboration
- Analytical
- Sell-side e-commerce
- Customer relationship management

function

A program that operates on values that are passed to it. Data Integrator functions are available through a function wizard in a **script**, **conditional**, or **Query transform**. Data Integrator also gives you access to functions provided by the **DBMS** you are using. In addition, you can define your own functions using the Data Integrator scripting language or by writing C-language programs.

IDoc

Intermediate Document. An SAP-specific format. Used for **EDI** and **ALE**.

IDoc type

Indicates the SAP format that is used to interpret the data of a business transaction. Consists of the following components:

- A control record

  Identical for each IDoc type.

- Several data records

  A single data record consists of a fixed key part and a variable data part. The data part is interpreted using segments, which differ depending on the IDoc type selected.

- Several status records

  Identical for each IDoc type. Describe the status states an IDoc has already passed through or the status an IDoc has attained.

import

The process of acquiring information for the **Data Integrator repository**. Import the following kinds of information into Data Integrator:

- The **metadata** for **source** and **target** databases
- Descriptions and code for user-defined and DBMS **functions** and **transforms**

InfoArea

In **SAP BW**, an arbitrary grouping of **InfoCube**s.

InfoCube

In **SAP BW**, a multidimensional star schema that is limited to one fact table and 16 dimensions.

InfoCubes store only numeric data at the fact level.

SAP BW reserves three of the dimensions for currencies, time, and units of measure. Dimensions have to be linked back to master data and hierarchy tables to gain access to the information stored there.

When a user query is initiated, BW creates a temporary **OLAP** cube that stores a constrained subset of the permanent OLAP cube. The temporary InfoCube persists until the user disconnects from BW or changes the query.

InfoObject

In **SAP BW**, an InfoObject is equivalent to a data element and its associated metadata (for example, description or business rules). Can be a **Key Figure** or a Characteristic.

InfoPackage

In **SAP BW**, a schedule for running BW updates.

Information resource (IR)

A producer or consumer of enterprise information. An information resource can be a flat file, a **back-office application**, an EAI system, or a Web application.

InfoSource

In **SAP BW**, a logical construct that defines a set of **InfoObjects** and their mappings to source data elements. InfoSources do not store data, but rather serve as conduits via **Transfer Structures**.

Integration Rapid Mart

An **Data Integrator application** that provides point-to-point content mapping and data movement mechanisms between different **back-office applications**.

interactive debugger

A Designer feature that allows you to step through the data of a job one row at a time using filters and breakpoints on a line. Like executing a job, you can start the interactive debugger from the Debug menu when a job is active in the workspace. While in debug mode, all other Designer features are set to read-only.

interface

Data Integrator offers two types of interfaces:

An internal **Data Integrator interface**, allows you to create datastore connections to natively supported applications.

An external **Data Integrator interface** (or **adapter**), allows Data Integrator to communicate with information exchange technologies such as Web Services and MQ Series.

IR

See **Information resource (IR)**.

job

The unit of work that can be scheduled independently for execution by the **Administrator**. Jobs are special **work flows** that can be called by other work flows or jobs.

Job Server

A process that receives requests from the **Designer** and the **Administrator** to start and stop **jobs**. To start batch or real-time jobs, the Job Server triggers the **Data Integrator engine**. Engine processes run on the same computer as the Job Server process that triggers them.

join rank

A value given to or calculated for all data sets in a **data flow**. Data Integrator uses the join rank to determine which **source** to read first when assembling the **data set** in a join. Data Integrator uses the source with the lower join rank as the inner source of the join and uses the source with the higher join rank as the outer source of the join. If more than one of the sources is located in the same **DBMS**, the highest rank is used for all of the sources from that DBMS. When a source is processed through a **Query transform** or another **transform**, and then used in a join, the maximum join rank defined for the source is used as the join rank in the query.

Key Figure

In **SAP BW**, an **InfoObject** that represents a numeric fact.

legacy system

An information or transaction processing system used to store data such as bank balances, inventories, payroll, and manufacturing parts.

license-controlled feature

A Data Integrator feature that is enabled or disabled based on the product license. The product license controls which icons and settings are available in Data Integrator as an internal **Data Integrator interface**.

line

The line you use to connect objects in a workspace diagram. Lines represent a left-to-right flow path for data moving between two objects during job execution.

memory datastore

A datastore connection/container for memory tables.

memory table

Internal Data Integrator table used to store a data set in memory while a job runs. Use instead of staging tables to improve performance of a real-time job built with multiple data flows. Use a memory table to move a data set between data flows.

message

Represents hierarchical data, such as a header with line items, for document-oriented transactions, such as a purchase order.

metadata

In Data Integrator, information acquired and maintained to describe tables in **source** and **target** databases. This information includes the names of tables and their columns, and the data types of the columns. All the metadata for all user tables is imported at one time for a given **datastore**.

In general, metadata typically includes a description of data models, a description of the layouts used in database design, the definition of the system of record, the mapping of data from the system of record to other places in the environment, and specific database design definitions.

multiple data flow model

One of three data flow models you can use in real-time jobs. Use this model to develop real-time jobs in separate logical object segments (data flows) rather than in a single data flow. Allows for any number of data flows in a real-time processing loop under these conditions: The first data flow must have a single message source, the last data flow must have a single message target. Other data flows in the same real-time job cannot have a message source or message target. See also **put/acknowledge data flow model** and **single data flow model**.

nested data

Data in one **table** that is related to a single row of another table. A nested table appears in Data Integrator as a column in a parent table. Columns in the nested table can themselves contain tables.

object

Any item that you create in the **Designer**. Data Integrator distinguishes two classes of objects: **reusable objects** that are complete and can be reused in your projects (such as **data flows**) and **single-use objects** that only appear as components of other objects (such as a **try/catch block**). This distinction affects how you create and retrieve each type of object.

object definition

The options that describe the operation of an **object**. To view and modify an object definition, open the object so that its definition appears in the **workspace**.

object library

A tool in the **Designer** that gives you access to reusable objects.

ODBC

Open database connectivity. A standard for **database** access initiated by the SQL Access Group consortium and now owned by Microsoft.

ODS

Operational data store. An OLAP-designed relational **database** that an enterprise has designated as the operational database of record (for example, a finance department might use an ODS to close its books).

OLAP

Online analytical processing. A relational **database** design that is optimized for random access. OLAP systems are used in a query environment, such as for a **front-office application**.

OLTP

Online transaction processing. A relational database design optimized for operational use. OLTP systems are used in an operational environment, such as for a **back-office application**.

operation code

A flag associated with a row in a **data set** that indicates the status of the data in the row. The operation codes are INSERT, UPDATE, DELETE, and NORMAL.

operational statistics

A category of reports within the metadata reporting tool that provides you with job and data flow performance information over a given period of time or number of executions.

parameter

A value passed to a **work flow** or **data flow** when that flow is called.

partition

To divide table data into sets based on a criteria such as a range or list of values in each row. You can configure Data Integrator to read and write partitioned table data in parallel threads. Designing jobs with partitioned table data can improve job performance if a Job Server's computer memory and number of CPUs supports the job's parallel-processing configuration settings.

project

The collection of **jobs** available in the **Designer** at a given time. A project provides a way to organize the **objects** you create.

property

Detailed descriptive information about **objects**.

put/acknowledge data flow model

One of three data flow models you can use in real-time jobs. Similar to the multiple data flow model, this model does not require a data flow with a target message. Data Integrator creates a target message to automatically acknowledge a request message. Designed for use with IDoc sources in SAP R/3 systems. See also **multiple data flow model** and **single data flow model**.

Query transform

A **data transformation object** that you can use to map columns from a **source** to a **target** schema, add new columns to the target schema, determine the data to extract, and perform operations on the data. Like a SQL SELECT statement, a query creates a **data set** that satisfies the conditions you specify.

Rapid Mart

A prepackaged **Data Integrator application** that delivers data for a specific business purpose. Easily deployed and customized, BusinessObjects™ Rapid Marts include the extraction, transformation, and data loading logic to support analysis or **e-commerce**. There are three types of Rapid Mart:

- **Analytic Rapid Mart**
- **Commerce Rapid Mart**
- **Integration Rapid Mart**

R/3 data flow

A **data flow** that extracts data from an **SAP R/3** source table. Data Integrator translates steps you define in an R/3 data flow into **ABAP** and then passes the ABAP program back to your SAP R/3 system for execution. The resulting **table** or **file** resides on the SAP R/3 system to be used as a **source** in the parent data flow.

real-time job

A group of objects (data flows, work flows, conditionals, scripts, and so forth) that execute on-demand as a "request-response" system. You design real-time jobs in the **Designer**, then configure them as real-time services and associate them with an **Access Server** in the **Administrator**, where they are started, managed and monitored. When a real-time service receives a request from a caller, it processes the request and returns a reply. This type of job is recommended for operations on small amounts of data.

relational data

A **data set** in which data in each column contains a scalar value. Data Integrator can process relational data; it can also process **nested data**.

repository

See **Data Integrator repository**.

reusable object

An **object** that can be defined, stored, and reused independent of other objects. Create reusable objects from the **object library**.

RFC server

Remote Function Call server. The Data Integrator RFC server allows third-party programs, including SAP R/3, to schedule and initiate Data Integrator jobs and return the results to Data Integrator.

SAP BW

Business Information Warehouse, an SAP product.

SAP R/3

An **ERP system**.

script

A step in a **work flow** that allows you to calculate values to pass to other parts of the work flow. The script can call **function**s, execute if-then-else statements, and assign values to **variable**s. Write a script in the Data Integrator scripting language.

segment

Format with which the data records of **IDocs** are interpreted.

server group

A defined collection of Job Servers on different computers. A server group automatically measures resource availability on each Job Server in the group and distributes scheduled batch jobs to the Job Server with the lightest load at run-time. Use the Server Groups node in the Administrator's navigation tree to group Job Servers that are associated with the same repository into a server group.

service request

Any message sent from a Web client that requires processing by a **real-time job**.

session

Associated with the **Adapter Properties** tab in an adapter datastore, a session is created by a stream-oriented adapter to group adapter components and maintain state for interactivity between Data Integrator and the adapter.

single data flow model

One of three data flow models you can use in real-time jobs. Allows for one data flow in a real-time processing loop. This data flow must have a single message source and a single message target. See also **multiple data flow model** and **put/acknowledge data flow model**.

single-use object

A step in a **work flow** or **data flow** that cannot be saved independently of the flow. Create single-use objects from the tool palette.

smart editor

A flexible editing tool in Data Integrator used for creating scripts, expressions, and custom functions without having to type the names of existing elements like column, function, and variable names.

SNMP

System Network Management Protocol helps network administrators manage network routing hardware. The protocol can manage a variety of hardware and software devices.

source

An **object** from which Data Integrator reads data.

SQL

Structured Query Language. A query language for accessing relational, **ODBC**, DRDA, or non-relational **database** systems.

SQL query tool

An end-user tool that accepts **SQL** to be processed against one or more relational databases.

star schema

A **database** design you can use to format data in a **data mart**. This design is based on a single fact table to which any number of dimensional tables may be joined. This type of database design supports multi-dimensional database analysis.

step

An **object** that is part of the definition of a **work flow** or **data flow**. Each step is represented by an icon in the diagram of the flow and is connected to other steps to indicate the flow of data through the data flow or the order of execution in the work flow.

table

A **database** table that Data Integrator reads data from or loads data into. The path and mechanisms for reading and loading data and apportioning the data among rows and columns are defined in the **datastore** that the table is associated with. Writing a **data set** to a database table means sending a combination of rows with appropriate **operation code** to the database table.

target

An object in which Data Integrator loads extracted and transformed data in a **data flow**. Data Integrator only loads rows flagged as INSERT or UPDATE.

TCP/IP

Transmission Control Protocol/Internet Protocol. The basic communication protocol of the internet, and often intranets and extranets. A computer having direct access to the internet contains a copy of the TCP/IP program. TCP/IP makes it possible for computers to communicate with each other.

Tdpid

(Teradata Director Program ID) The server name Data Integrator uses when loading with the bulk loader option. Data Integrator uses tdpid as a Teradata Warehouse Builder operator attribute.

Transfer Structure

In **SAP BW**, a data structure that defines a set of **InfoObjects** to be moved into BW. To an ETL tool like Data Integrator, a Transfer Structure looks like a table.

transform

A step in a **data flow** that acts on a **data set**. Data Integrator transforms are available through the **object library**. You can also import user-defined transforms and transforms from your **DBMS**.

Transformation Rule (Transfer Rule)

In **SAP BW**, a data manipulation rule between a **Transfer Structure** and a **Communication Structure**. The results of these transformation are globally available to all **InfoCubes**.

try/catch block

A combination of a try object and one or more catch objects that define alternate execution paths in case an error occurs during the execution of a **job**. You can tune try/catch blocks to trap specific errors and to provide general alerts or messages if an error occurs.

Update Rule

In **SAP BW**, a data manipulation rule between a **Communication Structure** and an **InfoCube**. The result of the data manipulation is specific to one InfoCube.

variable

A symbolic placeholder for a value. Data Integrator lets you define local variables and global variables.

Local variables pertain to the **work flow** in which they are defined. You can pass the value into another **work flow** or **data flow** using a **parameter**.

Global variables pertain to the **job** in which they are defined. With global variables, there is no need to define parameters between **objects** in the job. Global variables can also be selected at execution time. This eliminates the need to open the Designer to set global variable values.

Web Services

A standard platform for integrating applications. Web Services allow different programs, constructed in different languages, on different platforms to communicate with each other. DATA INTEGRATOR supports three XML web service technologies: SOAP, WSDL, and **XML Schema**.

work flow

A **reusable object** containing **steps** to define the order of **job** execution. Work flows call **data flow**s, but cannot manipulate data themselves.

You call work flows from inside other work flows or **job**s. You pass information into or out of work flows using **parameter**s. You reuse work flows by dragging existing work flows from the **object library**.

workspace

The window inside the **Designer** in which you define, display, and modify **objects**. The workspace for a **data flow** contains an area to build a **diagram** representing the data flow definition. The workspace for a **transform** contains an editor for modifying the transform options.

WSDL

Web Services Definition Language. Web services are self-contained, modular business process applications based on open Internet standards.

XML

Extensible Markup Language. This markup language is like HTML (Hypertext Markup Language) in that it specifies a standard with which you can define your own markup languages with their own sets of tags. XML allows you to define various tags with various rules, such as tags that represent business rules, tags that represent data description, or tags that represent data relationships.

XML Schema

The XML format used by Data Integrator to support message processing that includes **Web Services**. XML Schemas describe the data structure of an **XML** file or message. Data flows can read and write data to messages or files based on a specified XML Schema format. You can use the same XML Schema to describe multiple XML sources or targets. XML Schema properties include: Name, Description, Imported from, Root element name, and Namespace.

Index

Business Objects, Americas
3030 Orchard Parkway
San Jose, CA 95134
USA

info@businessobjects.com
www.businessobjects.com

Data Integrator™

Supplement for SAP

Version 6.5

Document Number DI-65-0107-001

January 30, 2004

Contents

1 Welcome

Welcome to the *Data Integrator Supplement for SAP*. This guide contains information about how to use Data Integrator with SAP R/3.

This chapter contains the following sections:

SAP Interfaces

Data Integrator has four SAP interfaces:

- SAP R/3 ABAP
- SAP R/3 BAPI
- SAP R/3 IDoc
- SAP BW

Each interface is a license-controlled feature that can be purchased separately and installed with Data Integrator. When you purchase the SAP R/3 ABAP interface, you can use an additional type of data flow called an R/3 data flow.

The following table lists SAP interfaces, their intended functionality, and the types of data flows you can use with each.

Licensed controlled feature	How is this feature designed to be used?	Which data flow should be used?
SAP R/3 ABAP	Use an SAP R/3 datastore to import metadata (tables, files, hierarchies, and functions) to create batch jobs to extract data from SAP R/3 sources. See "SAP R/3 ABAP Interface options" on page 216 for a list of options added to the Designer when this feature is installed.	R/3 data flow
SAP R/3 BAPI	Use an SAP R/3 datastore to import BAPI function metadata to be included in batch or real-time jobs. BAPI functions can be used to update SAP R/3 and SAP BW sources.	data flow
SAP R/3 IDoc	Use an SAP R/3 datastore to import IDoc metadata which can be used to create the following for batch and real-time jobs: For batch jobs • IDoc file sources • IDoc message sources • IDoc message targets For real-time* jobs • IDoc file sources • IDoc message sources • IDoc file targets • IDoc message targets	data flow
SAP BW	Use an SAP BW datastore to import BW metadata to create batch jobs that can be run from SAP BW or Data Integrator to update SAP BW targets.	data flow (batch jobs only)

About this document

This book is organized to mirror the information sequence in the *Data Integrator Designer Guide*. The final chapter is designed to mirror the information in the *Data Integrator Reference Guide*.

This book includes the following chapters:

- Chapter 2, "Data Integrator in the SAP R/3 Environment"
- Chapter 3, "SAP Datastores"
- Chapter 4, "SAP R/3 and File Formats"
- Chapter 5, "SAP R/3 and R/3 Data Flows"
- Chapter 6, "IDocs and Batch Jobs"
- Chapter 7, "SAP R/3 and Real-Time Jobs"
- Chapter 8, "Executing Batch Jobs that Contain R/3 Data Flows"
- Chapter 9, "Executing Batch Jobs in the SAP BW Environment"
- Chapter 10, "Capturing Changed Data in SAP R/3"
- Chapter 11, "Reference information"

More Data Integrator product documentation

Consult the *Data Integrator Getting Started Guide* for:

- An overview of Data Integrator products and architecture
- Data Integrator installation and configuration information
- A list of product documentation and a suggested reading path

After you install Data Integrator, you can view technical documentation from many locations. To view documentation in PDF format, you can:

- Select **Start** > **Programs** > **Data Integrator** ***version*** > **Data Integrator Documentation** and choose:
  - Release Notes
  - Release Summary
  - Technical Manuals
- Select one of the following from the Designer's **Help** menu:
  - Release Notes
  - Release Summary
  - Technical Manuals
- Select **Help** from the Data Integrator Administrator

You can also view and download PDF documentation by visiting Business Objects Customer Support online. To access this Web site, you must have a valid user name and password. To obtain your user name and password, go to http://www.techsupport.businessobjects.com and click **Register**.

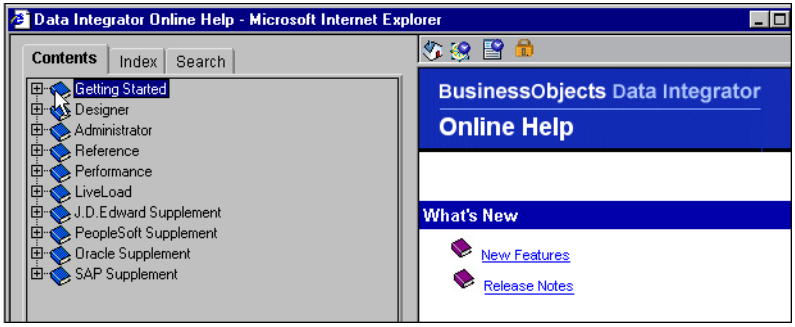
After installing Data Integrator Designer you can also view documentation in HTML format. **Online Help** opens automatically the first time you open the Designer.

You can also open **Help,** using one of the following methods:

- Choose **Contents** from the Designer's Help menu.
- Click objects in the object library or workspace and press **F1**.

  Online Help opens to the subject you selected.

Use **Online Help's** links and tool bar to navigate:



Online help tool bar

Icon	Icon Name	Description
	Home	Returns to the Online Help home page.
	Show/Hide Navigation	Shows and hides the Table of Contents for the core suite of Data Integrator Technical Manuals. Includes a master **Index** and **Search** tabs.
	Show/Hide Descriptions	The Online Help window automatically updates its information when you click an object in the Designer window. In addition, you can click any object in the Designer window then click this button to show or hide an object's local and global variables.
	Start/Stop Help Updates	When the lock appears open, each time you click an object in the Designer window, the Online Help information automatically updates.
		When the lock icon appears closed, the Online Help window does not automatically update when you click objects in the Designer window; you must press F1.

2 Data Integrator in the SAP R/3 Environment

This chapter covers the following SAP R/3 interfaces:

- SAP R/3 ABAP
- SAP R/3 BAPI
- SAP R/3 IDoc

These interfaces must use an Data Integrator SAP R/3 datastore connection.

If you are only planning to use the SAP BW interface, skip forward to "SAP BW datastores" on page 55.

This chapter includes the following sections:

Using Data Integrator in SAP R/3 environments

Data Integrator provides three basic methods for moving data into and out of SAP R/3:

- ABAP interface — Reads data from SAP R/3, using R/3 data flows, and primarily supports tables (transparent, pool and cluster).
- RFC/BAPI interface — Loads data to SAP R/3 (since a function is handled as a nested column within a query, it can return SAP R/3 tables, structures or scalar parameters, which can be loaded into any target) and allows direct access to R/3 tables outside of the R/3 data flows through RFC function calls.
- IDoc interface — Reads from and loads data to SAP R/3.

You can search an SAP R/3 application server for tables, IDocs, functions, and hierarchies. You can import metadata for them into Data Integrator using an SAP R/3 datastore connection.

Once metadata is imported, Data Integrator provides:

- R/3 data flows — Move table and hierarchy data between SAP R/3 and Data Integrator using ABAP/4 (SAP' s proprietary fourth-generation language).
- Data flows in batch jobs — Can contain R/3 data flows (as sources), tables, or IDocs (as sources and targets), and SAP R/3 functions as needed.
- Data flows in real-time jobs — Move table and IDoc data (using SAP R/3 functions as needed) between SAP R/3 and Data Integrator.

R/3 data flow processing using ABAP

In Data Integrator, R/3 data flows extract SAP R/3 data from tables and follow this general execution process:

- Data Integrator generates the ABAP code

- Data Integrator connects to the SAP R/3 application server through remote function call (RFC).
- The SAP R/3 application server executes the ABAP code.
- The ABAP program results are generated and communicated back to Data Integrator, which then loads the target data cache.

Batch and real-time processing using IDocs

An SAP R/3 IDoc is an intermediate document that can be used to send and receive business documents between R/3 application servers and other servers.

SAP R/3 offers numerous IDocs that describe different business documents. The data in an IDoc supports a particular transaction. An IDoc can contain data from multiple tables and can contain hierarchical data.

Data Integrator processes outbound and inbound IDocs from and to the SAP R/3 application server. SAP outbound IDocs are called, *IDoc sources* in Data Integrator. SAP inbound IDocs are called *IDoc targets*. As with other target objects in Data Integrator, IDoc targets can be used primarily to update a data cache and to update a back-office system, such as SAP R/3, on a conditional basis.

The method of processing IDocs depends upon the type of job in which you define the IDoc and whether the IDoc is handled as a message or a file.

In Data Integrator, an IDoc can be used as a source or target object and processed as a message or as a file. The only limitation is that batch jobs cannot process IDoc messages as sources.

	IDoc file	IDoc message
Batch job source?	Yes	No
Batch job target?	Yes	Yes
Real-time job source?	Yes	Yes
Real-time job target?	Yes	Yes

For example, Data Integrator real-time processes IDoc source messages (outbound IDocs from the SAP R/3 application server) as follows:

- Your SAP R/3 system administrator includes the Access Server as a TCP, remote function call (RFC) destination.
- The SAP R/3 application server (*client*) publishes an IDoc after processing a transaction. For example, when an operator enters a sales order.
- The Access Server receives the published IDoc and dispatches it to an available service provider for processing.
- If no service providers are available for this IDoc type, the Access Server queues the request and dispatches it when a service provider becomes available.
- When available, a service provider processes the IDoc. The IDoc data can trigger queries to the SAP R/3 application server or cached data as required to enrich the IDoc data. The real-time job flattens and loads the required information into a data cache and/or an IDoc message target as specified in its data flow designs.
- The service provider returns a response to the Access Server indicating the successful IDoc processing.

- The Access Server receives the response and sends an acknowledgment to the SAP R/3 application server so the IDoc can be marked as processed.

For more information about using IDocs with Data Integrator, see the following chapters:

SAP Guide topics	Link to IDoc information
Chapter 2, SAP R/3 environment	"SAP R/3 IDoc interface connectivity" on page 15
Chapter 3, SAP R/3 datastores	"Browsing, searching and importing metadata from SAP R/3" on page 48
Chapters 6 and 7	Chapter 6, "IDocs and Batch Jobs" Chapter 7, "SAP R/3 and Real-Time Jobs"
Chapter 11, Reference Information (under Objects)	"IDoc file" on page 232 "IDoc message" on page 234 "Source" on page 250 "Target" on page 259

Working with functions

SAP R/3 remote function calls (RFCs) can be used within queries created in data flows. Data Integrator supports remote function calls. In addition, Data Integrator offers the SAP R/3 BAPI interface to support the use of remote function calls designed for business transactions (BAPIs).

If you are using a license that includes the BAPI interface, you can use an SAP R/3 datastore to import BAPI function metadata. See "SAP R/3 RFC and BAPI function calls" on page 269 for more information.

SAP *normal functions* cannot be used in data flows. Instead, Data Integrator supports the use of normal functions and RFC functions in *R/3 data flows* with the following restrictions:

- The function can have only scalar, multiple input parameters. The function cannot have structure input or table parameters.
- The output parameter must be a single scalar parameter.

  In the following SAP normal function, valid input and output parameters are shown.

  ```
  RP_PERIOD_SALARY_FROM_ANSAL
  ```

  Input parameters:

  ```
  F_PERNR LIKE P0008-PERNR
  F_ANSAL LIKE P0008-ANSAL
  F_DATE LIKE Q0008-IBBEG
  ```

  Output parameters:

  ```
  F_PERIOD_SALARY LIKE P0008-BET01
  F_PERIODS_OUT TYPE P
  ```

Data Integrator cannot use normal functions in data flows because an SAP R/3 normal function is not an RFC function.

However, you can write a wrapper for a normal function, which can change the function to RFC enabled. In this case, a normal function would be supported in data flows.

SAP R/3 ABAP interface connectivity

Use the following information to install and configure Data Integrator when working in an SAP R/3 environment:

- Data transport requirements
- SAP R/3 permissions
- Security profile
- Server connectivity—host names

Data transport requirements

When you configure R/3 datastores in Data Integrator, you must specify the data transport method. This method defines how data that is extracted from the SAP R/3 server becomes available to the Data Integrator server (the computer on which the Data Integrator Job Server runs). The four options are:

- Custom Transfer — SAP R/3 server loads the **Working Directory on SAP server** with the transport file. The file is read by a third-party file transfer (custom transfer) program and loaded to the **Local directory**.
- Direct download — SAP R/3 server transfers the data directly to the **Local directory**.
- FTP — SAP R/3 server loads the **Working Directory on SAP server** with the transport file. The file is read using the **FTP relative path to the SAP working directory** and written to the **Local directory**.
- Shared directory — SAP R/3 server loads the **Working Directory on SAP server** with the transport file. The file is read using the **Data Integrator Path to the shared directory**.

See "Data transport methods" on page 169 for more information.

SAP R/3 permissions

Data Integrator uses different execution methods to execute generated ABAP in design, development, test, and production environments. Each method has different security requirements. See "SAP R/3 security levels", for detailed information and planning worksheets. Also see "SAP R/3 security levels" if you have security requirements for extracting from SAP BW.

Security profile

ABAP programs generated by Data Integrator can verify the permissions of the user executing the program. If a security profile is provided, the generated ABAP verifies that the user executing the program has the permissions specified by that profile. This security profile should be transported to the production system prior to running Data Integrator against it.

Security Profile Name: _____

Server connectivity—host names

The computer on which the Data Integrator Designer is installed should be able to access the computer on which the Data Integrator Job Server is installed. Similarly, the Data Integrator Job Server computer should be able to access the Data Integrator Designer computer *by name*. Use the `ping` command to verify that both can access each other by name, not IP address.

Host Name: _____

SAP R/3 IDoc interface connectivity

Your SAP R/3 system should be configured to send and receive IDocs from Data Integrator. Refer to the SAP R/3 implementation guide (transaction SPRO) and other SAP R/3 documentation for more information.

SAP R/3 versions 3.0 and higher are supported in Data Integrator. Extended/reduced IDoc Types are supported from SAP R/3 application versions 4.0A and higher.

If you are using version 3.x, contact Business Objects Technical Support for help in adjusting your Data Integrator configuration.

IDoc message *sources* (SAP R/3outbound IDocs) must be configured in the Data Integrator Designer and in the RFC client branch of the Data Integrator Administrator. See "Source" on page 250 and "RFC clients" on page 77 of the *Data Integrator Administrator Guide* for more information.

However, configuring RFC clients for IDoc message *targets* (SAP R/3 inbound IDocs) is unnecessary. The partner number, partner type, message type, and IDoc type settings you save in the Designer's IDoc Target Editors are the ones needed to send messages to SAP R/3 via transactional remote function calls (TRFCs).

For more information on importing and using IDocs in Data Integrator jobs, refer to IDocs in Chapter 3, "SAP Datastores".

The remainder of this chapter covers Data Integrator functions, and SAP R/3 security levels and user authorizations. These sections all apply to the ABAP, BAPI and IDoc interfaces.

Installing Data Integrator functions on SAP R/3

The default Data Integrator installation process places two Data Integrator function module files for SAP R/3 in the `Data Integrator\Admin\R3_Functions\transport` directory. These files are later loaded to SAP R/3 using the SAP Correction and Transport System (CTS). Using CTS allows for version control as Data Integrator functions are improved. There are two different upload files used for installing the Data Integrator functions that support the use of the ABAP, BAPI and IDoc interfaces. They are associated with different SAP R/3 versions.

See the `readme.txt` file under the Data Integrator/admin/R3_Functions/transport directory to get the name of the latest transport file used to install or upgrade SAP R/3 and SAP BW functions.

You will need some or all of these functions in normal operation of Data Integrator in an SAP R/3 environment. These functions perform such operations as dynamically loading and executing ABAP programs from Data Integrator, efficiently running pre-loaded ABAP programs, allowing for seamless metadata browsing and import from SAP R/3, and reporting the status of running jobs.

Functions and the CTS system

If you are installing Data Integrator for the first time, or updating to a new release, the Data Integrator functions must be added to the SAP R/3 environment. We recommend using the CTS system.

The following instructions work for updates and for new installations. These instructions anticipate that the user installing the functions is familiar with CTS and has the authorization to perform the transport.

➢ To install Data Integrator-supplied functions

1. Copy the Data Integrator supplied transport files to the appropriate directories on your SAP R/3 server.

   NOTE: In these steps, `900XXX.SXX` is a variable. To substitute the correct file name for the current release of Data Integrator, see the `readme.txt` file under the Data Integrator/admin/R3_Functions/transport directory.

   - Copy `R900XXX.SXX` to `/usr/sap/trans/data` directory.
   - Copy `K900XXX.SXX` to `/usr/sap/trans/cofiles` directory.

2. Open SAP GUI and run `/nSE37` to verify that function group `ZAW0` does not exist.
   - If function group `ZAW0` exists and contains previously installed Data Integrator functions, add an unconditional mode 2 option (parameter "U2") to the `tp import` command described in step 4 below (`tp import S<xx>K900<xxx> <SID> U2`).
   - If function group `ZAW0` exists and is used for another purpose, modify the transport file to use another function group to install Data Integrator functions. Make sure the function group you use does not already exist in your system.

3. From a command window, run the following transport command:

   `tp addtobuffer S<xx>K900<xxx> <SID>`

   (where SID is the SAP System ID)

   You will receive a response:

   ```
   This is tp version <SAP TP and SAP versions>
   for <database type> database.

   Addtobuffer successful for S<xx>K900<xxx>
   ```

```
tp finished with a return code: 0

meaning:

  Everything OK
```

4. Run the next transport command:

   ```
   tp import S<xx>K900<xxx> <SID>
   ```

   You will receive a response:

   ```
   This is tp version <SAP TP and SAP versions>
   for <database type> database.

   This is \\<SAP servername>\
   sapmnt\<SID>\SYS\exe\run\R3trans.exe
   version 6.03 <SAP release number>

   tp finished with return code: 0

   meaning:

     Everything OK
   ```

5. Run `/nSE37` again to verify that the functions were uploaded.

Manually uploading Data Integrator functions to SAP R/3

If you are not able to use the CTS to upload Data Integrator functions you can manually install them. In this case, you will need definitions for the following structures:

- ZACTA
- ZTAB2048

The table ZACTA is created as follows:

Name	ZACTA Transparent table

Short text Data Integrator Table

Field name	Key data element	Data type	Length	Check table	Short text
NAME	X ZAW_NAME	CHAR	120	Acta Name	Data Element for Table ZACTA
TOTAL_ROW	ZAW_ROW	INT4	10	Acta Row	Data Element for Table ZACTA
UPDATE_DAT	ZAW_DATE	DATS	8	Acta Date	Data Element for Table ZACTA

The structure ZTAB2048 is created as follows:

Structure name ZTAB2048

Short text Table with a 2048 byte field

Field name	Type name	Data Type	Length	Short text
WA	ZAWCH2K	CHAR	2048	Data Element for ZTAB2048

Data Integrator function modules

Data Integrator function modules include:

Z_AW_ABAP_RUN	Z_AW_READ_TEXT
Z_AW_AUTH_IMPORT	Z_AW_RFC_ABAP_INSTALL_AND_RUN
Z_AW_BW_QUERY	Z_AW_RFC_READ_TABLE
Z_AW_COLUMN_SEARCH	Z_AW_SYNTAX_CHECK
Z_AW_FILE_ROWCOUNT	Z_AW_TABLE_IMPORT
Z_AW_FUNCTION_GET	Z_AW_TABLE_SEARCH
Z_AW_IDOC_IMPORT	Z_AW_TREE_IMPORT
Z_AW_IDOC_SEARCH	Z_AW_TREE_IMPORT40
Z_AW_JOB_LOG	Z_AW_TREE_NAVIGATE
Z_AW_JOB_RUN	Z_AW_TREE_NAVIGATE40
Z_AW_JOB_STATUS	Z_AW_TREE_SEARCH
Z_AW_MODEL_NAVIGATE	Z_AW_TREE_SEARCH40

The appropriate Data Integrator function modules for your SAP R/3 version are installed.

Packaging of Z_AW_RFC_ABAP_INSTALL_AND_RUN

For SAP R/3, Data Integrator provides a function called `Z_AW_RFC_ABAP_INSTALL_AND_RUN` in its transport file. This function is used in the **generate_and_execute** mode (set in an SAP R/3 datastore) to generate ABAP code on Data Integrator and execute it on an SAP server. It allows dynamic program generation and is particularly useful in a development environment. However, it also provides a level of write access to an SAP server that may not be acceptable to your security environment. If this is a concern, remove this function from your SAP R/3 server. For more information about this function's parameters see, "Interface information and source code table" on page 21.

Sample Data Integrator function module

Following is the Data Integrator function module Z_AW_SYNTAX_CHECK:

```
FUNCTION Z_AW_SYNTAX_CHECK.

*"----------------------------------------

*"*"Local interface:

*"    EXPORTING
*"         VALUE(ERRORMESSAGE) LIKE SY-MSGV1

*"    TABLES
*"         PROGRAM STRUCTURE PROGTAB OPTIONAL

*"----------------------------------------
```

```
DATA: MESS(72), LIN(72), WRD(72).
SYNTAX-CHECK FOR PROGRAM MESSAGE MESS LINE LIN
WORD WRD.
IF MESS <> SPACE.

  ERRORMESSAGE = MESS.

  EXIT.
ENDIF.

ENDFUNCTION.
```

Interface information and source code table

The following table lists information about the Data Integrator functions for SAP R/3, their interfaces, and function source code. Descriptions include these characters to indicate parameter types:

- I - Import Parameter
- E - Export Parameter
- T - Table Parameter

Data Integrator functions to upload to SAP R/3

Function name	Parameter type / Parameter name / Reference field Exception
Z_AW_ABAP_RUN	Z_AW_ABAP_RUN executes generated or preloaded ABAP programs. I / PROGRAMNAME / SY-REPID E / ERRORMESSAGE / SY-MSGV1 T / SELTAB / RSPARAMS LOG / LISTZEILE **Exception(s):** PROGRAM_DOES_NOT_EXIST
Z_AW_AUTH_IMPORT	Z_AW_AUTH_IMPORT imports SAP R/3 authorization profiles used by Z_AW_ABAP_RUN. I / AUTHPROF / USR10-PROFN T / AUTHOBJ / LISTZEILE **Exception(s):** SECURITY_PROFILE_DOESNOT_EXIST
Z_AW_BW_QUERY	Z_AW_BW_QUERY gets metadata information from an SAP BW system. I / P_CODE / DD01D-DATATYPE I / P_IAREA / DD01D-DDTEXT I / P_ICUBE / DD01D-DDTEXT I / P_ODSO / DD01D-DDTEXT T/ ENTRIES / TAB512
Z_AW_COLUMN_SEARCH	Z_AW_COLUMN_SEARCH is currently not used and is reserved for future implementation. I / P_TNAME / DD03VT-DDTEXT I / P_CNAME / DD03VT-DDTEXT I / P_CDESC / DD03VT-DDTEXT I / P_LANG / DD03VT-DDLANGUAGE I / P_MAXROWS / SY-TABIX T / P_RETURN / LISTZEILE
Z_AW_FILE_ROWCOUNT	Z_AW_FILE_ROWCOUNT puts/gets the row count information for each transported file in Data Integrator R/3 data flow into/from ZACTA table. I / NAME/ ZACTA-NAME I / ROWCOUNT/ ZACTA-TOTAL_ROW I / SET / SONV-FLAG E / TOTAL_ROW / ZACTA-TOTAL_ROW
Z_AW_FUNCTION_GET	Z_AW_FUNCTION_GET gets the function interface I / FUNCNAME/TFDIR-FUNCNAME T / PRMTAB/ CATFU

Data Integrator functions to upload to SAP R/3 (Continued)

Function name	Parameter type / Parameter name / Reference field Exception
Z_AW_IDOC_IMPORT	Z_AW_IDOC_IMPORT imports IDocs from SAP R/3. I / IDOCTYP / EDISYN-DOCTYP I / LANG / EDISEGT-LANGUA T / SEGMENTS / LISTZEILE **Exception(s):** IDOC_TYPE_DOESNOT_EXIST
Z_AW_IDOC_SEARCH	Z_AW_IDOC_SEARCH searches existing IDocs in SAP R/3. I / P_NAME / EDIDOT-DOCTYP I / P_DESC / EDIDOT-DESCRP I / P_LANG / EDIDOT-LANGUA I / P_MAXROWS / SY-TABIX I / P_FLAG / SY-TABIX T / P_RETURN / LISTZEILE
Z_AW_JOB_LOG	Z_AW_JOB_LOG is an Data Integrator-supplied function that provides error handling and retrieves the SAP R/3 job log when a job is cancelled. This function is required for the "execute preloaded" access method you can define for an R/3 datastore. I / JOBCOUNT / TBTCO-JOBCOUNT I / JOBNAME / TBTCO-JOBNAME I / HOST / TBTCO-BTCSYSREAX T / LOG / BTCTLE **Exception(s):** JOB_NOT_FOUND JOB_LOG_NOT_FOUND JOB_NUMBER_NOT_VALID LOGNAME_WRONG_FORMAT LOGNAME_MISSINGLOG_HANDLE_ERROR LOG_IS_EMPTY LOG_NOT_FOUND TEMSE_CONVERSION_NOT_POSSIBLE TEMSE_ERROR TEMSE_FUNCTION_ERROR

Data Integrator functions to upload to SAP R/3 (Continued)

Function name	Parameter type / Parameter name / Reference field Exception
Z_AW_JOB_RUN	Z_AW_JOB_RUN submits ABAP jobs for background execution. I / PROGRAMNAME / SY-REPID I / JOBGROUP / TBTCO-JOBGROUP I / JOBNAME / TBTCO-JOBNAME I / HOST / TBTCO-BTCSYSREAX I / JOBCLASS / TBTCO-JOBCLASS E / JOBCOUNT / TBTCO-JOBCOUNT E / JOB_RELEASED / BTCH0000-CHAR1 T / SELTAB / RSPARAMS **Exception(s):** ABAP_PROGRAM_SYNTAX_ERROR CANT_CREATE_JOB INVALID_JOB_DATA JOBNAME_MISSING CANT_START_IMMEDIATE INVALID_STARTDATE JOB_CLOSE_FAILED JOB_NOSTEPS JOB_NOTEX LOCK_FAILED ABAP_PROGRAM_DOES_NOT_EXIST
Z_AW_JOB_STATUS	Z_AW_JOB_STATUS queries ABAP job status after a job is successfully submitted. I / JOBNAME / TBTCO-JOBNAME I / JOBCOUNT / TBTCO-JOBCOUNT I / HOST / TBTCO-BTCSYSREAX E / STATUS / TBTCO-STATUS **Exception(s):** JOB_NOT_FOUND
Z_AW_MODEL_NAVIGATE	Z_AW_MODEL_NAVIGATE allows you to navigate through the SAP data model, helping to identify tables. I / P_OID / DM41S-DMOID I / P_LANG / DD02T-DDLANGUAGE T / P_RETURN / LISTZEILE

Data Integrator functions to upload to SAP R/3 (Continued)

Function name	Parameter type / Parameter name / Reference field Exception
Z_AW_READ_TEXT	Z_AW_READ_TEXT allows extraction of text from SAP R/3. This function is an RFC-enabled version of the SAP R/3 function module, READ_TEXT. I/CLIENT/SY-MANDT I/ID/THEAD-TDID I/LANGUAGE/THEAD-TDSPRAS I/NAME/THEAD-TDNAME I/OBJECT/THEAD-TDOBJECT I/ARCHIVE_HANDLE/SY-TABIX E/HEADER/THEAD T/LINES/TLINE **Exception(s)**: ID — Text ID invalid LANGUAGE — Invalid language NAME — Invalid text name NOT_FOUND — Text not found OBJECT — Invalid text object REFERENCE_CHECK — Reference chain interrupted WRONG_ACCESS_TO_ARCHIVE — Archive handle invalid
Z_AW_RFC_ABAP_INSTALL_AND_RUN	Z_AW_RFC_ABAP_INSTALL_AND_RUN allows third-party tools such as Data Integrator to dynamically generate and run programs on an SAP server. I / MODE / SY-MSGTY I / PROGRAMNAME / SY-REPID E / ERRORMESSAGE / SY-MSGV1 T / PROGRAM / PROGTAB T / WRITES / LISTZEILE

Data Integrator functions to upload to SAP R/3 (Continued)

Function name	Parameter type / Parameter name / Reference field Exception
Z_AW_RFC_READ_TABLE	Z_AW_RFC_READ_TABLE allows extraction of SAP R/3 3.x and 4.x table data from data flows. I / QUERY_TABLE/DD02L-TABNAME I / DELIMITER/SONV-FLAG DEFAULT SPACE I / NO_DATASONV-FLAG DEFAULT SPACE I / ROWSKIPS/SOID-ACCNT DEFAULT 0 I / ROWCOUNT/SOID-ACCNT DEFAULT 0 T / OPTIONS/RFC_DB_OPT T / FIELDS/RFC_DB_FLD T / DATA/ZTAB2048 **Exception(s):** TABLE_NOT_AVAILABLE TABLE_WITHOUT_DATA OPTION_NOT_VALID FIELD_NOT_VALID NOT_AUTHORIZED DATA_BUFFER_EXCEEDED
Z_AW_SYNTAX_CHECK	Z_AW_SYNTAX_CHECK performs a syntax check for generated or preloaded ABAP. This function is required for the "execute preloaded" access method you configure in an R/3 datastore. E / ERRORMESSAGE / SY-MSGV1 T / PROGRAM / PROGTAB

Data Integrator functions to upload to SAP R/3 (Continued)

Function name	Parameter type / Parameter name / Reference field Exception
Z_AW_TABLE_IMPORT	Z_AW_TABLE_IMPORT imports SAP R/3 table definitions into the Data Integrator repository. I / TABNAME / DD03L-TABNAME I / LANG / DD02T-DDLANGUAGE E / DESCRIPTION / DD02T-DDTEXT E / TABCLASS / DD02L-TABCLASS T / COLUMNS / LISTZEILE T / INDEXES / LISTZEILE T / FKEYS / LISTZEILE **Exception(s):** NO_TABLECLASS_FOR_TABLE
Z_AW_TABLE_SEARCH	Z_AW_TABLE_SEARCH allows searches for SAP R/3 tables. I / P_NAME / DD02-DDTEXT I / P_DESC / DD02VV-DDTEXT I / P_LANG / DD02V-DDLANGUAGE I / P_MAXROWS / SY-TABIX I / P_FLAG / SY-TABIX T / P_RETURN / LISTZEILE

Z_AW_TREE_NAVIGATE	`Z_AW_TREE_NAVIGATE40` allows GUI-based navigation of an SAP R/3 version 4.x environment.
	`I / P_LANG / DD04T-DDLANGUAGE`
	`I / P_TREEAREA / DD02D-DDTEXT`
	`I / P_TREEGROUP / DD04L-ROLLNAME`
	`I / P_TREETABLE / DD02D-TABNAME`

Data Integrator functions to upload to SAP R/3 (Continued)

Function name	Parameter type / Parameter name / Reference field Exception
Z_AW_TREE_IMPORT	Z_AW_TREE_IMPORT provides a GUI method to import tables and fields.
	I / P_LANG / DD04T-DDLANGUAGE
	I / P_TREEAREA / DD02D-DDTEXT
	I / P_TREEGROUP / DD04L-ROLLNAME
	I / P_TREETABLE / DD02D-TABNAME
	E / DESCRIPTION / DD04T-DDTEXT
	E / CONTROLLING_AREA / DD02D-DDTEXT
	E / CHART_OF_ACCT / DD02D-DDTEXT
	E / TABLE_NAME / DD02D-TABNAME
	E / FIELD_NAME / DD03D-FIELDNAME
	E / DOMAIN_NAME / DD03L-DOMNAME
	E / CHECK_TABLE / DD03L-CHECKTABLE
	E / CONT_AREA_LIST / TAB512-WA
	T / COLUMNS / LISTZEILE
Z_AW_TREE_SEARCH	Z_AW_TREE_SEARCH is an Data Integrator-supplied function that allows GUI-based search and select in an SAP R/3 environment.
	I / TREENAME / DD08T-MESTEXT
	I / TREEDESC / DD04T-DDTEXT
	I / LANG / SY-LANGU
	I / MAX_ROWS / SY-TABIX
	I / FLAG / SY-TABIX
	T / P_RETURN / LISTZEILE
Z_AW_TREE_NAVIGATE40	Z_AW_TREE_IMPORT40 provides a GUI-based method to import tables and fields for SAP R/3 version 4.x environments.
	I / P_LANG / DD04T-DDLANGUAGE
	I / P_TREEAREA / DD02D-DDTEXT
	I / P_TREEGROUP / DD04L-ROLLNAME
	I / P_TREETABLE / DD02D-TABNAME
	T / P_RETURN / LISTZEILE

Data Integrator functions to upload to SAP R/3 (Continued)

Function name	Parameter type / Parameter name / Reference field Exception
Z_AW_TREE_IMPORT40	Z_AW_TREE_IMPORT40 provides a GUI-based method to import tables and fields for SAP R/3 version 4.x environments. I / P_LANG / DD04T-DDLANGUAGE I / P_TREEAREA / DD02D-DDTEXT I / P_TREEGROUP / DD04L-ROLLNAME I / P_TREETABLE / DD02D-TABNAME E / DESCRIPTION / DD04T-DDTEXT E / CONTROLLING_AREA / DD02D-DDTEXT E / CHART_OF_ACCT / DD02D-DDTEXT E / TABLE_NAME / DD02D-TABNAME E / FIELD_NAME / DD03D-FIELDNAME E / DOMAIN_NAME / DD03L-DOMNAME E / CHECK_TABLE / DD03L-CHECKTABLE E / CONT_AREA_LIST / TAB512-WA T / COLUMNS / LISTZEILE
Z_AW_TREE_SEARCH40	Z_AW_TREE_SEARCH40 allows GUI-based search and select in an SAP R/3 version 4.x environment. I / TREENAME / DD08T-MESTEXT I / TREEDESC / DD04T-DDTEXT I / LANG / SY-LANGU I / MAX_ROWS / SY-TABIX I / FLAG / SY-TABIX T / P_RETURN / LISTZEILE

SAP R/3 security levels

You can choose from three security levels based on your system requirements. Typically, SAP R/3 development environments have four application phases: development, consolidation, test, and production. ABAP programs are generated in the development phase, consolidated from multiple developers, transported to the test phase for testing, and finally transported to the production phase. The Data Integrator security mechanisms were designed using the SAP R/3 paradigm. The following table displays the various authorization levels and application phases:

User	Authorization level	Application phase
DEVUSER	High	Design or Design and Consolidation
TESTUSER	Middle	Test or Consolidation and Test
PRODUSER	Low	Production

A typical application passes through four phases:

1. In the *design phase*, a high-security profile user designs the data and work flows, and generates ABAP programs. The ABAP programs are dynamically communicated to SAP R/3 and executed.

   After execution, ABAP programs are automatically deleted.

   To do this, the developer must have authorization that could be designated DEVUSER.

2. In the *consolidation phase*, developers upload generated ABAP programs into their SAP R/3 system with predefined program names and preregistered correction numbers. A user in that phase can run these programs with a lower security profile, TESTUSER, for example.

   Consolidation can also occur in the design phase.

Here TESTUSER can create ABAP programs, install SAP R/3 functions, and schedule jobs.

3. In the *test phase*, the Data Integrator repository must be exported to a new repository, and the generated ABAP programs must be transported to the SAP R/3 test system. This is usually performed by TESTUSER.
4. The *production phase* begins when no additional program modifications are required.

   The user involved in the move to the production phase has the lowest security profile. The user in this phase could be called PRODUCER.

SAP R/3 authorization can be managed two ways.

1. Profile generation method

   Every user is assigned to activity groups. Each activity group contains information about allowed activities (such as transaction and menu path). Only users with an activity authorization can perform that activity.

2. Profiles assigned to users method

   Every profile contains authorizations. The user administrator creates new authorizations, assigns them to profiles, and assigns profiles to users.

Data Integrator provides various authorization choices to meet your organization's needs:

- Some organizations use DEVUSER during all the application phases.
- Others do not want to assign DEVUSER authorization, which requires a high security profile, to any individual. In Data Integrator you can use the TESTUSER authorization to:
  - Generate ABAP on the development system
  - Upload it to SAP R/3, then execute

- PRODUSER typically has the lowest authorization. This user can usually only execute ABAP programs to which TESTUSER has been granted authorization.

Data Integrator provides the checking mechanisms necessary to prevent unauthorized users from executing Data Integrator generated programs in your SAP R/3 system. If you specify a security profile in R/3 datastores, (see "Defining SAP R/3 datastores" on page 44) Data Integrator checks authorization before executing the program.

If you do not specify a security profile, the Data Integrator programs do not perform a security check prior to execution.

SAP R/3 user authorizations

This section describes how to define a profile using the profile generation method and the specific user authorization level method. It also lists recommended SAP R/3 security profiles and authorizations for using Data Integrator.

➢ To create an SAP R/3 profile for Data Integrator using profile generation

1. Create a user (for example for SAP DEV, TEST, or PRD systems).
2. Open the SAP Profile Generator (transaction PFCG) available in SAP R/3 versions 4.6 and above.
3. Create an Activity group (Role since SAP 4.6C), for example ZBODI_ROLE.
4. Enter a description for the role.
5. Go to the **Authorizations** tab and click **Change authorization data**.
6. On the **Change Role: Authorizations** screen, click the **Manually**, toolbar icon.
7. The **Manual Selection of Authorizations** window opens.
8. Type in the following authorization objects.

```
S_ADMI_FCD*

S_BTCH_JOB

S_DEVELOP*

S_DATASET

S_PATH

S_RFC

S_TABU_DIS
```

```
S_TCODE

S_RS_ADMWB — for SAP BW
```

* Required if you set the ABAP execution option to generate_and_execute rather than execute_preloaded in a production system.

9. Click OK
10. Return to the **Change Role: Authorizations** screen.
11. Manually configure components by entering the values documented in the following section of this guide: "Authorizations for Data Integrator" on page 37.
12. To complete the security profile, click the **Back** icon (or press F3), select the **User** tab, enter your SAP user ID for Data Integrator and click the **Save** icon.

➢ To define a Data Integrator-specific authorization level

1. Determine which SAP R/3 profile you need.

   Required profiles depend on how you use Data Integrator. See "SAP R/3 profiles" on page 35.

2. In SAP R/3, create the required authorizations for your profiles.

   The required authorizations for your profiles are listed in "SAP R/3 profiles" on page 35. Settings for each authorization are listed in "Authorizations for Data Integrator" on page 37.

   For example, you might create one or more of the following authorizations for Data Integrator operations:

   ```
   ZBODIBTCH
   ZBODIDMI
   ZBODIDEV
   ```

3. Create an SAP R/3 profile.

For example, you might create a profile specific to developing applications in Data Integrator: `BODI_DEV`.

4. Assign the SAP R/3 authorizations to the SAP R/3 profile.
5. Assign the profile to an SAP R/3 user for Data Integrator.

SAP R/3 profiles

The profiles you need to define depend on your use of Data Integrator:

- Development and test profile
- Production profile
- SAP BW loading profile

Development and test profile

To support development and test, create an SAP R/3 profile, such as BODI_DEV, that defines authorizations for DEVUSER or TESTUSER. This profile requires the following authorizations:

- Administration
- Batch
- Development
- File access
- File system access
- RFC calls
- Table source access
- Transactions

Production profile

To support production, create an SAP R/3 profile, such as BODI_PROD, that defines authorizations for PRODUSER. This profile requires the following authorizations:

- Administration[1]
- Batch
- Development[1]
- File access
- File system access
- RFC calls
- Table source access
- Transactions

SAP BW loading profile

If you are using Data Integrator to load data into SAP BW during production, create an SAP R/3 profile, such as BODI_BW_PROD, that defines additional authorizations for PRODUSER. This profile requires the following authorizations:

- BW loading
- RFC calls in BW

1. Required if you set the **ABAP execution option** to **generate_and_execute** rather than **execute_preloaded** in a production system

Authorizations for Data Integrator

Authorizations that support Data Integrator operations include:

- Administration
- Batch
- BW loading
- Development
- File access
- File system access
- RFC calls
- RFC calls in BW
- Table source access
- Transactions

Administration

This authorization allows Data Integrator to perform administrative operations.

Class: Basis

Object: System administration functions

Field	Value
System administration functions	MEMO[a], SUTL

a. Valid for SAP R/3 version 4.5a and later.

Suggested name: **ZBODIDMI**.

Batch

This authorization allows Data Integrator to run ABAP programs in batch mode in SAP R/3.

Class: Basis

Object: Batch processing

Field	Values
Job operation	RELE
Summary of jobs for a group	Any or no value

Suggested name: **ZBODIBTCH**.

BW loading

This authorization allows Data Integrator to load data in SAP BW.

Class: Business Information Warehouse

Object: Administrator Workbench - Objects

Field	Value
Administrator Workbench object	Any or no value
Activity	Any or no value

You could call this authorization **ZBODIBW**.

Development

This authorization sets development workbench settings in SAP R/3.

Class: Basis

Object: ABAP/4 Development Workbench

Field	Value
Activity	03
Authorization group ABAP/4 programs	Any or no value
Development class for transport systems	Same as previous
Object type	PROG, FUGR
Object name	Any or no value

Suggested name: **ZBODIDEV**.

File access

This authorization allows Data Integrator to access files on SAP R/3.

Class: Basis

Object: Authorization for file access

Field	Value
ABAP program name	Any or no value
Activity	Any or no value
Physical file name	Any or no value

Suggested name: **ZBODIFLE**.

File system access

This authorization allows Data Integrator to access the SAP R/3 network and file system.

Class: Basis

Object: File system access via ABAP

Field	Value
Activity	Any or no value
Authorization group for ABAP/4 access to file system	Any or no value

Suggested name: **ZBODIPTH**.

RFC calls

This authorization allows Data Integrator to access SAP R/3 using remote function calls.

Class: Cross-application authorization object

Object: Authorization check for RFC access

Field	Values
Activity	16
Name of RFC to be protected	BAPI, CADR, RFC1, SCAT, SLST, SUNI, SUTL, SDTX, SYST, ZAW0, and any other required function group
Type of RFC object to be protected	FUGR

Suggested name: **ZBODIRFC**.

RFC calls in BW

This authorization allows Data Integrator to load data into SAP BW using remote function calls.

Class: Cross-application authorization object

Object: Authorization check for RFC access (S_RFC)

Field	Value
Activity	16
Name of RFC to be protected	RSAB, CADR, RFC1, SCAT, SLST, SUNI, SUTL, SDTX, SYST, ZAW0
Type of RFC object to be protected	FUGR

NOTE: This is the minimal security requirement for a job loading SAP BW. You might need to create your own user to have the authorization to log in to SAP BW and schedule your Data Integrator BW load job.

Suggested name: **ZBODI_RFC_BW**.

Table source access

This authorization allows Data Integrator to access and extract content from SAP R/3 table sources.

Class: Basis

Object: Table Maintenance (via standard tools such as SM30)

Field	Value
Activity	03
Authorization group	*

Suggested name: **ZBODITABLE**.

Transactions

This authorization allows Data Integrator to complete transactions in SAP R/3.

Class: Cross-application authorization object

Object: Authorization check for transaction start

Field	Value
Transaction Code	SE37, SE38, SU53

You could call this authorization **ZBODITRANS**.

3 SAP Datastores

To use an SAP data source as a Data Integrator source you must define a Data Integrator datastore to serve as the logical link to SAP R/3 or SAP BW, then import your metadata into the Data Integrator datastore. This chapter discusses:

- SAP R/3 datastores
- SAP BW datastores

For general information about Data Integrator datastores, see Chapter 5, "Datastores," in the *Data Integrator Designer Guide*.

SAP R/3 datastores

This section discusses the steps required to use SAP R/3 as a Data Integrator data source:

- Defining SAP R/3 datastores
- Browsing, searching and importing metadata from SAP R/3

Defining SAP R/3 datastores

An SAP R/3 datastore includes the following information for working in an SAP R/3 environment:

- *Connection information*, including the R/3 application server name, the language used by the R/3 client application, and the client and system numbers.

  You supply these values based on your installation of SAP R/3.

- *Data transport method* used to exchange information between Data Integrator and SAP R/3.

  You select the method and supply the appropriate information based on decisions you make about running Data Integrator jobs.

  This section describes a data transport method that is ideal for a development environment. For complete information about available data transport methods and how they affect Data Integrator jobs, see "Data transport methods" on page 169.

- *Security information*, specifically the SAP R/3 security profile to be used by all connections instigated from this datastore between Data Integrator and SAP R/3.

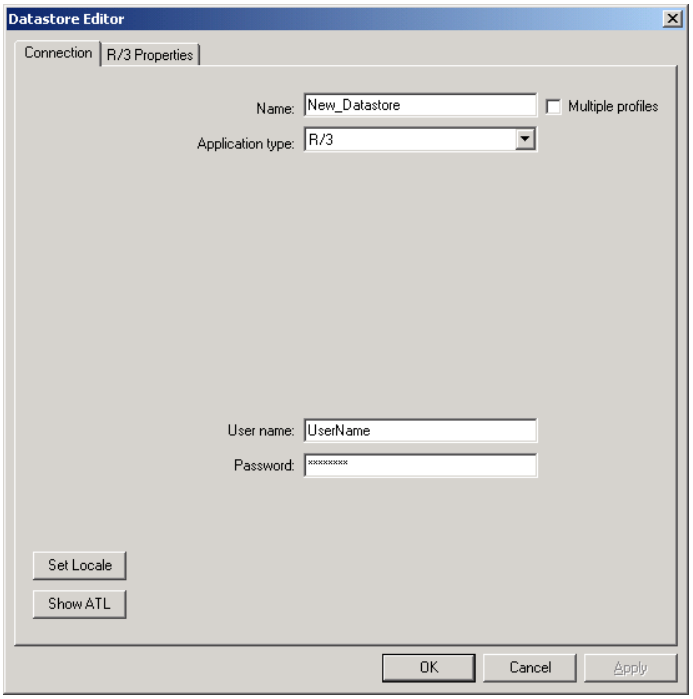
  You supply a security profile based on your installation of SAP R/3. For information, see "SAP R/3 security levels" on page 30.

➢ To define an SAP R/3 datastore

1. In the object library, go to the **Datastores** tab.
2. Right-click inside the object library window and choose **New**.

   The datastore editor opens.
3. Enter a new, unique name for the datastore in the **Name** box.

   The name can contain alpha-numeric characters and underscores (_). It cannot contain spaces.
4. Choose **R/3** in the **Application type** box.



5. Select the **Multiple profiles** check box if you want to specify multiple profiles and be able to change configuration information at runtime.

   After creating a datastore with multiple profiles, you cannot disable this option.

6. Enter the **User name** and **Password** for the R/3 data source.
7. Click to select the **R/3 Properties** tab.

   For initial development, use the following execution and data transport method options:

Option	Description
ABAP execution option	`generate_and_execute` The ABAP created by the Data Integrator job resides on the same computer as the Data Integrator Job Server. It is submitted to R/3 using SAP's RFC_ABAP_INSTALL_AND_RUN.
Data transport method	`shared_directory` Both Data Integrator and SAP R/3 have direct access to the directory where data is stored. This method works best when both systems are on Windows NT and security is not an issue.
Working directory on SAP server	A directory on the SAP R/3 application server where Data Integrator can write intermediate files. For example: `C:\temp\bodi\ds_development`
Data Integrator path to the shared directory	The path to the working directory as accessible through the computer on which the Data Integrator Job Server is running. This path may indicate a mapped network drive. For example: `V:\temp\bodi\ds_development`
Generated ABAP directory	A directory into which Data Integrator writes ABAP files. The path is relative to the Data Integrator Job Server. For example: `C:\bodi\ABAP_programs`

   As a development option, you can connect to an application server from your GUI using a routing string using the specified syntax in the **R/3 application server** field to ensure connection.

The syntax for using an SAP routing string is:

```
/H/"IP Address of local SAP router"/H/"IP Address of target
SAP router"/H/IP Address of target application server
```

For example:

Your SAP routing string (local and target) is
/H/10.10.1.7/H/204.79.199.5
Your application server IP address is 147.204.76.41

Therefore, your routing string would look like this:
/H/10.10.1.7/H/204.79.199.5/H/147.204.76.41

8. Click **OK** to save the datastore.

SAP R/3 security profiles

You can associate a security profile with a datastore so that R/3 data flows that access SAP R/3 sources defined by the datastore include appropriate authorization checking.

R/3 security profiles are useful for:

- Creating profiles of users that can access particular ABAP programs
- Moving ABAP programs from development to test and production

Specify a security profile in the **R/3 Properties** tab of a datastore. By default, Data Integrator does not use any SAP R/3 security profile.

Specify any security profile defined in R/3 (a predefined profile or a profile you defined —see "SAP R/3 user authorizations" on page 33.) Authorization checks are embedded in generated ABAP code.

Browsing, searching and importing metadata from SAP R/3

The processes for browsing, searching, and importing metadata from SAP R/3 are similar to the processes you would use for any source system. To find information on general operations see Chapter 5, "Datastores," in the *Data Integrator Designer Guide*.

There are four categories of SAP R/3 items for which you can import metadata into the Data Integrator object library. You can import metadata for each one by name. You can import metadata for hierarchies, IDocs and tables by searching. You can also browse SAP R/3 to find hierarchies and tables.

SAP R/3 item	Methods for importing metadata
Functions	Name
Hierarchies	Name, search, browse
IDocs	Name, search
Tables	Name, search, browse

➢ To import metadata by name

1. Open the object library.
2. Go to the **Datastores** tab.
3. Right-click the SAP R/3 datastore you want to use and select **Import by Name**. The Import by Name window opens.
4. In the **Type** box select the SAP R/3 item you want to import.
5. In the **Name** box enter the name SAP R/3 uses for the item you want to import. For example, enter SISC001 for an IDoc.

   NOTE: Hierarchies require entering a controlling area and set table.

6. Click **OK**.

Tables and hierarchies

Tables and hierarchies can be imported by browsing the data available through the R/3 datastore.

For more information about importing R/3 hierarchies, see "Extracting data from SAP R/3 hierarchies" on page 89.

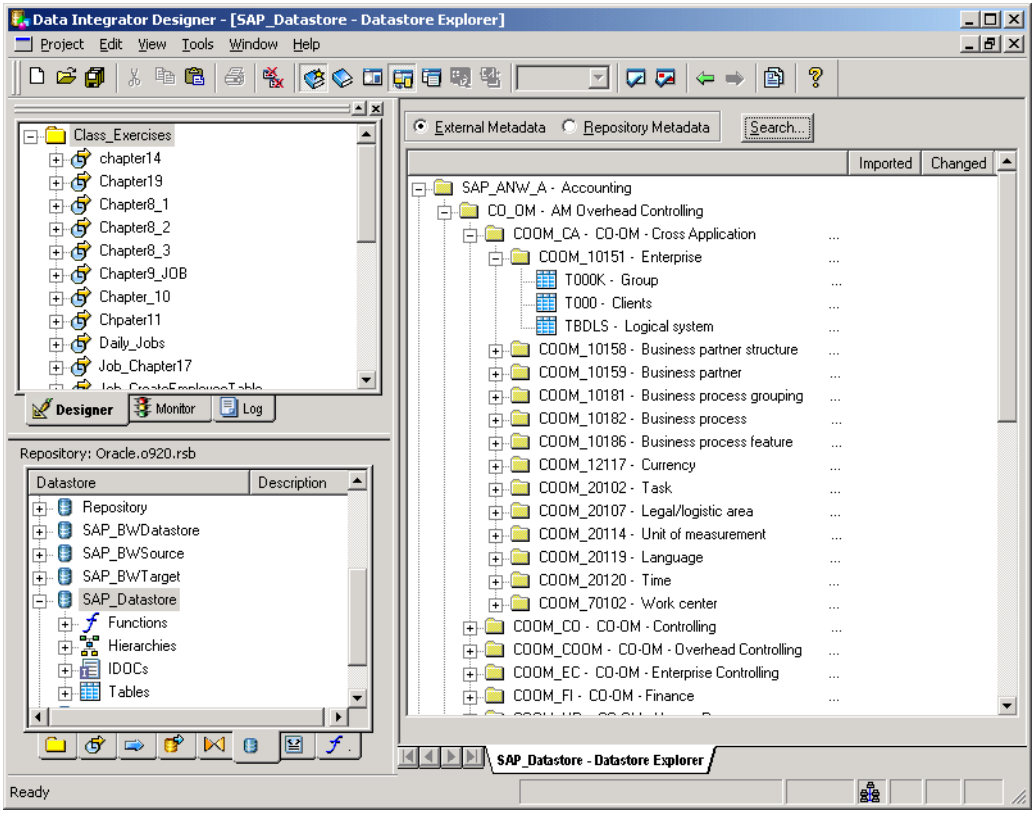
➢ To import table and hierarchy metadata by browsing

1. Open the object library.
2. Go to the **Datastores** tab.
3. Select the datastore you want to use.
4. Right-click and choose **Open**.

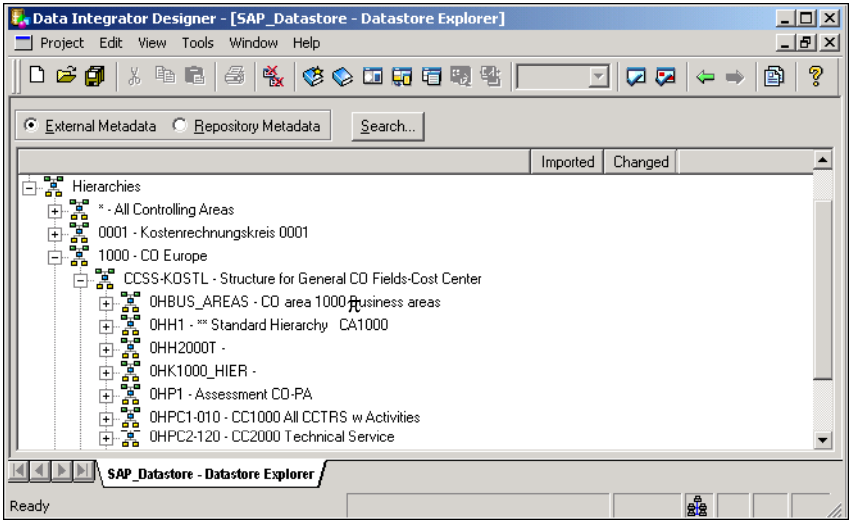
   The items already imported from the datastore appear in the workspace.

5. Right-click in the workspace, and select **External Data**.

   Click the plus sign next to a folder icon to navigate the structure of tables or hierarchies. Tables appear with a table icon.



Hierarchies are also nested according to their use in SAP R/3.



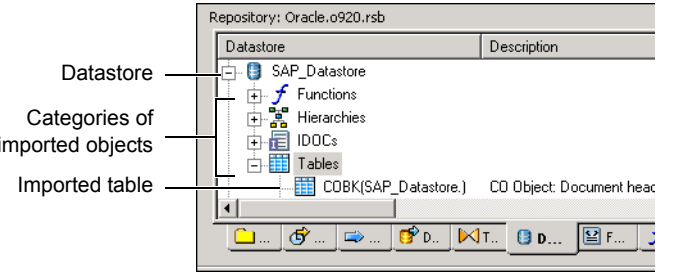
6. Select the items for which you want to import metadata.

   For example, to import a table, you must select a table rather than a folder that contains tables. To import a hierarchy, select the hierarchy group that contains the hierarchy.

7. Right-click and choose **Import**.

8. In the object library, open the datastore list to display the list of imported objects.

   In this example, the COBK table has been imported into datastore SAP_ds.



IDocs

To use IDocs, you must first import the metadata from SAP R/3 to your repository. IDocs can be imported by name or by using the search option.

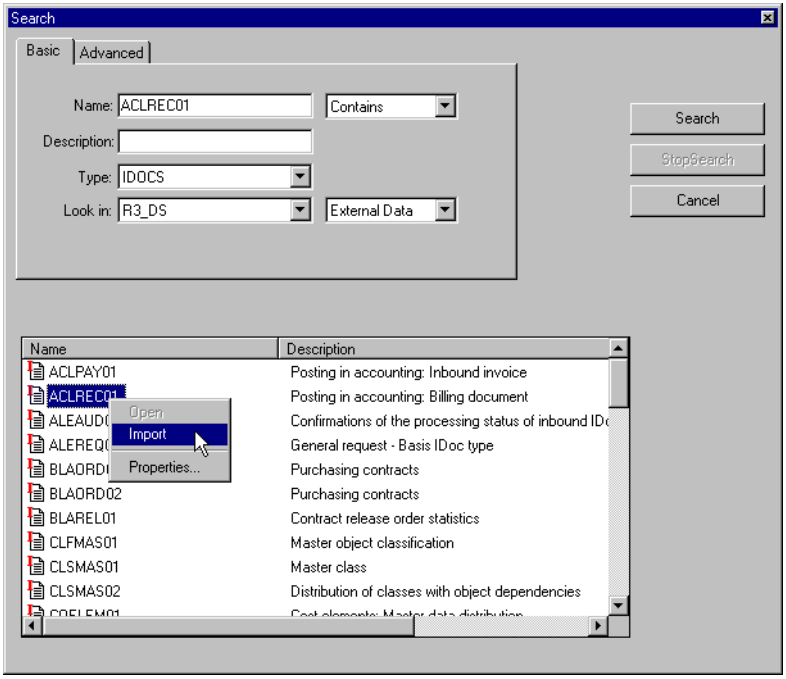
➢ To import IDoc metadata using the search option

1. In the object library, click the **Datastores** tab.
2. Right-click an SAP R/3 datastore and select **Search**.

   The Search window opens.
3. Enter data in either the **Name** or **Description** box.
   - In the **Name** box enter the name or part of the name of the IDoc you want to find, or
   - In the **Description** box, enter a full or partial description instead of a name.
4. In the type box select **IDocs**.
5. In the **Look in** box select an SAP R/3 datastore name.

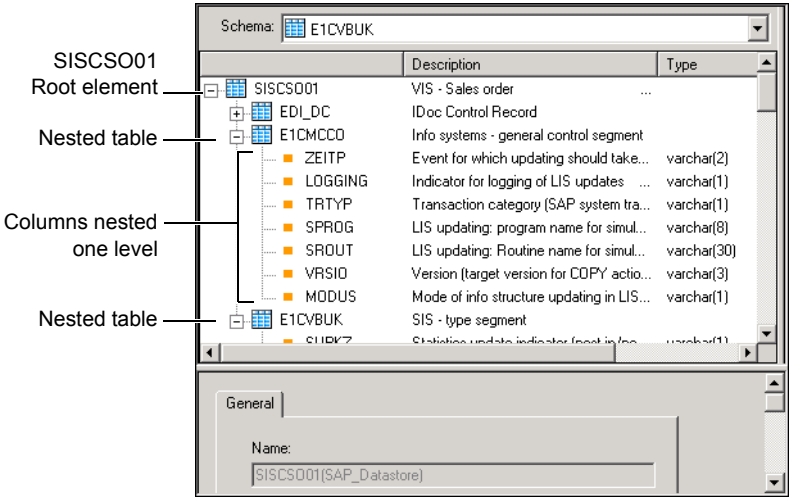
6. To the right of the **Look in** box, select **External Data**.
7. Click **Search**.

   The data returned appears in the window.
8. Right-click the name of the IDoc you want to import and select **Import**.



The IDoc appears in the object library under its datastore and under the IDoc category.

Data Integrator imports the schema for the IDoc type, maintaining hierarchical relationships among the data fields. Double-click the IDoc type in the object library to display the IDoc schema.



➢ To place an IDoc in a job

1. Select an imported IDoc from the object library.
2. Drag the IDoc into a data flow.
3. Select an appropriate option from the popup menu: **Make IDoc file source**, **Make IDoc file target**, **Mark IDoc message source**, or **Make IDoc message target**.

For more information about using IDocs see:

- Chapter 6, "IDocs and Batch Jobs"
- Chapter 7, "SAP R/3 and Real-Time Jobs"

SAP BW datastores

SAP Business Information Warehouse (SAP BW) is a type of SAP R/3 system that runs on the SAP R/3 application server.

The SAP BW interface for Data Integrator allows you to extract data from various sources, including SAP R/3 and SAP BW, transform it as specified in a Data Integrator batch job, and load it into various targets including a predefined BW InfoSource. You must purchase a license for this interface to use SAP BW datastores.

Data Integrator provides two datastores for SAP BW. This is different from other systems Data Integrator supports. Normally, you create a datastore, import objects, and then create sources and targets using those objects. For SAP BW, separate source and target datastores are created as an initial step because they expose different objects.

SAP BW as a source

SAP BW can be used as a source in Data Integrator by using a SAP BW Source datastore.

Using an SAP BW Source datastore is similar to using an R/3 datastore to extract BW data. You can import the same objects, except hierarchies. In addition, you can browse and import InfoAreas, InfoCubes, and ODS objects.

To use SAP BW data as a Data Integrator source:

- Define a Data Integrator SAP BW Source datastore that will serve as the logical link to your BW system
- Import BW metadata into Data Integrator using your SAP BW Source datastore

Defining an SAP BW Source datastore

Use a SAP BW Source datastore in Data Integrator when extracting data from SAP BW.

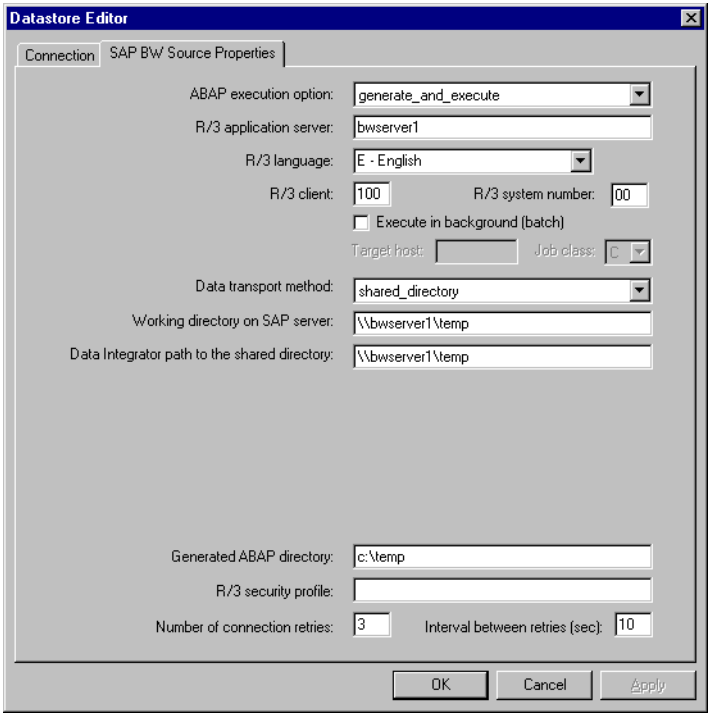
➢ To define an SAP BW Source datastore

1. In the Designer's object library, go to the **Datastores** tab.
2. Right-click in a blank area of the object library window and select **New**.

   The Datastore Editor window opens.
3. Enter a name for the datastore in the **Name** box.

   The name can contain any alpha or numeric characters or underscores (_). It cannot contain spaces.
4. In the **Application type** box, select **SAP BW Source**.
5. (Optional) Select the **Multiple profiles** check box if you want to specify multiple connection profiles for this datastore.
6. Enter the **User name** and **Password**.
7. Click the **SAP BW Source Properties** tab.

8. Enter properties for the R/3 server on which the BW application is running.



These properties are the same as those used by an R/3 datastore. For detailed descriptions, see SAP R/3 datastores on page 262. Like an SAP R/3 datastore, extraction capabilities are provided using either RFC functions or ABAP/4.

9. Click **OK** to create your SAP BW Source datastore.

   SAP BW Source datastores are displayed in the Data Integrator object library.

   You can now browse and import the metadata that will serve as your SAP BW source.

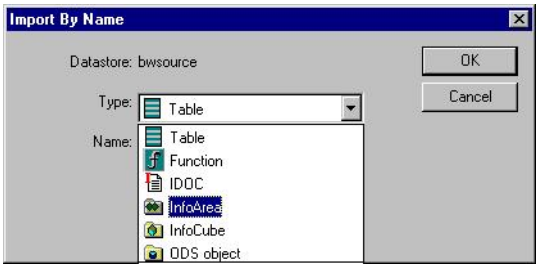
Browsing and importing metadata for an SAP BW source

You can import metadata from the object library in three ways:

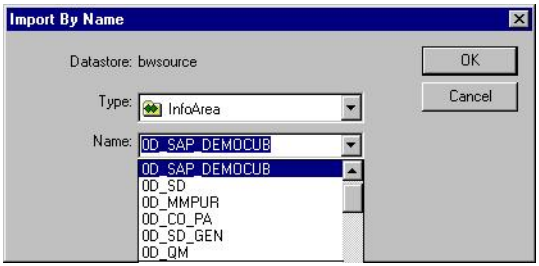
- By name
- By browsing
- By searching

➢ To import SAP BW source metadata by name

1. In the object library, select the name of your SAP BW Source datastore.
2. Right-click and choose **Import By Name**.
3. In the Import by Name window, select the type of metadata that you want to import.



If you select **InfoArea**, **InfoCube** or **ODS object**, the **Name** field offers a list of available objects.



4. Select the name of the object you want to import.

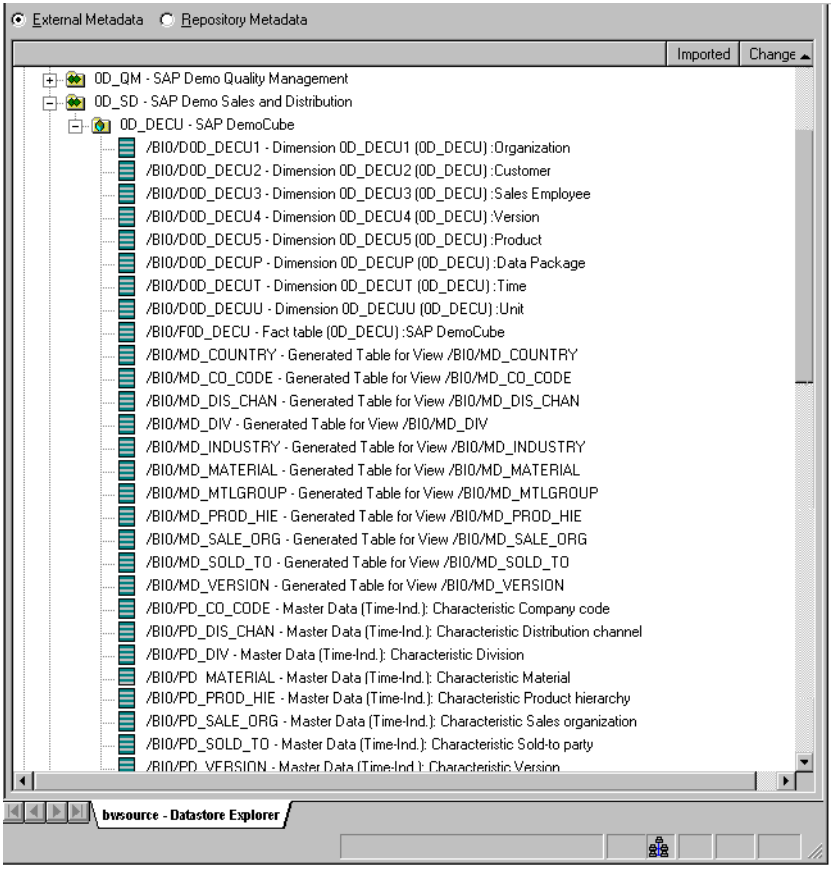
5. Click **OK**.

   The specified source appears in the object library.

➢ To import SAP BW source metadata by browsing

1. In the object library, right-click the name of the BW datastore.
2. Select **Open**.

   The BW external metadata browser opens in the workspace.



From this view you can expand an InfoArea, InfoCube and ODS object and browse its lower level content to identify the objects that provide data to it.

3. Select the structures to import.

   Select a:

   - Parent category (imports the parent and all its children)
   - Single category
   - Sub-set of objects by using the CTRL or CTRL + SHIFT keys

4. Right-click and choose **Import**.

   The imported structures appear in the object library. You can also open the datastore to see the contents of the datastore in the workspace by using the **Repository Metadata** option.

➢ To import SAP BW source metadata by searching

1. In the object library, select the name of your SAP BW Source datastore.
2. Right-click and choose **Search**.
3. Use the Search dialog to find and import metadata.

SAP BW as a target

Data Integrator can use SAP BW as a target. by using an SAP BW Target datastore.

The process for defining a batch job using SAP BW as a target is similar to defining a job for any target environment. The major difference is that job execution is initiated by SAP BW or Data Integrator using BAPI functions, through the Data Integrator RFC Server.

After the job finishes processing, SAP BW picks up the data from the InfoSource into which it was loaded, performs additional processing, as needed, and loads it into one or more BW InfoCubes.

➢ To use a SAP BW InfoSource as a Data Integrator target

- Define a Data Integrator SAP BW Target datastore that will serve as the logical link to your BW system
- Import your BW InfoSource metadata into the Data Integrator BW datastore

The following sections describe the steps to complete these tasks:

- Defining an SAP BW Target datastore
- Browsing and importing metadata for an SAP BW target

For information about executing SAP BW batch jobs, see Chapter 9, "Executing Batch Jobs in the SAP BW Environment"

Defining an SAP BW Target datastore

Use an SAP BW Target datastore in Data Integrator when loading data to SAP BW.

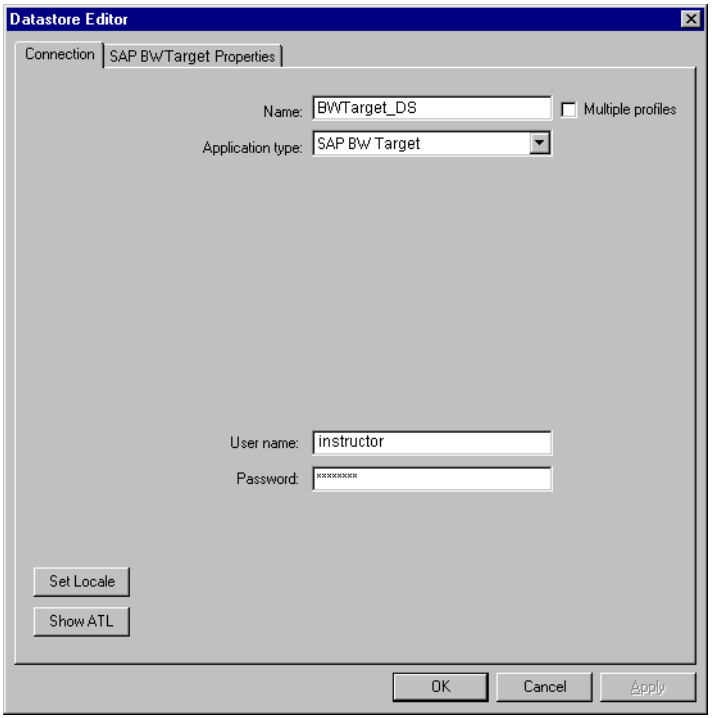
➢ To define an SAP BW Target datastore

1. In the Designer's object library, go to the **Datastores** tab.
2. Right-click in a blank area of the object library window and select **New**.

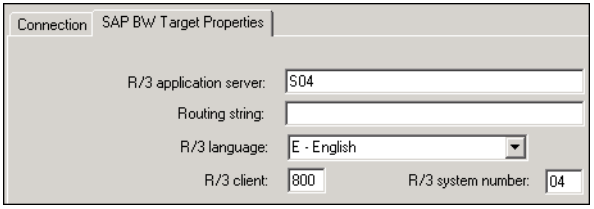
   The Datastore Editor window opens.
3. Enter a name for the datastore in the **Name** box.

   The name can contain any alpha or numeric characters or underscores (_). It cannot contain spaces.
4. In the **Application type** box, select **SAP BW Target**.

5. (Optional) Select the **Multiple profiles** check box if you want to specify multiple connection profiles for this datastore.
6. Enter the **User name** and **Password**.



7. Click the **SAP BW Target Properties** tab.
8. Enter the properties for the R/3 server on which the BW application is running.



You need to include routing string information if your application server is a remote system accessed through a gateway. No SAP R/3 routing permissions are required, but you must use the specified syntax in the **Routing string** field to ensure connection.

The syntax for using an SAP R/3 routing string is:

```
/H/"IP Address of local SAP router"/H/"IP Address of target
SAP router"/H/IP Address of target application server
```

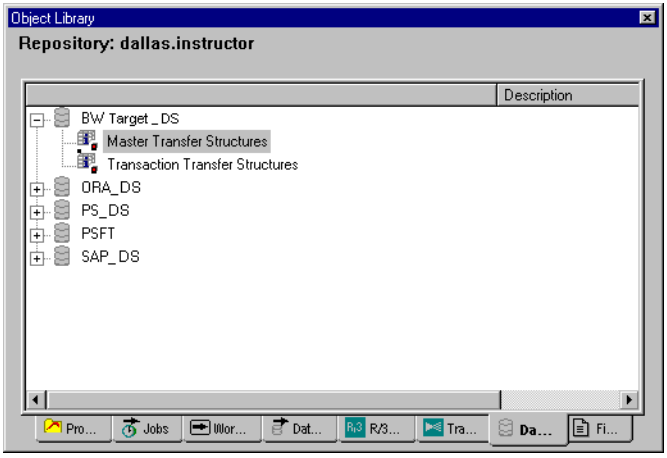
For example:

Your SAP routing string (local and target) is
/H/10.10.1.7/H/204.79.199.5
Your application server IP address is 147.204.76.41

Therefore, your routing string would look like this:
/H/10.10.1.7/H/204.79.199.5/H/147.204.76.41

9. Click **OK** to create your BW datastore.

   SAP BW Target datastores are displayed in the Data Integrator object library as shown.



You can now browse and import the metadata for the Master Transfer Structures and Transaction Transfer Structures that will serve as your SAP BW target. Information about how to do that is in the following sections.

Browsing and importing metadata for an SAP BW target

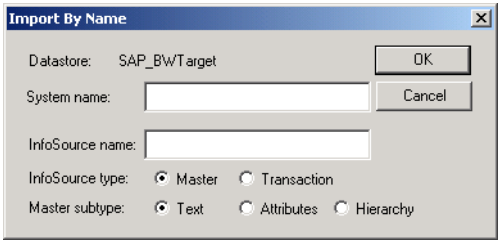
To access InfoSource metadata in Data Integrator, import its metadata into the object library.

You can import InfoSource metadata in one of two ways:

- By name
- By browsing

➢ To import InfoSource metadata by name

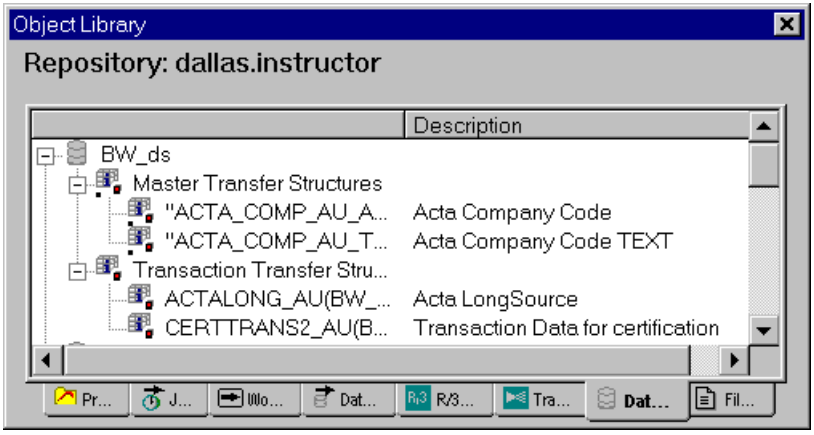
1. In the object library, select the name of your BW datastore.
2. Right-click and choose **Import By Name**.
3. In the Import by Name window, specify the Source System and InfoSource names.



Also, specify whether the InfoSource type is **Master** or **Transaction**. If the InfoSource type is **Master**, you must specify whether the subtype is **Text**, **Attributes**, or **Hierarchy**.

4. Click **OK**.

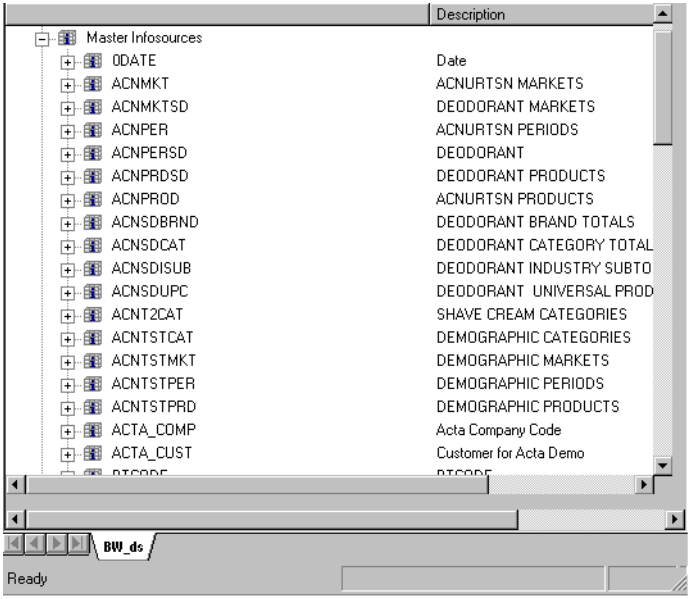
   The specified source appears in the object library. In the following example, two Master Transfer Structures and two Transaction Transfer Structures have been imported into Data Integrator.



➢ To import InfoSource metadata by browsing

1. In the object library, right-click a BW datastore.
2. Select **Open**.

   The BW external metadata browser opens in the workspace.



   From this viewer you can browse both Master and Transaction InfoSources. Both are sorted in alphabetical order.

   Master Transfer Structures contain dimension data, and Transaction Transfer Structures contain fact data.

   A single InfoSource can be loaded from multiple Source Systems. If you expand an InfoSource you can see all the Source Systems that provide data to it.

3. Select the structures to import.
4. Right-click and choose **Import**.

   The imported structures appear in the object library.

4 SAP R/3 and File Formats

This chapter contains the following sections:

- The Transport_Format
- Defining SAP R/3 file formats

For information about Data Integrator file formats not specific to the SAP R/3 environment, see Chapter 6, "File Formats," in the *Data Integrator Designer Guide*.

The Transport_Format

The Transport_Format is a Data Integrator predefined file format object for reading flat files from SAP R/3. It is automatically used to define data transport objects added to R/3 data flows.

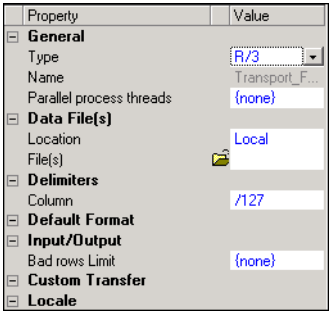
The Transport_Format appears in the **Format** tab of the object library under **Flat Files**.

You can modify it for your own use. You cannot delete it.

➢ To edit the Transport_Format

1. From the object library, right-click Transport_Format and select **Edit**.

   The file format editor opens.



2. Notice that the **Type** is set to `R/3`, the **Name** is set to `Transport_Format` and the **Delimiter**, which indicates the end of one column and the beginning of another, uses the ASCII characters `/127` as the column marker.

   The default delimiter for the Data Transport object is `/127`. If you used the Transport_Format in earlier versions of Data Integrator, then after updating you have three editing options:

- If your transport delimiter is already `/127`, do nothing
- Change the **Delimiter** to match your previous configuration and run your existing ABAP programs
- Use the new configuration which means you will need to also re-create ABAP programs, test, and move these into a production environment.

ABAP programs that use an ASCII delimiter provide improved performance.

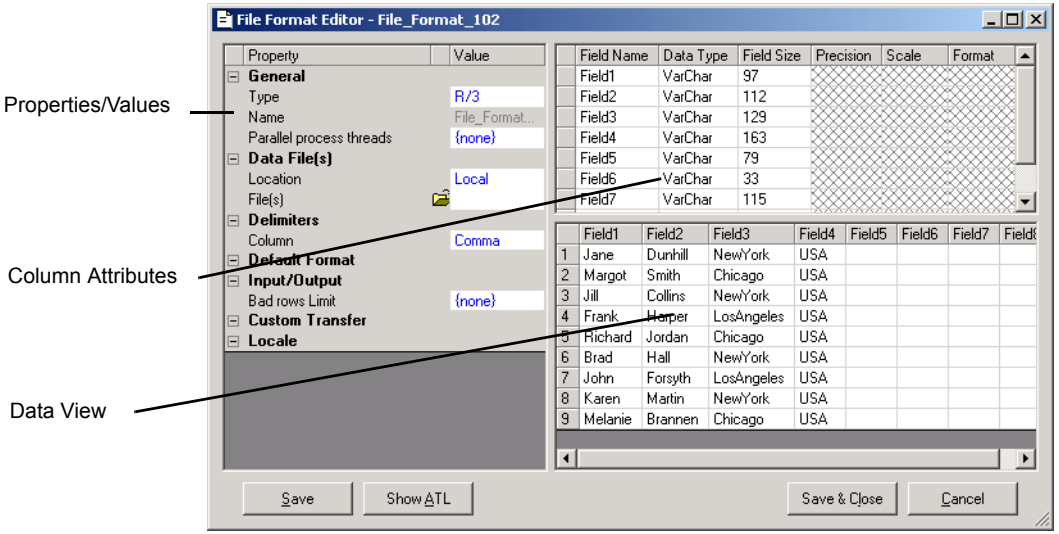
Defining SAP R/3 file formats

When you want to read from or write to a file in an R/3 data flow and do not want to use the predefined Transport_Format, you can create your own SAP R/3 file format. For details about SAP file format options, see "File format" on page 246. For general file format information, see Chapter 6, "File Formats," in the *Data Integrator Designer Guide*.

➢ To define an SAP R/3 file format

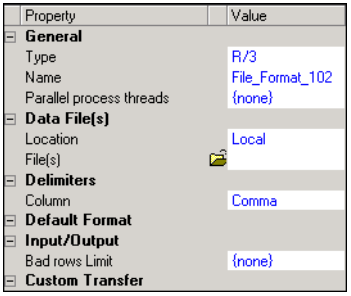
1. Open the object library.
2. Click the **Formats** tab.
3. Right-click **Flat Files** and choose **New**.

   The file format editor opens.
4. Set the **Type** to **R/3**.



5. In the **Name** box, enter a name to identify your format inside Data Integrator.

The name can include any alphanumeric characters and underscores (_). It cannot contain blank spaces.



6. (Optional) To use an existing file as a template for a file format, point to a file using the **Data File(s)** attribute.

   a. Under **Location**, select **Local** to indicate that you want to use a file on the computer where the Data Integrator Designer is running, or select **Job Server** to choose a file residing on the same computer as the default Job Server you associated with the Designer.

   b. When you select **Job Server**, the **Browse** button is disabled, so you must type the absolute path to the file. (For example, a path on UNIX might look like this:

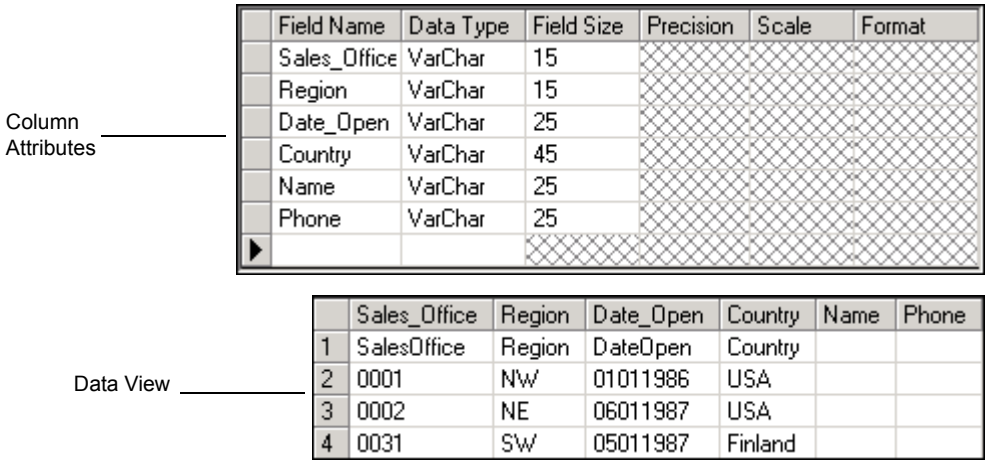
      - `/user/data/abc.txt`

      A path on NT might look like this:

      - `C:\DATA\abc.txt`

NOTE: The **Data File(s)** attribute is a design-time tool that allows you to view data and create metadata schema using a data file located on the Designer computer or on the Job Server computer. When the file format you create is dragged into a data flow as a source or target, you might need to change the **File** name and **Root Directory** attributes to reflect the data files used in your job. Your data file(s) must be located relative to the Job Server computer on which your job will be executed.

When you specify the file, the editor populates the Column Attributes work area for editing and places sample data from the file in the Data View work area. As you edit column attributes, the Data View work area changes to reflect those modifications.



	Field Name	Data Type	Field Size	Precision	Scale	Format
	Sales_Office	VarChar	15			
	Region	VarChar	15			
	Date_Open	VarChar	25			
	Country	VarChar	45			
	Name	VarChar	25			
	Phone	VarChar	25			

	Sales_Office	Region	Date_Open	Country	Name	Phone
1	SalesOffice	Region	DateOpen	Country		
2	0001	NW	01011986	USA		
3	0002	NE	06011987	USA		
4	0031	SW	05011987	Finland		

Click any cell to edit it. Use mouse or keyboard navigation (tab, arrow keys) to move through the fields and view lists from which you can choose data. Right-click to view a popup menu to insert or delete new rows.

7. (Optional) Modify the file format delimiter.
   a. Under **Delimiters** click the **Column** box.
   b. Click the down arrow to view menu options.
   c. Click to choose a delimiter type (**Comma**, **Semicolon**, **Space**, or **Tab**). Alternately, enter an ASCII character. Valid ASCII characters range from `/0` to `/254`.
   d. If defining a file format for a source, verify that the column delimiter matches the file from which you are reading.

If defining a file format for a target, you can keep the default value.

NOTE: The data in the file cannot contain the character(s) used as delimiters.

8. (Optional) You can examine the ATL file associated with this file format. Click **Show ATL** to view a read-only version of the ATL file.

   To search for specific words or phrases within the file, click **Find** and enter a key word or phrase, then click **Find Next** to find occurrences of the word or phrase. Click **Cancel** when you are finished finding words or phrases. Click **Cancel** again to return to the file format editor.

9. After you edit your column attributes, click **Save & Close**.

10. If you are using an existing file as a model, verify that the fields are delimited and have the data type you expect.

5 SAP R/3 and R/3 Data Flows

SAP R/3 consists of three key components:

- SAPGUI client — A front-end application that provides a logical view of the entire SAP system
- Application layer — An application server that interprets *ABAP/4*, SAP's proprietary fourth-generation language, and generates and executes SQL statements against the database
- Database layer — A database manager that manages the SAP R/3 data in its tables

Interacting directly with the SAP R/3 database layer can be problematic because:

- It is complex and difficult to understand.
- The data contained in pool and cluster tables can be accessed only through ABAP—not through SQL.

The Data Integrator SAP R/3 ABAP interface allows you to extract SAP R/3 data, through the application layer, using ABAP. This interface also provides the R/3 data flow and its associated Data Integrator Designer options. This means that:

- You do not need to understand many of the complexities of the SAP R/3 database layer.
- You can use Data Integrator to access not only transparent tables, but also pool and cluster tables.

To use an R/3 data flow, create a batch job, create a data flow inside the job, then create an R/3 data flow inside the data flow. This type of batch job uses the following general execution process:

- Data Integrator generates the ABAP code.
- Data Integrator connects to the SAP application server via remote function call (RFC).
- The ABAP code executes on the SAP application server.
- The ABAP program results are generated and communicated back to Data Integrator, which then loads the target data cache.

This chapter describes the following topics:

- What is an R/3 data flow?
- Defining R/3 data flows
- Extracting data from SAP R/3 hierarchies
- Creating custom ABAP transforms
- Optimizing ABAP

For generic information about Data Integrator data flows, see Chapter 7, "Data Flows," in the *Data Integrator Designer Guide*.

What is an R/3 data flow?

An *R/3 data flow* extracts and transforms data from SAP R/3 tables, files, and hierarchies. The R/3 data flow produces a data set that you can use as input to other transforms, save to a file that resides on an SAP R/3 server, or save to an SAP R/3 table. When Data Integrator executes R/3 data flows, it translates the extraction requirements into ABAP programs and passes them to SAP R/3 to execute.

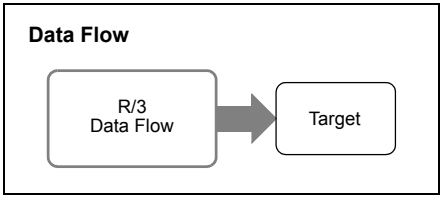
When you plan to use the R/3 extracted data outside the R/3 data flow, the R/3 data flow can place the extracted data set in a file. A Data Integrator *data transport* passes the data in the file to the parent data flow. A data transport indicates how data is communicated between the SAP R/3 server and Data Integrator.

When the target inside the R/3 data flow is the final destination for the data, you do not need a data transport. The R/3 data flow stores the data in a file saved on the SAP R/3 server. In this case, the R/3 data flow must be the only object in a data flow.

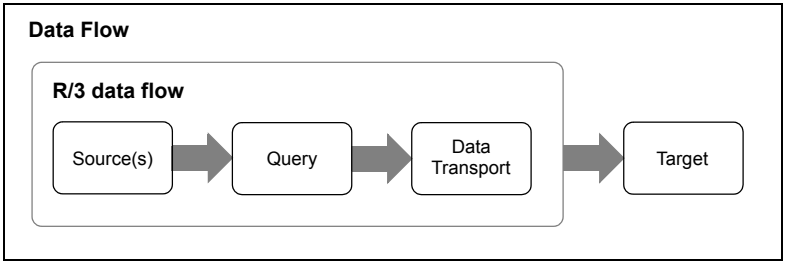
You can also use a data flow to extract data from an SAP R/3 table. When you use a data flow, Data Integrator extracts data from SAP R/3 using an RFC (remote function call). However, when you use an R/3 data flow, Data Integrator extracts data using ABAP and performance is generally better.

Because SAP R/3 performs all operations in an R/3 data flow, you want the operations in an R/3 data flow to make best use of SAP R/3 system capabilities. Therefore, the preferred method of use involves ABAP.

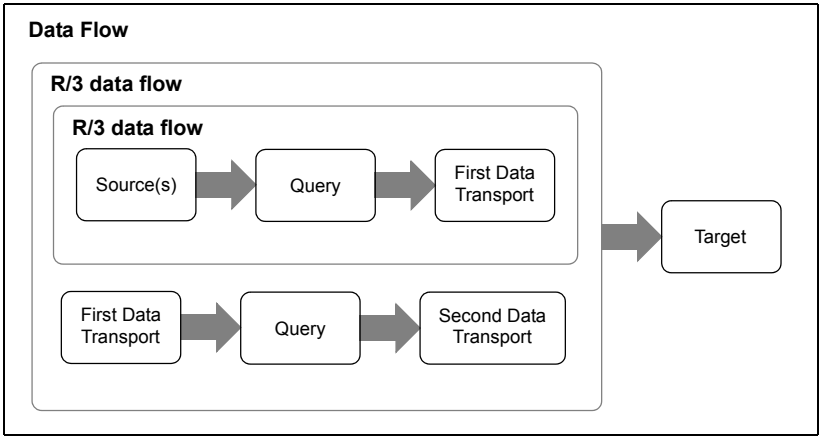
Consider the simple case of extracting data from an SAP R/3 table and loading it into a non-SAP R/3 target. The parent, data flow contains two objects: an R/3 data flow and a target.



The R/3 data flow defines the specific source tables from which you want to extract data and contains a data transport for the resulting data set.



In some cases, the transformations you need to perform in SAP R/3 are too complex to include in just one R/3 data flow. For example, the transformation might require multiple passes over the data. Data Integrator allows you to nest an R/3 data flow inside another R/3 data flow so that you can simplify each data flow. You can keep the output from the inner data flow in a file on the R/3 server and use the file in the outer data flow.



NOTE: Although R/3 data flows are used in data flows, data flows cannot be placed in R/3 data flows. As shown above, only other R/3 data flows can be placed inside R/3 data flows. Also, R/3 data flows cannot be used as *embedded data flows*.

Defining R/3 data flows

There are four procedures to complete when defining an R/3 data flow:

- Creating R/3 data flows.
- Specifying sources
- Defining a query
- Specifying a target

The following sections describe these steps in detail.

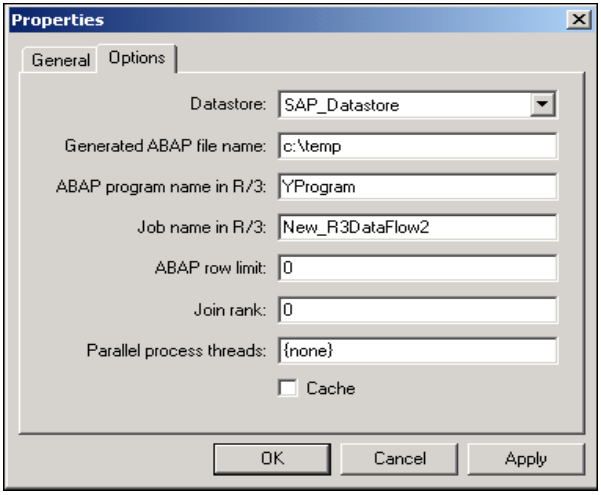
Creating R/3 data flows

Before creating an R/3 data flow, create the data flow that will contain the R/3 data flow and open it.

➢ To place R/3 data flow information into a data flow

1. On the tool palette, click the icon for an R/3 data flow  .
2. Click a location in the workspace.

The Properties window appears prompting for SAP R/3 options.



3. In the **Datastore** list, select the R/3 datastore you want to use for the extraction.
4. In the **Generated ABAP File name** box, enter the file name for the ABAP program that this R/3 data flow generates.

   Data Integrator writes this file in the Generated ABAP directory specified for the datastore.
5. Enter other options in the window.
   - **ABAP program name in R/3**

     Specifies the name Data Integrator uses for the ABAP program name that runs in SAP R/3. It must begin with Y or Z and can be up to 40 characters when you use any SAP 4.x version. If you are using an SAP R/3 3.x version, Data Integrator truncates the ABAP program name and displays only the first 8 characters. A message in the trace log indicates that an ABAP program name is truncated.
   - **Job name in R/3**

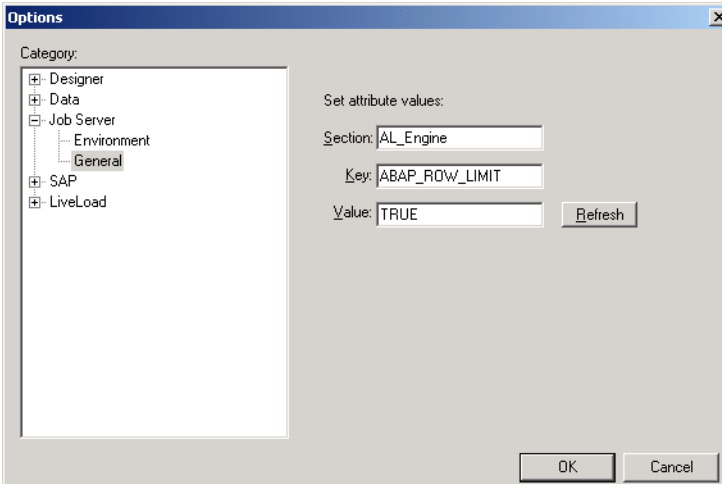
     Specifies the name used for the job that runs in SAP R/3. It defaults to the name of the data flow.

- **ABAP row limit**

  Can be used for the test phase of development to reduce the number of data rows sent to the transport file. The default value of zero indicates that all data rows will be output to the transport file. When this value is set to a number greater than zero, only the specified number of data rows is sent to the transport file. After you enter a number here, enable/disable this feature from the Designer.

  NOTE: The Designer and Job Server must be on the same computer. **ABAP row limit** is enabled/disabled globally for all R/3 jobs associated with this Job Server. Remember to disable this feature before moving the jobs into a production environment.

  Select **Tools** > **Options** > **Job Server** > **General,** enter data as shown in the following graphic; then click **OK**. Entries are case sensitive.



- **Join rank**

  If the calling data flow uses output from this R/3 data flow (in the form of a data transport) in a join, this rank determines the order of the join operation. The highest ranked source is the outermost SELECT loop in the join.

- **Cache**

  If the calling data flow uses output of the R/3 data flow as the inner source for a join, you can cache the output to improve the performance of the join. Use caching only if you are sure the data is small enough to fit in the memory allotted to the cache.

  The option is ignored if the R/3 data flow is nested inside another R/3 data flow.

6. Click **OK**.

   An R/3 data flow appears in the workspace and the project area.

7. Right click the R/3 data flow and select **Edit Name**. Type a new name.

8. Open the workspace for the R/3 data flow.

   Click the R/3 data flow name. Data Integrator opens the workspace where you define the R/3 data flow.

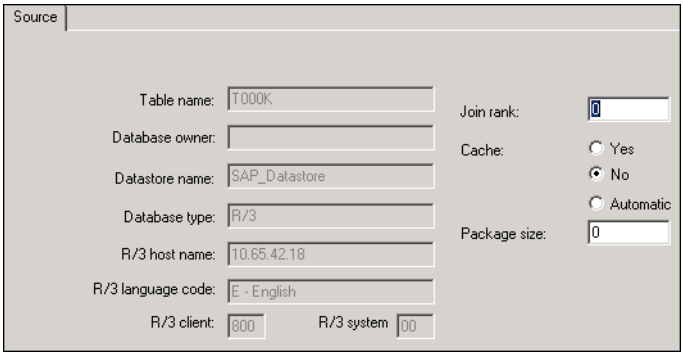
Specifying sources

You can add SAP R/3 tables or hierarchies to the R/3 data flow as sources. This example adds a table. See "Extracting hierarchy set data in Data Integrator" on page 90 for more information about using a hierarchy as a source.

Do not use non-SAP or IDoc sources here; add those sources to the parent data flow and use a query (including BAPI functions as needed) to join those sources with the data set produced by the R/3 data flow.

➢ To specify data sources

1. From an SAP R/3 datastore, click an R/3 table and drag it into the workspace of an open R/3 data flow.
2. Click the name of the source table in the work flow to open the source table editor.



You can set three options:

- **Join rank**

  Determines the join order when this table is joined with another table in this data flow. The default value is zero. To override the default value, set to a non-zero positive integer. For information on how Data Integrator uses this field with SAP R/3 version 4.0, see "Optimizing inner joins" on page 110. For information on how Data Integrator uses this field with SAP R/3 versions earlier than 4.0, see "Manually setting join order" on page 117.

- **Cache**

  Indicates whether Data Integrator should read the required data from the table and load it into memory.

There are three options:

Cache Option	Description
Yes	The table is always cached unless it is the outer-most table in a join.
No	The table is never cached. Package size operation is bypassed.
Automatic	Data Integrator determines whether or not to cache the table based on table size and number of filtered records. See "Tables used in a join" on page 122 for more information.

The default option is **No**. See "Source table options" on page 253 for more detailed information about caching source tables.

- **Package size**

  The package size option limits the number of rows brought into memory when building internal tables. By limiting the number of rows, less memory is used to construct internal tables.

  You should use this option only for cached tables whose size may lead to excessive memory consumption.

  See "Source table options" on page 253 for more detailed information about package size.

Defining a query

Define a query to produce the desired data set. With a query, you can manipulate SAP R/3 sources:

- Join the data sets from more than one source
- Select the rows to extract from the source
- Define the schema of the extracted data set
- Perform calculations on column values

➢ To define a query

1. Add a query to the R/3 data flow.
2. Draw the data flow connections.
3. Use the query editor to specify the desired data set.

NOTE: Data Integrator supports using the results of an R/3 data flow inside another R/3 data flow. If you are using the data in another R/3 data flow, make sure to:

- Keep the data on the SAP R/3 application server rather than on the Data Integrator server
- Use FTP or shared directory as the data transport method

You specify the location of the output file and data transport method in the R/3 datastore associated with the R/3 data flow.

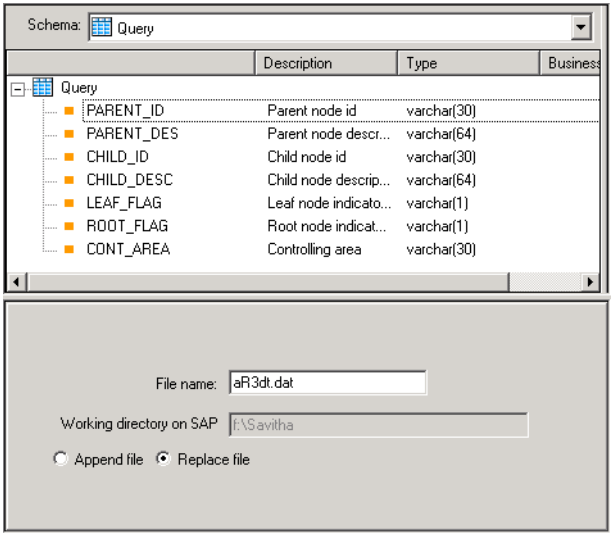
Specifying a target

Specify a target for the resulting data set. Potential targets include a data transport, which makes the data set available to the calling data flow, or a file on the SAP R/3 server.

➢ To define a data transport

1. Add a data transport to the R/3 data flow.
2. Click the name of the data transport.

   The data transport editor appears in the workspace.



3. In the data transport editor, enter the name of the transport file.

   The file is placed in the SAP R/3 working directory specified in the datastore definition.

4. If necessary, you can specify that the output file will be appended to an existing file with the same name (presumably created in a previous run of this same job). The default behavior is to replace any existing file.

5. Click the back arrow to return to the R/3 data flow.
6. Connect the components of the R/3 data flow.

   At execution, the R/3 data flow populates the file described by the data transport from step 3.

Extracting data from SAP R/3 hierarchies

The Data Integrator hierarchy object can extract set hierarchy information from the following set types:

- Basic sets — Sets containing values from one dimension. For example, in a basic set such as Assets that uses the dimension Accounts, all values in Assets are account numbers or ranges of account numbers.
- Single-dimension sets — Combinations of basic sets and/or other single-dimension sets that contain values from the same dimension. For example, the basic set Assets can be combined with the basic set Liabilities because they both use the same dimension Accounts.

In particular, Cost Center and Cost Element hierarchies in the Controlling module of SAP R/3 are supported set hierarchies.

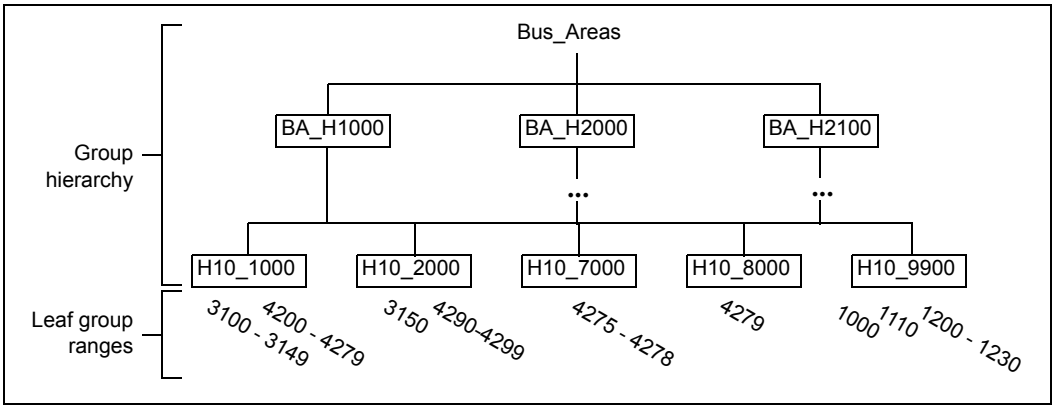
The following sections describe how hierarchies are defined in SAP R/3 and how Data Integrator translates the hierarchical relationships into horizontal or vertical representations:

- SAP R/3 hierarchy sets
- Extracting hierarchy set data in Data Integrator
- Example data flow with an SAP R/3 source

SAP R/3 hierarchy sets

Hierarchy set information from SAP R/3 includes hierarchy nodes and ranges of values associated with the leaf nodes. The possible values are defined by the domain upon which the hierarchy is based. The values themselves are stored in the set table associated with the hierarchy.

The set data represented in this hierarchy is as follows:



Extracting hierarchy set data in Data Integrator

Data Integrator provides a source reader—a hierarchy object—that extracts group nodes and leaf group ranges from supported SAP R/3 set hierarchies. When Data Integrator extracts the hierarchy using a hierarchy object, the result includes a row for each parent-child relationship in the hierarchy and associated leaf group ranges.

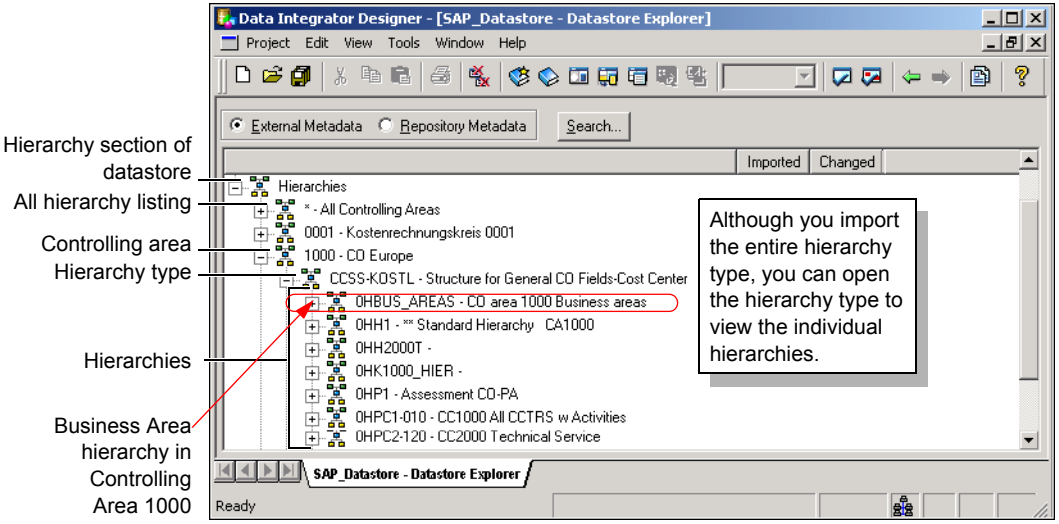
A hierarchy object produces the following output when extracting the Business area hierarchy from Controlling area 1000 shown previously.

	PARENT_ID	CHILD_ID	VALUE_FROM	VALUE_TO
NULL parent to root node		0HBUS_AREAS		
Root node to first child	0HBUS_AREAS	0HBA_H1000		
First child (now parent) to its first child	0HBA_H1000	0HH10_1000	0000003100	0000003149
	0HBA_H1000	0HH10_1000	0000004200	0000004279
First child (now parent) to its second child	0HBA_H1000	0HH10_2000	0000003150	0000003150
	0HBA_H1000	0HH10_2000	0000004290	0000004299
First child (now parent) to its third child	0HBA_H1000	0HH10_7000	0000004275	0000004278
First child (now parent) to its fourth child	0HBA_H1000	0HH10_8000	0000004279	0000004279
First child (now parent) to its fifth child	0HBA_H1000	0HH10_9900	0000001000	0000001000
	0HBA_H1000	0HH10_9900	0000001110	0000001110
	0HBA_H1000	0HH10_9900	0000001200	0000001230
Root node to second child	0HBUS_AREAS	0HBA_H2000		
Second child (now parent) to its first child	0HBA_H2000	0HH20_3000	2-4200	2-4200
	0HBA_H2000	0HH20_3000	2-4210	2-4210
Root node to third child	0HBUS_AREAS	0HBA_H2100		
Third child (now parent) to its first child	0HBA_H2100	0HH21_9000	21-3100	21-3100
			...	

When you view the SAP R/3 hierarchy types in Data Integrator, you do not see the data associated with the hierarchy. Instead you see all of the hierarchy types grouped in two ways:

- Together in a single section (all hierarchy types organized across controlling areas by extraction logic)
- By controlling area

The names of the hierarchy types are made up of the SAP R/3 set table name and the hierarchy group name. The hierarchies are named after the root node.



To extract data using a hierarchy in Data Integrator, import the hierarchy metadata as part of an SAP R/3 datastore definition.

➢ To import a hierarchy type from SAP to your repository

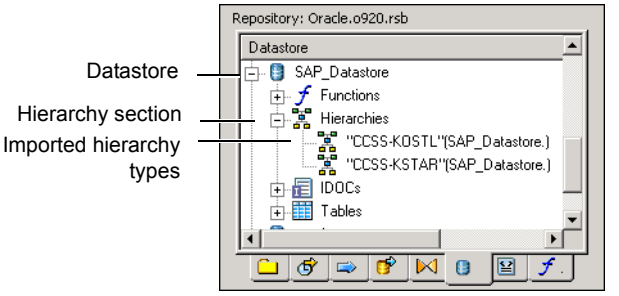
1. In the object library, **Datastore** tab, open the datastore corresponding to the SAP instance from which you want to import the hierarchy type.
2. Right-click in the workspace and choose **External Data**.
3. Open the Hierarchy section of the datastore.
4. Select the hierarchy type to import.

   The available hierarchies are listed according to the controlling areas with which they are associated. You can also browse through all hierarchies regardless of controlling area in the section labeled "All Hierarchies."
5. Right-click and choose **Import**.

The internal view of the datastore shows the hierarchy type.

The imported hierarchy

After a hierarchy type is imported into a datastore, it appears in the object library.



A hierarchy instance in an R/3 data flow

Use a hierarchy object as you would a table from the object library: select the hierarchy, then drag it into the definition of the R/3 data flow open in the workspace.

The instance of the hierarchy object in the data flow has options you can set to specify:

- A single controlling area or all controlling areas from which the hierarchy data is extracted.

  If you imported the hierarchy type from a controlling area in the browser, or if you specified a controlling area when importing the hierarchy by name, a single controlling area is already specified in the hierarchy definition. If you imported the hierarchy type from the **All Hierarchies** browser section or did not specify a controlling area, you can specify the controlling area for each instance of the hierarchy.

  Choose a single controlling area from the list for best performance.

- A single hierarchy or all hierarchies in the hierarchy type.

Choose a single hierarchy for best performance. Enter the hierarchy name as it is shown in the external browser of the datastore.

Example data flow with an SAP R/3 source

This section describes one technique for extracting and transforming the hierarchy so you can easily report on any level of the hierarchy.

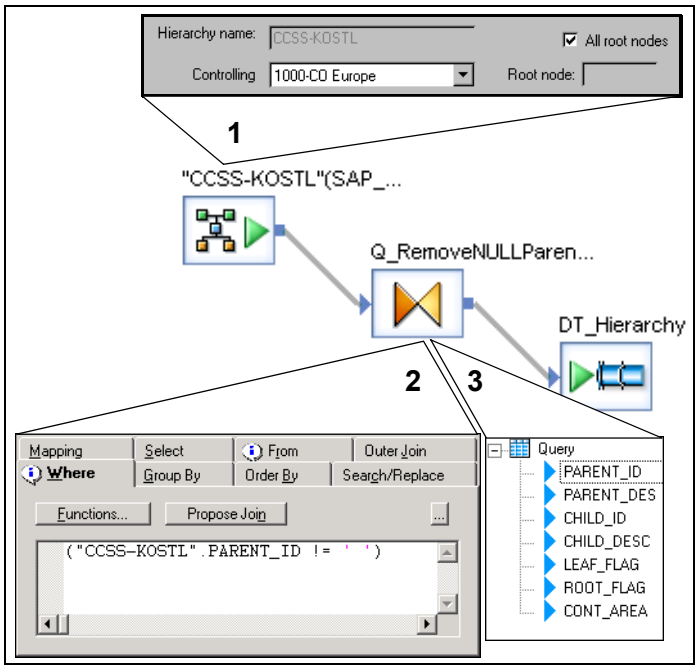
The job in this example consists of three data flows:

1. R3_Hierarchy_Extraction: an R/3 data flow that extracts the Business Areas hierarchy for Controlling Area 1000.
2. R3_Hierarchy_Leaf_Values: an R/3 data flow that associates the leaf data with the child nodes of the hierarchy.
3. DF_Hierarchy_Flattening: a data flow that horizontally flattens the hierarchy.

R3_Hierarchy_Extraction data flow

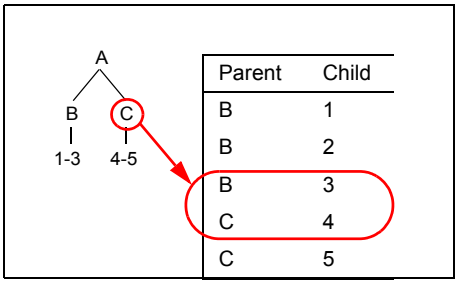
The R3_Hierarchy_Extraction data flow extracts the hierarchy from SAP R/3. It performs these operations:

1. Extracts the hierarchy type from SAP R/3.
2. Filters NULL parent rows from the output.
3. Maps the parent, child, leaf and root flags, and controlling area columns to the output.



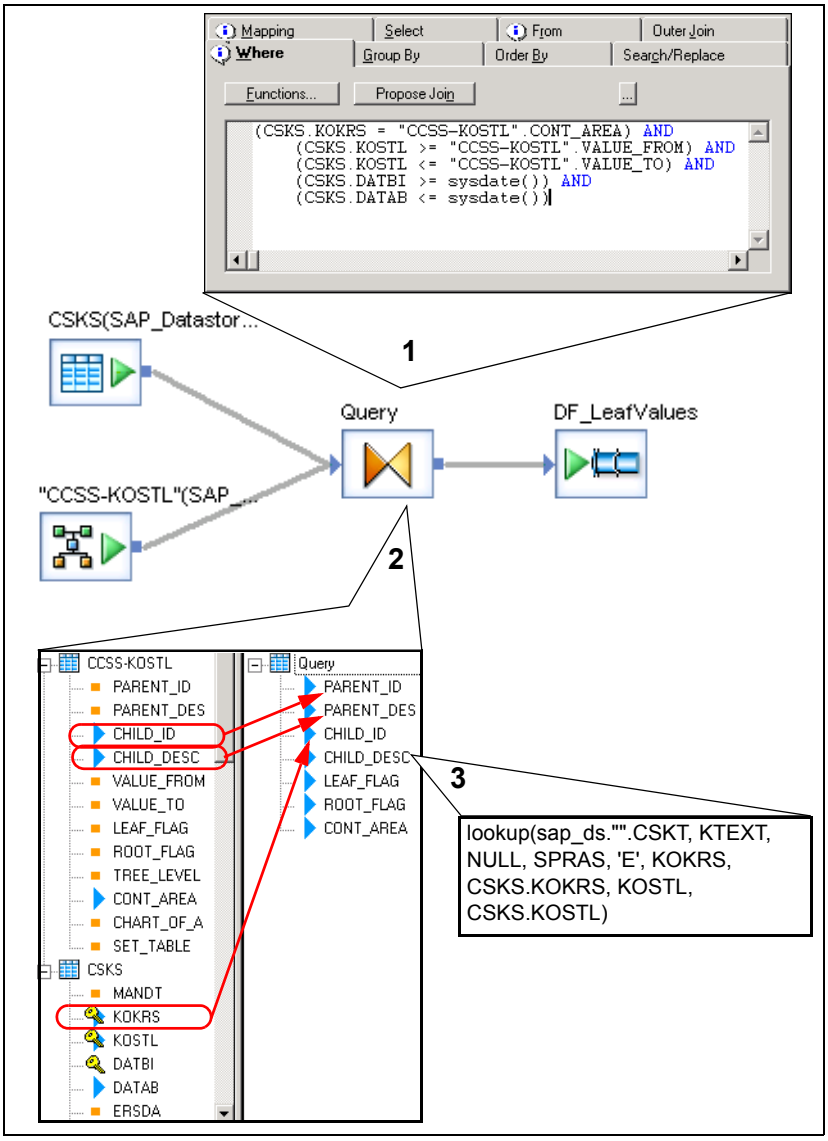
R3_Hierarchy_Leaf_Values data flow

The R3_Hierarchy_Leaf_Values data flow extracts values from the set table that correspond to the leaf nodes in the hierarchy. It produces a new set of parent/child pairs where the new parents are the leaf nodes from the hierarchy and the new children are the values associated with the leaf nodes. The result of this data flow is essentially another level to the hierarchy.



Specifically, the data flow does the following:

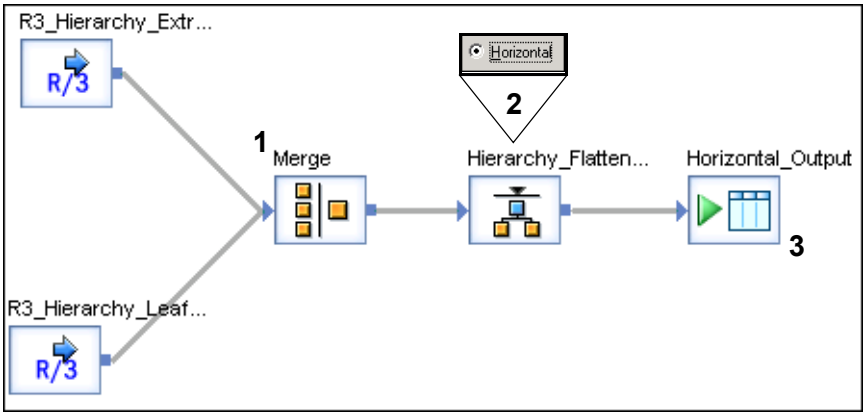
1. Joins the hierarchy with the set table to produce a row for each leaf group value, filtering for valid cost centers and Controlling Area 1000.
2. Maps the child_id and child_desc from the hierarchy to the parent_id and parent_desc in the output. Maps the cost center id (KOKRS) from the set table to the child_id in the output.
3. Looks up a text description of the cost center id and maps the result to the child_desc.



DF_Hierarchy_Flattening data flow

The DF_Hierarchy_Flattening data flow merges the leaf group values with the hierarchy and horizontally flattens the hierarchy as follows:

1. Merges the output from the two R/3 data flows.
2. Horizontally flattens the data using the Hierarchy_Flattening transform.
3. Loads the data into a target.



The output contains a row for each node in the hierarchy, including each leaf group value. The table below shows a section of the output including the BA_H2000 cost center group:

Current leaf		Level 0	Level 1	Level 2	Level 3
0HBUS_AREAS	0	0HBUS_AREAS			
0HBA_H2000	1	0HBUS_AREAS	0HBA_H2000		
0HH20_3000	2	0HBUS_AREAS	0HBA_H2000	0HH20_3000	
2-4200	3	0HBUS_AREAS	0HBA_H2000	0HH20_3000	2-4200
2-4210	3	0HBUS_AREAS	0HBA_H2000	0HH20_3000	2-4210
21-3100	3	0HBUS_AREAS	0HBA_H2000	0HH20_3000	21-3100
21-3110	3	0HBUS_AREAS	0HBA_H2000	0HH20_3000	21-3110
21-4200	3	0HBUS_AREAS	0HBA_H2000	0HH20_3000	21-4200
21-4210	3	0HBUS_AREAS	0HBA_H2000	0HH20_3000	21-4210
21-4280	3	0HBUS_AREAS	0HBA_H2000	0HH20_3000	21-4280
0HH20_4000	2	0HBUS_AREAS	0HBA_H2000	0HH20_4000	
2-3100	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-3100
2-3110	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-3110
2-3200	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-3200
2-4100	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-4100
2-4120	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-4120
2-4130	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-4130
2-4200	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-4200
2-4210	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-4210
21-3100	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	21-3100
21-3110	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	21-3110
21-4200	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	21-4200
21-4210	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	21-4210
21-4280	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	21-4280
0HH20_9900	2	0HBUS_AREAS	0HBA_H2000	0HH20_9900	
2-1110	3	0HBUS_AREAS	0HBA_H2000	0HH20_9900	2-1110
2-1200	3	0HBUS_AREAS	0HBA_H2000	0HH20_9900	2-1200
2-1210	3	0HBUS_AREAS	0HBA_H2000	0HH20_9900	2-1210
...		...	...	...	...

Creating custom ABAP transforms

You can create ABAP programs and incorporate them into R/3 data flow as custom ABAP transforms. This section gives you information about creating and using custom transforms based on ABAP programs. This section discusses:

- ABAP program requirements
- Using ABAP logic blocks in transforms
- Troubleshooting ABAP programs.

ABAP program requirements

You can map a custom-written ABAP FORM logic block to a Data Integrator custom ABAP transform. The ABAP logic block needs to perform the following functions:

- Extract data out of SAP R/3 using custom logic that is not currently supported by Data Integrator ABAP generation logic—for example, import statements and submit statements.

  NOTE: A subset of call function statements is supported: Functions returning scalar values are supported; ones returning tables are not.

- Load the resulting data into an output schema defined for the custom ABAP transform. You can also define and pass parameters to a custom ABAP transform. To include executing the ABAP logic block as part of your Data Integrator data movement, define a custom ABAP transform via the transforms tab. Then add it to an existing R/3 data flow.

Modifying custom ABAP for Data Integrator

Use an `SAP FORM` to modify a custom ABAP program. The custom ABAP must be enclosed in a `FORM` block to work with Data Integrator. In the structure shown below, notice the beginning of the `FORM` section after the comments. Use a FORM to pass the parameters to the output.

```
Comments
Start "FORM" block
Customer's custom ABAP
Start "OTAB1" block
End "OTAB1" block
End "FORM" block
```

The following pages show an example of a custom ABAP program and includes notes about modifying it and creating an `OTAB1` block.

This ABAP logic block returns any profit center hierarchy with expanded leave nodes (i.e., instead of returning a from_value/ to_value, this logic block looks up the master data table and returns a list of all possible values). This ABAP logic block is for an SAP 3.1 system. You can modify it for SAP 4.X.

NOTE: Make sure no line length is greater than 71.

Sample custom ABAP program

```
FORM <<<FORMNAME>>> USING
   $CONT_AREA LIKE TKA01-KOKRS
   $ROOTNODE LIKE SETHDR-SETID
```

* Local tables declaration. Should be moved by ABAP generation to global so that there is no conflict. To convert this logic block into a cost center extraction logic block, change all occurrences of CEPCT to CSKT, GLPCT to CCSS and PRCTR to KOSTL.

```
TABLES: CEPCT.
```

* Local variables declaration.

```
DATA: INDEX TYPE I.
DATA: LEVEL TYPE I, P_LEVEL TYPE I, NODE TYPE I.
DATA: VCOUNT TYPE I.
DATA: IDX TYPE I.
DATA: BEGIN OF SET_HIERARCHY OCCURS 10. INCLUDE STRUCTURE SETHIER.
DATA: END OF SET_HIERARCHY.
DATA: TYPE LIKE SET_HIERARCHY-TYPE.
DATA: BEGIN OF SET_VALUES OCCURS 10. INCLUDE STRUCTURE SETVALUES.
DATA: END OF SET_VALUES.
```

*End of local variables declaration.

*Program continues on next page

*Logic block to extract data from SAP.

* Call function to extract Cost Center hierarchy given a root node

```
      CALL FUNCTION 'G_SET_TREE_IMPORT' EXPORTING

* CLIENT                         = SY-MANDT
* FIELDNAME                      = ' '
* LANGU                          = SY-LANGU
* NO_DESCRIPTIONS                = ' '
* REPORT_ENVIRONMENT             = ' '
          SETID                      = $ROOTNODE
          TABNAME                    = 'GLPCT'
          NO_VARIABLE_REPLACEMENT    = 'X'
          ROOT_HEADER_ONLY           = ''
*   NO_SETID_CONVERSION           = 'X'
*   NO_TABLE_BUFFERING            = 'X'
      TABLES
          SET_HIERARCHY              = SET_HIERARCHY
          SET_VALUES                 = SET_VALUES
      EXCEPTIONS
          SET_NOT_FOUND              = 1
          ILLEGAL_FIELD_REPLACEMENT  = 2
          ILLEGAL_TABLE_REPLACEMENT  = 3
          OTHERS                     = 4.
```

*Logic block to move data into Data Integrator custom ABAP output schema OTAB1.

```
      NODE = 0.
      IDX = 1.
      LOOP AT SET_HIERARCHY.
          NODE = NODE + 1.
          VCOUNT = SET_HIERARCHY-VCOUNT.
          LEVEL = SET_HIERARCHY-LEVEL.
          TYPE = SET_HIERARCHY-TYPE.

          IF LEVEL = 0.

              MOVE 'NULL' TO <<<OTAB1>>>-PARENT_SET.
              MOVE SET_HIERARCHY-SETID TO <<<OTAB1>>>-CHILD_SET.
              MOVE 'NULL' TO <<<OTAB1>>>-VALUE.
              APPEND <<<OTAB1>>>.
          ELSE.
```

* Pop the stack to find parent node.

```
            INDEX = NODE.
            P_LEVEL = LEVEL - 1.
            WHILE INDEX > 0.
               READ TABLE SET_HIERARCHY INDEX INDEX.
               IF SET_HIERARCHY-LEVEL = P_LEVEL.
                  MOVE SET_HIERARCHY-SETID TO <<<OTAB1>>>-PARENT_SET.
                  EXIT.
            ENDIF.
            INDEX = INDEX - 1.
         ENDWHILE.
         IF TYPE = 'S'.
            MOVE 'NULL' TO <<<OTAB1>>>-VALUE.
            APPEND <<<OTAB1>>>.
         ENDIF.
      ENDIF.
      WHILE VCOUNT > 0.
         READ TABLE SET_VALUES INDEX IDX.
         SELECT * FROM CEPCT WHERE SPRAS = 'E'
            AND   KOKRS = $CONT_AREA
            AND   DATBI >= SY-DATUM
            AND   PRCTR >= SET_VALUES-FROM
            AND   PRCTR <= SET_VALUES-TO.
         MOVE CEPCT-PRCTR TO <<<OTAB1>>>-VALUE.
         APPEND <<<OTAB1>>>.
      ENDSELECT.
      VCOUNT = VCOUNT - 1.
      IDX = IDX + 1.
   ENDWHILE.
ENDLOOP.
```

** End of logic block move data into Data Integrator custom ABAP output schema OTAB1.

ENDFORM.

Custom ABAP programming tips

FORMNAME should be entered as it is shown at the beginning of the program on page 102. It will be replaced when the final ABAP program is generated.

$CONT_AREA and $ROOTNODE are parameters. Their data types are referenced in the ABAP. These parameters can be passed to the Transform. You would have to go to the variables/parameters window inside Designer and define them first, they will also have to be initialized. These parameters are optional.

After the FORM block, add a OTAB1 logic block immediately after the custom ABAP and before the "ENDFORM." line. The custom ABAP extracts the results from the SAP R/3 server. The OTAB1 logic block takes these results and moves them into an internal table called OTAB1. You should not declare the OTAB1 in the FORM since it will be replaced in the final generated ABAP. The internal table OTAB1 is replaced by ITAB1 and made accessible to Data Integrator. Use the OTAB1 logic block on page 103 as a guide, but it must be modified to follow the logic of your custom ABAP.

Using ABAP logic blocks in transforms

If you have ABAP logic blocks you want to use in your Data Integrator data movement, you can make them available as transforms.

➢ To import an ABAP logic block

1. Open the object library to the **Transforms** tab.
2. Right-click and choose **New ABAP Transform**.

   A new ABAP transform appears in the object library.
3. Right-click the transform and choose **Properties**.
4. Change the name of the transform so users can find it in the object library.
5. Click **OK**.
6. Right-click the transform and choose **Open** to define the custom ABAP transform.

7. Enter the full path name of the file containing the ABAP logic block.

   You can also click the button next to the box to navigate to the file.

8. Enter a weight to indicate the rank of this data set in a join. You can also set this value when the transform is used in an SAP R/3 data flow.

➢ To define the output schema created by the transform

1. In the output schema, right-click and choose **New Output Column**.

   A Column Properties window opens.

2. Enter the name of the column.
3. Enter the data type of the column.
4. Enter a description of the column.
5. Click **OK**.
6. Repeat steps 1 through 5 for each column in the schema produced by the ABAP logic block.

➢ To define the input and output parameters for the transform.

1. Choose **Tools** > **Variables** to open the Variables and Parameters window.

   The transform is the current context in the window.

2. Right-click in the window and choose **Insert**.
3. Right-click the new parameter and choose **Properties**.
4. Change the name of the parameter.

   Choose a name that identifies this parameter so users can map appropriate data to it.

5. Set the data type of the parameter.

6. Repeat steps 1 through 5 to define additional parameters.

➢ To use the transform in an R/3 data flow

1. Open an R/3 data flow in the workspace.
2. Drag the transform from the object library into the workspace.

   The transform appears in the R/3 data flow as a source.
3. Construct the rest of the r/3 data flow using the data schema produced by the transform.

➢ To pass values to the transform

1. With the R/3 data flow open in the workspace, choose **Tools** > **Variables**.

   The Variables and Parameters window appears, with the R/3 data flow as its context.
2. Go to the **Calls** tab.
3. Double-click a parameter and specify an expression or value in the **Value** field.
4. Repeat step 3 to set values for other parameters.

Troubleshooting ABAP programs

Custom ABAP functions can cause a FUNCTION_CONFLICT runtime error executing generated ABAP if a parameter data type specified in the custom ABAP is not the same as expected by SAP R/3.

For example, you write a custom ABAP program that writes a varchar(10) value to an SAP R/3 system variable. SAP R/3 expects a char(30) value for the same variable. When you execute a job that contains a call to the custom ABAP program, the job fails at run time and adds a function conflict error to the error log for the execution.

Optimizing ABAP

Data Integrator generates ABAP code to extract data from SAP R/3. Data Integrator optimizes the generated code to make the best use of SAP R/3 resources. When optimized, the generated ABAP program executes as fast as possible with minimal impact on SAP R/3. By producing the best possible ABAP program, Data Integrator minimizes execution time and maximizes total throughput.

Data Integrator makes use of the features offered by SAP R/3 version 3.0 and version 4.0. Therefore, the ABAP code Data Integrator generates depends on the version of SAP R/3 you are running. This section describes:

- Optimizations using SAP R/3 open SQL features
- Optimizations using nested SELECT statements
- Optimizations for table caching

Optimizations using SAP R/3 open SQL features

Data Integrator optimizes ABAP using the open SQL features introduced in SAP R/3 versions 3.0 and version 4.0. The open SQL features include:

- Inner joins
- Outer joins
- Aggregate functions
- Group by clause
- Order by clause
- Distinct statement

SAP R/3 version 3.0 introduced some of the open SQL features and SAP R/3 version 4.0 introduced others.

Features introduced in SAP R/3 version 3.0

SAP R/3 version 3.0 introduced four open SQL features:

- Aggregate functions (AVG, MAX, MIN, SUM, COUNT*)
- Group by clause
- Order by clause
- Distinct statement

When possible, Data Integrator generates ABAP code that pushes these operations to the SAP R/3 database. To push the operation to the database, several conditions must be met:

- System runs SAP R/3 version 3.0 or newer.
- Query can be completed by a single ABAP SELECT statement
- Select list does not contain count() aggregate.
- No expression is inside an aggregate function.
- Parameter column of the aggregate function is a Data Integrator numeric data type.

  NOTE: Data Integrator numeric data types include integer, real, and double. The SAP R/3 NUMERIC data type is mapped to Data Integrator type VARCSTR, which is not a numeric data type.

When any of these conditions is not met, Data Integrator generates ABAP code that completes the operation within ABAP.

Features introduced in SAP R/3 version 4.0

SAP R/3 version 4.0 introduced two open SQL features:

- Inner joins
- Outer joins

Optimizing inner joins

SAP R/3 version 4.0 and later can retrieve related data from multiple tables with a single SELECT statement using the JOIN command. You no longer need to use nested SELECT loops. Because a single SELECT statement requires fewer database accesses, performance improves.

When you create R/3 data flows that join transparent tables and that use at least one simple join predicate in the SELECT list or WHERE clause, Data Integrator creates a single SELECT statement using the JOIN command and pushes the operation to the SAP R/3 database layer. For inner joins, simple join predicates are AND, =, <>, <, >, <=, and >=; these predicates must evaluate simple columns. Any complex join predicates are evaluated inside the select loop.

Data Integrator uses the table's join rank to determine which tables to join into a single SELECT statement and the order to join the tables. (See "To specify data sources" on page 84 for information about setting the join rank.) Specifically, tables that have the same join rank are joined together with a single SELECT statement that is pushed to SAP R/3. Tables with higher join ranks are joined before tables with lower join ranks. Tables or groups of tables with different join ranks are joined with nested SELECT loops. These joins are not pushed to SAP R/3. For information about joins using nested SELECT loops, see "Optimizations using nested SELECT statements" on page 115.

When a table has the default join rank (zero), Data Integrator determines the best join rank for that table. Data Integrator decides based on which fields the table shares with other tables, using the fields that link the tables in the WHERE clause. Other tables in the join keep their assigned join rank.

When joining tables with a single SELECT statement, SAP R/3 determines which table is the inner table and which table is the outer table of the join. When joining tables with nested SELECT loops, Data Integrator uses the table or joined tables with the higher join rank as the outer table and the table or joined tables with the lower join rank as the inner table.

For example, suppose you want to join five transparent tables, A through E, with three different join ranks.

Table	Join Rank
A	10
B	10
C	5
D	3
E	3

Data Integrator creates a single SELECT statement joining A and B, and pushes the operation to SAP R/3 (these tables have the same join rank and the highest value).

Next, Data Integrator creates a single select statement joining D and E, and pushes the operation to SAP R/3 (these tables have the same join rank but a lower value). In both cases, SAP R/3 determines which table is the inner table and which table is the outer table.

Finally, Data Integrator creates a nested SELECT loop that joins the joined tables A and B with table C and joins the result with the joined tables D and E. The joined tables A and B are used as the outer table in the first join, and the joined tables A, B, and C are used as the outer table in the second join (the table with the higher rank is used as the outer table).

Now, suppose you specified four different ranks.

Table	Join Rank
A	10
B	4
C	5
D	3
E	3

In this case, Data Integrator creates a single SELECT statement joining D and E, and pushes the operation to SAP R/3. Next, Data Integrator creates a nested SELECT loop that joins table A with table C, joins the result with table B, and joins that result with the joined tables D and E. Table A is the outer table in the first join, the joined tables A and C are the outer tables in the second join, and the joined tables A, C, and B are the outer tables in the third join.

Finally, suppose you specified five different ranks for the five tables.

Table	Join Rank
A	10
B	4
C	5
D	3
E	8

In this case, none of the joins are done with a single SELECT statement that is pushed to SAP R/3. Instead, Data Integrator creates a nested SELECT loop. The loop joins A with E, the result with C, that result with B, and that result with D.

When Data Integrator determines which table is the outer table and which table is the inner table of a join, it uses information about the table size and estimates the effect any filters will have on the number of rows retrieved. Data Integrator uses the smaller data set as the outer table in a join. For general information about determining the best join order, see "Determining the best join order" on page 117.

Data Integrator determines the table size based on the **Estimated row count** attribute for the table. You can enter a value for any table. See "Table" on page 98 of the *Data Integrator Reference Guide* for more information.

For joins containing nested relations, Data Integrator uses the statistics of the embedding table when determining the best join order. For joins containing clustered fields, Data Integrator uses the statistics of the transparent table when determining the best join order.

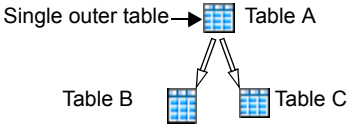
Optimizing outer joins

SAP R/3 version 4.0 and later can perform an outer join between two tables using the LEFT OUTER JOIN command. Using this ABAP command avoids awkward code that leads to unnecessary database table scans and poor performance.

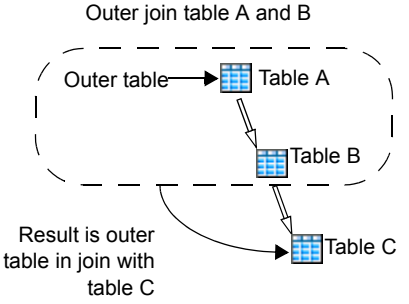
SAP R/3 can only use the LEFT OUTER JOIN command when joining transparent tables with at least one simple expression in the SELECT list or WHERE clause. For outer joins, simple expressions specify fields using AND commands with an equals condition. If the join also specifies a complex expression, Data Integrator will evaluate that expression inside the SELECT loop. Furthermore, SAP R/3 only supports outer joins when one table is used as the outer table in the join. In other words, within SAP R/3 you cannot have an outer join between two tables and use the result in another outer join.

When you create R/3 data flows with an outer join that joins transparent tables and that uses simple expressions in the SELECT list or WHERE clause, Data Integrator attempts to use the LEFT OUTER JOIN command and push the operation to the SAP R/3 database layer.

When the outer join only uses one table as the outer table in the join, Data Integrator is able to create a single SELECT statement using the LEFT OUTER JOIN command and push the operation to the SAP R/3 database layer.



When the outer join uses more than one table as the outer table in the join—such as the result of a previous outer join—then Data Integrator is unable to create a single SELECT statement. Instead, Data Integrator chooses one outer join to complete using the LEFT OUTER JOIN command and uses a nested SELECT loop to complete the join. Data Integrator randomly chooses which tables to join using the LEFT OUTER JOIN command.



You can use the join rank to control the order that tables are joined.

Optimizations using nested SELECT statements

When working with SAP R/3 versions earlier than 4.0, you must use nested SELECT statements to join tables. When working with SAP R/3 version 4.0, you must occasionally use a nested SELECT statement because of restrictions on using the JOIN command and WHERE clause. When joining tables using a nested select statement, Data Integrator creates ABAP code that joins the tables in the order it anticipates is most efficient. You can override the decision with manual settings.

When joining data in a nested SELECT statement, Data Integrator extracts data in loops—the first table is read in an "outer" loop and subsequent tables are read in an "inner" loop. The number of loops depends on the number of tables joined. For best performance, you want the table that limits the most records read in the outer loop. By defining the smaller data set first, you will limit database operations when searching for matching records in the inner table. Data Integrator uses rules to ensure that this happens. You can override these rules and manually specify the join order.

During a join, only two tables are read at one time. However, outer joins that are split into multiple query transforms can reduce extraction performance. Therefore, join performance is better if outer joins that include more than two tables are specified in a single query transform.

This section discusses:

- Automatically setting join order
- Manually setting join order
- Determining the best join order

For information about setting up joins and outer joins, see "Query transform overview" on page 173.

Automatically setting join order

Data Integrator automatically identifies the most appropriate table to read first in the join (that is, the table to use as the outer table). Data Integrator applies several rules:

1. Use the smaller source

   Data Integrator estimates the table size using the Estimated_Row_Count attribute for the table. You can change this value to better reflect the actual size of the data set. For more information how to change this value, see "Table" on page 98 of the *Data Integrator Reference Guide*.

2. Use the table with fewer key fields specified in a WHERE clause

   When joining tables, the WHERE clause specifies key fields that must match. Some tables, such as a detail table that is joined with a header table, have additional key fields that are not used in the match. Typically, the table with additional key fields is larger; the table with fewer key fields defines a smaller data set.

   Data Integrator detects the number of keys, and uses the table with fewer key fields in the outer SELECT loop and the table with the extra keys in the inner SELECT loop.

   NOTE: It is important to use all the SAP-specified keys when joining two related tables. If they are not included in the join, the underlying database used by SAP R/3 may scan the entire table rather than use the primary index.

3. Use the table with a constraint in the WHERE clause

   When the WHERE clause in the joining query transform includes a constraint, Data Integrator assumes that the selection based on a literal constraint will result in fewer records selected and this table should be the table in the outer SELECT loop. This may or may not be the desired result.

For joins containing nested relations, Data Integrator uses the statistics of the embedding table when determining the best join order. For joins containing clustered fields, Data Integrator uses the statistics of the transparent table when determining the best join order.

Manually setting join order

In some cases, you may not want Data Integrator to automatically determine the join order. For a particular join, the WHERE conditions or keys may lead to less-than-optimal performance when Data Integrator applies its rules to determine join order. In these cases, you can specify the join order manually.

To specify the join order manually, set the join rank to a positive non-zero integer for each table in the join. Set the join rank using the table editor; see "To specify data sources" on page 84. The table with the largest join rank is the outer table in the join. The other tables are included in the order of their join ranks.

If the join rank is omitted or set to zero on all tables in the joining query transform, Data Integrator chooses the join rank of all tables in the join automatically, based on the rules described in "Automatically setting join order".

Even though the join ranks specify tables in the proper order, the underlying SAP R/3 access methods may still prevent optimal performance. In this case, consider alternatives to the multiple table join, including using a two-way join with a table lookup using caching or small tables.

Determining the best join order

Use the information available for the data sets involved in a join to determine the best join order. This sections shows two types of common joins and the techniques for determining the appropriate join orders.

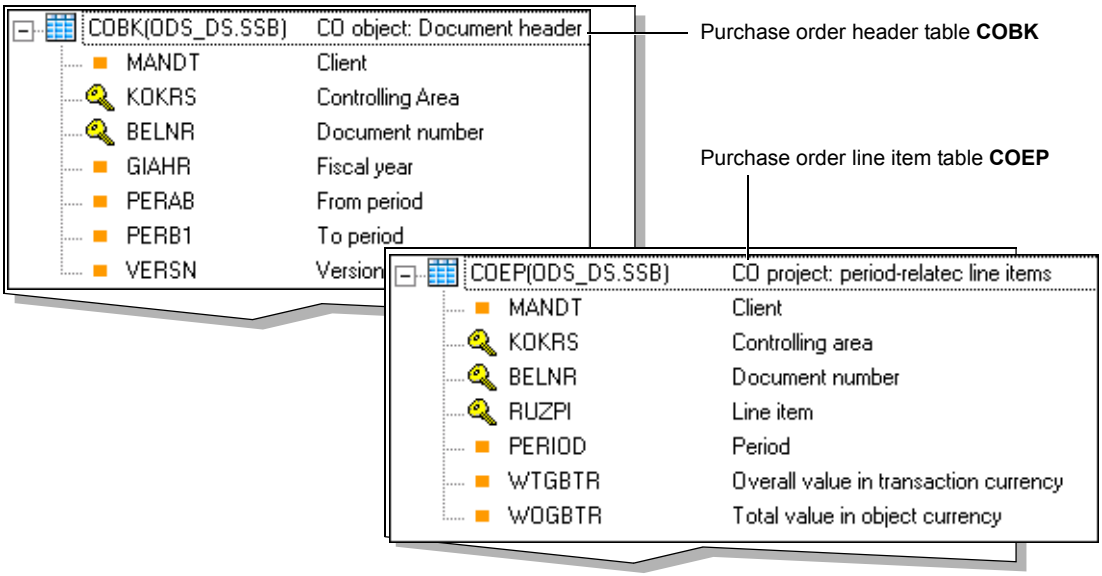
Header and detail tables

For "header" and "detail" table joins, you usually want the header table to be the outer table and the detail table to be the inner table.

Because there are typically many line items for each header, the header table defines the smaller data set. Therefore, you want to use the header table as the outer table in the join and the line item table as the inner table in the join. The join operation reads all of the rows of the detail table into memory. Then for each row of the header table, the join matches multiple rows from the detail table. Producing the output requires reading through each table one time, with comparisons made for each row in the *smaller* table.

The alternative would be to read the header table first, then for each row of the detail table, read through the header table to find the matching details. Producing the output in this alternative requires reading through each table one time, with comparisons made for each row in the *larger* table. The cost of the join operation is higher in this case.

For example, when extracting rows from a purchase order line item table, you would join them with the corresponding row from the purchase order header table.



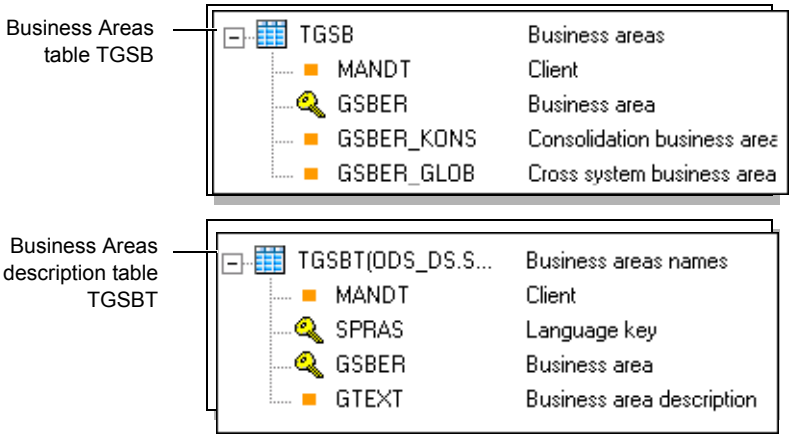
This join would produce a row for each controlling area and document number identified in the header table. You might limit the results by setting bounds on when the document was created, using document creation dates from the header table.

Identifier and description tables

When joining tables to extract text "descriptions" for "identifiers," you often find that the description table is much larger than the identifier table. In that case, you want to use the identifier table as the outer table and the description table as the inner table.

The join operation reads all of the rows of the description table into memory. Then for each row of the identifier table, the join matches a row from the description table. Producing the output requires reading through each table one time, with comparisons made for each row in the smaller table.

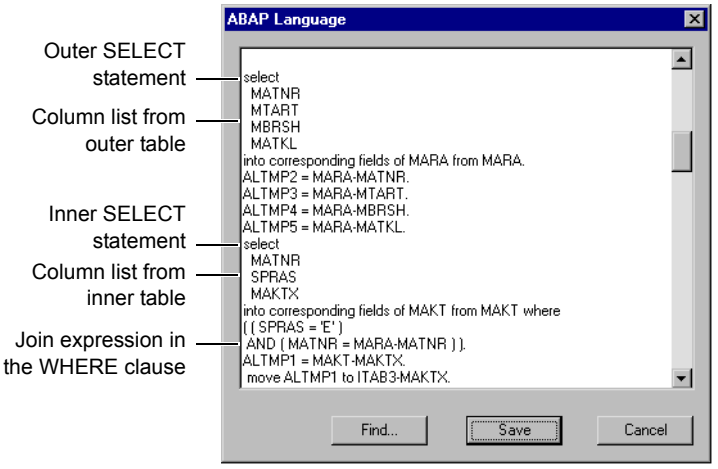
For example, when extracting business areas and descriptions, you might join areas from the business area table (the identifier) with text descriptions from the business areas description table (the description).



Because the description table is larger than the identifier table, you want the join to use the identifier table, TGSB, as the outer table and the description table, TGSBT, as the inner table. This join would produce a row for each business area identifier in the TGSB table. You might limit the results by language values from the TGSBT table.

Checking join order in ABAP

Before actually running a job, you can check the order in which tables will be joined by examining the ABAP that Data Integrator generates for the R/3 data flow. In the ABAP, look for the appropriate SELECT loop description. The outer table is described before the inner table. For example, for a join between two tables, MARA with join rank 2 and MAKT with join rank 1, the ABAP contains two SELECT statements indicating the outer table (MARA) and the inner table (MAKT).



➢ To generate and view ABAP

1. Open the R/3 data flow in the workspace.
2. Choose **Debug** > **Generate ABAP**.

   The ABAP Language window opens in the workspace.

   If there are errors in the R/3 data flow, the Output window opens with a list of any errors or warnings. There cannot be any errors in the R/3 data flow for ABAP to be generated.

3. Locate the correct SELECT statements.

   Use the **Find** button or browse through the code by scrolling.

The first table in the set is the outermost table. For SAP R/3 versions earlier than 4.0, there will only be one table in each loop.

Optimizations for table caching

SAP R/3 can cache a table (that is, load it into memory). You can cache tables in two cases:

- Tables used in a join
- Tables used in a lookup function

Tables used in a join

Caching tables used as inner tables in joins improves performance because fewer database accesses are required. You can manually set or override table caching.

Use the **Cache** property on the table's editor to set table caching. (See "To specify data sources" on page 84 for information about setting this property.) There are three options.

Cache Option	Description
Yes	The table is always cached unless it is always the outer-most table in a join.
No	The table is never cached.
Automatic	Data Integrator determines whether to cache the table or not.

The default option is **No**.

If you enable the cache on all tables in a join, inner and outer tables are read into internal tables prior to a join. This construct replaces SELECT LOOP processing. You should then use the Package Size option on the outer loop of the join to limit the size of the internal tables brought into memory.(See "To specify data sources" on page 84 for information about setting this property.)

The rows extracted from cached tables must fit into the production SAP R/3 system memory. When you select the **Automatic** option, Data Integrator estimates the table size and number of filtered records and projects the number of rows that must be extracted. When Data Integrator projects that 2,500 or fewer rows will be extracted and the table is used as the inner loop of a join, Data Integrator caches the table. To estimate the table size, Data Integrator uses the **Estimated row count** attribute for the table. You can enter a value for any table. See "Table" on page 98 of the *Data Integrator Reference Guide* for more information.

When the cached tables do not fit into memory, you will get an overflow or memory contention error when executing the ABAP program. This error can occur when the tables you select for caching (tables with the cache option set to **Yes**) do not fit into memory or when the tables Data Integrator selects for caching (tables with the cache option set to **Automatic**) do not fit into memory.

In either case, you have two options:

- Change the cache option to **No** for one or more tables, and re-run the program until you can execute it without an error.
- Use the Package size option to limit the number of rows read into memory.

Tables used in a lookup function

Caching the values that a lookup function needs from a translation table reduces the number of times the system must be queried and improves system performance.

When creating a lookup function, you can specify whether or not to cache the translation table. When you specify that a translation table is cached, Data Integrator is able to improve performance. There are three options:

- NO_CACHE — Data Integrator caches no values.
- PRE_LOAD_CACHE — Data Integrator applies filters and caches all the matching lookup columns in the translation table.

  Select this option if the table can fit into memory.
- DEMAND_LOAD_CACHE — Data Integrator searches the R/3 data flow for other lookup functions that use the same translation table and have the same cache setting. When Data Integrator finds two or more lookups, it combines those lookups and caches the results into an internal table. SAP R/3 reads the internal table to find the results of individual lookups.

  Select this option when you use the table in multiple lookups and the compare conditions are highly selective, resulting in a small subset of data.

Because a lookup may retrieve many records, you must be careful that the caching requirement does not exceed available memory. If you get an error message indicating a lack of memory, you should remove caching from one or more tables.

For more information about creating a lookup function, see "lookup" on page 276.

Making table size information available to jobs

When optimizing execution algorithms, Data Integrator makes decisions based on the size of tables being imported.

You can:

- Make table size information known to jobs for automatic tuning

- Use cost coefficients to further tune job performance

Optimizations for testing R/3 data flows

You can control the number of rows returned from an ABAP program when you run its job in test mode.

When you drop an R/3 data flow into the workspace the Properties window for that data flow appears. On the **Options** tab, use **ABAP Row Limit** to limit the number of rows that the R/3 data flow reads. Then, select the **Test Mode** option from the Execution Properties window when you execute this job. This feature allows you to quickly run jobs in a development environment using SAP R/3 data.

See "ABAP row limit" on page 82 for configuration information.

6 IDocs and Batch Jobs

As described in the previous chapter, R3 data flows can be placed into data flows in batch jobs.

In addition:

- R/3 tables can be used in batch jobs as sources.
- BW targets can be used in batch jobs. See Chapter 9, "Executing Batch Jobs in the SAP BW Environment"
- RFC/BAPI functions can be used in batch jobs

Another SAP R/3 object that can be used in batch jobs is an IDoc. This chapter covers the use of IDocs in batch jobs.

This chapter contains the following sections:

- IDoc sources in batch jobs
- IDoc targets in batch jobs
- Adding an IDoc to a batch job

IDoc sources in batch jobs

In data flows of batch jobs, IDoc file sources can hold one file or many individual files (to be processed at a later time). See "Source" on page 250 for the list of IDoc source editor options.

IDoc file sources are processed as follows:

- The IDoc file is opened and the IDoc record type is examined to see which type it is (SAP Release 3.0/3.1 or SAP Release 4.x).

  NOTE: IDoc file sources are validated against the IDoc Type, the IDoc Type related hierarchy, and the SAP Release. SAP R/3 release 4.x is used as the default release number. If you are not using SAP R/3 release 4.x, contact Business Objects Customer Support to have an adjustment made to your installation.

- Related IDoc metadata is read.
- The IDoc list (one or many IDocs) is built based on the information in the **IDoc File** box.

  If an error occurs, this message is displayed: "Cannot build IDoc from IDoc file: <%s>. Please see the error log." The process terminates, and the next file is processed (if multiple IDoc files are defined).

- Each IDoc is read one by one from the IDoc list and passed to the query. A trace message is recorded in the job's trace log: "Process IDoc number: <no. in the file: filename>."

Multiple file read

You can configure Data Integrator to read multiple files by listing the names of these IDoc files in the **IDoc File/s** box of the source editor in the Designer.

Examples

- If you specify the IDoc file, D://temp/`IDOC_R3*.txt`, Data Integrator will process data from all files in the D://temp directory with names beginning with `IDOC_R3` and the extension `.txt`.
- If you specify the IDoc file, D://temp/`IDOC_R3?.txt`, Data Integrator will process data from all files in the D://temp directory with:
  - names beginning with `IDOC_R3, and`
  - any character before the extension, and
  - the extension `.txt`.
- If you specify the IDoc file, `D://temp/IDOC_01.txt`, D://temp/`IDOC_02.txt`, D://temp/`IDOC_03.txt`, Data Integrator will process data from these three IDoc files.

Variable file names

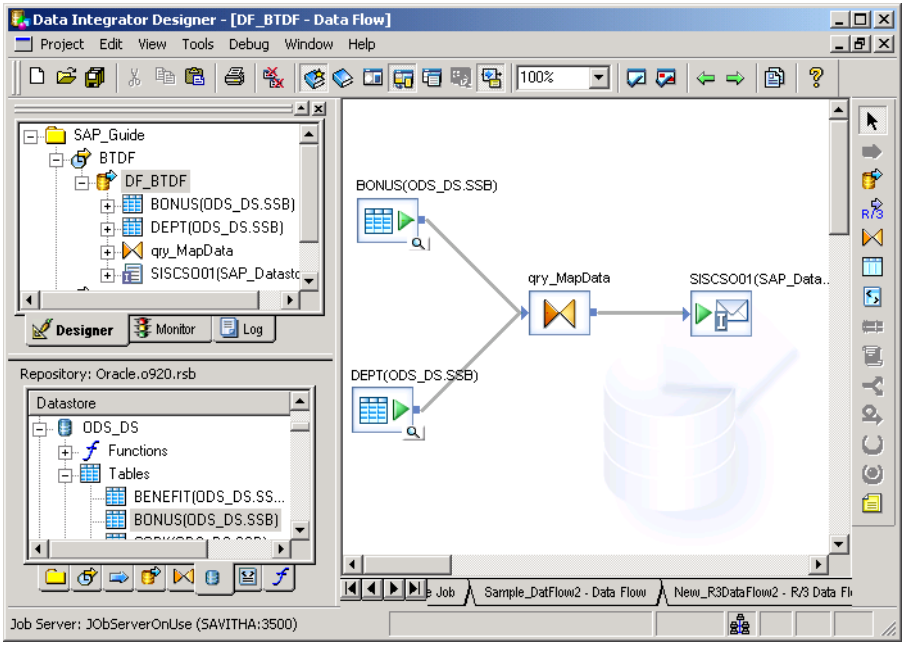
You can also use variable file names in the **IDoc File/s** box. Define a global or local variable for an initialization script of a job. For example:

```
$filepath = 'D:\temp\IDOC_R3*.txt'
```

`Then` enter `$filepath` in the **IDoc File/s** box. In this way you can avoid opening each IDoc object to configure the location of source files.

IDoc targets in batch jobs

In data flows of batch jobs, one or more sources can be transformed into an IDoc *message* target.



The sources may become active on a scheduled basis. As soon as all sources are present, the transform (a query for example) processes the data into the IDoc format configured in the query's output schema and the IDoc message/s are sent to SAP R/3 via a TRFC connection.

A data flow that contains an IDoc *file* target can be used to store IDocs. When you configure an IDoc file target, Data Integrator allows you to capture data for more than one IDoc. Each time a root segment passes into a file target (for example **IDoc File** `C:\temp\idoc`), a new file is created and named using an eight character suffix starting with `00000000.txt`. For example an **IDoc File** set to `C:\temp\idoc` can store `idoc00000000.txt` to `idoc99999999.txt`.

See "Target" on page 259 for a list of IDoc target attributes.

Adding an IDoc to a batch job

When you drag an IDoc into a data flow, Data Integrator displays a popup menu, with valid options: **Make IDoc file source**, **Make IDoc file target**, **Make IDoc message target**.

➢ To add an IDoc to a batch job

1. Select an IDoc from the object library.
2. Drag it into an open data flow.
3. Select the appropriate option.

NOTE: When an IDoc is first used in Data Integrator, part of the IDoc metadata is stored in Data Integrator directories on the computers you are using to run the Data Integrator platform. If you upgrade your SAP R/3 system, the metadata may describe old IDoc versions. Therefore, when you upgrade your SAP R/3 system, delete all IDoc metadata files (with extension .ido) from your Data Integrator computers. Perform a file search on your Data Integrator computers for *.ido and delete these files. The next time you run a job that includes an IDoc, Data Integrator will re-import the IDoc and create a new .ido file. There is no need to manually re-import IDocs if you upgrade your version of SAP R/3.

7 SAP R/3 and Real-Time Jobs

This chapter discusses real-time jobs in the SAP R/3 environment. For general information about creating and using real-time jobs, see Chapter 10, "Real-time jobs," in the *Data Integrator Designer Guide*. This chapter includes the following sections:

- Overview
- IDoc sources in real-time jobs
- R/3 table sources in real-time jobs
- Caching source data
- IDoc targets in real-time jobs
- Data cache targets and transactional loading
- Calling BAPI functions
- Calling RFC functions

Overview

Data Integrator supports real-time data transformation including receiving messages from ERP systems (such as SAP R/3) or XML-based, analytic applications. "Real-time" means that Data Integrator reacts to messages as they are sent, performing predefined operations and responding appropriately.

You define the operations performed to process messages and create responses by building real-time jobs using the Data Integrator Designer and then by converting those jobs to on-demand services using the Data Integrator Administrator.

IDoc sources in real-time jobs

For real-time data extraction from and loading to SAP R/3, Data Integrator uses SAP R/3 Application Link Enabling (ALE) technology and Intermediate Documents (IDocs) to capture and process transactions

Each real-time job can contain only one *message* source. For extracting data from SAP R/3, using Data Integrator, that message source must be an IDoc.

A real-time job can contain other sources. Data from these supplementary sources is extracted on-demand when the real-time job is processing data from the message source.

The message source object represents the schema of the expected messages. Messages received are compared to the schema. If a data flow receives a message that does not match the schema, the data flow throws an error at run time.

You must import the schema for each SAP R/3 IDoc message type your real-time system will process. See Chapter 3, "SAP Datastores" for information about creating a datastore and importing metadata for IDocs.

Creating real-time jobs using IDoc sources

To create and run a real-time job that uses an IDoc message source you will need to perform the following steps.

- See "SAP R/3 IDoc interface connectivity" on page 15 for information about configuring SAP R/3 to send IDocs to Data Integrator.
- In the Designer:
  - Create an R/3 datastore.
  - Import IDoc metadata into the Data Integrator repository.
  - Create a real-time job using one or more data flows.

- Drag an IDoc Type into the first data flow and select **Make IDoc message source**.

  NOTE: No XML message sources or targets can be configured in a real-time job that has an IDoc message source.

- Drop a query and a target.
- Connect the objects.
- Map schemas using the Query editor.
- Validate and execute the real-time job by running it in test mode. Results are saved to a file you specify.
- When the job runs as desired, open the Data Integrator Administrator to set up the job in a test/production environment.

- In the Administrator:
  - If you have not already done so, add a connection to the repository that contains the real-time job and add a connection to the Access Server that will manage the real-time service.
  - Add a service and service providers for the real-time job.
  - From the RFC Configuration page, add an RFC connection, then add supported IDocs by specifying the IDoc Type (used in the job) and the service (you just created for the real-time job) on the Add IDoc page.

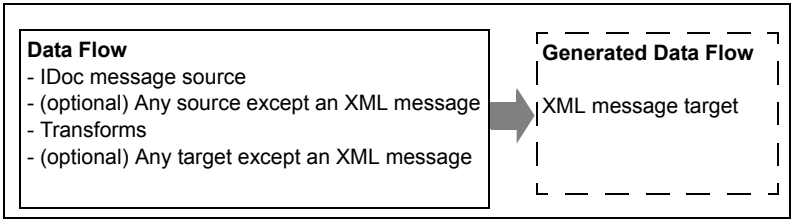
    In this step you are configuring an Access Server to communicate with SAP R/3 to process IDocs in real-time jobs.
  - From the Real-time Service Status page, start the service for the real-time job.
  - From the Real-time Service Status page monitor the service and access Job Server error and trace logs to verify the service is successfully processing IDocs from SAP R/3.

Request/acknowledge data flow model

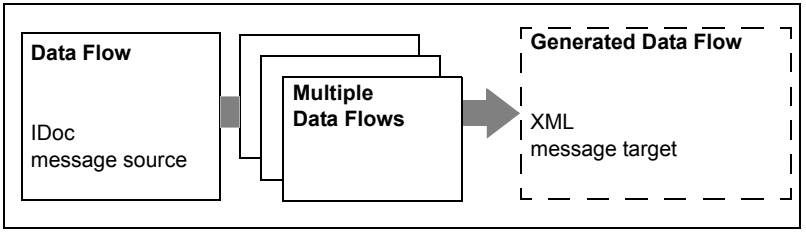
The request/acknowledge data flow model supports the use of an IDoc as a message source in the first data flow inside a real-time processing loop. Data Integrator automatically generates a final data flow, in any real-time processing loop that contains an IDoc message source. This final data flow contains an XML message target.

The XML message target sends an acknowledgement back to the Access Server indicating that the IDoc message source was received from SAP R/3. The Access Server returns the acknowledgement to SAP R/3.

Request/Acknowledge model with a single data flow



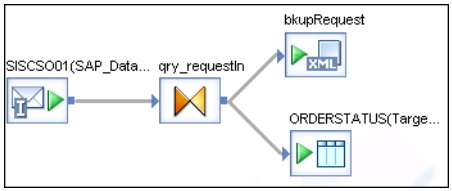
Request/Acknowledge model with multiple data flows



When using the request/acknowledge model, after the first data flow, you can configure additional data flows, work flows, scripts, conditionals, etc. The only limitation is that because an IDoc message source is used, XML message sources and targets cannot be configured in the job. Outside the real-time processing loop you can add initialization and/or clean-up objects as desired.

First data flow in a Request/Acknowledge model

In a request/acknowledge data flow model, the first data flow within a real-time processing loop contains an IDoc message source. To move a data set to succeeding data flows (optional), you can use a memory table as shown in the example below.



Adding an IDoc source to a data flow

Use the following procedure to add an IDoc source to a data flow.

➢ To create an IDoc source

1. Open the data flow in your workspace.
2. Open the object library and go to the **Datastores** tab.

3. Expand the datastore that defines the connection to the SAP R/3 application server.
4. Inside the datastore, expand the IDocs category.
5. Select the IDoc that the data flow will process.

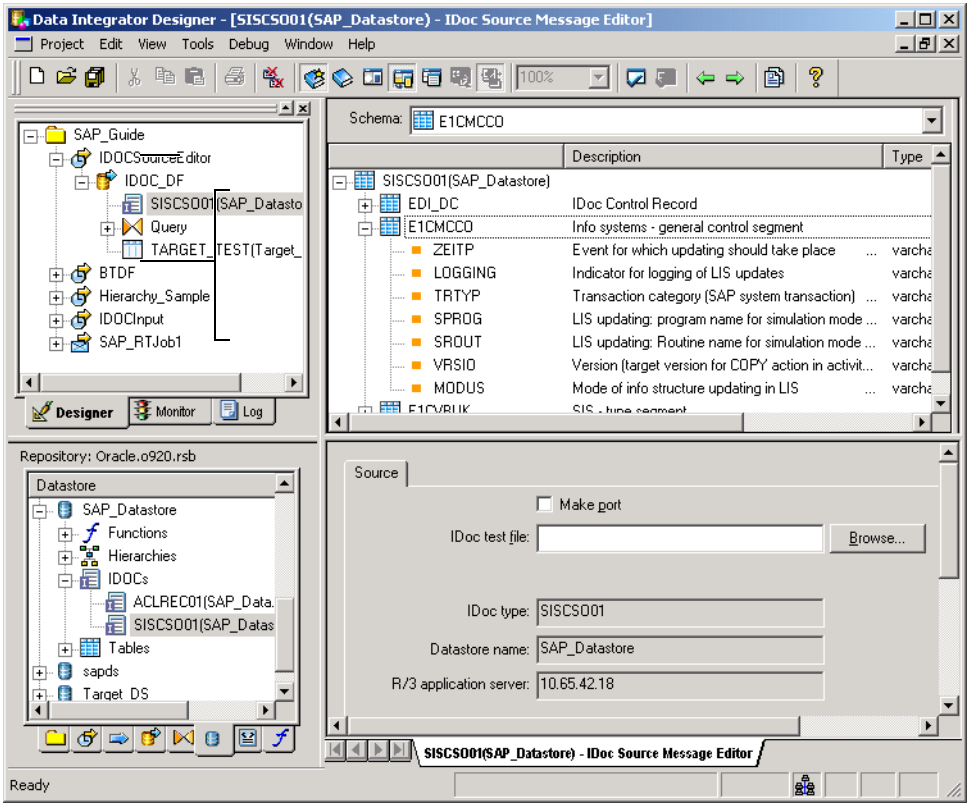
6. Drag the IDoc into the workspace and choose to use it as a file or message source by selecting **Make Idoc file source** or **Make Idoc message source** from the popup menu.

   NOTE: If your data flow is the first inside a real-time processing loop it can have one and only one message source. If another message source already exists in the data flow definition, Data Integrator will not allow you to place another in the workspace.

7. Drop a Query and a target (optional if your data flow is part of a real-time job) and connect the objects.

8. Define source editor options by clicking the name of the IDoc to open the source editor.

   If the IDoc source contains nested data, the schema displays nested tables to represent the relationships among the data.



IDocs in general, and the Sales Order IDoc (SISCSO01) shown above, contain many nested tables and potentially hundreds of columns across the levels. It is possible that you only need to load data from a fraction of the columns from the IDoc. To make working with the IDoc contents easier, consider deleting columns and nested tables that you don't need.

Options on the IDoc Source Editor are described in detail in Chapter 11, "Reference information" under "Source" on page 250

9. Define the output schema.

First double-click the Query object to open it.

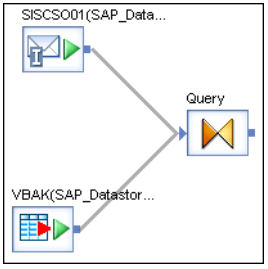
Columns can be dragged and dropped or copied and pasted from the input schema to the output schema. The icons show when a column is mapped. Here is sample input and output schema:

Input schema	Description	Output schema	Description
E1CVBAK-VBELN	Sales document number	E1CVBAK-VBELN	Sales document number
E1CVBAK-AUART	Sales document type	E1CVBAK-E1CVBP0-PARVW	Partner function
E1CVBAK-E1CVBAP-POSNR	Sales document item number		
E1CVBAK-E1CVBP0-PARVW	Partner function		

Considerations:

- All IDoc fields have type char.
- To use two columns with the same name, the first column's name must be changed before the second column is added.

10. (Optional) Join IDoc data with an SAP R/3 table using a query transform.



The data from the SAP R/3 table is extracted using an RFC function call. This is efficient for a small amount of data.

For more detailed information about IDoc file and message objects see "IDoc file" on page 232 and "IDoc message" on page 234.

R/3 table sources in real-time jobs

You can use R/3 tables as sources in real-time jobs after you import metadata for the table into your repository. When you drag an R/3 table to a data flow definition, it becomes a source.

When the data flow performs a query against the R/3 table, it executes an R/3 function call to extract the data through the SAP R/3 application server.

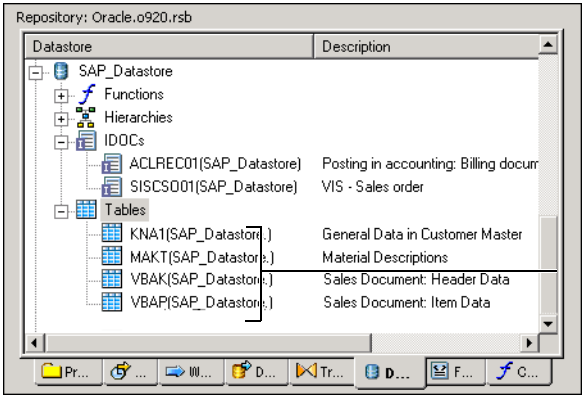
NOTE: This method of extracting data from SAP R/3 is particularly well suited to extracting a small amount of specific data (on the order of 1 to 10 rows). It is not recommended as a substitute to using R/3 data flows to produce ABAP programs to extract large amounts of data in a batch job

If you include an R/3 table in a join with a real-time message source, Data Integrator includes the data set from the real-time message source as the outer loop of the join. If more than one supplementary source is included in the join, you can control which table is included in the next outer-most loop of the join using the join ranks for the tables.

➢ To use an R/3 table in a data flow

1. Import the metadata for the table using a datastore.

   The table appears in the table category under the datastore in the object library.



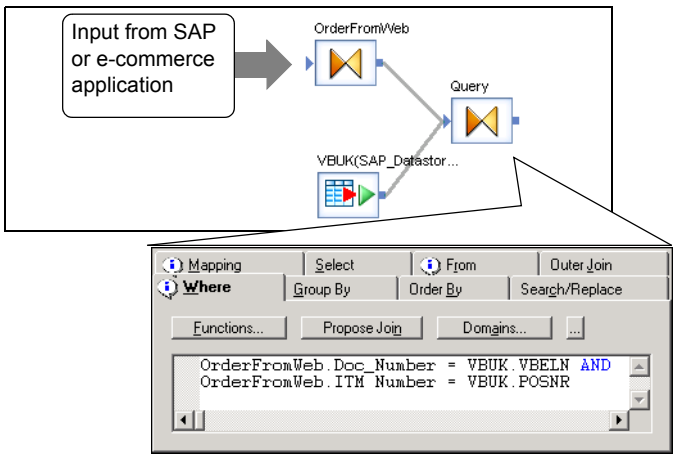
Imported R/3

2. Drag the table into the data flow.
3. Connect a query to the output of the table.

   Because R/3 tables are designed to extract a limited amount of data from SAP R/3, the first step after the source would most likely be a query with a WHERE clause that limits the data extracted.

4. In the WHERE clause of the query, add an expression to reduce the amount of data extracted from SAP R/3 by this source.

The expression in the WHERE clause is likely to be a join expression that includes specific data from the message that the data flow is processing. For example, if the data flow is using the R/3 table VBUK to supply delivery status values to respond to an e-commerce application, the join expression would restrict the values from VBUK to the specific sales order referred to in the incoming message. In this case, the join expression restricts the values to sales order number and line item numbers matching the input.



The WHERE clause joins the two inputs, resulting in output for only the sales document and line items included in the input from the e-commerce application.

Caching source data

All tables or files used as sources allow you to set a cache option. The option indicates that the data extracted is stored in memory until the data flow processing is complete.

In real-time jobs, you should not cache sources unless the data being cached is small and is unlikely to be updated in the life of the real-time process.

In batch jobs, caching can improve the performance of data flow processing by reducing the number of times a set of data is read from the database or file source. In real-time jobs, however, the improvement in performance provided by caching is minimized by the likelihood that the job reads only a small amount of data from the source for any given message. In addition, because the job reloads cached data only when the Access Server shuts it down and restarts it, cached data may become stale in memory.

IDoc targets in real-time jobs

Most actions that users perform through a Web enabled application require some response from the system in the form of data. The response can be as simple as an acknowledgment that a message was received. It can also be a specific set of data extracted from an ERP system or data cache. If you have configured a real-time job using an IDoc message source, you do not need to configure a target to send an acknowledgement because Data Integrator generates an acknowledgement (an XML message) automatically.

IDocs can also be configured as message or file targets in real-time jobs. IDoc message targets move data to SAP R/3. IDoc file targets can be used to collect data for testing, data flow validation and backup.

For example, you may want to create two, identical data flows. The first with an IDoc message target and the second with an IDoc file target. The second data flow could be used to store IDocs in a file temporarily. It could be used in a job that your system administrator will start only when it is known that the target server will be down. Later you can use a job to update the database with the data captured in the IDoc file target.

IDoc target messages can be sent from Data Integrator on a regular schedule or in real-time because you can choose to use IDoc message targets in either batch or a real-time jobs.

While there can only be one IDoc message source in a real-time job, any number of IDoc targets can be included in a data flow. In addition, if your source is not an SAP R/3 application server, a data flow can have an XML message as a source and an IDoc message or file as a target.

Any number of IDoc message targets can be added to a data flow. A data flow can also contain other targets such as a table or file.

Data flows in real-time and batch jobs that contain an IDoc message target send IDocs to SAP R/3 in the same way. The only difference is when the data is sent. Data can be sent to SAP R/3 on-demand or, if you use an IDoc message target in a batch job, data is sent when the job it scheduled to run.

When a job is executed, output statistics from the IDoc target object are written to the real-time job's monitor log. The number of IDocs written is recorded. The number of IDocs is incremented for every root-level row of data passed to SAP R/3 as IDocs contain only a single root segment.

Data and format considerations

IDoc targets can insert and update data in SAP R/3. However, you must follow SAP R/3 rules to do this successfully.

For example, to create an IDoc target in Data Integrator, you will import metadata from an existing IDoc; then re-use this metadata to create an IDoc target object in Data Integrator. This method limits the risk of creating data related errors in SAP R/3.

Data

The following examples list data consistency issues you should consider when planning to use IDoc targets.

Example 1

If your IDoc contains information to update a sales order, then the target IDoc (sales order) must:

- Be defined with a valid IDoc Type (name).

  In SAP R/3, a name for an IDoc is called an IDoc type. IDoc types are subsets of message types. An example of a message type is ORDERSP, while an example of an IDoc type is ORDERS04.

- Match the SAP R/3 order number
- Match SAP R/3 *segment field* formatting.

- Contain all mandatory IDoc segments for its message type

  SAP R/3 segments are sections of IDocs. They are equivalent to tables and nested tables in Data Integrator. Segment fields are equivalent to columns. Each SAP R/3 message type has required segments and segment fields without which, the IDoc will not be posted to SAP R/3.

  The Query Transform Definition window in Data Integrator displays SAP R/3 segments (as tables) and SAP R/3 segment fields (as columns with data types). However, the Query Transform Definition window does not indicate which segments are required, nor does Data Integrator enforce SAP's segment repetition rules. An attribute of a segment is that it has a minimum and maximum number of times that it can occur in a single IDoc.

To ensure you have required order numbers, segments and that your IDoc will not violate segment repetition rules, use SAP R/3 front-end applications to view this information.

Example 2

If your IDoc is a new sales order, then the sales order must:

- Be defined with valid a IDoc type, partner number, partner type, and message type names.

  Partner numbers and types are used to distribute IDoc data in SAP R/3. If they do not exist or you do not specify the correct values, your IDoc will not post to SAP R/3 successfully. Use SAP R/3 front-end applications to configure SAP R/3 to receive an inbound IDoc (IDoc target) from Data Integrator.

  SAP R/3 defines message types for different IDoc tasks. The message type you choose must be appropriate for the processing goal of the IDoc. Message types provide part of the required description of what is being passed to SAP R/3. Again, reference the list of message types using SAP R/3 front-end applications.

- Match SAP R/3 segment field formatting.
- Contain all mandatory IDoc segments for its message type

Data types and formats

Data Integrator supports several data types. IDoc Targets support only varchar.

- If an IDoc Target has columns configured in the varchar data type, no conversion occurs when it is sent to SAP R/3.
- If an IDoc Target column uses another data type, the data type is converted to varchar and a default ABAP/4 format.

The following table lists Data Integrator data types and their default formats:

Data Integrator data type	Is converted to...
date	ABAP/4 date (YYYYMMDD)
datetime	ABAP/4 date (YYYYMMDD)
decimal	ABAP/4 float
double	ABAP/4 float
int	ABAP/4 int
interval	ABAP/4 float
long	ABAP/4 int
real	ABAP/4 float
time	ABAP/4 time (HHMISS)
varchar	no conversion

These default formats are defined by Data Integrator to help with the data movement to SAP R/3. However, if the IDocs or the ABAP/4 modules are customized, you are responsible for modifying the default formats as needed to match the requirements.

- An IDoc can be customized in SAP R/3

Importing the metadata for your IDoc target can limit the risk of having a failed posting to SAP R/3 due to inconsistent formats.

- Input and posting modules can be customized

  IDocs are generated and read by ABAP/4 modules. IDocs are processed in SAP R/3 first by an input module and then by a posting module. ABAP/4 program modules control how data is interpreted. The configuration of these modules can be customized.

  For example, a posting module may be coded to receive a date formatted such as MMDDYYYY or DDMMYYYY instead of the standard ABAP/4 date type format (YYYYMMDD).

You must modify the metadata in Data Integrator or the format of the IDoc data generated in SAP R/3 to account for customized formats.

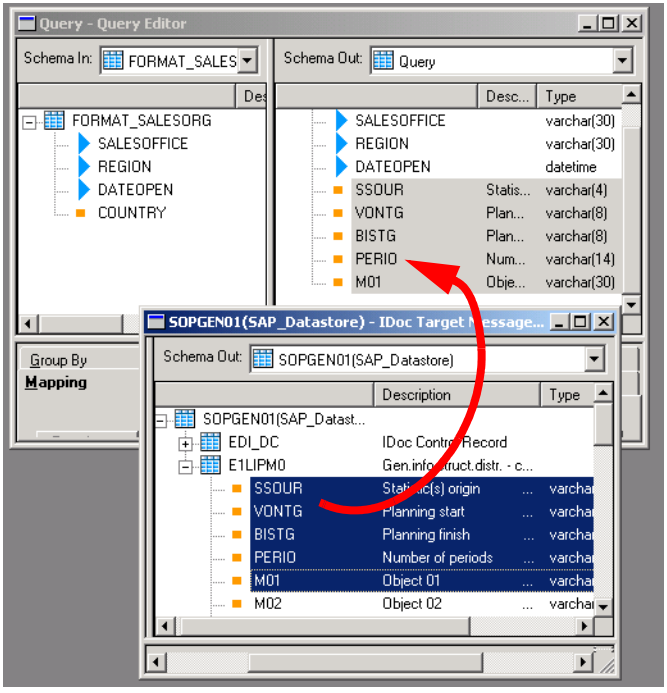
Adding an IDoc target to a data flow

When you drag an IDoc into a data flow, Data Integrator produces a popup menu of valid options for the type of data flow you are using.

➢ To add an IDoc target to a data flow

1. Select an IDoc from the object library
2. Drag it into an open real-time or batch dataflow.
3. Select **Make IDoc file target** or **Make IDoc message target**.
4. Drop a source object/s and a query.

You must copy imported IDoc file and message targets from the IDoc message target definition window into the target schema of a query before using them in Data Integrator data flows.

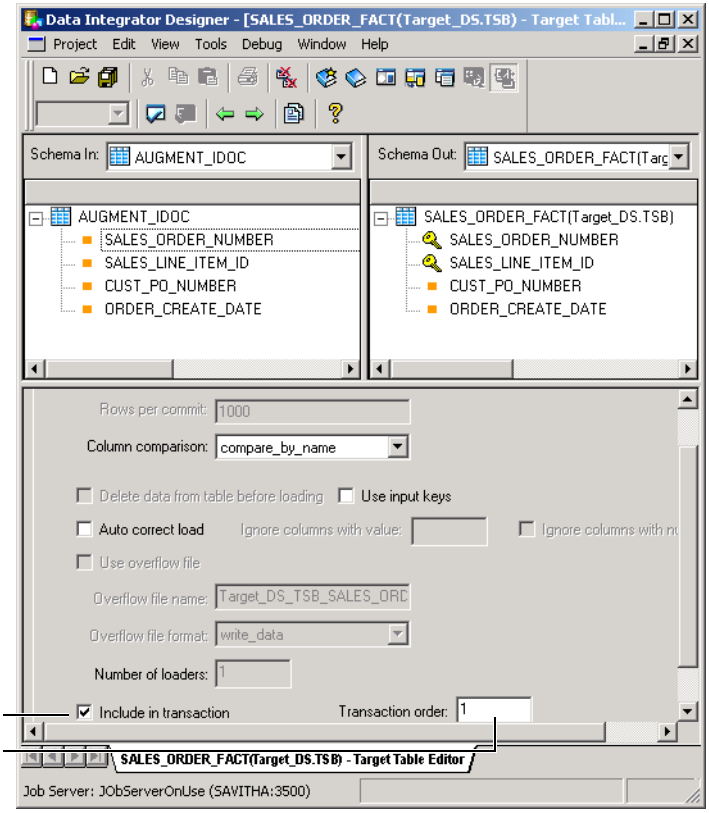


5. Connect the source object/s to the query and the query to the IDoc target object.
6. Click the name of the IDoc target.
7. Complete the IDoc target editor boxes. See "Target" on page 259 for more information.
8. Click **OK**.

Data cache targets and transactional loading

In addition to IDocs, you can use tables and files as targets in real-time jobs.Target tables in data flows also support *transactional loading*, in which the data resulting from the processing of a single message can be loaded into multiple tables as a single transaction. No part of the transaction is applied if any part fails.

You can specify the order that tables are included in the transaction through an option in each target table editor.



Loading targets as a single transaction only applies to targets in a single datastore. If the real-time job loads tables in more than one datastore, targets in each datastore are loaded independently.

In addition, the options to bulk load a table or to include pre-load or post-load commands are not available for tables used as targets in transactional loading.

See the description of transactional loading options for targets in Chapter 2, "Data Integrator Objects," in the *Data Integrator Reference Guide*.

Calling BAPI functions

A real-time job can apply data to SAP R/3 using a real-time job remote function call (RFC). The RFCs that are included in SAP R/3 releases, called Business Application Programming Interface (BAPI) function calls, such as BAPI_SALESORDER_CREATEFROMDAT1, can be particularly useful.

The metadata for the function—displayed through the function wizard in a query—describes the input and output parameters of the function, including the columns and tables included in the transaction applied to SAP R/3. See "Browsing, searching and importing metadata from SAP R/3" on page 48.

See "Calling RFC functions" on page 156 for step-by-step instructions for including calls to RFC functions in a real-time job.

Calling RFC functions

A real-time job can use RFC-enabled functions to retrieve information from and apply information to SAP R/3. With nested tables, you can include tables as input or output parameters to the function.

Functions allow you to:

- Return a specific response based on specific input you provide to the function
- Apply data to or retrieve data from more than one SAP R/3 table at a time

RFC functions can require input values for some parameters; SAP R/3 supplies default values for other inputs, and some can be left unspecified. You must determine the requirements of the function to prepare the appropriate inputs.

NOTE: To avoid returning errors from RFC calls, make sure input is formatted as required by SAP R/3. For example, all character data must be in uppercase; some values require padding to fill out the length of the data type.

To make up the input, you can specify the top-level table, top-level columns, and any tables nested one-level down relative to the tables listed in the FROM clause of the context calling the function. If the RFC includes a structure as an input parameter, you must specify the individual columns that make up the structure.

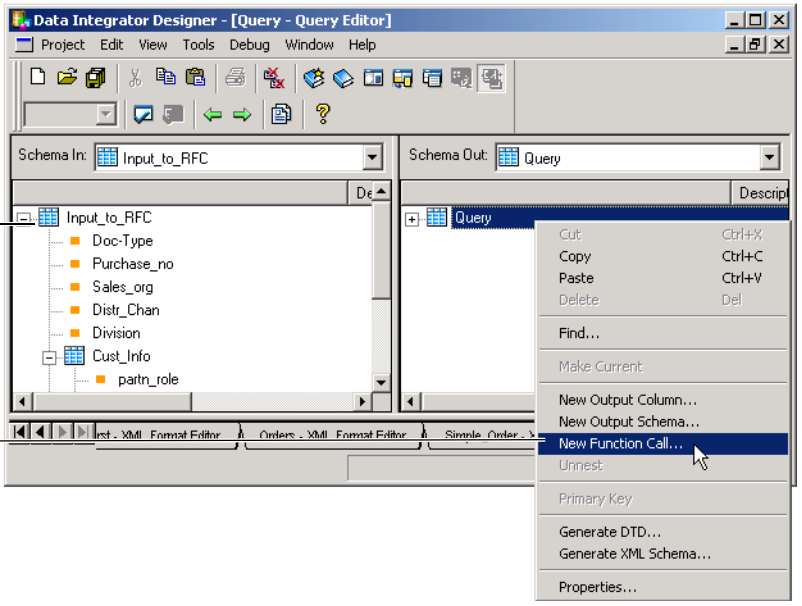
A data flow may contain several steps that call a function, retrieve results, then shape the results into the columns and tables required for a response.

➢ To call a function that includes tables as input or output parameters

1. Create a table that includes the columns required as input to the function as the input data set to a query.

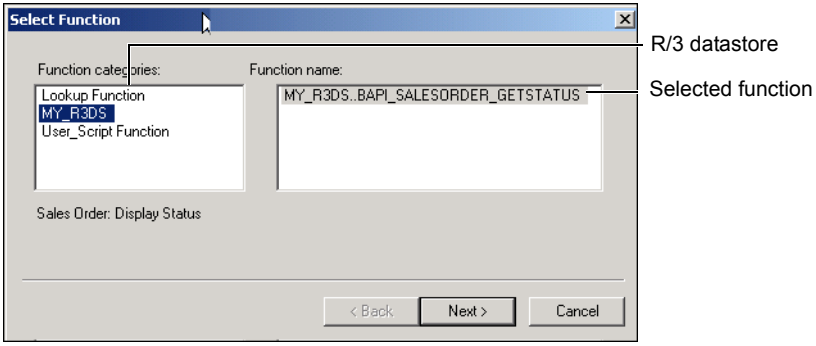
   This example uses the RFC BAPI_SALESORDER_CREATEFROMDAT1. It takes data for a sales order as input and returns a new sales order number and other status information for the items ordered.

2. In the output schema of the query, right-click and choose **New Function**.



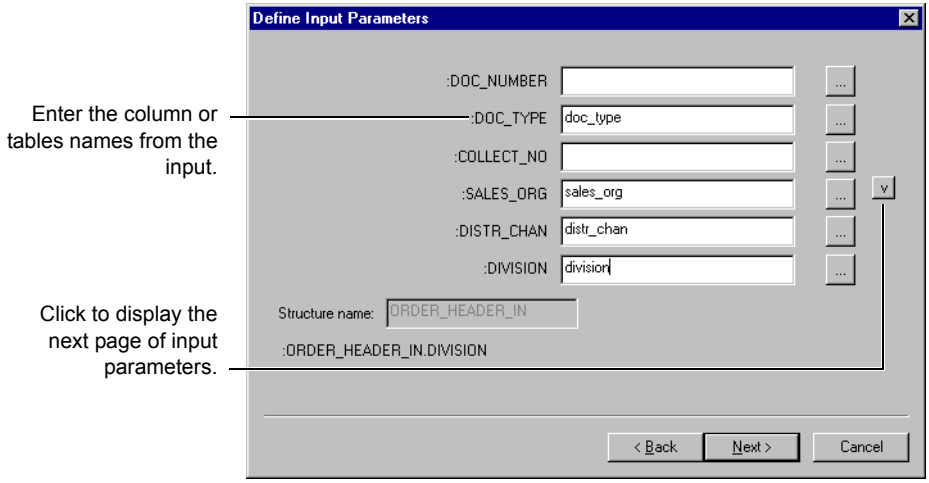
3. In the function wizard, select the datastore corresponding to the SAP R/3 application server where you want to apply the function.

4. Select the function to call and click **Next**.



5. Enter the input parameters for the function and click **Next**.

   If there are more than fit on a page in the function wizard, click the down arrow to display the next page of input parameters.



6. Select which parameters to return and click **Finish**.

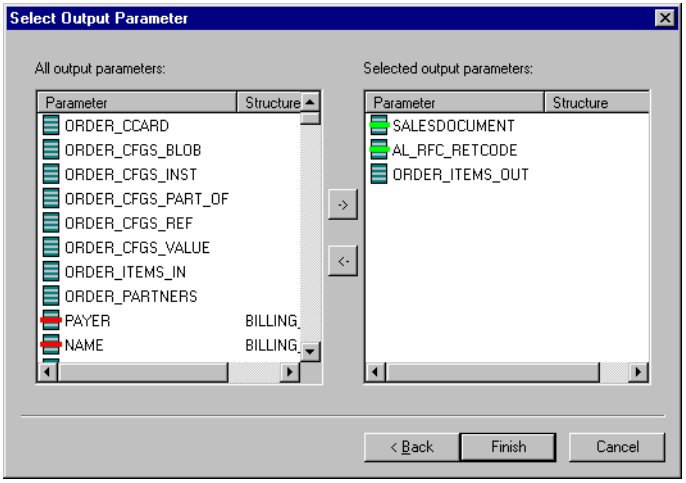
   The function wizard shows the columns and tables available as output parameters from the function:

   Columns (blue bar)

   Tables

   Columns from structures (red bar)

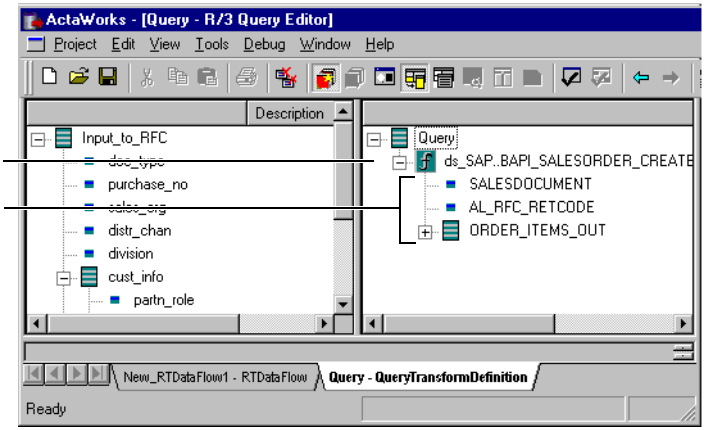
   Select a column or table you want to be included in the output, then click the right-pointing arrow in the center of the page to move it to the list on the right.



The function appears as a nested table in the output of the query. The output parameters you chose are listed as columns or tables under the function.

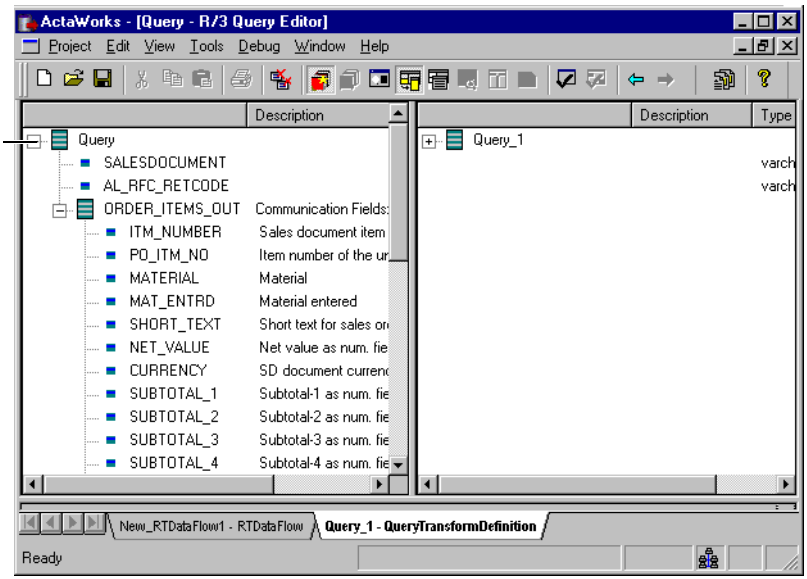
The function appears nested under the top-level output.

Output parameters appear nested under the function.



7. The function is automatically marked to unnest in the output of the query, so the nesting level indicated by the function name is not included in the input to the next step in the data flow.

The function does not appear in the input to the next step.



8 Executing Batch Jobs that Contain R/3 Data Flows

This chapter contains specific information about executing batch jobs that contain R/3 data flows; it covers the following topics:

- Data Integrator functions supporting R/3 data flows
- Validating and generating ABAP code
- ABAP job execution scenarios
- Sample execution sequence for a ABAP job
- Scheduling Data Integrator jobs from SAP R/3
- Monitoring ABAP jobs in SAP R/3
- Execution modes of R/3 data flows
- Debugging and testing ABAP jobs
- Generating ABAP code

For more information about executing jobs, see Chapter 13, "Executing Jobs," in the *Data Integrator Designer Guide* and Chapter 5, "Batch Jobs," in the *Data Integrator Administrator Guide*.

NOTE: This chapter assumes you are an experienced SAP R/3 user and have already followed similar procedures. Your system may be configured differently than the examples, so you should treat the steps in the procedures as high-level information—you may need to fill in additional details or you may need to do the steps in a somewhat different order.

Data Integrator functions supporting R/3 data flows

Using R/3 data flows, Data Integrator extracts data from SAP R/3 by directly accessing the SAP R/3 application server with Data Integrator-generated ABAP. There are two methods for Data Integrator to execute the ABAP:

- `generate_and_execute`
- `execute_preloaded`

The first method uses the R/3 datastore ABAP execution option `generate_and_execute` which calls the SAP R/3 `RFC_ABAP_INSTALL_AND_RUN` function. This method requires that users submitting the job have `S_DEVELOP` and `S_ADMI_FCD` authorizations. Most SAP R/3 production systems do not provide this combination of authorizations to one user. Additionally, the `RFC_ABAP_INSTALL_AND_RUN` function is not well suited to execute in parallel. When multiple users access the system at the same time, some user jobs will fail due to ABAP program name conflicts.

The second method uses the R/3 datastore ABAP execution option `execute_preloaded`. It uses Data Integrator-provided functions to overcome the security and stability issues incumbent with the first method.

While both methods return successful results, the Data Integrator method provides an easier and more secure solution.

See "ABAP execution options" on page 167 for more information.

Validating and generating ABAP code

Data Integrator enables you to check generated ABAP syntax errors before you run a job. The errors Data Integrator detects include mismatches between strings and numerics in parameter passing.

To request ABAP validation, you need to have the Data Integrator function module `Z_AL_SYNTAX_CHECK` installed on your SAP R/3 system. For information about the function modules and how to install them, see "Installing Data Integrator functions on SAP R/3" on page 16.

➢ To validate ABAP code created by Data Integrator

1. Select a R/3 data flow object (R/3 data flow, R/3 table, transport) in the project area or open a R/3 data flow object in the workspace.
2. Select **Debug** > **Validate** > **Validate ABAP**.

   Data Integrator checks for potential ABAP errors. If the code contains errors, a popup window indicates what the errors are. Otherwise, the window indicates that there are no errors.

➢ To generate ABAP independent of execution

1. Select a R/3 data flow object (R/3 data flow, R/3 table, data transport) in the project area or open a R/3 data flow object in the workspace.
2. Select **Debug** > **Generate ABAP**.

   NOTE: The ABAP code is saved to the directory you specified when you created the SAP R/3 datastore used to import the object's metadata.

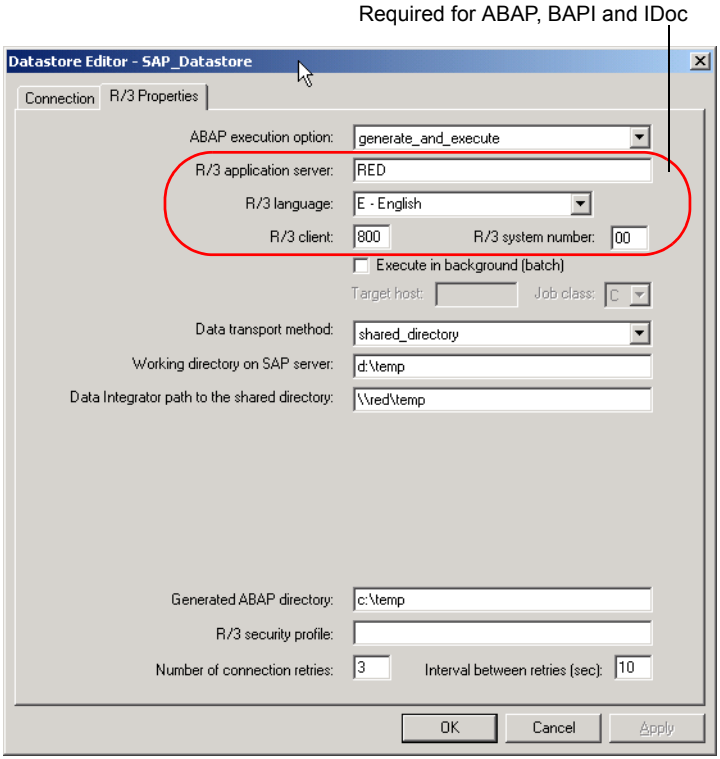
➢ To change the Generated ABAP directory

1. In the object library, select an SAP R/3 datastore.

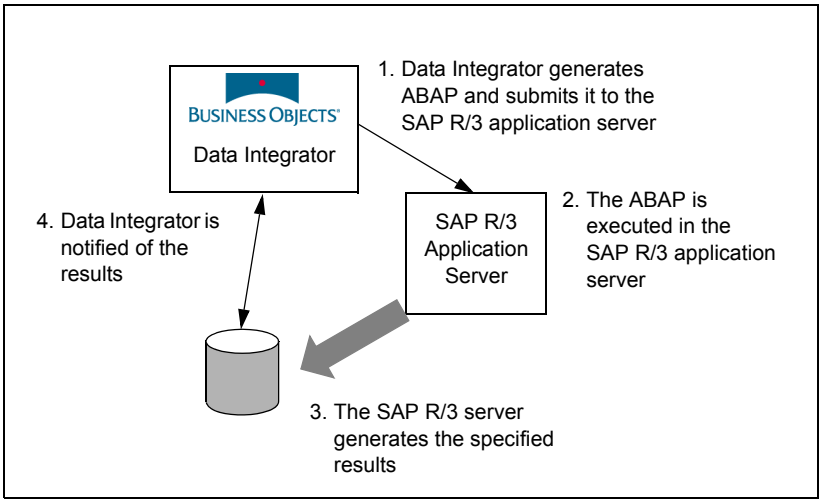
2. Right-click and select **Edit**.
3. Click the **R/3 Properties** tab.
4. Enter the new directory path in the **Generated ABAP directory** box.
5. Click **OK**.

ABAP job execution scenarios

Much of the SAP R/3 datastore definition determines how the R/3 data flow portion of an Data Integrator batch job executes and returns data. The illustration below shows the portion of the **R/3 Properties** tab that is required for all three SAP R/3 interfaces. It is circled in red. The remainder of the options on this tab are required for the ABAP interface and its R/3 data flow objects.



The following diagram shows a generic R/3 data flow execution scenario. (In this scenario, batch job execution is initiated in Data Integrator.) The diagram lists some of the key options that determine the specifics of the scenario.



The following sections explain in more detail how the datastore definition affects batch job execution when the job contains an R/3 data flow.

ABAP execution options

There are two options:

- generate_and_execute

  ABAP is sent to the SAP R/3 application server via SAP-supplied `RFC_ABAP_INSTALL_AND_RUN`.

This option is designed to be used during the development stage of a project. When a job is executed using an R/3 datastore set to the **generate_and_execute** mode, Data Integrator generates the a random ABAP program name. Use the Designer's **Tools** > **Options** > **SAP** > **Environment** option called **Prefix to ABAP program name** to assign a prefix to the random name generated by Data Integrator. Generated files are stored in the directory you defined for the datastore.

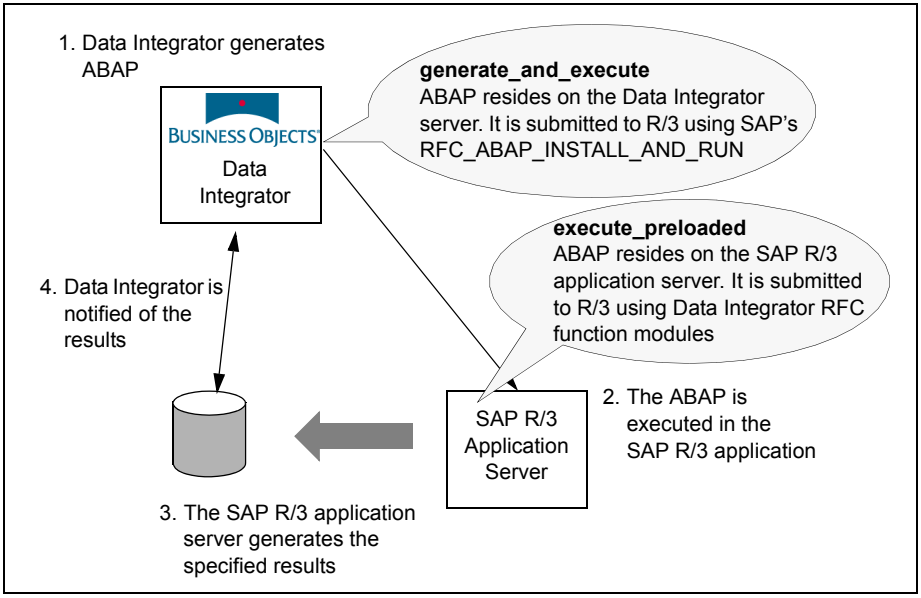
- execute_preloaded

  Communication between Data Integrator and SAP R/3 is handled by the Data Integrator RFC function modules, which must have been pre-installed on SAP R/3.

  This option is generally used only in the test and production stages of an application project. When you select **execute_preload**, Data Integrator doesn't generate ABAP to run the job. It simply calls an ABAP program name on the SAP R/3 application server. This program was created when you ran the job in **generate_and_execute** mode. Load the previously generated ABAP program onto SAP R/3, rename it as desired, and, from the **Options** tab of the dataflow's Properties window, enter its name in the **ABAP program name in R/3** box.

  The execute_preload option is unique to the Data Integrator ABAP interface. It improves performance by sending a call to the program and running it on the SAP R/3 server instead of generating the program each time the job runs.

The following diagram illustrates the key differences between generate_and_execute and execute_preloaded.



Data transport methods

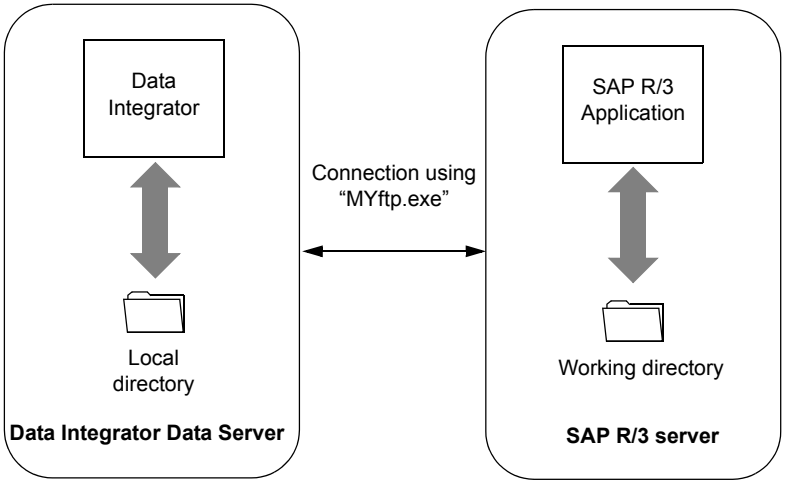
R/3 data flows in Data Integrator allow you to automatically create ABAP programs. After an ABAP program runs, it creates a flat file (transport file) on the SAP R/3 host. This file is later transferred from the SAP R/3 host to a directory local to the job server executing the Data Integrator job.

To use a data transport method for this outbound SAP R/3 file, select an option when configuring a datastore for your SAP R/3 host. Data transport options specified in an SAP R/3 datastore editor determine the method by which the results (from batch jobs that contain R/3 data flows) are returned to Data Integrator. The four options are:

- Custom Transfer — SAP R/3 server loads the **Working Directory on SAP server** with the transport file. The file is read by a third-party file transfer program and loaded to the **Local directory**.
- Direct download — SAP R/3 server transfers the data directly to the **Client Download directory**.
- FTP — SAP R/3 server loads the **Working Directory on SAP server** with the transport file. The file is read using the **FTP relative path to the SAP working directory** and written to the **Client download directory**.
- Shared directory — SAP R/3 server loads the **Working Directory on SAP server** with the transport file. The file is read using the **Data Integrator Path to the shared directory**.

Custom Transfer method

Using the custom transfer method, you specify a third-party file transfer program which Data Integrator uses to transport data between the remote SAP R/3 server and a local directory on an Data Integrator Job Server machine. The transfer from SAP R/3 to Data Integrator is enabled using a data transport object (used in R/3 data flows).



Data transport method:	custom_transfer
Working directory on SAP server:	\\temp
Local directory:	d:\temp
Custom transfer program executable:	MYftp.exe
Custom transfer user name:	ARJones
Custom transfer password:	*******
Custom transfer arguments:	$AW_USER $AW_PASSWORD $AW_LOCAL_DIR $AW_FILE_NAME $AW_SAP_WORK_DIR

The Local and the Generated ABAP directory can be the same directory. Both directory paths are relative to the Data Integrator Job Server.

When you select the Custom Transfer option, provide:

- A transfer program (to be invoked during job execution)
- (optional) Additional arguments used by the transfer program such as security, compression, or Data Integrator system variables.

Like other datastore settings, custom transfer program settings can be overridden if they are changed in a particular R/3 data flow. You can also edit the custom transfer option when exporting an R/3 datastore.

See "File transfers" on page 139 of the *Data Integrator Designer Guide* for an introduction to the use of the custom transfer option and Data Integrator system variables.

Datastore profiles support the custom transfer option. See "Defining profiles" on page 111 of the *Data Integrator Designer Guide* for an introduction to the use of the Datastore profiles.

Data Integrator system variables for transferring a file from

SAP R/3

System variables can be used to provide the connection data you enter into Data Integrator to the custom transfer program. The following system variables can be used in the **Custom Transfer Arguments** box:

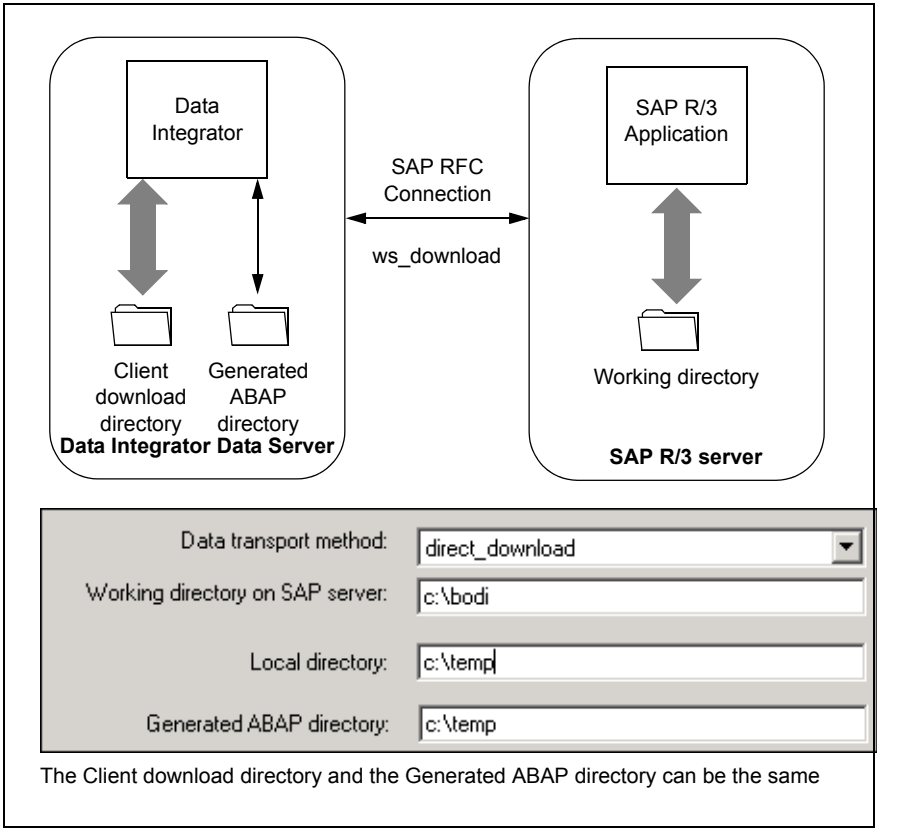
Data entered in this field:	Is substituted for this variable if it is defined in the Arguments box
R/3 application server	$AW_SAP_HOST
Working directory on SAP server	$AW_SAP_WORK_DIR
Custom transfer user name	$AW_USER
Custom transfer password	$AW_PASSWORD
Local directory *(Local* denotes the job server computer)	$AW_LOCAL_DIR
File name This field is on the data transport object and it is part of an R/3 data flow.	$AW_FILE_NAME

Direct_download transport method

With direct_download, the data produced by SAP R/3 after running the ABAP program is returned by the Data Integrator Job Server computer via a ws_download function call. The data is stored in the **Client download directory**.

You should avoid using this method except when you need a quick method to return a small amount of data — it may not be reliable and does not scale well.

The direct_download execution scenario looks like the following.

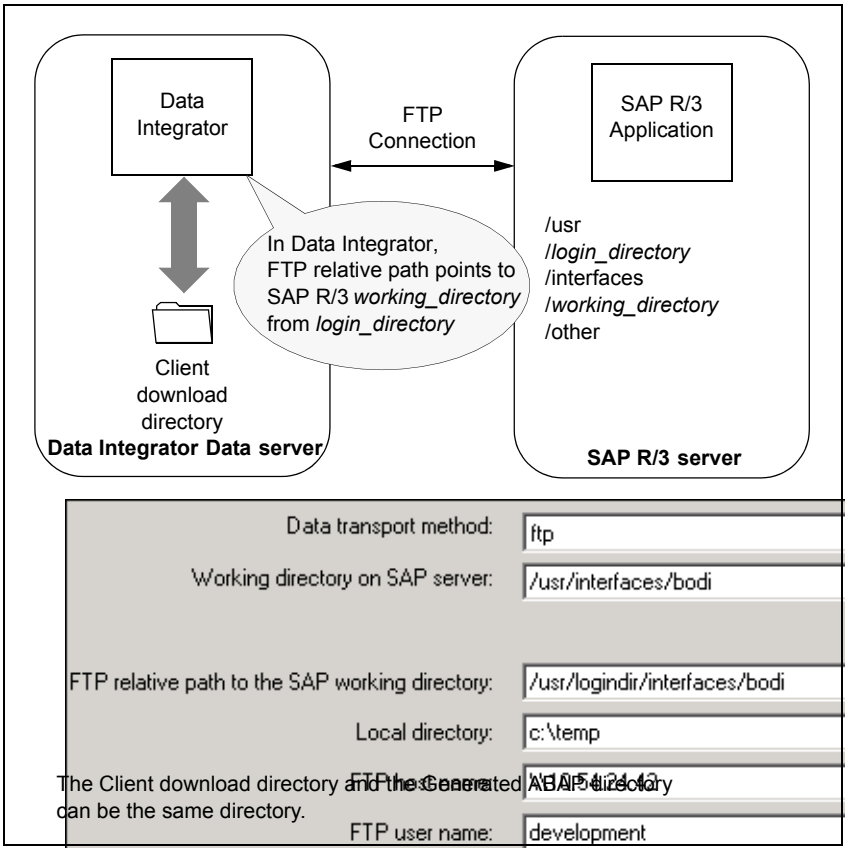


The Client download directory and the Generated ABAP directory can be the same

FTP transport method

The FTP method is most useful in heterogeneous environments (Windows NT and UNIX). It scales well and provides a good level of security.

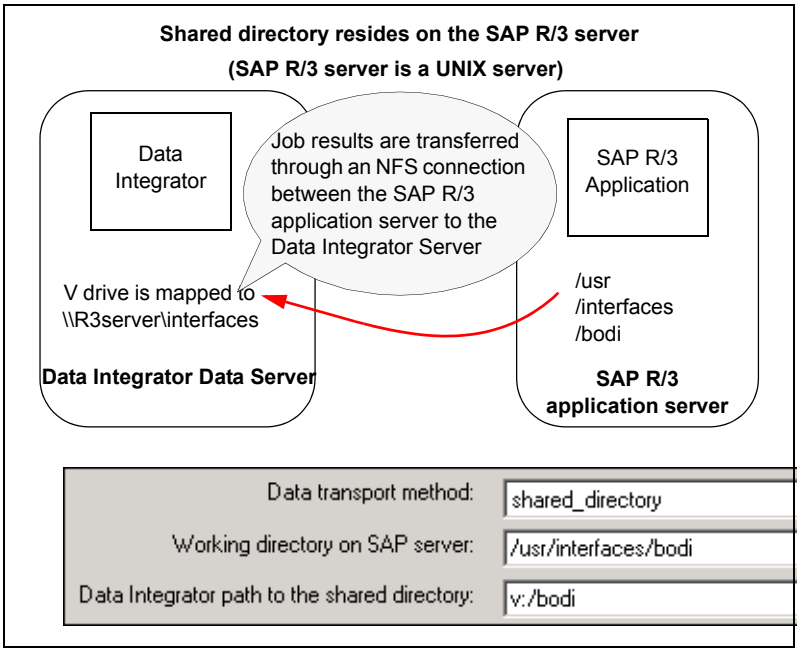
The FTP execution scenario looks like the following.



Shared directory transport method

With shared directory transport method, both Data Integrator and SAP R/3 have direct access to the directory where data is stored. This method works best in a homogeneous environment (both systems are on Windows NT). The security risk inherent in a shared directory may pose a problem for some organizations. If this is the case, FTP is probably a better choice.

The shared directory execution scenario looks like the following.



Shared directory requirements

To use the shared directory data transport method, set up a shared directory between the SAP R/3 application server and Data Integrator Job Server computer:

- If the SAP R/3 application server is on UNIX, use NFS or a similar disk sharing process. Use these steps to verify that the Job Server can access files generated by SAP R/3:
  1. Log into the SAP R/3 application server computer.
  2. Sign onto the NFS system.
  3. Navigate to the directory you specified in the datastore as the "Working directory on SAP server."
  4. Enter the command `touch bodi`.

Note that you can use any term instead of `bodi`. The touch command creates a file with zero length in the current directory.

5. Log into the Data Integrator Job Server computer.
6. Navigate to the directory you specified in the datastore as the "Data Integrator path to the shared directory."
7. Look for the file `bodi`.
8. Use the command `file bodi` to verify that the Job Server can read the file.

- If the SAP R/3 application server is on Windows NT or Windows 2000, use Windows NT or Windows 2000 directory sharing to provide mutual access to the staging area. Include paths to:
  - The shared device on the application server
  - The shared device on the Data Integrator Data Server

The most common data transport method is the shared directory method. However, since both the Data Integrator and the SAP R/3 application server can run on either Windows or UNIX platforms, mapping between computers can be broken if one of the computers is rebooted.

Business Objects recommends:

- If you are running SAP R/3 and the Data Integrator on UNIX, make sure that your network will automatically mount the volumes you want to use when any computer is re-booted.
- If you are running the SAP R/3 application server on Windows and the Data Integrator Data Server on UNIX and you re-boot the UNIX computer, you may need to re-map Data Integrator to the SAP R/3 application server. If your jobs return error messages that read, "The SAP Job Canceled Host ... File xxx cannot be opened....," disconnect then remap the NFS connection in SAP.

For more information, see "ABAP job execution scenarios" on page 166.

Sample execution sequence for a ABAP job

This section describes how a batch job that contains an R/3 data flow (also called an ABAP job) executes.

The following execution sequence is for a hypothetical job with the following characteristics:

- The batch job contains a data flow that, in turn, calls an R/3 data flow.
- The R/3 data flow extracts data from a subset of the columns in an SAP R/3 table.
- The calling data flow passes the output from the R/3 data flow into a query transform and loads a target table or file.
- The datastore options selected are **generate_and_execute** and **shared_directory**.
- The scenario is valid for jobs initiated in Data Integrator as well as jobs initiated in SAP R/3.
- Oracle is assumed for this example, but the process is similar for other environments.

The execution steps

1. Data Integrator retrieves the job specification from the repository.
2. Data Integrator validates the job specification.
3. Data Integrator performs any required optimizations.
4. Data Integrator compiles the job specification.

   As part of this step, Data Integrator generates an ABAP program corresponding to the R/3 data flow specification and writes it to the path specified in the R/3 datastore options.

All the parameters are replaced with data before ABAP is submitted to SAP R/3 to execute.

5. Data Integrator invokes `RFC_ABAP_INSTALL_AND_RUN` with the ABAP program as a parameter.
6. Depending on whether the ABAP job is running in the background or foreground (See "Execution modes of R/3 data flows" on page 191), one of the following happens:
   - If the R/3 batch option is not selected, then the ABAP program runs as a foreground job in dialog user mode in SAP R/3. The `RFC_ABAP_INSTALL_AND_RUN` return code is checked for status. (`RFC_OK` return status implies successful completion of the ABAP program.)
   - If the R/3 batch option is selected (see "Debugging and testing ABAP jobs" on page 194), then the ABAP program runs in the background in SAP R/3. A special Data Integrator-internal ABAP program runs in SAP R/3 and polls for status from the job status table TBTCO based on the job name that was assigned. The status column TBTCO-STATUS provides status information.
7. The ABAP portion of Data Integrator SAP R/3-related jobs cannot run in parallel, so Data Integrator serializes the requested processes.

   NOTE: ABAP jobs can run in parallel when you specify execute_preloaded and the job executes with Data Integrator RFC.
8. Data Integrator schedules the job by calling SAP R/3 functions JOB_OPEN and JOB_CLOSE.
9. The job is scheduled as a class C job.
10. Data Integrator queries the job status every 15 seconds.
11. When the job finishes, the ABAP is deleted, but the job is still listed with a status of "finished."

12. When the ABAP program completes successfully, it writes output to the file name specified in the data transport in the path specified as **Working directory on SAP server** in the datastore definition.
13. The R/3 data flow is done and returns to the calling data flow.
14. Data Integrator reads the output file generated from R/3 data flow execution and processes it.

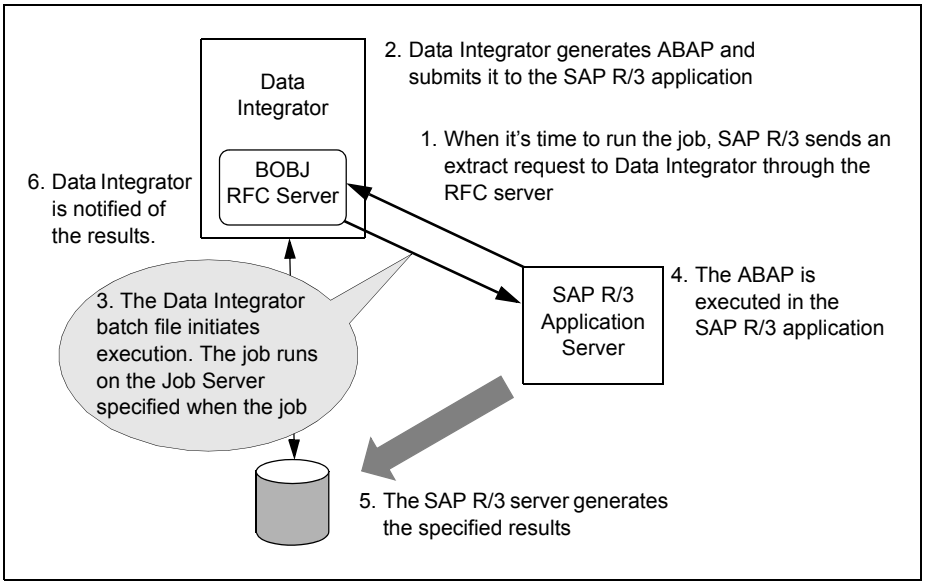
    Processing may include performing joins with other R/3 data flows or tables from other source databases or source files. It may also include any column transformations and/or transforms applied to the entire R/3 output data set. Once all transformations are done, Data Integrator is ready to place the resulting data set in an output file or target table.
15. If the output is to a target table, these are the options available for loading the table:
    - If Oracle is the target database, then bulk load is the usual choice (see Chapter 4, "Using Bulk Loading," in the *Data Integrator Performance Optimization Guide*).

      If bulk loading is turned on, Data Integrator generates the control file (`.CTL`) and the input (to the SQL*Loader) data file (*generated_filename*`.dat`) in the `C:/Data Integrator/Log/BulkLoader/` directory or the bulk loader directory specified in the Oracle datastore. It then invokes SQL*Loader on the client side (the system running Data Integrator) to load the data according to the control file specification.
    - If bulk loading is not used, Data Integrator builds and sends batch INSERT SQL statements to the Oracle server to execute and load the rows. A batch size option specifies the commit size.

Scheduling Data Integrator jobs from SAP R/3

This section describes the sequence of steps you need to follow to prepare jobs initiated in SAP R/3.

The following diagram shows an overview of the execution process for jobs initiated in SAP R/3.



Overview of preparation steps

For jobs where the schedule is set and the job is initiated by SAP R/3, you must do the following.

- In Data Integrator:
  1. Export the job to a batch file (`.bat` file for Windows NT environments, and `.sh` file for UNIX environments). See "To export a job for scheduling" on page 48 of the *Data Integrator Administrator Guide*.
  2. Ensure that the Data Integrator Job Server is running (or will be running when the job begins to execute). See "Ensuring that the Job Server is running" on page 318 of the *Data Integrator Designer Guide*.
  3. Start the RFC server, or ensure that it is running (or will be running when the job begins to execute). See "Using the Data Integrator RFC server" on page 184.
- In SAP R/3:
  1. From transaction SM59, register the Data Integrator RFC server with SAP R/3 (see "Registering the Data Integrator RFC server in SAP R/3" on page 185).
  2. From transaction SE38, create the ABAP program that will execute the job schedule (see "Creating the program that will execute the schedule" on page 186).
  3. From transaction SM36, create the job schedule (see "Creating a job schedule in SAP R/3" on page 189).
  4. (Optional, at the time the job runs) From transaction SM37, monitor the job as it runs (see "Monitoring ABAP jobs in SAP R/3" on page 190).
- In Data Integrator:

  (Optional, at the time the job runs) Monitor the job as it runs.

The Data Integrator Designer does not need to be running. If it is running, the execution window opens when the job begins to run.

Using the Data Integrator RFC server

The Data Integrator RFC (remote function call) server is the execution link for Data Integrator jobs initiated by SAP R/3 or SAP BW.

Starting the RFC server

The RFC Server is a command-line program.The syntax for the command that starts the server (in both the Windows NT and UNIX environments) is:

```
RfcSvr -a progid -g gateway_host -x sapgwxx
```

Where:

progid	The Data Integrator program ID used in the RFC destination definition in SAP R/3.
gateway_host	The SAP R/3 application server name.
sapgwxx	The SAP R/3 gateway system number.

➢ To start the Data Integrator RFC Server

1. Open a Microsoft command prompt or go to the UNIX command line.
2. Type a command similar to the following:

   ```
   RfcSvr -a MYPROGID -g SAPS04 -x SAPGW04
   ```

   Where `MYPROGID` is the Data Integrator program ID used in the RFC destination definition in SAP R/3, `SAPS04` is the SAP R/3 application server name, and `SAPGW04` is the SAP R/3 gateway system number.

Ensuring that the RFC server is running

Ensure that the Data Integrator RFC Server is running by checking the running processes in any of these environments:

- SAP R/3
- Windows NT (if your RFC server has been installed in the NT environment)
- UNIX (if your RFC server has been installed in the UNIX environment)

If the Data Integrator RFC server is running, it appears among the running processes listed.

In the prior example, `RfcSvr` should be listed among the running processes.

Registering the Data Integrator RFC server in SAP R/3

➢ To register the RFC Server in SAP R/3

1. Run transaction **SM59**.
2. If you have not yet created the TCP/IP connection between Data Integrator and SAP R/3, do so from the TCP/IP connection choice on this screen.
3. When the TCP/IP connection has been created and is active, click the **Create** button on the Display and Maintain RFC Destinations screen.
4. In the RFC destination box, add an RFC destination name.

   This can be any name you choose. The name used in the example is Data Integrator Sales Rapid Mart.
5. Under **Technical Settings** > **Connection type**, choose **T** (for TCP/IP).
6. (Optional) Under **Description**, add descriptive text.

7. Click or press **Enter**.

   The RFC Destination screen appears.

8. In the **Registration — Program ID** text box, fill in the name of the Program ID.

   This is the job name you specified under the `-a` option of the `RfcSvr` command you used to start the Data Integrator server (see "Starting the RFC server" on page 184).

9. Click the **Registration** button.

   The Data Integrator RFC server is now registered to SAP R/3.

10. Click the **Test connection** button to check if the connection is successful.

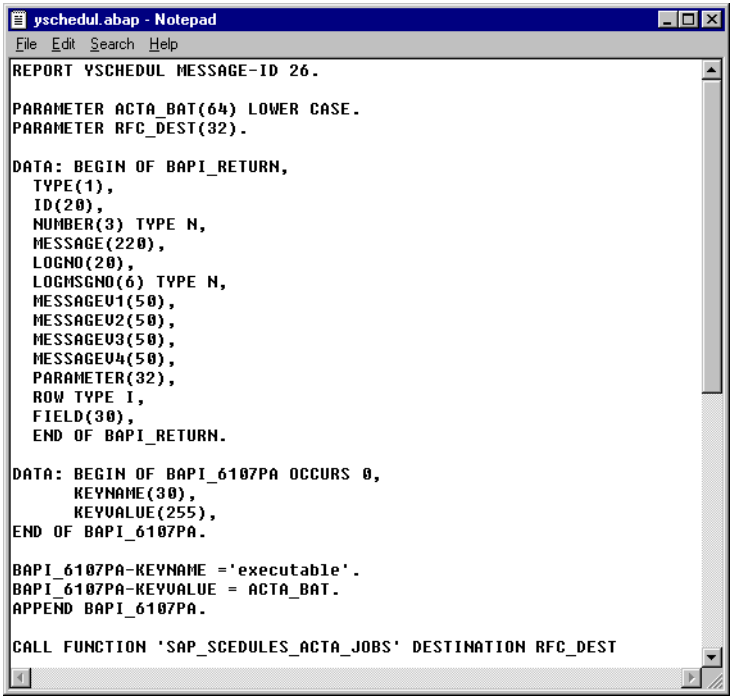
    If the connection is not successful, you receive an error message. You need to debug the problem before proceeding.

Creating the program that will execute the schedule

You need to create the ABAP program that will initiate communications with the Data Integrator RFC server to execute the job schedule.

We recommend that you use the model ABAP code (`yschedul.abap`) provided for you in the `Data Integrator\Admin\R3_Functions\schedule` directory (and the procedures in this section assume you are doing that). You can also create your own, if you like.

The following screen shows a portion of the `yschedul.abap` program.



➢ To create an ABAP program in SAP R/3

1. Move program `yschedul.abap` from the Data Integrator directory (or wherever it is stored) to your SAP R/3 system.
2. Run transaction **SE38**.
3. Select **yschedul** in the **Program** drop-down box.
4. Click the **Create** button.

   The Program Attributes screen appears.
5. From the Program Attributes screen:
   - In the **Title** box, add a title.
   - In the **Type** box, enter **1** (online).

6. Click **Save**.
7. Click the **Source code** button.
8. In the **Create object catalog entry** screen, enter the development class for this job.

   This can be any development class you have created for this purpose. We recommend using `zact`.
9. Click **Save**.

   An ID number has been assigned to this job (the correction and transport number).
10. Click **Save**.

    The ABAP/4 Editor: Edit Program screen appears.
11. Choose **Upload** from the **Utilities** > **Upload/Download** menu.
12. From the Upload dialog, navigate to `yschedul.abap`.

    (The default location is `c:\Data Integrator\Admin\R3_Functions\Schedule`).
13. Click **OK**.

    This returns you to the ABAP/4 Editor: Edit Program screen.
14. Click **Save**.
15. Click the back arrow to get back to the ABAP/4 Editor: Initial Screen.
16. From the ABAP/4 Editor: Initial Screen, click **Execute**.
17. Add the fully qualified path to the batch file containing the job you want to run, and the RFC destination name.
18. Click **Save**.
19. Enter a variant name and (optionally) a description.

    A variant name is a collection of parameters that determine how a job is run. Use any name that is meaningful to you.
20. Click or press **Enter**.

The SE38 procedure is finished.

Creating a job schedule in SAP R/3

➢ To create a job schedule in SAP R/3

1. Run transaction **SM36**.

   View the Define Background Job screen.

2. From the Define Background Job screen:
   a. Add a job name.
   b. Add the target host (the SAP R/3 application server).
3. Click the **Steps** button.

   The Create Step 1 screen appears.

4. In the Create Step 1 screen, click the **ABAP/4** button.

   Items in the ABAP program section are now available.

5. Add the program **Name** (`yschedul`).
6. Add the appropriate variant information (`AWRM`, in the prior example).
7. Click **Save**.
8. Return to the **Define Background Job** screen.
9. Click the **Start date** button.
10. Add the date and time for the job to run.
11. Fill in additional information, as needed, and click **Save** to complete the job schedule.

The job runs at the time you scheduled it.

If Data Integrator is active when the job begins to run, the execution window (with the trace button active) appears.

Monitoring ABAP jobs in SAP R/3

You can monitor an ABAP job generated by Data Integrator just as you would any R/3 ABAP job, through the SM37 batch monitoring window.

The Data Integrator job name appears in the list of jobs in the SM37 window.

Execution modes of R/3 data flows

There are two execution modes for R/3 data flows:

- Dialog execution mode
- Batch execution mode

NOTE: You cannot use the dialog and batch capabilities at the same time.

Dialog execution mode

In *dialog execution mode*, Data Integrator connects to SAP R/3, submits and executes an ABAP program as a foreground job in SAP R/3, and then disconnects from SAP R/3 after the program execution finishes.

Dialog is the default execution mode.

An ABAP program executed in dialog mode is subject to the dialog execution time constraint (typically set to be 5 minutes) defined in the SAP R/3 system profile. If the execution time of a program executing in dialog mode exceeds the specified time-out period, SAP R/3 aborts execution of the program, producing an ABAP data flow runtime error and an exception error in Data Integrator.

The execution time constraint is set using the `vdisp/max_wprun_time` parameter in the profile corresponding to the system executing the job.

Batch execution mode

In *batch execution mode*, Data Integrator connects to SAP R/3, submits the ABAP program as a background job in SAP R/3, and then disconnects from SAP R/3 after the program is submitted. Data Integrator schedules and executes the submitted program as a background job.

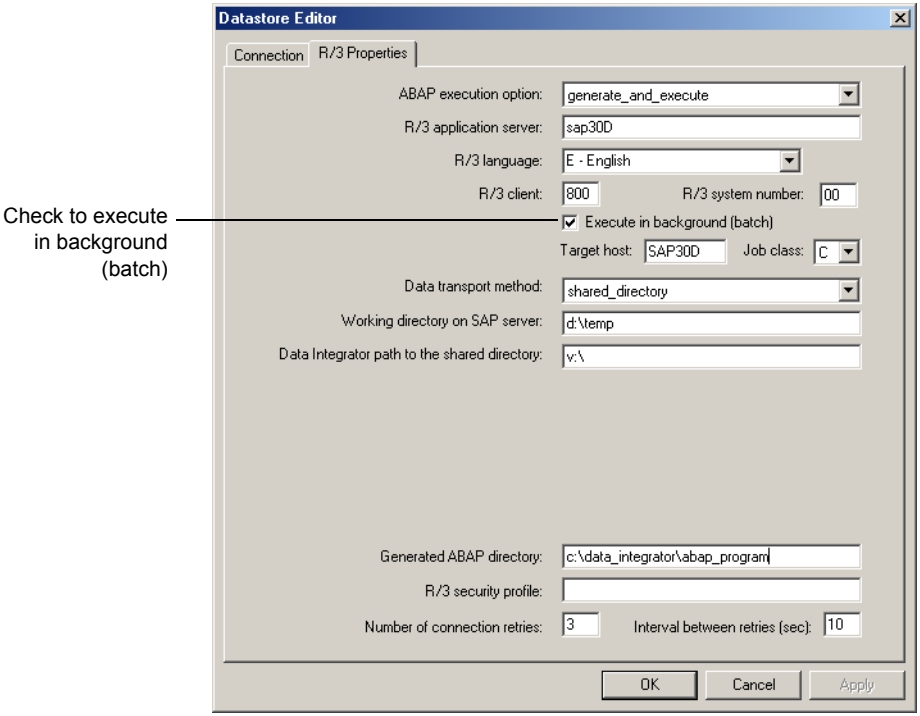
An ABAP program executed in batch mode is not subject to the dialog execution time constraint mentioned above. However, there is a slight performance overhead due to background scheduling in SAP R/3 and execution status checking by Data Integrator.

NOTE: If you kill a Data Integrator job containing an R/3 data flow defined to run in batch mode, the batch program still executes. You must use SAP R/3 facilities to terminate the batch program.

To run the ABAP generated by the R/3 data flow in batch execution mode, select the **Execute in Background (batch)** check box in the R/3 properties tab of an R/3 datastore. Make sure that the Data Integrator SAP R/3 functions are uploaded to the SAP R/3 system running the job. See "Installing Data Integrator functions on SAP R/3" on page 16 for more information.

When more than one R/3 data flow is running concurrently in batch mode, the batch job submissions are serialized. However, after the batch jobs are submitted, they run in parallel in the SAP R/3 application server if the server has more than one batch processor available.

You specify background (batch) processing for the ABAP job on the SAP R/3 datastore definition screen (R3 Properties tab), as shown in the following screen.



NOTE: The R/3 application server name is not case sensitive, but the Target host name must be entered as it is actually registered in the SAP R/3 system.

Debugging and testing ABAP jobs

Generating and executing ABAP programs from Data Integrator can result in the following kinds of errors:

Task	Type of error	Result
Design	Validation errors	Execution is halted. Information is available interactively (in a popup error window).
ABAP generation	Generation errors	Execution is halted. Information is available interactively (in a popup error window).
ABAP validation	ABAP syntax errors	Execution is halted. For errors detected by Data Integrator, detailed information is provided in the error log. For errors detected by R/3, the message in the error log is RFC_CLOSED. In this case you need to run the ABAP program manually in R/3 to identify the problem more precisely.
ABAP execution (dialog mode)	ABAP execution errors	Execution is halted. Information is available interactively (in a popup error window).
ABAP execution (batch mode)	Job execution errors (job was cancelled)	Execution is halted. The job log from R/3 is available in the Data Integrator error log.
ABAP execution	Function exceptions	Causes an exception. Information is available in the Data Integrator error log if the Data Integrator function modules have been installed on SAP R/3.

For additional information about the function modules including how to install them, see "Installing Data Integrator functions on SAP R/3" on page 16.

Generating ABAP code

You can generate ABAP code to be executed manually on SAP R/3.

➢ To generate ABAP code

1. Open the R/3 data flow.
2. Choose **Debug > Generate ABAP**.

   When the generation is complete, Data Integrator opens the text file containing the ABAP program.

   This file name and location were specified when you defined the R/3 data flow.

9 Executing Batch Jobs in the SAP BW Environment

This chapter contains the following sections:

See Scheduling Data Integrator jobs from SAP R/3 for information about how to set up the RFC server and schedule batch jobs with BW targets (also called SAP BW jobs). SAP BW jobs can be scheduled from SAP R/3 or from Data Integrator (by using BW BAPI functions).

NOTE: This chapter assumes you are an experienced SAP BW user and have already followed similar procedures. Your system may be configured differently than the examples, so you should treat the steps in the procedures as high-level information—you may need to fill in additional details or you may need to do the steps in a somewhat different order.

Overview of the job execution process

To prepare to move data into BW using Data Integrator:

- Define the appropriate SAP R/3 security profiles and authorizations.

  See "SAP R/3 user authorizations" on page 33.

- Install the Data Integrator Designer and Job Server, if they are not already installed.
- Create Data Integrator datastores for:
  - The system where your source data resides.
  - The system where your BW InfoSource (Transfer Structure) resides. See "SAP BW as a target" on page 60.
- Import the source metadata.
- Import the target metadata.
- In the Data Integrator Designer, construct the job that will extract the data from the data source to the BW InfoSource.
- In the Data Integrator Designer, validate the job and all its lower-level objects (work flows, data flows, etc.).
- In the Data Integrator Administrator, add a connection to the repository that is storing the BW job, then from the Batch Job Configuration page, export the job to a batch file.

  See "To export a job for scheduling" on page 48 of the *Data Integrator Administrator Guide*.

- Start the Data Integrator RFC server (see "Starting the RFC server" on page 184).
- Ensure that the Data Integrator RFC server is running (see "Ensuring that the RFC server is running" on page 185).
- Go to SAP BW to schedule the load job or user the BW BAPI function to schedule the job from within Data Integrator.

Setting up SAP BW InfoCubes and InfoSources

You will use the BW Administrator Workbench to set up SAP BW.

➢ To set up the environment

1. Create and activate the InfoCube or InfoCubes where the extracted data will ultimately be placed.

   Your InfoCubes should be defined to hold data that extracted using Data Integrator.

2. Create and activate the InfoSource where Data Integrator will place the extracted data:

   a. Go to the InfoSource window (**InfoSources** tab).

   b. Right-click InfoSources at the top of the hierarchy and select **Create Application Component**.

   c. Complete the window that appears with appropriate information. So for **Application comp**, you might enter: MYAPPPCOMP. For **Long description**, you might enter: My application component.

   d. Click **Enter**.

      The application component is created and appears in the hierarchy list.

   e. Right-click the name of your new application component in the component list and select **Create InfoSource**.

      The Create InfoSource: Select Type window appears.

   f. Select the type of InfoSource you are creating — **Transaction data** or **Master data** — and click **Enter**.

      When you select **Transaction data**, the Create InfoSource (Transaction data) window appears.

   g. Enter the appropriate information and click **Enter**.

      The new InfoSource appears in the hierarchy under the application component name.

Designating Data Integrator as a source system

➢ To designate Data Integrator as a source system with an active status

1. Do one of the following:
   - If the source system does not exist, right-click and choose **Create source system**, then create the new source system as explained in the following steps.
     a. From the BW Administrator Workbench window, right-click **Source systems** at the top of the hierarchy, and choose **Create source system**.

        The Select Source System Type window appears.

     b. Select the last check box on the window and click **Enter**.

        The Create source system window appears.

     c. Enter information about your system, then click **Enter**.

        The RFC Destination window appears.

     d. Click **Registration**.

     e. Enter the program ID.

     f. Click the **Save** icon to complete registration.

     g. Exit this menu and return to the Administrator Workbench InfoSource screen.
   - If the Source System already exists, right-click the InfoSource and select **Assign Source System**.

     The Transaction data - InfoSource: Assign source system window appears. Enter the information required and click **Enter**.
2. Click the button to the right of the InfoSource name to view the Maintain Communication Structure window.
3. Add or change information, as required.

4. Activate the Transfer Structure for this Source System. Follow these steps:

   a. Right-click the InfoSource name and select **Update InfoSource Metadata**.

      The Maintain Meta Data window appears.

   b. Enter or change information as required, then click **Save**.

   NOTE: If you will be performing both full and delta updates, check **Delta update**.

5. Map the transfer structure fields to the communication structure fields.

   NOTE: You must successfully complete all previous steps to begin this step.

   a. Right-click the name of the external source system and select **Maintain Transfer Rules**.

      The Maintain Transfer Rules for Communication Structure window appears.

   b. Map the fields and add transformation information as required.

   c. Click **Activate**.

      The transfer rule is now activated.

Creating the job schedule

You create the job schedule using BW Administrator Workbench. The schedule runs automatically at the times and dates indicated.

If you have Data Integrator Designer or Administrator open when the job runs, the trace log appears automatically and remains open and active while the job is processing, just as it does for a job whose execution is initiated within Data Integrator.

➢ To create the job schedule (InfoPackage)

1. Right-click the Source System name in the InfoSource view and select **Create InfoPackage**.

   The Create InfoPackage window appears.

2. Complete the InfoPackage name (Long description) and click **Enter**.

   The Scheduler (Maintain InfoPackage) appears.

3. Select the **3rd Party selections** tab.
4. Complete the information required (the name of the Data Integrator batch job) and click **Check** to activate.
5. Select and view the **Update parameters** tab.
6. Select and view the **Schedule** tab.
7. Check **Instant loading of data online** or **Background processing**.
8. Click **Start** to create the job.

Job execution process

BW initiates the request that causes Data Integrator to extract data from the specified sources and make it available to SAP BW.

NOTE: Alternately, You can use BW BAPI functions BAPI_IPAK_CREATE and BAPI_IPAK_START to send a load request to the Data Integrator RFC server from within Data Integrator. However, when you use these functions, you should also include authorizations for RSAB and BAPT function groups.

The batch job execution process consists of the following steps:

- The SAP BW Scheduler executes code that sends the load request to the Data Integrator RFC server.

  The load request contains the following information:

  - The name of the batch file you exported from Data Integrator
  - The request ID, request date, request time, and the full/delta update flag

- The Data Integrator RFC server calls the Data Integrator Job Server which then invokes Data Integrator to extract and load the BW InfoSource.
- Request information is passed to Data Integrator which executes the job and loads data to BW with the request ID to identify into which transfer structure it is loading.
- Data Integrator extracts the data.
- Before loading to BW (through the BW-STA BAPI interface), Data Integrator first checks whether BW InfoSource metadata changed since the last data import.

  If the metadata changed, Data Integrator produces an error message indicating that you must re-import the Transfer Structure metadata into the Data Integrator repository before proceeding to import the target data into the Transfer Structure.

The following must all be running for job execution to complete successfully:

- The BW application server where the BW Scheduler resides
- The Data Integrator RFC server
- The Data Integrator Job Server

You can monitor the jobs in both Data Integrator and BW, and look at the trace and error logs in Data Integrator. When Data Integrator Designer is running, the log opens automatically, as it does when you run jobs initiated in Data Integrator.

SAP BW load options

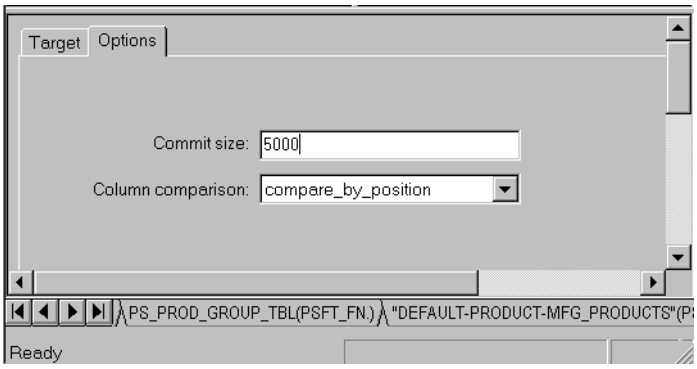
There are two BW load options:

- **Commit size**

  The default is 5000.

- **Column comparison**

  Compare by position or compare by name.



NOTE: You cannot use overflow files when loading to a BW Transfer Structure.

10 Capturing Changed Data in SAP R/3

This chapter describes changed-data capture using SAP R/3 sources in the following sections:

- Changed-data capture for SAP R/3
- Design guidelines for SAP R/3 source-based CDC
- Using SAP R/3 source-based CDC

Changed-data capture for SAP R/3

The following methods can be used in Data Integrator:

- IDoc-based extraction

  This method does incremental extraction of data using the SAP IDoc (Intermediate Document) architecture.

  IDocs capture data when a transaction is being processed. (An IDoc can be generated for each transaction processed.) This can be an effective means for capturing changed data when the underlying tables do not contain date and time stamps or when it is important to capture each changing state between extractions.

  For more information on how to use IDocs, see "Batch and real-time processing using IDocs" on page 9.

- SAP R/3 change log-based extraction

  DBMS change logs keep a history of changes in a single table or file and provide a way to correlate the change log records to the data.

  You can get information about data changes by examining the log.

Capturing only changes

After the initial data warehouse load is complete, changed-data capture can be used to extract only new or modified data and update the data warehouse. This is incremental data that has changed since that last refresh cycle. Data Integrator acts as a mechanism to locate and extract only the incremental data that has changed since the last refresh.

The most important reasons for loading only the changed data are performance and history:

- *Performance* — The less data to extract, transform, and load, the less time the job typically takes.

- *History* — A data warehouse or a data mart has to track the history of changes to data so that the data can be correctly analyzed over time.

  For example, if a customer moves from one sales region to another, simply updating the customer record to reflect the new region negatively affects any analysis by region over time because all the purchases made by the customer before the move are attributed to the new region.

Identifying and loading only the changed data is called *changed-data capture*.

This chapter discusses both general concepts and specific procedures for doing changed-data capture in Data Integrator.

Changed-data capture can be either source-based or target-based:

- *Source-based changed-data capture* extracts only the changed rows from the source.

  This method (also known as *incremental extraction*) is usually preferred for the obvious performance benefit gained by extracting the fewest number of rows.

  Data Integrator provides you with the Date and time field comparison for source-based changed-data capture.

  This technique uses a creation and/or modification timestamp on every row to allow a comparison with the time of the last update.

- *Target-based changed-data capture* extracts all the data from the source, but loads only the changed rows into the target.

  Target-based changed-data capture is useful when you care only about capturing history and aren't concerned about performance.

  Data Integrator offers *table comparison* to support this method.

Design guidelines for SAP R/3 source-based CDC

General source-based change data capture guidelines are as follows.

Use source-based CDC when:

- Date and time fields are available.
- You have a large table, a large percentage of it changes between extracts, and there is an index on the date and time.
- You are not concerned about capturing the results of each transaction between extracts.

Don't use source-based CDC when:

- You have a large table, a large percentage of it changes between extracts, and there is no index on the timestamps.
- You need to capture row deletes.
- You need to capture multiple events occurring on the same row between extracts.

Also note that:

- This method results in additional overhead at extract time.

The remainder of this section describes when you should use each SAP R/3 CDC method:

- IDoc-based extraction
- SAP R/3 change log capturing

IDoc-based extraction

Use IDoc-based extraction when:

- The required tables don't support date and time fields.
- IDocs are already configured and started a given data set.

  This prevents your having to access data twice.

- You must capture multiple transactions between batch runs.
- You have a large table, and a small percentage of it changes between extracts.

Disadvantages of this method:

- It is difficult to tune the IDoc algorithms and mechanisms.
- There can be significant transactional and disk space overhead for generating IDocs.
- IDocs may not support hierarchies.
- SAP R/3-supplied IDocs do not provide many of the columns that are often used for data warehousing.

  Even though this is true, Data Integrator gives you the ability to still use IDoc-based data capture, providing you join IDocs with tables.

SAP R/3 change log capturing

Use SAP R/3 change log-based extraction when:

- You must capture all changes (inserts, deletes, and updates) to master data.
- You cannot use IDocs or table comparison to capture data from tables without timestamps.

Disadvantages of this method:

- Captures only perform updates and deletes, not inserts of data, when used with sales orders.
- The change log is a large cluster table that is inefficient to access.

Using SAP R/3 source-based CDC

This section discusses:

- Capturing IDocs
- Capturing change logs

Capturing IDocs

IDocs were originally designed for EDI (Electronic Data Interchange), but have gradually evolved to be used in SAP's ALE (Application Link Enabling) distributed application architecture.

How IDocs are used

In an SAP R/3 system where IDocs are configured, they are created when a transaction is committed. This can be an effective means for capturing changed data when the underlying tables do not contain date and time stamps.

Overcoming IDoc Limitations using Data Integrator

One limitation is that standard IDocs capture only a subset of the data often required by a data warehouse. IDocs are not recommended for full refresh operations. Since IDocs are typically used for EDI and not data warehousing, standard IDocs do not fully meet requirements for data warehouse source columns.

Although you can create custom IDocs, they are difficult to build and maintain.

Use Data Integrator to extend the IDoc. By joining data from related SAP R/3 tables, you can overcome these limitations.

Using IDocs in Data Integrator

You can use IDoc functionality in Data Integrator in one of two ways:

- If not all the detailed information you need to extract from SAP R/3 is available in IDocs, use the IDoc to capture the object identifiers of the changed data and then use the Data Integrator query transform to join it with other transaction tables to extract the details.
- If all the detailed information you need to extract is available in the IDocs, get all the changed data directly from the IDocs.

Capturing change logs

SAP R/3 keeps all changes to master data in the CDHDR table. For SAP R/3, change information is available only for fields with change document enabled.

The difference between using timestamps and using change logs is small: the timestamp information is maintained in a separate table rather than the source table, and your work flow must extract the timestamp from the change log.

As in the scenario described in "Create and update timestamps" on page 523 of the *Data Integrator Designer Guide*, the change log gives access to both new and updated rows. The following example shows the extraction required for new and updated rows from an SAP R/3 table, KNA1, for customer master data. The change log object class for customer is DEBI, which is stored in field OBJECTCLAS in table CDHDR.

The change indicator is recorded in the field CHANGE_IND. The row extracted from the change table contains the customer number in the field OBJECTID, which the second SELECT command uses to extract the new and updated rows.

You accomplish these extractions in Data Integrator by adding the WHERE clause from the following SQL commands into a query transform in an R/3 data flow:

- Find new rows:

```
SELECT *
  FROM CDHDR
  WHERE OBJECTCLAS = 'DEBI'
  AND UPDATE >= $StartDate
  AND UPDATE <= $EndDate
  AND CHANGE_IND = 'I'

SELECT SINGLE *
  FROM KNA1
  WHERE KUNNR = CDHDR-OBJECTID
```

- Find updated rows:

```
SELECT *
  FROM CDHDR
  WHERE OBJECTCLAS = 'DEBI'
  AND UPDATE >= $StartDate
  AND UPDATE <= $EndDate
  AND CHANGE_IND = 'U'

SELECT SINGLE *
  FROM KNA1
  WHERE KUNNR = CDHDR-OBJECTID
```

11 Reference information

This chapter contains SAP R/3-specific reference information about SAP R/3 ABAP interface options, objects, data types, and functions. Specific sections include:

- SAP R/3 ABAP Interface options
- Objects
- Data Integrator objects
- Data types
- Functions
- Data Integrator Designer functions

Refer to the *Data Integrator Reference Guide* for general Data Integrator reference information.

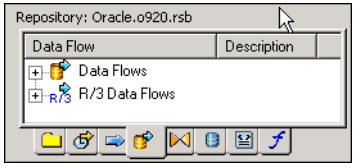
For conceptual and procedural information about Data Integrator objects, data types and expressions, and functions see the *Data Integrator Designer Guide*.

SAP R/3 ABAP Interface options

If you install Data Integrator Designer using a license that includes the SAP R/3 ABAP interface, you will find several additions to the Designer's graphical user interface.

Object Library and tool palette objects

For example, after you open Designer, you will find the **R/3 data flows** object, in addition to **Data Flow** object, in the object library.



Also, in the object library, on the **Transforms** tab, right-click and select **New ABAP transform** to create a custom ABAP transform. On the **Format**s tab, under **Flat Files**, a built-in format called the **Transport_Format** appears.

R/3 data flows can be placed into batch jobs. R/3 data flows allow you to use SAP R/3 as a database source in batch jobs. R/3 data flow objects generate ABAP code. A unique object is sometimes needed instead of a target when you use an R/3 data flow. It is called a data transport.

R/3 data flow and data transport objects also appear active in the tool palette when your installation of Data Integrator Designer includes the SAP R/3 ABAP interface:

Icon	Object Name	Description (class)	Can be used in:
	R/3 data flow	Creates a new R/3 data flow. (reusable)	batch jobs
	Data transport	Specifies a staging file for data extracted from SAP R/3. (single-use)	R/3 data flows

See "Objects" on page 220 for more information about SAP R/3 objects.

R/3 data flow commands

The following commands appear in the **Debug** menu when your installation of Data Integrator Designer includes the SAP R/3 ABAP interface and when the object open in the workspace is an R/3 data flow:

- **Validate ABAP**

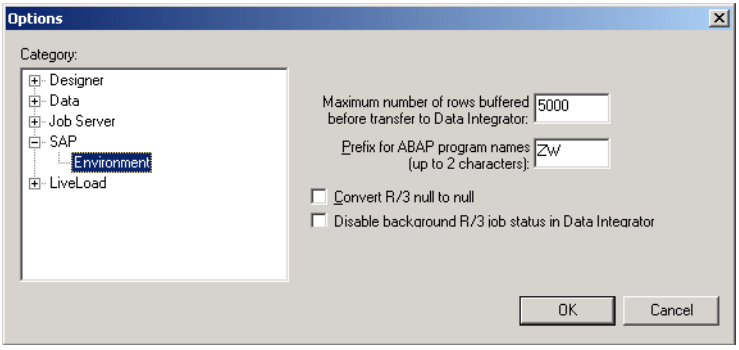
  Validate the ABAP code Data Integrator produces for the current SAP R/3 object

- **Generate ABAP**

  Display the ABAP generated for the current R/3 data flow

SAP R/3 environment options

The following appear in Data Integrator Designer, under the Tools menu, when your installation of Data Integrator Designer includes the SAP R/3 ABAP interface.



Buffer Size

The size of the SAP R/3 buffer that is used to hold data prior to writing it into the data transport file. This option is provided to "throttle" the amount of memory used by generated ABAP programs. The value indicates the number of rows held in memory prior to flushing them to the transport file. When the number of rows reaches the specified buffer size, the ABAP program writes the rows held in memory to the file and then reuses the memory for subsequent data.

Without this mechanism, an extraction of 10 million rows from a single table would potentially cause the SAP R/3 application server to run out of memory, since the ABAP program's buffer would continue to grow attempting to hold all 10 million rows. The default value for this option is 5000 rows. Set this value once to be used for all SAP R/3 connections initiated by Data Integrator.

Prefix for ABAP program names

You can define this parameter to separate Data Integrator-generated ABAP program names from other ABAP program names. Since ABAP program names must begin with a "Y" or a "Z", this option provides a default that begins with ZW.

Convert R/3 null to null

Converts NULL values from SAP R/3 sources in R/3 data flows to database-specific NULL. "Working with null values from SAP R/3" on page 265 for more information.

Disable background R/3 job status in Data Integrator log

Disables the trace log for ABAP jobs (batch jobs that contain R/3 data flows).

Objects

The following table lists the names and descriptions of SAP-specific objects available in Data Integrator.

Object	Class	Description
Custom ABAP transform	Reusable	Inserts custom ABAP logic blocks into a generated ABAP program for an R/3 data flow.
Data transport	Single-use	Identifies a file to hold a data set extracted by an R/3 data flow.
Hierarchy	Reusable	Browse, search, and import hierarchy metadata
IDoc file	Reusable	An IDoc is an intermediate document that SAP R/3 produces to move data between servers. In Data Integrator, an IDoc file can be used in a real-time or batch job as a source or a target.
IDoc message	Reusable	An IDoc is an intermediate document that SAP R/3 produces to move data between servers. In Data Integrator, an IDoc message can be used in a real-time job as a source or a target and in a batch job as a target.
R/3 data flow	Reusable	Extracts data from SAP R/3 sources.
transport_format	Reusable	The Transport_Format is an Data Integrator predefined file format object for reading flat files in SAP R/3. It is automatically used to define data transport objects added to R/3 data flows. For specific information about this format, see "The Transport_Format" on page 68. For general information about file formats see "File format" on page 246.

Custom ABAP transform

CLASS

Reusable

ACCESS

- In the object library, click the **Transforms** tab then right-click and select **New ABAP Transform**. After renaming your custom ABAP Transform, drag it into an R/3 data flow. From the **Tools** menu, select **Variables** to define parameters.
- You can reuse an existing transform by dragging and dropping the object from the **Transforms** tab.

DESCRIPTION

A custom ABAP transform executes a piece of ABAP logic to produce a set of rows to be used by subsequent transforms in the same R/3 data flow.

A custom ABAP transform does not take any input sets and produces one and only one output data set. It allows for any number of input parameters, but no output parameters.

A custom ABAP transform has the following properties:

Properties	Description
Name	Specifies the name for the custom ABAP program that appears in the object library and in diagrams. This name cannot contain spaces.
Description	Allows you to document the transform's operation.

A custom ABAP transform has the following options:

Option	Description
ABAP language file name	Indicates the name of the file containing the ABAP program to execute.
Join rank	Indicates the rank of the output data set relative to other data sets included in a join. For more information, see "Query transform overview" on page 173 of the *Data Integrator Designer Guide*.
Output schema	Lists the columns in the schema produced by this ABAP program. To add a column to the schema, right-click and choose **New Column**.
Column type	Indicates the data type of the selected column. This option appears after you create a column, click outside the column name, and then select the column.

Data transport

CLASS

Single-use

ACCESS

- To insert, open an R/3 data flow diagram in the workspace, select the data transport icon in the tool palette, and click on the workspace.
- To view options, click the name of the data transport. This opens the data transport editor.

DESCRIPTION

Data transports are targets in R/3 data flows. A data transport takes the data SAP R/3 extracts during an R/3 data flow and stores it in a file defined by the Transport_Format file format.

Use a data transport when you want to use the extracted data in a data flow outside the R/3 data flow. In this case, you must use the R/3 data flow as a source in the parent data flow.

Do not use a data transport when you want the extracted data kept in a file on the SAP R/3 server or in an SAP R/3 table. In that case, use a file or SAP R/3 table as the target in the R/3 data flow.

The data transport editor shows the schema produced by the R/3 data flow and the options that define the data transport. These options specify the location of the file and indicate whether to append to or overwrite the data in the file.

You can set the delimiter SAP R/3 uses to write data to the file. The default delimiter is a tab character. Change the delimiter in the Transport_Format file format definition. Valid ascii character delimiters range from /0 to /254.

The following table lists the options in the data transport editor:

Data transport editor option	Description
File Name	The name of the saved staging file.
Working directory on SAP	The location of the saved staging file. This is the **Working directory on SAP server** specified for the datastore associated with the current R/3 data flow.
Append file	Adds the data set to the existing contents of the file.
Replace file	Clears the file before writing the data set to it.

Data transports have one attribute:

Attribute	Description
Name	The name of the object. This name appears on the object in the diagram.

Hierarchy

CLASS

Reusable

ACCESS

In the object library, click the **Datastores** tab and expand a datastore listing (click the plus sign next to the datastore name).

DESCRIPTION

Hierarchy objects allow you to browse, select, and import hierarchical information from a source database. Hierarchy objects can import the following hierarchical information:

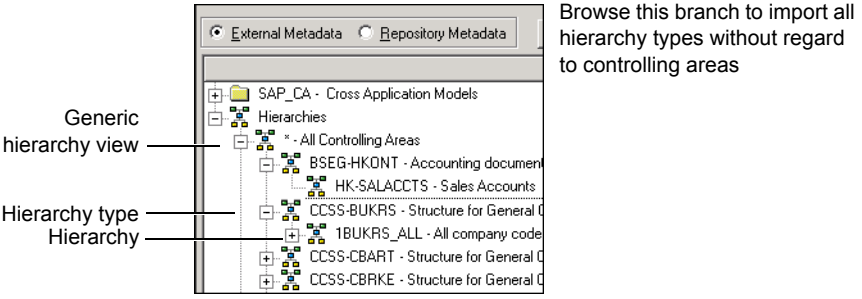
Source application	Hierarchies extracted
SAP R/3	Data Integrator represents SAP R/3 sets as hierarchies. In SAP R/3, sets are group of objects such as cost element groups or cost center groups. The sets allow you to see an individual object or all similar objects in a related area. Hierarchy objects can import set information from SAP R/3 basic or single-dimension sets. In particular, you can import cost element and cost center hierarchies.

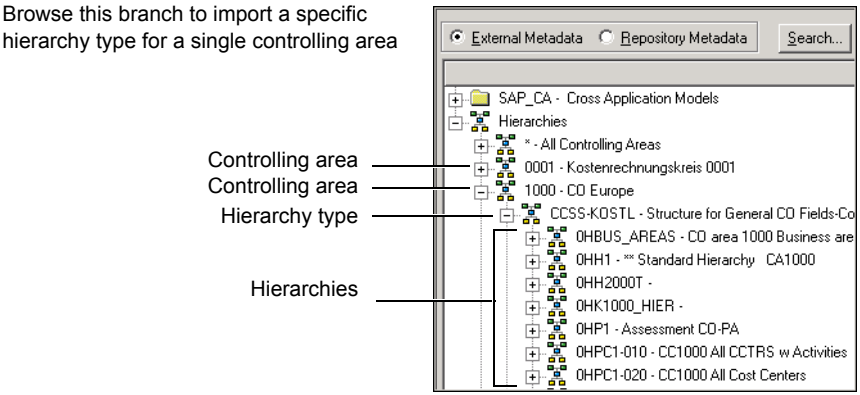
To extract hierarchical information, select a hierarchy type, import its metadata into your repository, then drag the hierarchy icon from the object library into a data flow. Importing metadata is described in "Importing metadata through a custom datastore" on page 90 of the *Data Integrator Designer Guide*.

SAP R/3 hierarchies

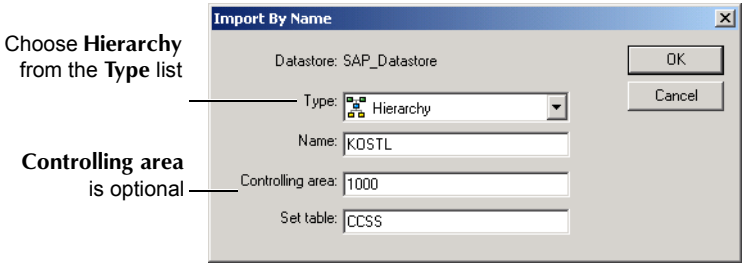
EXTERNAL VIEW

Hierarchy types are listed in the external browsing view for an SAP R/3 datastore. From this view, you can choose to import a hierarchy type from a specific controlling area, or you can import a hierarchy type regardless of its association with a controlling area.



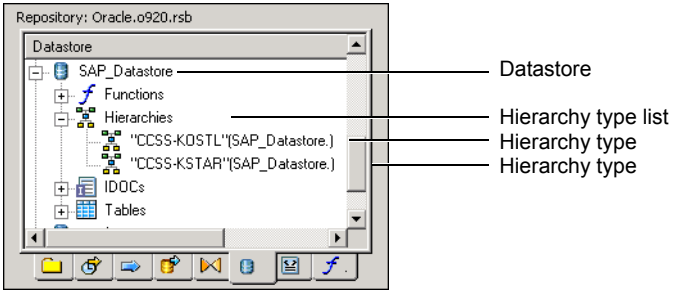


You can also import a hierarchy by specifying the hierarchy type by name. Select the datastore in the object library, right-click and choose **Import By Name**. The following window opens.



IMPORTED HIERARCHY TYPE

After you import the hierarchy type, it appears in the object library nested under the datastore name.



The hierarchy object has the following properties:

Property	Description
Name	The name of the hierarchy object. The name is constructed as follows: `set_table-hierarchy_group(datastore)`
Description	The description of the hierarchy type as included in the SAP R/3 table.
Controlling_Area	The controlling area from which you imported the hierarchy type. You could specify the controlling area by selecting the hierarchy type in the controlling area view or by entering the controlling area name in the Import by Name window when you imported the hierarchy type. If you did not specify a controlling area, the value in Controlling_Area is an asterisk (*). If you import a chart of account hierarchy type, the controlling area will be inferred. The Controlling_Area value will never be NULL.
Set_Table	The table from which the hierarchy data is extracted.
Chart_Of_Account	The chart of account from which you imported this hierarchy type. This value is NULL for many hierarchy types associated with controlling areas but not chart of accounts.
Hierarchy_Group	The hierarchy type imported.
Check_Table	The location of the range of values for the hierarchy type leaf nodes.
Domain_Name	The data type and range definitions of the leaf node values.

Selecting the hierarchy in the object library and choosing **Open** displays the object properties and the source schema that results from the hierarchy extraction. Each row of the hierarchy object output describes a single relationship in the hierarchy.

Column name	Description
Parent_id	The parent node in the relationship described by this row.
Parent_desc	A text description of the parent node.
Child_id	The child node in the relationship described by this row.
Child_desc	A text description of the child node.
Value_from	The start of the range of values associated with a leaf node. The range is found in the check table, which is recorded in the hierarchy object attributes. NULL if the row does not describes a leaf node.
Value_to	The end of the range of values associated with a leaf node. The range is found in the check table, which is recorded in the hierarchy object attributes. NULL if the row does not describes a leaf node.
Leaf_flag	A value of 1 indicates that the child node is a leaf node. A value of 0 indicates that the child node is not a leaf node.
Root_flag	A value of 1 indicates that the parent node is the root node. A value of 0 indicates that the parent node is not the root node.
Tree_level	The level of the child node in the hierarchy relative to the root node.
Cont_area	The controlling area associated with the hierarchy type. If this hierarchy type was selected without specifying a controlling area, the value of Cont_area is an asterisk (*).
Chart_of_acct	The chart of account associated with the hierarchy type.
Set_table	The table from which the hierarchy data is extracted.

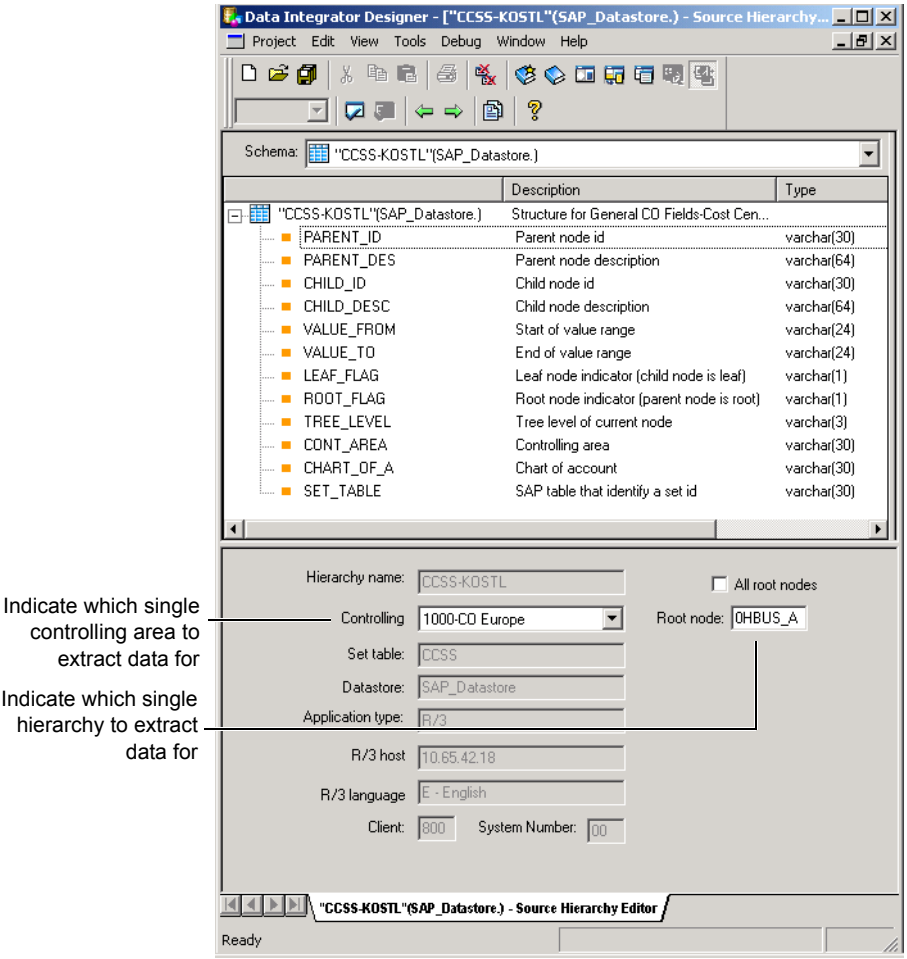
NOTE: For every root node in the hierarchy, the hierarchy object produces a row with NULL as the root node. Depending on the final use of the hierarchy data, you may want to filter the NULL parent rows out of the hierarchy data.

HIERARCHY INSTANCE

You can drag a hierarchy from the object library into an R/3 data flow definition. The hierarchy object appears in the data flow definition as follows:

The hierarchy editor displays the datastore information for the hierarchy and provides options for the instance of the hierarchy object:

Option	Description
All root nodes or Root node	Indicates that all hierarchies (root nodes) will be extracted from the specified hierarchy type. Turn off this option to specify a single hierarchy to extract. Look in the external view of the datastore to determine specific hierarchy names.
All controlling areas or Controlling area	Indicates that all hierarchy types for all controlling areas in which the groups appear will be extracted. Turn off this option to specify a single hierarchy type for a single controlling area to extract. **Note:** For best performance, make sure to specify a single controlling area.



Indicate which single controlling area to extract data for

Indicate which single hierarchy to extract data for

IDoc file

CLASS

Reusable

ACCESS

In the object library, click the **Datastores** tab, click an SAP R/3 datastore, and click the list of available IDocs.

DESCRIPTION

IDoc file sources are handled the same way in data flows of real-time and batch jobs.

See "Source" on page 250 for information about IDoc source message and file attributes.

See "Target" on page 259 for information about IDoc target message and file attributes.

A real-time job that contains an IDoc file source processes its files one at a time. A real-time job that contains an IDoc message source can process messages one at a time or in parallel. See "IDoc message" on page 234 for more information.

MULTIPLE FILE READ IN BATCH JOBS

One or more IDoc files can be saved into one file or into many individual files to be processed at a later time.

You can configure Data Integrator to read files in groups by listing the names of these IDoc files in the **IDoc File/s** box of the source editor in the Designer.

The following list provides three examples:

- If you specify the IDoc file, D://temp/`IDOC_R3*.txt`, Data Integrator will process data from all files in the D://temp directory with names beginning with `IDOC_R3` and the extension `.txt`.
- If you specify the IDoc file, D://temp/`IDOC_R3?.txt`, Data Integrator will process data from all files in the D://temp directory with:
  - names beginning with `IDOC_R3`
  - any character before the extension
  - the extension `.txt`.
- If you specify the IDoc file, `D://temp/IDOC_01.txt`, D://temp/`IDOC_02.txt`, D://temp/`IDOC_03.txt`, Data Integrator will process data from these three IDoc files.

The Multiple File Read feature is supported in the following IDoc rec.type releases:

- SAP Release 3.0/3.1
- SAP Release 4.x

IDoc file sources are validated against the IDoc Name, its related hierarchy, and the SAP Release.

VARIABLE FILE NAMES IN BATCH JOBS

You can also use variable file names in the **IDoc File/s** box. Define a global or local variable for an initialization script of a job. For example,
`$SOPGEN01_filepath = "D:\temp\IDOC_R3*.txt".`

Then enter `$filepath` in the **IDoc File/s** box. In this way you can avoid drilling down to each IDoc object to configure the location of source files.

IDoc message

CLASS

Reusable

ACCESS

In the object library, go to the **Datastores** tab, open the SAP R/3 datastore, and open the list of available IDocs.

DESCRIPTION

An IDoc is an intermediate document that SAP R/3 produces to communicate transaction data between R/3 application servers. SAP R/3 offers numerous IDocs that describe different transaction types. The data in an IDoc supports a particular transaction. An IDoc can contain data from multiple tables and can contain hierarchical data.

You can use an IDoc message as a source or target in a real-time job. You can use an IDoc message as a target in a batch job. You cannot use an IDoc in an R/3 data flow.

You can use the multiple file read and variable file name features with IDoc message sources in real-time jobs. However, these features can only be used when the job is run in test mode. "IDoc file" on page 232 for more information.

IDoc message sources must be configured in the Data Integrator Administrator as well as in Data Integrator Designer. See "Creating real-time jobs using IDoc sources" on page 135 for a high level overview of this process. Then refer to the *Data Integrator Administrator Guide* for more information about how services and service providers for real-time jobs are configured.

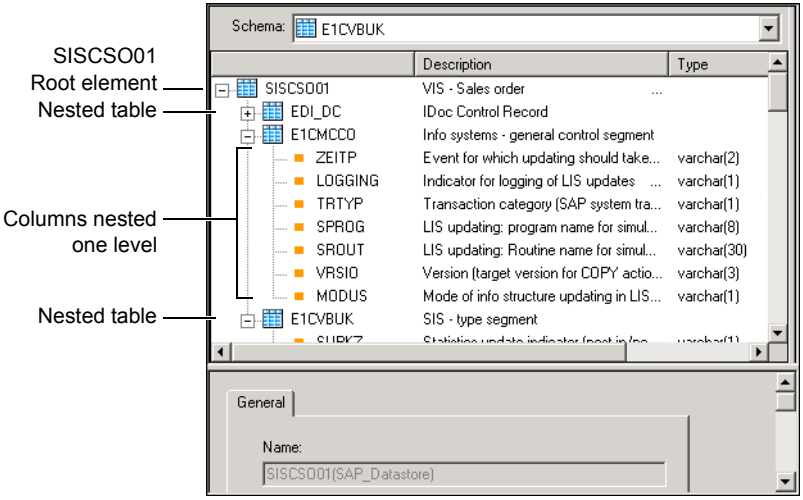
Real-time jobs that contain an IDoc message source can be processed one at a time or in parallel. To enable parallel processing, select **Administrator** > **Real-time** > **Client Interface** in the Data Integrator Administrator.

The **Parallel Processing** option allows you to increase the number of IDoc source messages processed per minute for the IDoc type specified. This option is disabled by default.

The parallel processing option allows the Access Server to send an IDoc to a service queue (where it waits for a service provider) and continue with the next IDoc without waiting for reply. The maximum number of outstanding IDoc requests in the queue is the number of IDocs received or 4, whichever is smaller.

NOTE: Where a strict IDoc processing sequence is required, do not use the Parallel Processing option.

To use IDocs, you must first import the metadata from SAP R/3 to your repository. Data Integrator imports the schema for the IDoc, maintaining hierarchical relationships among the data fields. You can display the IDoc schema by double-clicking the IDoc name in the object library.



When IDoc metadata is imported into Data Integrator or when IDocs are sent and received, the control data appears as the first segment for all IDoc objects. The control data is called EDI_DC on SAP R/3 version 3.x and EDI_DC40 on SAP R/3 Version 4.x. The control data contains IDoc administrative information, such as IDoc number and sender or receiver information.

See "To import IDoc metadata using the search option" on page 52 for more information about importing IDocs.

See "Source" on page 250 for a description of IDoc source editor options.

See "Target" on page 259 for a description of IDoc target editor options.

R/3 data flow

CLASS

Reusable

ACCESS

- To insert an existing R/3 data flow, go to the object library **Data Flows** tab, open the list of R/3 data flows, select the R/3 data flow you want to add, and drag it into the workspace.
- To insert a new R/3 data flow, select the R/3 data flow icon in the tool palette, and click in the workspace.

DESCRIPTION

An R/3 data flow extracts data from SAP R/3 sources. You can insert an R/3 data flow into any other data flow (R/3, batch, or real-time). An R/3 data flow produces a data set you can use as a source in other data flows.

You can define parameters to pass values into the R/3 data flow. If you cannot express the required extraction logic using the Data Integrator interface, you can embed ABAP logic into an R/3 data flow by creating a new ABAP transform. See "Creating custom ABAP transforms" on page 100.

When Data Integrator executes R/3 data flows, it translates extraction requirements into ABAP programs and passes them to SAP R/3 to execute. An R/3 data flow can store the extracted data set in an SAP R/3 table or in text file saved on the SAP R/3 server. To use the data outside the R/3 data flow, save the data in a file and use a data transport object to pass the data to the parent data flow. The data transport object defines a file format and indicates how data is transferred from the SAP R/3 server to Data Integrator.

When you use the R/3 data flow as a source in a join (either with other R/3 data flows, SAP R/3 sources, or with non-SAP R/3 sources), you can assign a join rank to the R/3 source. When joining sources, Data Integrator reads sources with higher join ranks before reading those with lower join ranks. Set join rank in the R/3 data flow properties when you first define the R/3 data flow.

NOTE: Join rank must be a positive integer. Zero is the default value. With a zero join rank, Data Integrator determines join order.

An R/3 data flow can contain the following objects:

- Sources (SAP R/3-compatible files or SAP R/3 tables)
- Targets (SAP R/3-compatible files or SAP R/3 tables)
- Transforms (Custom ABAP or Row_Generation)
- Queries
- R/3 data flows

R/3 data flows have several options.

Option	Description
Datastore	Specifies the datastore corresponding to the SAP R/3 tables or files this R/3 data flow accesses.
Generated ABAP file name	Specifies the file name containing ABAP program code that Data Integrator generates from this SAP R/3 data flow. This file is written to the directory specified in the datastore definition.
ABAP program name in R/3	Specifies the name Data Integrator uses for the ABAP program name that runs in SAP R/3. It must begin with Y or Z and be 8 characters or fewer.
Job name in R/3	Specifies the name used for the job that runs in SAP R/3. It defaults to the name of the data flow.
ABAP row limit	Limits the number of rows that will be read by this R/3 data flow. Use this option for testing purposes to limit the time that the parent batch job will run. See "ABAP row limit" on page 82. This option can also be used to limit the number of rows read using the Designer's Data Scan feature. R/3 data flows return one file of data from scans. Individual objects inside an R/3 data flow cannot be read using Data Scan.
Join rank	Determines the order that Data Integrator reads this data set when joining it with another source. When joining sources, Data Integrator reads sources with higher ranks before reading sources with lower ranks.
Cache	Indicates whether Data Integrator should cache the data set produced by this R/3 data flow for use in other operations. If the result is part of a another R/3 data flow, the cache option is ignored.
Always execute during recovery	Indicates that Data Integrator should re-execute this R/3 data flow during an automatic recovery when it is part of a re-executed work flow that is a recovery unit. In this case, Data Integrator re-executes the R/3 data flow even if it completed successfully in an earlier run.

R/3 data flows have the following built-in attributes:

Attribute	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object.
Description	Your description of the R/3 data flow.

If you delete an R/3 data flow from the object library, calls to the object are replaced with an icon indicating that the calls are no longer valid, and are deleted from the project area.

Data Integrator objects

The following table lists names and descriptions of general reference objects that are associated with SAP R/3 supplemental reference information. While the supplemental information is listed in this document, see Chapter 2, "Data Integrator Objects," in the *Data Integrator Reference Guide* for basic information about these objects.

Object	Class	Description
Datastore	Single-use	Describes the connection information needed for Data Integrator to access a database or an SAP R/3 system. Cannot be dropped.
File format	Reusable	Indicates how data is arranged in a source or target file.
Job	Reusable	A job is a set of objects that you can execute together.
Source	Single-use	An object from which Data Integrator reads data in a data flow, or an R/3 data flow.
Table	Reusable	Indicates an external DBMS table for which metadata has been imported into Data Integrator, or the target table into which data is or has been placed. A table is associated with its datastore; it does not exist independently of a datastore connection. A table retrieves or stores data based on the schema of the table definition from which it was created.
Target	Single-use	An object in which Data Integrator loads extracted and transformed data in a data flow.

Datastore

CLASS

Reusable

ACCESS

In the object library, click the **Datastores** tab.

DESCRIPTION

A datastore provides a connection to a database. Data Integrator uses the connection defined by the datastore to import descriptions of the database and its tables (metadata). When you specify tables, hierarchies, or other objects as sources or targets in a data flow, Data Integrator uses the datastore to determine how to read data from or load data to those tables.

In addition, some transforms and functions require a datastore name to qualify the tables they access.

For an SAP R/3 datastore, specifying an IP address rather than a name in the host name field usually results in a faster connection.

If you delete a datastore from the object library, you must remove references to the datastore from the following locations:

- Source or target tables using this datastore in your diagrams
- The lookup and key_generation functions and Key_Generation, History_Preserving, Table_Comparison, and SQL transform references
- R/3 data flows for R/3 connections

Datastores have the following attribute:

Attribute	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object.

SAP R/3 DATASTORES

Set the following options to define an SAP R/3 datastore:

SAP R/3 datastore options

Tab	Option	Description
Connection		
	Name	Specify the datastore name. This name is used to reference the datastore from other object definitions.
	Application type	Choose R/3 to display the options for R/3 datastores. This option cannot be edited after the datastore is created.
	User name	Enter the name of the account through which Data Integrator accesses the SAP R/3 application server.
	Password	Enter the password for the user name.
	LiveLoad	Configure a LiveLoad-controlled datastore. For more information, see "Datastore object" on page 18 of the *Data Integrator LiveLoad User's Guide*.
R/3 Properties		
	ABAP execution option	Select the job execution strategy. There are two options: • generate_and_execute — ABAP resides on the Data Integrator server and is submitted to R/3 using the SAP R/3 RFC_ABAP_INSTALL_AND_RUN function. • execute_preloaded — ABAP resides on the SAP R/3 application server and is submitted to R/3 using Data Integrator RFC function modules. If the Data Integrator job changes between scheduled executions, choose generate_and_execute. If the job does not change, choose execute_preloaded. Your choice affects the required authorizations. See "Authorizations for Data Integrator" on page 37.
	R/3 application server	Name of the remote SAP R/3 application host that Data Integrator connects to (is using as an R/3 datastore).
	R/3 language	Indicate the login language from the drop-down list.

SAP R/3 datastore options (Continued)

Tab	Option	Description
	R/3 client	Indicate the SAP R/3 client.
	R/3 system number	Indicate the SAP R/3 system number.
	Execute in background (batch)	Specify that the generated ABAP programs created by R/3 data flows defined with this datastore will execute in batch mode on the SAP R/3 application server. Batch mode operation is slower than the normal console mode; however, choose batch mode if the application is too long to run during the console mode time window.
	Target host	Specify the target host if you choose to execute ABAP programs in background.
	Job class	Specify the job class if you choose to execute ABAP programs in background.
	Data transport method	Define how to get data from the SAP R/3 server to the Data Integrator server. There are four options: • Custom Transfer — Use a third-party program • ftp — Use FTP • shared_directory — Use NFS (shares one network drive or directory) • direct_download — Use SAP R/3 WS_DOWNLOAD function These methods are described in Chapter 8, "Executing Batch Jobs that Contain R/3 Data Flows". When you select a data transport method, the appropriate options for that method appear in the tab.
	Working directory on SAP server	Indicate where ABAP intermediate file, R/3 source, and R/3 target are written. All the files used by the R/3 data flow should be placed in this directory. This directory also stores the transport file used by the FTP, shared-directory, and custom transfer data transport methods.
	Client download (Local) directory	Indicate the Data Integrator working or local directory. This directory is local to the job server used by the job.
	Data Integrator path to the shared directory	(Shared directory) Indicate the path from the Data Integrator server to the SAP R/3 server's working directory.
	FTP relative path to the SAP working directory	(FTP) Indicate the path from the FTP root directory to the SAP R/3 server's working directory. When you select FTP, this directory is required.
	FTP host name	FTP host name

SAP R/3 datastore options (Continued)

Tab	Option	Description
	FTP user name	FTP user name
	FTP password	FTP password
	Custom Transfer Program	The name of the third-party file transfer program you want to use to transfer files from the SAP working directory to the local directory. For example: "`MyTransferProgram.exe`"
	Custom Transfer User name	(optional) Log in ID for the SAP R/3 server to which the custom transfer program connects.
	Custom Transfer Password	(optional) Password for the SAP R/3 server to which the custom transfer program connects. Passwords entered into this option are encrypted by Data Integrator.
	Custom Transfer Arguments	(optional) Specify arguments for your custom transfer program. Arguments can add security or compression mechanisms to your program or include Data Integrator system variables. See "Data Integrator system variables for transferring a file from SAP R/3" on page 172 for more information.
	Generated ABAP directory	Indicate the directory into which generated ABAP files are written. Can be the same directory as the local/direct download directory.
	R/3 security profile	Specify the security profile you want to use in the generated ABAP program. The user of the Data Integrator R/3 datastore must have the required profile.

File format

CLASS

Reusable

ACCESS

In the object library, click the **Formats** tab.

DESCRIPTION

A file format describes the structure of an ASCII file. A file format can be specific to a single file, or it can be a generic description used for many data files. Construct file formats from the properties found in the file format editor.

See "File format" on page 62 of the *Data Integrator Reference Guide* for general information about file formats. The following options are for the R/3 file format type only.

File format options

Property	Possible values	Description
General		
Type	R/3	The format of the data in the text file. Available properties change based on the selected file format type. For information about delimited and fixed width file format types, see "File format" on page 62 of the *Data Integrator Reference Guide*.
Name	Any alphanumeric character and underscores (_)	A descriptive name for the file format. This name appears in the object library.
Data File (s)		
Location	Local, Job Server	(Optional) The computer on which the data file(s) are stored.

File format options (Continued)

Property	Possible values	Description
File/s	File name, file name including full path name, or blank	In new and edit modes, specify an existing file on which you base the file format description. Data from this file appears in the Column Attributes area. In these modes, you can leave this property blank. In source and target modes, specify the location of the actual file for this source or target. In these modes, you cannot leave this property blank. For added flexibility, you can enter: • A variable that is set to a particular file with full path name. • A list of files, separated by commas, or a file name containing a wild card. In this case, Data Integrator reads all these files as a single source.
Delimiters		
Column	Tab, Semicolon, Comma, Space	Specify the character that indicates the end of one column and the beginning of the next. Alternatively, enter an ASCII character. Valid ASCII characters range from /0 to /254.

The SAP R/3 ABAP interface includes a built-in file format called the Transport_Format which can be seen on the **Formats** tab, under **Flat Files**, in the object library. See "SAP R/3 and File Formats" on page 67 for more information.

Job

CLASS

Reusable

ACCESS

- In the object library, click the **Jobs** tab.
- Right-click a job in the project area.

DESCRIPTION

A job is a set of objects that you can execute together. For Data Integrator to execute the steps of any object, the object must be part of a job. A single job must include all the steps you want executed.

See "Log" on page 75 of the *Data Integrator Reference Guide* for general information about jobs. The following are the additional references to SAP R/3.

- Data Integrator retrieves results from the following types of steps:
  - R/3 data flows
  - R/3 data transport files

- Jobs have properties that determine what information Data Integrator collects and logs when running the job. Use Trace properties to select the information that Data Integrator monitors and writes to the trace log file during a job. You can turn several R/3 related traces on and off.

Trace	Description
ABAP Query	Writes a message when an ABAP program is submitted and when the ABAP program completes. Also writes the ABAP job status every 15 seconds.
RFC Function	Writes messages when Data Integrator calls an RFC function or BAPI. Specifically, this trace writes messages when: • An RFC function or BAPI starts • Processing for an individual record starts • Processing for an individual record finishes • New connections to SAP R/3 are established during real-time jobs • An RFC function or BAPI finishes
SAP Table Reader	Writes messages when Data Integrator reads from an SAP R/3 table source. Specifically, this trace writes messages when: • Data Integrator establishes a connection with SAP R/3 • Data Integrator begins reading from the SAP R/3 table • The RFC call to retrieve table content starts • The RFC call to retrieve table content finishes • Data Integrator finishes reading from the SAP R/3 table

Source


Table


SAP R/3 table in batch job


File

SAP R/3 data flow

XML message

XML file


IDoc message


IDoc file

CLASS

Single-use

ACCESS

- To insert an SAP R/3 table as a source, open the object library, go to the **Datastores** tab, select the table, drag it into a batch or R/3 data flow in the workspace, and select **Make Source**.
- To insert a file as a source, open the object library, go to the **Formats** tab, select a flat file format for the file, drag it into the workspace, and select **Make Source**. Use the file format editor to specify the file name.
- To insert an R/3 data flow as a source, open the object library, go to the **Data flows** tab, select the R/3 data flow, drag it into a batch or R/3 data flow in the workspace.
- To insert an XML message or file as a source, open the object library, go to the **Formats** tab, select the DTD or XML Schema format for the XML file, drag it into the workspace, and select **Make XML message source** or **Make XML file source**. Use the source editor to specify the test file name for the message or the source file name for the file.
- To view options of tables or files, click the name of the table or file in the workspace or in the project area. This opens the table or file editor.
- To view options of R/3 data flows, right click the R/3 data flow in the workspace or project area, select **Properties**, and go to the **Options** tab.

- To insert an IDoc message or file as a source, open the object library, click the **Datastores** tab, click an SAP R/3 datastore, click the IDoc category, select an IDoc, drag it into a data flow, and select **Make IDoc message source** or **Make IDoc file source**. Use the source editor to specify object attributes.

  NOTE: A real-time job source can be one and only one XML or IDoc message. If another message source is already in the data flow definition, Data Integrator will not allow you to place the IDoc in the workspace.

DESCRIPTION

A source is an object from which Data Integrator reads data.

In a batch job, a source can be a table, an SAP R/3 table, a previously defined template table, a file, an XML file, an IDoc file or an R/3 data flow.

In a real-time job, a source can be a table, an SAP R/3 table, a previously defined template table, a file, an IDoc file, an XML file, an IDoc message, or an XML message. Each real-time job must have exactly one real-time data source—either an IDoc message or an XML message.

In an R/3 data flow, a source can be another R/3 data flow, an SAP R/3 table, or a file located on the SAP R/3 server. Files must be in a file format that SAP R/3 can read. Data Integrator provides a predefined flat file format, called Transport_Format, which SAP R/3 can read.

You can select an SAP R/3 table as a source in any data flow. However, when you use an SAP R/3 table as a source in a data flow, Data Integrator extracts data using the Data Integrator-provided RFC (remote function call) Z_AW_RFC_READ_TABLE. The function call is limited to extracting 2048 bytes per row. If this function has not been loaded to the SAP R/3 server, Data Integrator extracts data using the SAP R/3-supplied function, RFC_READ_TABLE. This function call limits extracted data to 512 bytes per row.

When you use an SAP R/3 table as a source in an R/3 data flow, Data Integrator extracts data using ABAP. For large amounts of data (larger than a few rows), performance is generally better when you extract data from SAP R/3 using an R/3 data flow.

IDocs are used as message sources in real-time jobs and as file sources in batch or real-time jobs. Data flows in real-time jobs that use IDocs as message sources can be processed one at a time or in parallel. IDoc message sources are generated directly from SAP R/3. See Chapter 7, "SAP R/3 and Real-Time Jobs".

Also, an IDoc file source in a data flow of a batch job can be configured to read files in groups and to use variable file names. See Chapter 6, "IDocs and Batch Jobs".

NOTE: IDocs cannot be used as source or target objects in R/3 data flows.

A table source has options in addition to the datastore information:

Source table options

Option	Description
Join rank	Indicates the rank of the table relative to other tables in the data flow when creating a join. Data Integrator joins tables with higher join ranks before joining tables with lower join ranks. For more information, see "Join ordering" on page 24 of the *Data Integrator Performance Optimization Guide*. **Note:** The join rank must be a positive integer. Zero is the default value. When the join rank is zero, DATA INTEGRATOR determines the join order.
Cache	Indicates whether Data Integrator should read the required data from the table and load it into memory. Because an inner table of a join must be read for each row of an outer table, you may want to cache a table when it is used as an inner table in a join and when the data retrieved will fit into available memory. For non-SAP R/3 tables, there are two options: • **Yes**: The table is always cached unless it is the outer-most table in a join. • **No**: The table is never cached. For SAP R/3 tables in an R/3 data flow, there are three options: • **Yes**: The table is always cached unless it is the outer-most table in a join. • **No**: The table is never cached. • **Automatic**: DATA INTEGRATOR determines whether to cache the table or not. The default option is **No**. When two SAP R/3 tables are involved in an inner join and you cache both tables, DATA INTEGRATOR improves performance by generating ABAP using the array operations SELECT FOR ALL ENTRIES and IN_MEMORY MERGE JOIN. These operations are exclusive to inner joins, specifically to-way inner joins. To optimize join performance the outer and inner tables are sorted then merged. To enable these operations, set the cache option for both tables to **Yes**. The result is that both inner and outer tables are read into internal tables prior to the join. Use **Package size** on the outer loop of the join to limit the size of internal tables.

Source table options (Continued)

Option	Description
Package size	Indicates the maximum number of rows loaded into memory. Limiting rows reduces memory consumption. Package Size is an extension of the SELECT_INTO_TABLE operation, which uses one SELECT statement to retrieve blocks of rows to create an internal table. Use this option when caching a table that might consume excessive memory. DATA INTEGRATOR does not limit rows when you enter 0, a negative number, or when you set **Cache** to **No**.

NOTE: Data Integrator automatically uses the SELECT_INTO_TABLE operation when you cache a database table and the content of the cache populates an internal table, or when you cache one or more LOOKUP functions against one or more tables and the contents of the LOOKUP cache populates an internal table.

A file source also has the **Join Rank** and **Cache** options in addition to the file format information. Data Integrator interprets **Join Rank** and **Cache** the same in a file as a table.

An XML message source has these options in addition to read-only XML Schema or DTD format information:

Option	Description
XML test file	The location relative to the Job Server of an XML-formatted file to use as the message source when you execute the job in test mode.
Enable validation	A check box to turn on the comparison of the incoming message to the stored XML Schema or DTD format. When this option is selected, the real-time job throws an exception if the incoming message is not valid. When you are developing a real-time job, this validation helps you to make sure sample data is both valid and well-formed. If you select this option in production, make sure to include appropriate error handling either in the Data Integrator job or the client application to process an error caused if a data flow in the real-time job receives data that does not validate against the imported format.
implied: Join rank	The XML message source is always the outer table in a join. You cannot change its join rank. This option is implied and does not appear in the editor.

An XML file source has these options in addition to read-only XML Schema or DTD format information:

Option	Description
XML file	The location relative to the Job Server of an XML-formatted file to use as the source.
Enable validation	A check box to turn on the comparison of the incoming data to the stored XML Schema or DTD format. When this option is enabled, the data flow throws an exception if the incoming source is not valid.
implied: Join rank	The XML file source is always the outer table in a join. You cannot change the join rank for it. This option is implied and does not appear in the editor.

An IDoc message source in a real-time job has the following options:

Option	Description
Make Port	(Optional) Select an IDoc as an input port for an embedded dataflow. See Chapter 11, "Embedded Data Flows," in the *Data Integrator Designer Guide*.
IDoc test file	Browse the network to find a test file to read.
IDoc Type	(Read-only) Data Integrator will only process IDoc files (listed in the IDoc Test File box) if they are of this IDoc type. For example, SOPGEN01. This information is entered when you import metadata for the IDoc Type.

An IDoc file source, in a real-time job, has the following options.

Option	Description
Make Port	(Optional) Select an IDoc as an input port for an embedded dataflow. See Chapter 11, "Embedded Data Flows," in the *Data Integrator Designer Guide*.
IDoc File/s	Browse to a network file that will contain IDocs generated in SAP R/3. Data Integrator reads the contents of this file/s when this data flow runs. See also: "Multiple File Read in batch jobs" on page 232.
IDoc Type	(Read-only) Data Integrator will only process IDoc files (listed in the IDoc File box) if they are of this IDoc type. For example, SOPGEN01. This information is entered when you import metadata for the IDoc Type.
Join Rank	(optional) Determines the join order when this IDoc is joined with another object in this data flow. The default value is zero. To override the default value, set to a non-zero positive integer. For information on how Data Integrator uses this field, see "Join ordering" on page 24 of the *Data Integrator Performance Optimization Guide*.

Table

CLASS

Reusable

ACCESS

In the object library, click the **Datastores** tab.

DESCRIPTION

You can use a table from a non-SAP R/3 datastore as a source or target in a data flow. You cannot use a table from a non-SAP R/3 datastore in an R/3 data flow.

You can use an SAP R/3 table as a source or target in an R/3 data flow or as a source in a data flow. You cannot use an SAP R/3 table as a target in a data flow.

Tables have the following built-in attributes:

Attribute	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object.
Description	Your description of the table.
Business_Name	Business name imported with table metadata.
Table_Usage	How the object is used when it was accessed most recently. Expect values such as Source, Target, Lookup Table, or Comparison Table.
Total_Number_Of_Rows_Processed	The number of rows loaded into the table in the last successful load.
Date_last_loaded	The time the table was last successfully loaded by Data Integrator.
Number_Of_Rows_Rejected	The number of rows rejected in the last successful load.
Number_Of_Inserts	The number of rows inserted in the last successful load.

Attribute	Description
Number_Of_Updates	The number of rows updated in the last successful load.
Date_Created	The date that the object was created.
Estimated_Row_Count	A configurable estimate of the table size used in calculating the order tables are read to perform join operations. Used for SAP R/3 tables only.
Number_Of_Deletes	The number of rows deleted in the last successful load.
Elapsed_Time_For_Load	The time it took to load this table in the last successful load.
Table_Type	The type of datastore object. Most tables are type Table.
SAP_Table_Class_Name	The table class name as designated by the SAP_Table_Class value.
SAP_Table_Class	Table type information. Used for SAP R/3 tables only. 1 — Transparent table 2 — INTTAB 3 — Cluster 4 — Pool 5 — View 6 — Append (Structure)

Target

Table

Template table

File

Data transport

XML message

XML file

IDoc message

IDoc file

CLASS

Single-use

ACCESS

- To display target options, click the name of the target in the workspace or in the project area. This opens the object editor.
- To display target properties, right click a target and choose **Properties**.

DESCRIPTION

A target is an object in which Data Integrator loads data.

In a batch job, a target can be a flat file, a table or template table, an XML file, an IDoc file, an IDoc message, or an R/3 data flow.

In a real-time job, a target can be a file, a table, a template table, an XML file, an XML message, an IDoc file, or an IDoc message.

In an R/3 data flow, a target can be a file or a data transport object.

TARGET FILES

You can use any flat file format in a batch or R/3 data flow as a target. You can use any DTD or XML Schema format for XML file and XML message targets.

If the schema defined in the format doesn't match the schema that is input to the target, Data Integrator provides validation errors to identify the mismatch.

Target tables

No tables can be used in R/3 data flows as targets.

NOTE: For target file, table and XML object information in general, see "Target" on page 103 of the *Data Integrator Reference Guide*.

Target IDocs

IDoc messages can be sent from Data Integrator on a regular schedule or in real-time. Data Integrator allows you to use a batch or real-time job to configure IDoc message targets. Any number of IDoc targets can be included in a single data flow.

IDoc targets support reduced message types. When Data Integrator detects a reduced message type, it retrieves the appropriate metadata from SAP R/3 and creates a reduced IDoc, removing unused segments and giving unused fields the value of '/'.

NOTE: Reduced message type functionality is only available for some message types (e.g., DEBMAS, CREMAS, etc.).

The attributes of an IDoc target depend on the type of target and the type of data flow. To see the IDoc attributes, double-click an IDoc target object in a data flow diagram.

- IDoc message targets in a data flow of a batch job

Attribute	Description
Make Port	(Optional) Select an IDoc as an input port for an embedded dataflow. See Chapter 11, "Embedded Data Flows," in the *Data Integrator Designer Guide*.
IDoc Type	(Read-only) Displays the IDoc Type you imported into Data Integrator for this object. For example, SOPGEN01.
Partner Number	This is an SAP term for where to send the IDoc in SAP R/3. Enter the partner number of the IDoc you will be processing as a message target in DATA INTEGRATOR. If you do not know the partner number, consult your SAP R/3 administrator.
Partner Type	This is an SAP R/3 term for an attribute of a partner number. Enter the partner type. For example, LS for logical system.
Message Type	This is an SAP R/3 term for an EDI message for this IDoc Type. Enter the message type.

- IDoc message targets in a data flow of a real-time job

Attribute	Description
Make Port	(Optional) Select an IDoc as an input port for an embedded dataflow. See Chapter 11, "Embedded Data Flows," in the *Data Integrator Designer Guide*.
IDoc Test File	Browse to a network file that will contain IDoc/s generated in Data Integrator. For example: C:/temp/filename. Use this location to test the output of the IDoc message target before starting a real-time job in the Data Integrator Administrator. Data Integrator writes to this file when this real-time job is executed in test mode.
IDoc Type	(Read-only) Data Integrator will only process IDoc files (listed in the IDoc Test File box) if they are of this IDoc type. For example, SOPGEN01. This information is entered when you import metadata for the IDoc Type.
Partner Number	This is an SAP R/3 term for where to send the IDoc in SAP R/3. Enter the partner number. If you do not know the partner number, consult your SAP R/3 administrator.
Partner Type	This is an SAP R/3 term for an attribute of a partner number. Enter the partner type. For example, LS for logical system.
Message Type	This is an SAP R/3 term for an EDI for this IDoc Type. Enter the message type.

- IDoc file targets in data flows

Attribute	Description
Make Port	(Optional) Select an IDoc as an input port for an embedded dataflow. See Chapter 11, "Embedded Data Flows," in the *Data Integrator Designer Guide*.
IDoc File	Browse to a network file that will contain IDocs generated in Data Integrator. Data Integrator writes to this file when the data flow is executed. When you configure an IDoc File target, Data Integrator allows you to capture data for more than one IDoc. Each time a root segment passes into a file target (for example **IDoc File** C:\temp\idoc), a new file is created and named using an eight character suffix starting with 00000000.txt. For example an **IDoc File** set to C:\temp\idoc could store idoc00000000.txt to idoc99999999.txt.
IDoc Type	(Read-only) The IDoc type you imported from SAP R/3 into Data Integrator for this object. For example, SOPGEN01.

Data types

For more information on data types in general, see Chapter 4, "Data Types," in the *Data Integrator Reference Guide*.

This section discusses:

- Conversion to/from Data Integrator internal data types
- Working with null values from SAP R/3
- Design considerations for SAP R/3 null behavior
- Null dates

Conversion to/from Data Integrator internal data types

The following table lists the conversions from SAP R/3 data types to Data Integrator internal data types.

Source	External data type	Converts to Data Integrator data type
SAP R/3	date (D)	date
	time (T)	time
	float (F)	double
	integer (I)	int
	numeric (N)	varchar
	packed (P)	decimal
	char (C)	varchar
	hexadecimal (X)	varchar

The following table lists the conversions from Data Integrator internal data types to SAP R/3 data types.

Internal data type	Is converted to SAP R/3 data type
date	date (D)
datetime	date (D)
decimal	decimal (P)
double	float (F)
int	int (I)
interval	int (I)
numeric	char (C)
real	float (F)
time	time (T)
varchar	char (C)

Working with null values from SAP R/3

SAP R/3 uses a different standard for null values than Data Integrator or the target databases that Data Integrator supports. You can configure Data Integrator to account for the difference in one of two ways:

- Always converting the SAP R/3 equivalent of a null value to the Data Integrator NULL constant
- Reading the SAP R/3 null value literally

Each method has consequences for how you can use null-value tests in analytic processing. Data in a given warehouse must be loaded over time using the same method.

The default configuration for Data Integrator is to read the SAP R/3 value literally. You can change this configuration in the Designer option **Convert R3 null to null**. Go to the **Tools** menu and choose **Options**. Under **Administrator Options**, choose **Server Options** to see the **Convert R3 null to null** check box.

Design considerations for SAP R/3 null behavior

When choosing which method to use to handle SAP R/3 null values and when designing extraction logic in Data Integrator, consider the following issues:

- Primary keys built from concatenated values

  Spaces are often used in SAP R/3 where null values would cause an error in a primary key built from several key values concatenated together. If you choose to have Data Integrator automatically convert these spaces to null values, you may have to include null-value replacement logic in your data flows to avoid errors in primary keys for extracted tables.

- Null date values

  If dates are left to Data Integrator and the target database conventions to determine the conversion of an SAP R/3 null date value, the result may not be useful for constructing reports. See "Null dates" for more details.

- Filters for null values

The value used to match an SAP R/3 null value depends on the data type of the value.

Data type	SAP R/3 null value
Strings	single space
Numbers	zero
Dates	00000000
Times	000000

Null dates

The concept of a null date in SAP R/3 is represented by a value called INITIAL that is a property of every SAP R/3 data element. You can see INITIAL dates displayed through SAPGUI as 00/00/0000 (month 0, day 0, year 0).

In the default configuration of Data Integrator (Convert R/3 nulls to Nulls is disabled), INITIAL dates are written to an R/3 transport file as NULLs. In R/3 transport files, NULLs are represented by empty strings. They are then read by the parent data flow, which converts every empty string to a single space. If passed through again without data type or value change, Data Integrator converts the single blank into 01-JAN-1900 before loading it into a target.

The single space is not a valid date value. When Data Integrator encounters a single space in date conversion operations (implicit or explicit) or when evaluating an expression, the results are not predictable. To avoid unpredictable results, the data flow must explicitly convert the INITIAL date to a valid date value.

Consider these techniques to convert INITIAL dates to valid date values:

- Convert INITIAL dates to valid dates using the nvl function in the R/3 data flow

  In an R/3 data flow, Data Integrator produces ABAP for the nvl function that looks for an INITIAL date rather than a null value. The mapping expression can be the following:

  ```
  nvl(date_column, valid_date)
  ```

  where the valid date is a parameter passed to the data flow or a string indicating the appropriate initial date for your data warehouse.

  An example mapping expression would be:

  ```
  nvl(VBAK.ANGDT, '19901231')
  ```

- Turn on the option **Convert R3 null to null** and use nvl in non-R/3 data flows

If an INITIAL date is extracted from SAP R/3 and loaded directly into a Data Integrator transport file, the INITIAL date is converted to a null value. If the **Convert R3 null to null** option is turned on, the null values in the transport file are brought into the data flow as null values. You can then use the nvl function in the data flow to detect the null date and substitute an appropriate valid date for it.

Functions

Data Integrator does not support functions that include tables as input or output parameters, except RFC and BAPI functions imported from SAP R/3.

For more information on functions in general, see Chapter 6, "Functions and Procedures," in the *Data Integrator Reference Guide*.

SAP R/3 RFC and BAPI function calls

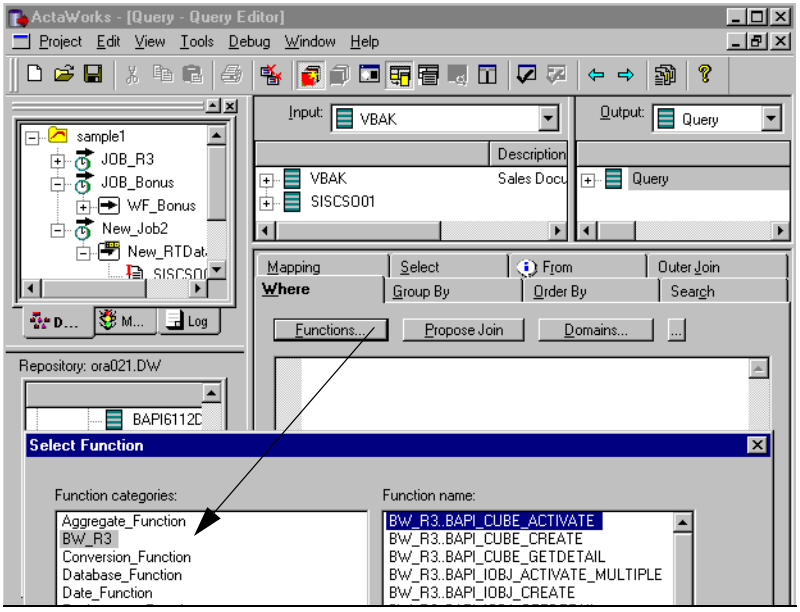
You can call SAP R/3 Remote Function Call (RFC) enabled functions, including Business Application Programming Interface (BAPI) functions, from queries inside data flows.

To make an RFC function available to call from Data Integrator data flows, import the metadata for the function from the SAP R/3 application server using an SAP R/3 datastore connection. Be aware that the requirements for RFCs and BAPIs, and therefore their metadata, may be different between versions of SAP R/3.

If you design data flows with BAPI calls against one version of SAP R/3, then change datastores to a later version of SAP R/3 Data Integrator allows this without the need to re-import the BAPI. Any new parameters added to the function call, including additional columns in table parameters, are added automatically to the call and filled with NULL values. Thus Data Integrator allows you to design jobs that are portable between SAP R/3 systems.

For a Data Integrator job to execute an RFC function, the login indicated by the datastore into which you imported the function must include the appropriate permissions required to execute the functions.

After you import the metadata for an SAP R/3 function, the function is listed in the Functions category of the R/3 datastore. You will also see the function in the function wizard, listed under the datastore name.



See "Calling application functions" on page 271 of the *Data Integrator Designer Guide*.

When a function can provide a single output value, you can call the function in a column mapping in a query. Data Integrator provides a function wizard (available by clicking the **Functions** button on the **Where** tab of a query) to help you include input parameters for the function call. Input parameters can include expressions.

When a function can provide more than one output value, you can call the function by including it as an object in the output schema of a query. Data Integrator provides a function wizard (available through a short-cut menu) to help you include input and output parameters for the function call. See "Calling application functions" on page 271 of the *Data Integrator Designer Guide*.

Data Integrator supports tables as input and output parameters for SAP R/3 RFC and BAPI functions. The function import process automatically includes the metadata for tables included as function parameters.

To specify a table as an input parameter to a function, the table must be an input to a query, either as a top-level input or nested under the top-level. The table must also be available in the FROM clause of the context where you call the function. Data Integrator maps columns in the input schema by name to the columns in the table used as the function input parameter. You need only supply the columns that are required by the function. At validation, if Data Integrator encounters type mismatches between supplied columns and the function signature, it attempts to convert the given type to the expected type. For type mismatches that it cannot resolve, Data Integrator produces validation errors. The implicit and illegal type conversions are described in "Type conversion" on page 208 of the *Data Integrator Reference Guide*.

One of the values that an RFC function can return is AL_RFC_RETCODE. This column contains a flag that identifies the success or failure of the function call. The possible values for AL_RFC_RETCODE are as follows:

Value	Description	Returned by
ACTA_RFC_OK	The RFC call succeeded. This value is replaced by the return value from the RFC call.	Data Integrator
ACTA_RFC_R3_CONN_EXCEPTION	The R/3 datastore connection cannot be created, because of a connection error, invalid user, password, system number, or host name.	Data Integrator
ACTA_RFC_CALL_ERROR	The connection completes, but the call fails in SAP R/3.	Data Integrator
RFC_OK	The function call succeeded. Look for the results or errors that it returns.	SAP R/3
RFC_FAILURE	The function call returned an error. If the function is a BAPI, details for the cause of the error are available in the RETURN structure available as an output from the function.	SAP R/3
RFC_EXCEPTION	The function call returned an error. If the function is a BAPI, details for the cause of the error are available in the RETURN structure available as an output from the function.	SAP R/3
RFC_SYS_EXCEPTION	The function call returned an error and closed the connection to Data Integrator. If the function is a BAPI, details for the cause of the error are available in the RETURN structure available as an output from the function.	SAP R/3
RFC_CALL	The function call was received by SAP R/3. If this value is left, the function failed to return a success flag after starting.	SAP R/3
RFC_CLOSED	SAP R/3 closed the connection and cancelled the function call.	SAP R/3

Value	Description	Returned by
RFC_MEMORY_INSUFFICIENT	SAP R/3 doesn't have enough memory available to process the function call.	SAP R/3
RFC_VERSION_MISMATCH	The version of the function call from Data Integrator is incompatible with the function expected by SAP R/3.	SAP R/3

The RETURN structure for BAPIs varies between releases of SAP R/3:

Field	Description
TYPE	blank[a] — success S — success E — error W — warning I — information A — abort
CODE	Error message numbers
MESSAGE	Error message text in the language chosen at login

a. This value is blank or NULL depending on the current setting of the Server option **Convert R3 null to null**. Check this option by choosing **Tools** > **Options** in the Designer.

In particular when calling BAPI functions, the data you provide through the BAPI call may be different from the data that you use to test a BAPI directly in the SAP R/3 GUI interface. The SAP R/3 interface automates data handling, where the BAPI operation undercuts the interface level.

Consider the following issues:

- All character values must be uppercase
- Padding values
- Assumed decimal values (QTY)
- Codes are language-specific
- Automatic type conversion
- SAP R/3 version-specific behavior

To determine the data requirements of various SAP R/3 functions, you can read the function requirements in the SAP GUI transaction screens:

- BAPI list by functional area: `bapi`
- BAPI and RFC source and input and output parameters: `se37`

You can also determine appropriate values, such as the language-specific code values, by looking at the table where the data is ultimately stored.

Data Integrator Designer functions

The following table lists names and descriptions of the Data Integrator functions available in the Designer that are either SAP R/3-specific or are associated with specific SAP R/3 supplemental reference information. While the supplemental information is listed in this document, see Chapter 6, "Functions and Procedures," in the *Data Integrator Reference Guide* for basic information about these and other Data Integrator functions.

Function	Category	Description
lookup	Miscellaneous	Finds a value in one table or file based on values in a second table or file.
substr	String	Returns a specific portion of a string starting at a given point in the string.
sy	Miscellaneous	Returns the value of an SAP R/3 system variable at run time. This function is only available through query transforms in R/3 data flows.

lookup

For SAP R/3 tables, no owner value is required, but the periods are. For example: `sap_ds..KNA1`

cache_spec The caching method the lookup operation uses. List within single quotes.

With regard to the three possible settings:

- NO_CACHE

  Reads values from the *translate_table* for every row without caching values.

- PRE_LOAD_CACHE

  For SAP R/3 tables, applies the filters and loads the *result_column* and the *compare_column* into an internal R/3 table. The function reads the values from the internal table.

  Select this option if the number of rows in the table is small or you expect to access a high percentage of the table values.

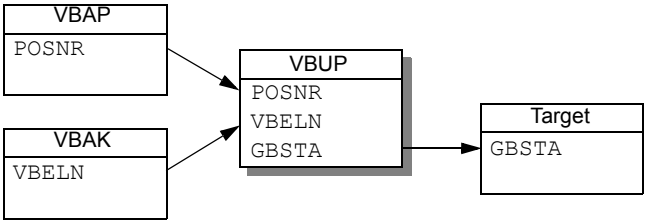
- DEMAND_LOAD_CACHE

  Select this option if the number of rows in the table is large and you expect to access a low percentage of the table values frequently.

  For SAP R/3 tables, loads the *result_column* and *compare_column* into an internal R/3 table when two or more lookups use the same translation table. So, Data Integrator creates a single SELECT statement to load the combined values from all the lookups into an internal R/3 table.

  Select this option when you use the table in multiple lookups and the compare conditions are highly selective, resulting in a small subset of data.

You can use the lookup function when retrieving data from SAP R/3 tables in an R/3 data flow. For example, suppose you are joining two tables, VBAK and VBAP, and you want to retrieve a value from a third table, VBUP.



You can create a lookup function with two *expression* and *compare_column* pairs:

```
lookup(sap_ds..VBUP, GBSTA, 'none',
'NO_CACHE', VBELN, VBAK.VBELN, POSNR,
VBAP.POSNR)
```

This function returns the value from the GBSTA column (the *result_column*) in the VBUP table (the *translate_table*) that corresponds to the VBELN value (first *compare_column*) in the VBAK table (first *expression*) and the POSNR value in (second *compare_column*) the VBAP table (second *expression*). When no corresponding value is found, the function returns "none."

When there are multiple matching rows in the *translate_table* and the translate table is an SAP R/3 table, the lookup function randomly selects a matching row and returns the value in the *result_column* for that row. To avoid random selection:

- Use the lookup_seq function if the table has a sequence column that discriminates between matching records
- Use an outer join and add criteria to the WHERE clause to discriminate between multiple matching records

substr

start	In an R/3 data flow, the first character is position 0; in other data flows, the first character is position number 1. If *start* is 0 in a non-R/3 data flow, the new string begins with the first character (that is, position 1). If *start* is negative, the new strings begins with the first character in the string (that is, position 0 in an R/3 data flow or position 1 in a non-R/3 data flow).

sy

Returns the value of an SAP R/3 system variable at run time. This function is only available through query transforms in R/3 data flows.

SYNTAX

```
sy('R3_variable')
```

RETURN VALUE

varchar(255)	The value of the SAP R/3 system variable. You may need to recast the return value to the actual data type of the system variable in SAP R/3.

WHERE

R3_variable	A string value containing the name of the SAP R/3 system variable. This value is not case-sensitive. Enclose the name in single quotation marks (').

When the sy function is executed, Data Integrator generates the appropriate function call in the ABAP for the R/3 data flow (appends `SY-` to the *R3_variable* that you specify) and returns the result of the function call at run time.

The table SYST in SAP R/3 lists the available system variable, their data types, and descriptions.

If the given *R3_variable* does not exists in SAP R/3, Data Integrator produces a run-time error:

```
ABAP program <Generated ABAP Program> syntax
error: <The data object "SY" has no component
called "ABC">.
(3.0) 06-04-99 11:51:36 (E) (260:354)
R3C-150504: Dynamically execute generated ABAP
failed. The ABAP file is <filename>. Upload the
program to SAP to find the problem.
```

No value is returned when this error occurs.

EXAMPLES

Function	Results
`sy('SUBRC')`	`0` if the subroutine call returned successfully
`sy('mandt')`	The client ID number of the current SAP R/3 client
`sy('UNAME')`	The logon name of the user
`sy('DATUM')`	The current date from the SAP R/3 application server
`sy('UZEIT')`	The current time from the SAP R/3 application server
`sy('TCODE')`	The current transaction code

A Glossary

This glossary defines terms used in this documentation set including common terms that have a special meaning with respect to Data Integrator. Within a definition, other terms that are defined in the glossary appear in **bold type**.

ABAP

Advanced Business Application Programming.
A fourth-generation programming language developed by SAP in which R/3 applications are written.

ABAP program

A program that executes database operations on an **SAP R/3** server. Data Integrator **R/3 data flows** generate ABAP programs.

Access Server

A process that starts, stops, and monitors real-time services (**real-time jobs**). The Access Server dispatches requests to real-time services, ensuring optimal load balancing and complete life cycle management. The Access Server also starts, stops, and monitors **adapters**.

adapter

An external Data Integrator **interface**. There are two types of adapters:

- Custom adapters — Adapters developed using the Data Integrator Adapter Development Kit
- Prepackaged adapters — Adapters prebuilt and purchased from Business Objects, such as the Data Integrator MQSeries Adapter

Administrator

A browser-based system administration application that supports Data Integrator. Use the Administrator to configure and manage:

- Services (**real-time jobs**)
- Service providers (instances of real-time services)
- Clients (Web application clients)
- Batch jobs
- Adapters
- Job logs

after-image

The values in an UPDATE row after the row changes. You might want to pair before- and after-images of UPDATE rows when designing log-based changed-data capture jobs which Data Integrator supports.

aggregate data

Data that results when a process combines elements. This data can be presented collectively or in summary form.

ALE

Application Link Enabling. An SAP R/3 programming-related interface designed to allow reliable communication across a distributed environment.

Analytic Rapid Mart

An **Data Integrator application** that produces a predefined **data warehouse** for a specific area of business analysis. Analytic Rapid Marts are typically based on dimensional modelling. With Analytic Rapid Marts, users can manage large volumes of data.

attribute

A **property** created for a type of **object**.

BAPI

Business Application Programming Interface. A standardized SAP R/3 programming interface that allows non-SAP R/3 applications to access specific business processes and data.

back-office application

A software system that an enterprise uses to run business operations, such as human resources, general ledger, and finance. Back-office applications store data in a relational **database** optimized for operational use.

There are several types of back-office applications:

- Enterprise resource planning (ERP) — For example, SAP R/3, BAAN, J.D. Edwards, PeopleSoft
- Supply chain management (SCM) — For example, i2
- Customer relationship management (CRM) — For example, Siebel, Clarify
- Legacy

Basis

The SAP infrastructure. Basis is the foundation for all SAP products.

batch job

A group of objects (data flows, work flows, conditionals, scripts, and so forth) that executes on a schedule. This type of job is recommended for operations on large amounts of data.

before-image

The values in an UPDATE row before the row changes. You might want to pair before- and after-images of UPDATE rows when designing log-based changed-data capture jobs which Data Integrator supports.

bulk loading

A software-based mechanism that moves large data files. This mechanism supports compression, blocking, and buffering to optimize transfer times.

business component

A set of tables Siebel applications use to create a logical object called a business object.

Business Objects Data Platform

A set of applications from Business Objects, Americas, that enterprises can use to manage all types of data—from data for supply chain partners to data for customers. The Business Objects Data Platform consists of **Data Integrator** and one or more **Rapid Marts**.

CDC datastore

A CDC datastore allows you to limit extracted data to changed data only. A CDC datastore connects a **changed-data capture** table on a source database to Data Integrator.

CDC checkpoint

A CDC checkpoint enables Data Integrator to restrict CDC subscription reads. After you enable a checkpoint, the next time the CDC job runs, it reads only the rows inserted into the CDC table since the last checkpoint.

CDC Subscription

A CDC subscription name allows different data flows to extract data from the same table without corrupting data extracted by other data flows. The datastore and table name can be the same; however, the table owner name must be different in each source CDC table.

changed-data capture

The process of retrieving changes made to a production data source. This process consolidates units of work, ensures data is synchronized with the original source, and reduces data volume in a warehouse environment.

Citrix MetaFrame XP

Citrix MetaFrame XP software provides an access infrastructure for enterprise applications. You can use this software to run Data Integrator on a server which publishes instances of the Designer and other Data Integrator interfaces to users on client computers.

client/server

A distributed technology approach where the processing is divided by function. The server performs shared functions (such as managing communications and providing database services), while the client performs individual user functions.

COM

Component Object Model. Provides a set of interfaces that allow clients and servers to communicate within the same computer (running on Windows 95 or later versions).

Commerce Rapid Mart

An **Data Integrator application** that produces a predefined **data warehouse** and that includes logic to support real-time requests for data from a **back-office application**. A Commerce Rapid Mart is usually based on different data model than an **Analytic Rapid Mart**. With a Commerce Rapid Mart, users can integrate a company's back-office system with an **e-commerce** application.

Communication Structure

In **SAP BW**, a data structure that defines a set of **InfoObject**s available from an **InfoSource** to put into **InfoCube**s.

conditional

A **single-use object**, available in **work flow**s**,** that allows you to branch the execution logic based on the results of an expression. The conditional takes the form of an if/then/else statement.

custom ABAP program

A custom ABAP program runs an **ABAP program** and generates a **data set**. With a custom ABAP program, you can run an existing ABAP program as part of a job. Use a custom ABAP program as a **source** in a **data flow** or an **R/3 data flow**.

data extraction

The process of moving data from a **database** or application source to a database target (either from a legacy database to a **data mart**, or from one data mart to another).

data flow

A **reusable object** containing **steps** to define the transformation of data from **source** to **target**. Data flows are called from inside a **work flow** or **job**. You can pass information into or out of data flows using **parameters**.

Data Integrator

A software system that allows users to build and execute applications with which they can create and maintain **data warehouses**.

Data Integrator consists of several components:

- **Access Server**
- **Data Integrator engine**
- **Data Integrator repository**
- **Data Integrator Service**
- **Administrator**
- **Designer**
- **Job Server**

Data Integrator application

A set of **jobs** that process batch data extractions and real-time requests using data from a variety of data sources.

Data Integrator engine

The core process that reads job information from the **Data Integrator repository** and sets up run-time processes that execute the job. The run-time processes extract, transform, and load relational and hierarchical data. The **Job Server** starts the Data Integrator engine to execute batch or real-time jobs.

Data Integrator interface

A program that Data Integrator uses to access data sources. Specific interfaces vary by installation. There are internal interfaces—those native to the installation—and external interfaces—those that you install separately. Internal interfaces allow Data Integrator to access applications like **SAP R/3** and **SAP BW**, **messages**, relational database systems, and **legacy systems**. An external interface is also known as an **adapter**. It allows Data Integrator to access applications using information exchange technologies such as Web Services and MQ Series.

Data Integrator repository

The database that contains information about an Data Integrator application. The repository contains information about defined **reusable objects**, the **metadata** for tables, and built-in **transforms** and **functions**. When you invoke Data Integrator, you log in to the repository containing the **objects** you want to use.

Data Integrator Service

The process that ensures that the **Access Server** and the **Job Server** are running. You can configure the Data Integrator Service to restart the Access Server and Job Server whenever the computer where they are located restarts.

data loading

The process of populating a **data warehouse**. Data loading is provided by **DBMS**-specific load processes, DBMS insert processes, and independent fast-load processes.

data mapping

The process of assigning a source data element to a target data element.

data mart

A highly-focused version of a **data warehouse**. Typically, created by a department or division of a company, data marts contain data for a specific subject area, such as finance or sales. Data Integrator can populate a data mart.

data transformation

The process of filtering, merging, decoding, and translating source data to create validated data for a **data warehouse**.

data set

Rows of data with a defined schema. A step in a **data flow**—such as reading data from a **source**, joining data in a **Query transform**, or transforming data though another **transform**—yields a data set. You can view individual data sets by placing a **target** table or file at that point in the **data flow**.

data transport

A step in an **R/3 data flow** that defines a **target** to store the **data set** extracted during the flow. You can locate the target file on the **SAP R/3** server or in a location accessible to both the SAP R/3 server and to Data Integrator across a network.

data type

The format used to store a value. Data types can imply a default format for displaying and entering the value. Data read from a **source** is converted to the appropriate Data Integrator data types; data loaded to a **target** is converted from its Data Integrator data type to the type appropriate for the target.

data warehouse

A relational **database** containing subject-oriented information that supports decision analysis. Typically, information in a data warehouse is integrated from multiple sources, including operational sources. Data warehouses provide a single, non-volatile point of reference for a particular time period. Data warehouses are composed of **data marts.** Data Integrator can populate a data warehouse.

database

A collection of **table**s managed by a **DBMS** such as Microsoft SQL Server or Oracle.

DataConnector

DataConnector operator instances are used to read data files generated by Data Integrator when performing bulk loading using the Teradata Warehouse Builder.

datastore

A logical channel connecting Data Integrator to a **database**. The datastore definition includes the name and location of the database as well as user authentication information. Data Integrator uses a datastore definition to qualify a table name wherever a table is indicated in a diagram or expression. You can access the datastore definition through the **object library**.

DBMS

Database management system. A software system that builds and maintains database tables.

debug mode

Allows you to debug a job using the **interactive debugger** features in the Designer.

declarative specification

A way of indicating the desired results of a **data transformation** without placing constraints on the method of reaching the results. When you create a **data flow** in Data Integrator, you specify what data you want to read from which **source**, what you want done to that data, and where you want it loaded. What you specify constitutes the declarative specification. Data Integrator then determines the most efficient way to reach the requirements you set, including pushing operations to the source **DBMS** for execution.

degree of parallelism

You can set a value at the data flow level that allows you to parallel-process a **transform** within a data flow.

For example, if from the Properties window of a data flow you set the **Degree of parallelism** to 5, then when the job executes, Data Integrator replicates each transform in the data flow five times. Each of these replicated transforms executes in parallel using a separate process thread. Using a value for degree of parallelism can improve job performance if a **Job Server**'s computer memory and number of CPUs supports a job's parallel-processing configuration settings.

dependency analysis

A category of metadata reports that produces reports based on search criteria you specify to find specific objects in the repository. For each object returned, dependency analysis reports provide object lineage as well as object impact information.

derived data

Data that results from a computational step applied to reference or event data. Derived data is the result of either relating two or more elements of a single transaction (such as aggregation), or relating one or more elements of a transaction to an external algorithm or rule.

Designer

A graphical user interface that allows you to design and test an **Data Integrator application**.

diagram

The icons and connections between the icons that make up the definition of a **job**, **work flow**, or **data flow**.

DMZ

Demilitarized zone. A perimeter network that is more accessible than your most protected network, but located inside a security layer protected from the internet.

drill down

A method of exploring detailed data that was used in creating a summary level of data. Drill down levels depend on the granularity of the data in the **data warehouse**.

DTD

Document type definition. A text file that describes the elements (tags) in an **XML** document and the relationship among them. When an XML document is used to describe a transaction, the DTD describes the data schema used in the transaction.

e-commerce

Buy and sell transactions conducted over the internet.

EDI

Electronic Data Interchange. Electronic exchange of structured data between businesses. This exchange is not dependent on hardware, software, or communication protocols.

element

A component found within XML Schemas and DTDs.

embedded data flow

A data flow created within one or more data flows.

enterprise data

Data that is defined for use across a corporate environment.

ERP system

Enterprise resource planning system. A **back-office application** from which Data Integrator can extract data.

exception

An error that occurs while executing a **job**. You can catch individual or groups of exceptions using a **try/catch block** inside a **work flow**. Catching an exception allows you to automatically execute a solution for the error.

Executive Information Systems (EIS)

Tools programmed to provide canned reports or briefing books to executives. They offer strong reporting and drill-down capabilities. These tools allow ad hoc querying against a multi-dimensional **database**, and most offer analytical applications along functional lines such as sales or financial analysis.

extract date

The date data was extracted.

extract frequency

The interval at which data is extracted, such as daily, weekly, monthly, or quarterly. The frequency that data extracts are needed in the **data warehouse** is determined by the shortest frequency requested through an order, or by the frequency required to maintain consistency of the other associated data types in the source data warehouse.

file

A **source** or **target** for data. To read or load data to a file within Data Integrator, define a **file format** that instructs Data Integrator about the structure of the data in the file.

file format

A description of how data is or should be organized in a **file** Data Integrator reads from or loads to. A file format can be specific to a single file or generic for many files.

front-office application

Software that companies use to support reporting, **e-commerce**, and other tasks. Types of front-office applications include:

- Supply chain collaboration
- Analytical
- Sell-side e-commerce
- Customer relationship management

function

A program that operates on values that are passed to it. Data Integrator functions are available through a function wizard in a **script**, **conditional**, or **Query transform**. Data Integrator also gives you access to functions provided by the **DBMS** you are using. In addition, you can define your own functions using the Data Integrator scripting language or by writing C-language programs.

IDoc

Intermediate Document. An SAP-specific format. Used for **EDI** and **ALE**.

IDoc type

Indicates the SAP format that is used to interpret the data of a business transaction. Consists of the following components:

- A control record

  Identical for each IDoc type.

- Several data records

  A single data record consists of a fixed key part and a variable data part. The data part is interpreted using segments, which differ depending on the IDoc type selected.

- Several status records

  Identical for each IDoc type. Describe the status states an IDoc has already passed through or the status an IDoc has attained.

import

The process of acquiring information for the **Data Integrator repository**. Import the following kinds of information into Data Integrator:

- The **metadata** for **source** and **target** databases
- Descriptions and code for user-defined and DBMS **functions** and **transforms**

InfoArea

In **SAP BW**, an arbitrary grouping of **InfoCube**s.

InfoCube

In **SAP BW**, a multidimensional star schema that is limited to one fact table and 16 dimensions.

InfoCubes store only numeric data at the fact level.

SAP BW reserves three of the dimensions for currencies, time, and units of measure. Dimensions have to be linked back to master data and hierarchy tables to gain access to the information stored there.

When a user query is initiated, BW creates a temporary **OLAP** cube that stores a constrained subset of the permanent OLAP cube. The temporary InfoCube persists until the user disconnects from BW or changes the query.

InfoObject

In **SAP BW**, an InfoObject is equivalent to a data element and its associated metadata (for example, description or business rules). Can be a **Key Figure** or a Characteristic.

InfoPackage

In **SAP BW**, a schedule for running BW updates.

Information resource (IR)

A producer or consumer of enterprise information. An information resource can be a flat file, a **back-office application**, an EAI system, or a Web application.

InfoSource

In **SAP BW**, a logical construct that defines a set of **InfoObjects** and their mappings to source data elements. InfoSources do not store data, but rather serve as conduits via **Transfer Structures**.

Integration Rapid Mart

An **Data Integrator application** that provides point-to-point content mapping and data movement mechanisms between different **back-office applications**.

interactive debugger

A Designer feature that allows you to step through the data of a job one row at a time using filters and breakpoints on a line. Like executing a job, you can start the interactive debugger from the Debug menu when a job is active in the workspace. While in debug mode, all other Designer features are set to read-only.

interface

Data Integrator offers two types of interfaces:

An internal **Data Integrator interface**, allows you to create datastore connections to natively supported applications.

An external **Data Integrator interface** (or **adapter**), allows Data Integrator to communicate with information exchange technologies such as Web Services and MQ Series.

IR

See **Information resource (IR)**.

job

The unit of work that can be scheduled independently for execution by the **Administrator**. Jobs are special **work flows** that can be called by other work flows or jobs.

Job Server

A process that receives requests from the **Designer** and the **Administrator** to start and stop **jobs**. To start batch or real-time jobs, the Job Server triggers the **Data Integrator engine**. Engine processes run on the same computer as the Job Server process that triggers them.

join rank

A value given to or calculated for all data sets in a **data flow**. Data Integrator uses the join rank to determine which **source** to read first when assembling the **data set** in a join. Data Integrator uses the source with the lower join rank as the inner source of the join and uses the source with the higher join rank as the outer source of the join. If more than one of the sources is located in the same **DBMS**, the highest rank is used for all of the sources from that DBMS. When a source is processed through a **Query transform** or another **transform**, and then used in a join, the maximum join rank defined for the source is used as the join rank in the query.

Key Figure

In **SAP BW**, an **InfoObject** that represents a numeric fact.

legacy system

An information or transaction processing system used to store data such as bank balances, inventories, payroll, and manufacturing parts.

license-controlled feature

A Data Integrator feature that is enabled or disabled based on the product license. The product license controls which icons and settings are available in Data Integrator as an internal **Data Integrator interface**.

line

The line you use to connect objects in a workspace diagram. Lines represent a left-to-right flow path for data moving between two objects during job execution.

memory datastore

A datastore connection/container for memory tables.

memory table

Internal Data Integrator table used to store a data set in memory while a job runs. Use instead of staging tables to improve performance of a real-time job built with multiple data flows. Use a memory table to move a data set between data flows.

message

Represents hierarchical data, such as a header with line items, for document-oriented transactions, such as a purchase order.

metadata

In Data Integrator, information acquired and maintained to describe tables in **source** and **target** databases. This information includes the names of tables and their columns, and the data types of the columns. All the metadata for all user tables is imported at one time for a given **datastore**.

In general, metadata typically includes a description of data models, a description of the layouts used in database design, the definition of the system of record, the mapping of data from the system of record to other places in the environment, and specific database design definitions.

multiple data flow model

One of three data flow models you can use in real-time jobs. Use this model to develop real-time jobs in separate logical object segments (data flows) rather than in a single data flow. Allows for any number of data flows in a real-time processing loop under these conditions: The first data flow must have a single message source, the last data flow must have a single message target. Other data flows in the same real-time job cannot have a message source or message target. See also **put/acknowledge data flow model** and **single data flow model**.

nested data

Data in one **table** that is related to a single row of another table. A nested table appears in Data Integrator as a column in a parent table. Columns in the nested table can themselves contain tables.

object

Any item that you create in the **Designer**. Data Integrator distinguishes two classes of objects: **reusable objects** that are complete and can be reused in your projects (such as **data flows**) and **single-use objects** that only appear as components of other objects (such as a **try/catch block**). This distinction affects how you create and retrieve each type of object.

object definition

The options that describe the operation of an **object**. To view and modify an object definition, open the object so that its definition appears in the **workspace**.

object library

A tool in the **Designer** that gives you access to reusable objects.

ODBC

Open database connectivity. A standard for **database** access initiated by the SQL Access Group consortium and now owned by Microsoft.

ODS

Operational data store. An OLAP-designed relational **database** that an enterprise has designated as the operational database of record (for example, a finance department might use an ODS to close its books).

OLAP

Online analytical processing. A relational **database** design that is optimized for random access. OLAP systems are used in a query environment, such as for a **front-office application**.

OLTP

Online transaction processing. A relational database design optimized for operational use. OLTP systems are used in an operational environment, such as for a **back-office application**.

operation code

A flag associated with a row in a **data set** that indicates the status of the data in the row. The operation codes are INSERT, UPDATE, DELETE, and NORMAL.

operational statistics

A category of reports within the metadata reporting tool that provides you with job and data flow performance information over a given period of time or number of executions.

parameter

A value passed to a **work flow** or **data flow** when that flow is called.

partition

To divide table data into sets based on a criteria such as a range or list of values in each row. You can configure Data Integrator to read and write partitioned table data in parallel threads. Designing jobs with partitioned table data can improve job performance if a Job Server's computer memory and number of CPUs supports the job's parallel-processing configuration settings.

project

The collection of **jobs** available in the **Designer** at a given time. A project provides a way to organize the **objects** you create.

property

Detailed descriptive information about **objects**.

put/acknowledge data flow model

One of three data flow models you can use in real-time jobs. Similar to the multiple data flow model, this model does not require a data flow with a target message. Data Integrator creates a target message to automatically acknowledge a request message. Designed for use with IDoc sources in SAP R/3 systems. See also **multiple data flow model** and **single data flow model**.

Query transform

A **data transformation object** that you can use to map columns from a **source** to a **target** schema, add new columns to the target schema, determine the data to extract, and perform operations on the data. Like a SQL SELECT statement, a query creates a **data set** that satisfies the conditions you specify.

Rapid Mart

A prepackaged **Data Integrator application** that delivers data for a specific business purpose. Easily deployed and customized, BusinessObjects™ Rapid Marts include the extraction, transformation, and data loading logic to support analysis or **e-commerce**. There are three types of Rapid Mart:

- **Analytic Rapid Mart**
- **Commerce Rapid Mart**
- **Integration Rapid Mart**

R/3 data flow

A **data flow** that extracts data from an **SAP R/3** source table. Data Integrator translates steps you define in an R/3 data flow into **ABAP** and then passes the ABAP program back to your SAP R/3 system for execution. The resulting **table** or **file** resides on the SAP R/3 system to be used as a **source** in the parent data flow.

real-time job

A group of objects (data flows, work flows, conditionals, scripts, and so forth) that execute on-demand as a "request-response" system. You design real-time jobs in the **Designer**, then configure them as real-time services and associate them with an **Access Server** in the **Administrator**, where they are started, managed and monitored. When a real-time service receives a request from a caller, it processes the request and returns a reply. This type of job is recommended for operations on small amounts of data.

relational data

A **data set** in which data in each column contains a scalar value. Data Integrator can process relational data; it can also process **nested data**.

repository

See **Data Integrator repository**.

reusable object

An **object** that can be defined, stored, and reused independent of other objects. Create reusable objects from the **object library**.

RFC server

Remote Function Call server. The Data Integrator RFC server allows third-party programs, including SAP R/3, to schedule and initiate Data Integrator jobs and return the results to Data Integrator.

SAP BW

Business Information Warehouse, an SAP product.

SAP R/3

An **ERP system**.

script

A step in a **work flow** that allows you to calculate values to pass to other parts of the work flow. The script can call **function**s, execute if-then-else statements, and assign values to **variable**s. Write a script in the Data Integrator scripting language.

segment

Format with which the data records of **IDocs** are interpreted.

server group

A defined collection of Job Servers on different computers. A server group automatically measures resource availability on each Job Server in the group and distributes scheduled batch jobs to the Job Server with the lightest load at run-time. Use the Server Groups node in the Administrator's navigation tree to group Job Servers that are associated with the same repository into a server group.

service request

Any message sent from a Web client that requires processing by a **real-time job**.

session

Associated with the **Adapter Properties** tab in an adapter datastore, a session is created by a stream-oriented adapter to group adapter components and maintain state for interactivity between Data Integrator and the adapter.

single data flow model

One of three data flow models you can use in real-time jobs. Allows for one data flow in a real-time processing loop. This data flow must have a single message source and a single message target. See also **multiple data flow model** and **put/acknowledge data flow model**.

single-use object

A step in a **work flow** or **data flow** that cannot be saved independently of the flow. Create single-use objects from the tool palette.

smart editor

A flexible editing tool in Data Integrator used for creating scripts, expressions, and custom functions without having to type the names of existing elements like column, function, and variable names.

SNMP

System Network Management Protocol helps network administrators manage network routing hardware. The protocol can manage a variety of hardware and software devices.

source

An **object** from which Data Integrator reads data.

SQL

Structured Query Language. A query language for accessing relational, **ODBC**, DRDA, or non-relational **database** systems.

SQL query tool

An end-user tool that accepts **SQL** to be processed against one or more relational databases.

star schema

A **database** design you can use to format data in a **data mart**. This design is based on a single fact table to which any number of dimensional tables may be joined. This type of database design supports multi-dimensional database analysis.

step

An **object** that is part of the definition of a **work flow** or **data flow**. Each step is represented by an icon in the diagram of the flow and is connected to other steps to indicate the flow of data through the data flow or the order of execution in the work flow.

table

A **database** table that Data Integrator reads data from or loads data into. The path and mechanisms for reading and loading data and apportioning the data among rows and columns are defined in the **datastore** that the table is associated with. Writing a **data set** to a database table means sending a combination of rows with appropriate **operation code** to the database table.

target

An object in which Data Integrator loads extracted and transformed data in a **data flow**. Data Integrator only loads rows flagged as INSERT or UPDATE.

TCP/IP

Transmission Control Protocol/Internet Protocol. The basic communication protocol of the internet, and often intranets and extranets. A computer having direct access to the internet contains a copy of the TCP/IP program. TCP/IP makes it possible for computers to communicate with each other.

Tdpid

(Teradata Director Program ID) The server name Data Integrator uses when loading with the bulk loader option. Data Integrator uses tdpid as a Teradata Warehouse Builder operator attribute.

Transfer Structure

In **SAP BW**, a data structure that defines a set of **InfoObject**s to be moved into BW. To an ETL tool like Data Integrator, a Transfer Structure looks like a table.

transform

A step in a **data flow** that acts on a **data set**. Data Integrator transforms are available through the **object library**. You can also import user-defined transforms and transforms from your **DBMS**.

Transformation Rule (Transfer Rule)

In **SAP BW**, a data manipulation rule between a **Transfer Structure** and a **Communication Structure**. The results of these transformation are globally available to all **InfoCube**s.

try/catch block

A combination of a try object and one or more catch objects that define alternate execution paths in case an error occurs during the execution of a **job**. You can tune try/catch blocks to trap specific errors and to provide general alerts or messages if an error occurs.

Update Rule

In **SAP BW**, a data manipulation rule between a **Communication Structure** and an **InfoCube**. The result of the data manipulation is specific to one InfoCube.

variable

A symbolic placeholder for a value. Data Integrator lets you define local variables and global variables.

Local variables pertain to the **work flow** in which they are defined. You can pass the value into another **work flow** or **data flow** using a **parameter**.

Global variables pertain to the **job** in which they are defined. With global variables, there is no need to define parameters between **objects** in the job. Global variables can also be selected at execution time. This eliminates the need to open the Designer to set global variable values.

Web Services

A standard platform for integrating applications. Web Services allow different programs, constructed in different languages, on different platforms to communicate with each other. DATA INTEGRATOR supports three XML web service technologies: SOAP, WSDL, and **XML Schema**.

work flow

A **reusable object** containing **steps** to define the order of **job** execution. Work flows call **data flow**s, but cannot manipulate data themselves.

You call work flows from inside other work flows or **job**s. You pass information into or out of work flows using **parameter**s. You reuse work flows by dragging existing work flows from the **object library**.

workspace

The window inside the **Designer** in which you define, display, and modify **objects**. The workspace for a **data flow** contains an area to build a **diagram** representing the data flow definition. The workspace for a **transform** contains an editor for modifying the transform options.

WSDL

Web Services Definition Language. Web services are self-contained, modular business process applications based on open Internet standards.

XML

Extensible Markup Language. This markup language is like HTML (Hypertext Markup Language) in that it specifies a standard with which you can define your own markup languages with their own sets of tags. XML allows you to define various tags with various rules, such as tags that represent business rules, tags that represent data description, or tags that represent data relationships.

XML Schema

The XML format used by Data Integrator to support message processing that includes **Web Services**. XML Schemas describe the data structure of an **XML** file or message. Data flows can read and write data to messages or files based on a specified XML Schema format. You can use the same XML Schema to describe multiple XML sources or targets. XML Schema properties include: Name, Description, Imported from, Root element name, and Namespace.

Index

Business Objects, Americas
3030 Orchard Parkway
San Jose, CA 95134
USA

info@businessobjects.com
www.businessobjects.com

Data Integrator

Supplement for Siebel

Version 6.5

Document Number DI-65-0117-001

January 30, 2004

Contents

1 Siebel Interface

The Data Integrator Siebel interface allows you to create Siebel datastores and import tables for use as sources in Data Integrator jobs. With this interface, you can use the Data Integrator Designer to:

- Browse and import Siebel tables grouped by type, business component, and Siebel repository
- Browse to tables in a Siebel repository using the following path through higher level data: applications > screens > views > business objects > business components > tables.
- Import function metadata

This document contains the following sections:

More Data Integrator product documentation

Consult the *Data Integrator Getting Started Guide* for:

- An overview of Data Integrator products and architecture
- Data Integrator installation and configuration information
- A list of product documentation and a suggested reading path

After you install Data Integrator, you can view technical documentation from many locations. To view documentation in PDF format, you can:

- Select **Start** > **Programs** > **Data Integrator** ***version*** > **Data Integrator Documentation** and choose:
  - Release Notes
  - Release Summary
  - Technical Manuals
- Select one of the following from the Designer's **Help** menu:
  - Release Notes
  - Release Summary
  - Technical Manuals
- Select **Help** from the Data Integrator Administrator

You can also view and download PDF documentation by visiting Business Objects Customer Support online. To access this Web site, you must have a valid user name and password. To obtain your user name and password, go to http://www.techsupport.businessobjects.com and click **Register**.

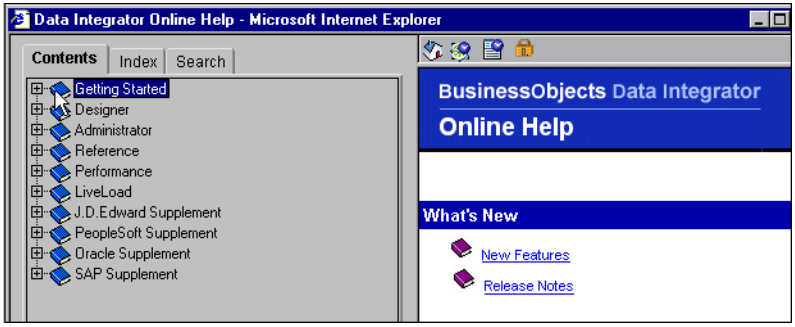
After installing Data Integrator Designer you can also view documentation in HTML format. **Online Help** opens automatically the first time you open the Designer.

You can also open **Help,** using one of the following methods:

- Choose **Contents** from the Designer's Help menu.
- Click objects in the object library or workspace and press **F1**.

  Online Help opens to the subject you selected.

Use **Online Help's** links and tool bar to navigate:



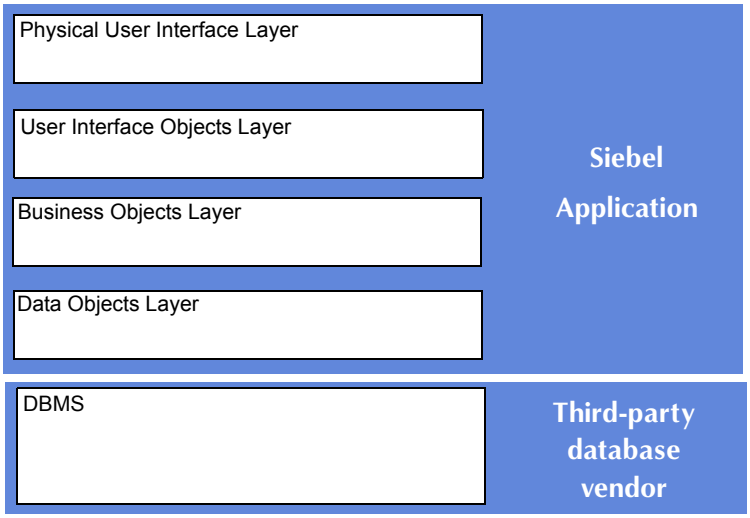
Online help tool bar

Icon	Icon Name	Description
	Home	Returns to the Online Help home page.
	Show/Hide Navigation	Shows and hides the Table of Contents for the core suite of Data Integrator Technical Manuals. Includes a master **Index** and **Search** tabs.
	Show/Hide Descriptions	The Online Help window automatically updates its information when you click an object in the Designer window. In addition, you can click any object in the Designer window then click this button to show or hide an object's local and global variables.
	Start/Stop Help Updates	When the lock appears open, each time you click an object in the Designer window, the Online Help information automatically updates.
		When the lock icon appears closed, the Online Help window does not automatically update when you click objects in the Designer window; you must press F1.

System requirements

The Siebel interface allows you to connect Data Integrator with a third-party database under the Siebel application layer. You can import tables (including table and column descriptions) and function metadata. Data Integrator also displays information from both the logical business objects and user interface layers of Siebel applications. This makes it easier to find and import tables.

Siebel Architecture



MS SQL Server is the only database that the Siebel interface supports.

Install the drivers you need to connect MS SQL Server with Data Integrator on the same computers on which you install the Data Integrator Designer and Job Server components. See "System requirements" on page 46 of the *Data Integrator Getting Started Guide* for general information about Windows software and hardware requirements.

The Siebel interface supports Siebel version 7.5.2 and higher compatible versions. It can be installed using the Data Integrator installer. The interface is license-controlled. For more information about licence-controlled interfaces for Data Integrator, see "Optional license-controlled features" on page 63 of the *Data Integrator Getting Started Guide*.

Security

Because Data Integrator connects to Siebel applications through a database connection, Data Integrator can see all Siebel application objects that the provided MS SQL Server database user can access without requiring the application's user login security information.

Datastores

Data Integrator uses datastore connections to link with other applications or databases.

- In a design environment, use a datastore to browse or import metadata that represents external tables and other database objects.
- In a production environment, Data Integrator uses datastore information to move data between source and target databases and applications.

After defining a Siebel datastore in Data Integrator, you can browse tables by type, or drill down on business components or Siebel repositories to find the tables you want to import. You can also import tables, functions, and business components by name.

This section discusses:

- Defining a Siebel datastore
- Browsing and importing metadata
- Extracting data from Siebel applications

Defining a Siebel datastore

With a Siebel datastore connection, you can browse Siebel application metadata.

➢ To define a Siebel datastore

1. In the object library of the Data Integrator Designer, go to the **Datastores** tab
2. Right-click inside the object library window and select **New**.
3. In the Datastore Editor window, enter a name for this datastore (`DS_SiebelApps`, for example).

4. In the **Application type** list, select **Siebel**.
5. Enter the connection information.

   See "Custom application datastores" on page 35 of the *Data Integrator Reference Guide* for MS SQL Server connection options and descriptions.
6. Click **OK**.

   The Siebel datastore appears in the object library.

Browsing and importing metadata

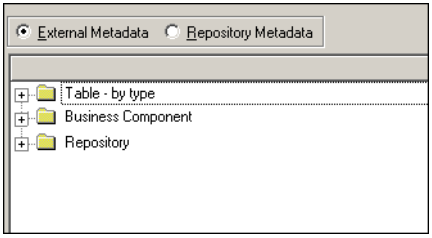
After creating a Siebel datastore, you can browse and import the metadata from the connected application.

NOTE: While you cannot browse functions, you can import them by name.

➢ To browse and import the metadata

1. View the three ways data is grouped and displayed in Siebel. You can:
   - Right-click the Siebel datastore name and select **Open**, or
   - Double-click the datastore name

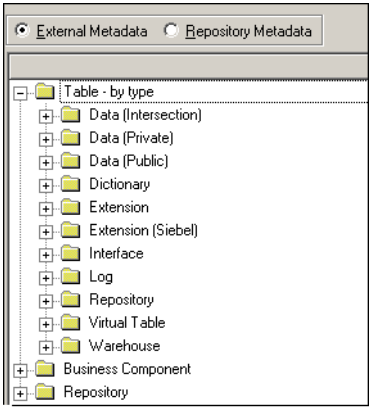
   The workspace displays the list of directories.



2. To find the tables you want to import, expand a table type, business component, or repository folder.

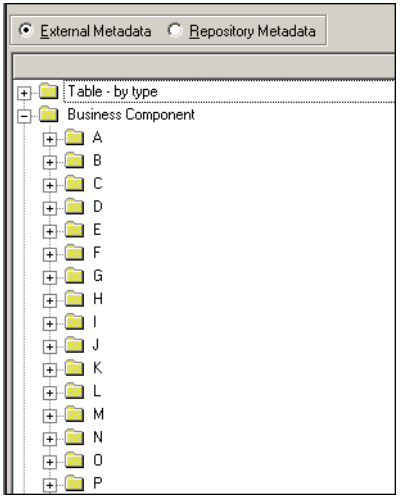
- **Table type**

Siebel applications use many table types. You can browse tables by looking at tables grouped by type.



- **Business component**

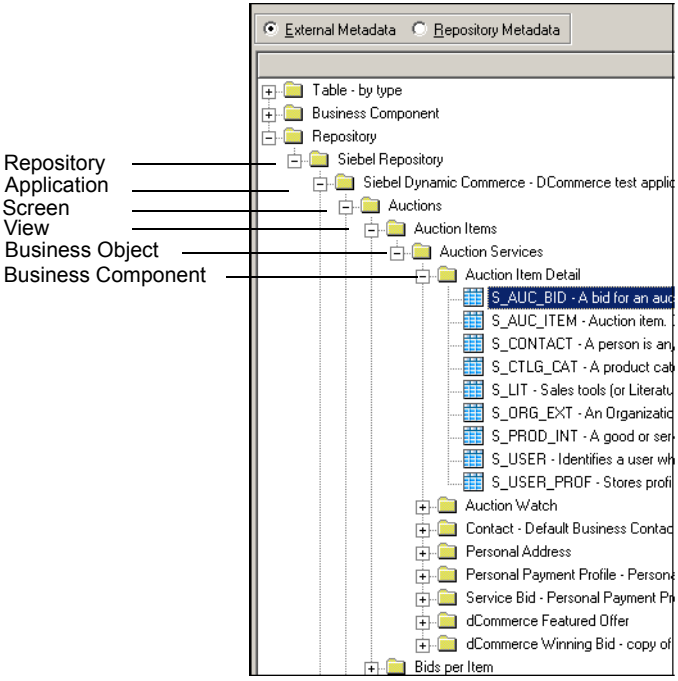
Business components are sets of tables used to create a logical Siebel application object called a business object. Business components are grouped in alphabetical folders.



You can select and import one or more business components or tables.

- **Repository**

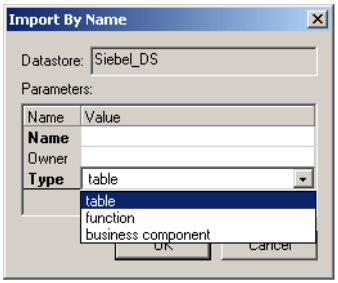
  Siebel repositories hold a variety of Siebel application objects. You can drill through them to find the tables you want to import.



3. To import table data, right-click one or more tables or business components and select **Import**.

   NOTE: When you import a table into Data Integrator via the Siebel interface, the source folder (table hierarchy) is not preserved. All tables are listed at the same level. For example, import all the tables in a business component then add select statements, joins, and mappings using the Designer to create the same associations used in your Siebel application.

You can also import Siebel metadata by right-clicking a Siebel datastore in the object library and selecting **Import by name**.



Once tables are imported, you can view column names, data types, and business names for use in creating jobs.

Schema: S_USER(Siebel_DS.dbo)

	Description	Type	Business_Name
S_USER(Siebel_DS.dbo)	Identifies a user who is able to log into the Sieb...		
ROW_ID	Siebel System Field	varchar(15)	Row Id
CREATED	Siebel System Field	datetime	Created
CREATED_BY	Siebel System Field	varchar(15)	Created By
LAST_UPD	Siebel System Field	datetime	Last Updated
LAST_UPD_BY	Siebel System Field	varchar(15)	Last Updated By
MODIFICATION_NUM	Siebel System Field	decimal(10,0)	Modification Number
CONFLICT_ID	Siebel System Field	varchar(15)	Conflict Id
PAR_ROW_ID	Siebel System Field	varchar(15)	Parent Row Id
LOGIN	SSE Login Name	varchar(50)	Login Name
USER_FLG	User Flag	varchar(1)	User Flag
LAST_LOGIN_TS	Last Time User Logged In	datetime	Last Time User Logged In
PW_LAST_UPD	Password Last Updated	datetime	Password Last Updated
CHALLENGE_ANS...	Correct answer to challenge question	varchar(100)	Challenge Answer
CHALLENGE_QUES...	Question user may be asked to access confide...	varchar(100)	Challenge Question
COMMENTS	Comments	varchar(250)	Comments
CTI_ACD_USERID	The login Id of this employee on the ACD middl...	varchar(15)	CTI ACD Login Id
CTI_PWD	The password of this employee on the ACD mid...	varchar(30)	CTI ACD Login Password
EXCHNG_PROF_NA...	MS Exchange profile name	varchar(254)	MS Exchange profile name
EXCHNG_PROF_P...	MS Exchange profile password	varchar(250)	MS Exchange profile password
EXCHNG_STOREID	MS Exchange Store Id	varchar(250)	MS Exchange Store Id
EXCHNG_SYNC_TY...	MS Exchange synchronization type	varchar(30)	MS Exchange synchronization type
LOGIN_DOMAIN	LOGIN_DOMAIN is reserved for future use by ...	varchar(50)	Login Domain
PASSWORD	Password - not currently used	varchar(30)	Password
RPT_SRVR_PWD	Report Server Password	varchar(30)	Report Server Password

Extracting data from Siebel applications

When extracting data from Siebel applications, Data Integrator processes and converts data types appropriately. For data type conversion details, see "Data type processing" on page 207 of the *Data Integrator Reference Guide*.

A Glossary

This glossary defines terms used in this documentation set including common terms that have a special meaning with respect to Data Integrator. Within a definition, other terms that are defined in the glossary appear in **bold type**.

ABAP

Advanced Business Application Programming.
A fourth-generation programming language developed by SAP in which R/3 applications are written.

ABAP program

A program that executes database operations on an **SAP R/3** server. Data Integrator **R/3 data flows** generate ABAP programs.

Access Server

A process that starts, stops, and monitors real-time services (**real-time jobs**). The Access Server dispatches requests to real-time services, ensuring optimal load balancing and complete life cycle management. The Access Server also starts, stops, and monitors **adapters**.

adapter

An external Data Integrator **interface**. There are two types of adapters:

- Custom adapters — Adapters developed using the Data Integrator Adapter Development Kit
- Prepackaged adapters — Adapters prebuilt and purchased from Business Objects, such as the Data Integrator MQSeries Adapter

Administrator

A browser-based system administration application that supports Data Integrator. Use the Administrator to configure and manage:

- Services (**real-time jobs**)
- Service providers (instances of real-time services)
- Clients (Web application clients)
- Batch jobs
- Adapters
- Job logs

after-image

The values in an UPDATE row after the row changes. You might want to pair before- and after-images of UPDATE rows when designing log-based changed-data capture jobs which Data Integrator supports.

aggregate data

Data that results when a process combines elements. This data can be presented collectively or in summary form.

ALE

Application Link Enabling. An SAP R/3 programming-related interface designed to allow reliable communication across a distributed environment.

Analytic Rapid Mart

An **Data Integrator application** that produces a predefined **data warehouse** for a specific area of business analysis. Analytic Rapid Marts are typically based on dimensional modelling. With Analytic Rapid Marts, users can manage large volumes of data.

attribute

A **property** created for a type of **object**.

BAPI

Business Application Programming Interface. A standardized SAP R/3 programming interface that allows non-SAP R/3 applications to access specific business processes and data.

back-office application

A software system that an enterprise uses to run business operations, such as human resources, general ledger, and finance. Back-office applications store data in a relational **database** optimized for operational use.

There are several types of back-office applications:

- Enterprise resource planning (ERP) — For example, SAP R/3, BAAN, J.D. Edwards, PeopleSoft
- Supply chain management (SCM) — For example, i2
- Customer relationship management (CRM) — For example, Siebel, Clarify
- Legacy

Basis

The SAP infrastructure. Basis is the foundation for all SAP products.

batch job

A group of objects (data flows, work flows, conditionals, scripts, and so forth) that executes on a schedule. This type of job is recommended for operations on large amounts of data.

before-image

The values in an UPDATE row before the row changes. You might want to pair before- and after-images of UPDATE rows when designing log-based changed-data capture jobs which Data Integrator supports.

bulk loading

A software-based mechanism that moves large data files. This mechanism supports compression, blocking, and buffering to optimize transfer times.

business component

A set of tables Siebel applications use to create a logical object called a business object.

Business Objects Data Platform

A set of applications from Business Objects, Americas, that enterprises can use to manage all types of data—from data for supply chain partners to data for customers. The Business Objects Data Platform consists of **Data Integrator** and one or more **Rapid Marts**.

CDC datastore

A CDC datastore allows you to limit extracted data to changed data only. A CDC datastore connects a **changed-data capture** table on a source database to Data Integrator.

CDC checkpoint

A CDC checkpoint enables Data Integrator to restrict CDC subscription reads. After you enable a checkpoint, the next time the CDC job runs, it reads only the rows inserted into the CDC table since the last checkpoint.

CDC Subscription

A CDC subscription name allows different data flows to extract data from the same table without corrupting data extracted by other data flows. The datastore and table name can be the same; however, the table owner name must be different in each source CDC table.

changed-data capture

The process of retrieving changes made to a production data source. This process consolidates units of work, ensures data is synchronized with the original source, and reduces data volume in a warehouse environment.

Citrix MetaFrame XP

Citrix MetaFrame XP software provides an access infrastructure for enterprise applications. You can use this software to run Data Integrator on a server which publishes instances of the Designer and other Data Integrator interfaces to users on client computers.

client/server

A distributed technology approach where the processing is divided by function. The server performs shared functions (such as managing communications and providing database services), while the client performs individual user functions.

COM

Component Object Model. Provides a set of interfaces that allow clients and servers to communicate within the same computer (running on Windows 95 or later versions).

Commerce Rapid Mart

An **Data Integrator application** that produces a predefined **data warehouse** and that includes logic to support real-time requests for data from a **back-office application**. A Commerce Rapid Mart is usually based on different data model than an **Analytic Rapid Mart**. With a Commerce Rapid Mart, users can integrate a company's back-office system with an **e-commerce** application.

Communication Structure

In **SAP BW**, a data structure that defines a set of **InfoObject**s available from an **InfoSource** to put into **InfoCube**s.

conditional

A **single-use object**, available in **work flow**s**,** that allows you to branch the execution logic based on the results of an expression. The conditional takes the form of an if/then/else statement.

custom ABAP program

A custom ABAP program runs an **ABAP program** and generates a **data set**. With a custom ABAP program, you can run an existing ABAP program as part of a job. Use a custom ABAP program as a **source** in a **data flow** or an **R/3 data flow**.

data extraction

The process of moving data from a **database** or application source to a database target (either from a legacy database to a **data mart**, or from one data mart to another).

data flow

A **reusable object** containing **steps** to define the transformation of data from **source** to **target**. Data flows are called from inside a **work flow** or **job**. You can pass information into or out of data flows using **parameters**.

Data Integrator

A software system that allows users to build and execute applications with which they can create and maintain **data warehouses**.

Data Integrator consists of several components:

- **Access Server**
- **Data Integrator engine**
- **Data Integrator repository**
- **Data Integrator Service**
- **Administrator**
- **Designer**
- **Job Server**

Data Integrator application

A set of **jobs** that process batch data extractions and real-time requests using data from a variety of data sources.

Data Integrator engine

The core process that reads job information from the **Data Integrator repository** and sets up run-time processes that execute the job. The run-time processes extract, transform, and load relational and hierarchical data. The **Job Server** starts the Data Integrator engine to execute batch or real-time jobs.

Data Integrator interface

A program that Data Integrator uses to access data sources. Specific interfaces vary by installation. There are internal interfaces—those native to the installation—and external interfaces—those that you install separately. Internal interfaces allow Data Integrator to access applications like **SAP R/3** and **SAP BW**, **messages**, relational database systems, and **legacy systems**. An external interface is also known as an **adapter**. It allows Data Integrator to access applications using information exchange technologies such as Web Services and MQ Series.

Data Integrator repository

The database that contains information about an Data Integrator application. The repository contains information about defined **reusable objects**, the **metadata** for tables, and built-in **transforms** and **functions**. When you invoke Data Integrator, you log in to the repository containing the **objects** you want to use.

Data Integrator Service

The process that ensures that the **Access Server** and the **Job Server** are running. You can configure the Data Integrator Service to restart the Access Server and Job Server whenever the computer where they are located restarts.

data loading

The process of populating a **data warehouse**. Data loading is provided by **DBMS**-specific load processes, DBMS insert processes, and independent fast-load processes.

data mapping

The process of assigning a source data element to a target data element.

data mart

A highly-focused version of a **data warehouse**. Typically, created by a department or division of a company, data marts contain data for a specific subject area, such as finance or sales. Data Integrator can populate a data mart.

data transformation

The process of filtering, merging, decoding, and translating source data to create validated data for a **data warehouse**.

data set

Rows of data with a defined schema. A step in a **data flow**—such as reading data from a **source**, joining data in a **Query transform**, or transforming data though another **transform**—yields a data set. You can view individual data sets by placing a **target** table or file at that point in the **data flow**.

data transport

A step in an **R/3 data flow** that defines a **target** to store the **data set** extracted during the flow. You can locate the target file on the **SAP R/3** server or in a location accessible to both the SAP R/3 server and to Data Integrator across a network.

data type

The format used to store a value. Data types can imply a default format for displaying and entering the value. Data read from a **source** is converted to the appropriate Data Integrator data types; data loaded to a **target** is converted from its Data Integrator data type to the type appropriate for the target.

data warehouse

A relational **database** containing subject-oriented information that supports decision analysis. Typically, information in a data warehouse is integrated from multiple sources, including operational sources. Data warehouses provide a single, non-volatile point of reference for a particular time period. Data warehouses are composed of **data marts.** Data Integrator can populate a data warehouse.

database

A collection of **table**s managed by a **DBMS** such as Microsoft SQL Server or Oracle.

DataConnector

DataConnector operator instances are used to read data files generated by Data Integrator when performing bulk loading using the Teradata Warehouse Builder.

datastore

A logical channel connecting Data Integrator to a **database**. The datastore definition includes the name and location of the database as well as user authentication information. Data Integrator uses a datastore definition to qualify a table name wherever a table is indicated in a diagram or expression. You can access the datastore definition through the **object library**.

DBMS

Database management system. A software system that builds and maintains database tables.

debug mode

Allows you to debug a job using the **interactive debugger** features in the Designer.

declarative specification

A way of indicating the desired results of a **data transformation** without placing constraints on the method of reaching the results. When you create a **data flow** in Data Integrator, you specify what data you want to read from which **source**, what you want done to that data, and where you want it loaded. What you specify constitutes the declarative specification. Data Integrator then determines the most efficient way to reach the requirements you set, including pushing operations to the source **DBMS** for execution.

degree of parallelism

You can set a value at the data flow level that allows you to parallel-process a **transform** within a data flow.

For example, if from the Properties window of a data flow you set the **Degree of parallelism** to 5, then when the job executes, Data Integrator replicates each transform in the data flow five times. Each of these replicated transforms executes in parallel using a separate process thread. Using a value for degree of parallelism can improve job performance if a **Job Server**'s computer memory and number of CPUs supports a job's parallel-processing configuration settings.

dependency analysis

A category of metadata reports that produces reports based on search criteria you specify to find specific objects in the repository. For each object returned, dependency analysis reports provide object lineage as well as object impact information.

derived data

Data that results from a computational step applied to reference or event data. Derived data is the result of either relating two or more elements of a single transaction (such as aggregation), or relating one or more elements of a transaction to an external algorithm or rule.

Designer

A graphical user interface that allows you to design and test an **Data Integrator application**.

diagram

The icons and connections between the icons that make up the definition of a **job**, **work flow**, or **data flow**.

DMZ

Demilitarized zone. A perimeter network that is more accessible than your most protected network, but located inside a security layer protected from the internet.

drill down

A method of exploring detailed data that was used in creating a summary level of data. Drill down levels depend on the granularity of the data in the **data warehouse**.

DTD

Document type definition. A text file that describes the elements (tags) in an **XML** document and the relationship among them. When an XML document is used to describe a transaction, the DTD describes the data schema used in the transaction.

e-commerce

Buy and sell transactions conducted over the internet.

EDI

Electronic Data Interchange. Electronic exchange of structured data between businesses. This exchange is not dependent on hardware, software, or communication protocols.

element

A component found within XML Schemas and DTDs.

embedded data flow

A data flow created within one or more data flows.

enterprise data

Data that is defined for use across a corporate environment.

ERP system

Enterprise resource planning system. A **back-office application** from which Data Integrator can extract data.

exception

An error that occurs while executing a **job**. You can catch individual or groups of exceptions using a **try/catch block** inside a **work flow**. Catching an exception allows you to automatically execute a solution for the error.

Executive Information Systems (EIS)

Tools programmed to provide canned reports or briefing books to executives. They offer strong reporting and drill-down capabilities. These tools allow ad hoc querying against a multi-dimensional **database**, and most offer analytical applications along functional lines such as sales or financial analysis.

extract date

The date data was extracted.

extract frequency

The interval at which data is extracted, such as daily, weekly, monthly, or quarterly. The frequency that data extracts are needed in the **data warehouse** is determined by the shortest frequency requested through an order, or by the frequency required to maintain consistency of the other associated data types in the source data warehouse.

file

A **source** or **target** for data. To read or load data to a file within Data Integrator, define a **file format** that instructs Data Integrator about the structure of the data in the file.

file format

A description of how data is or should be organized in a **file** Data Integrator reads from or loads to. A file format can be specific to a single file or generic for many files.

front-office application

Software that companies use to support reporting, **e-commerce**, and other tasks. Types of front-office applications include:

- Supply chain collaboration
- Analytical
- Sell-side e-commerce
- Customer relationship management

function

A program that operates on values that are passed to it. Data Integrator functions are available through a function wizard in a **script**, **conditional**, or **Query transform**. Data Integrator also gives you access to functions provided by the **DBMS** you are using. In addition, you can define your own functions using the Data Integrator scripting language or by writing C-language programs.

IDoc

Intermediate Document. An SAP-specific format. Used for **EDI** and **ALE**.

IDoc type

Indicates the SAP format that is used to interpret the data of a business transaction. Consists of the following components:

- A control record

  Identical for each IDoc type.

- Several data records

  A single data record consists of a fixed key part and a variable data part. The data part is interpreted using segments, which differ depending on the IDoc type selected.

- Several status records

  Identical for each IDoc type. Describe the status states an IDoc has already passed through or the status an IDoc has attained.

import

The process of acquiring information for the **Data Integrator repository**. Import the following kinds of information into Data Integrator:

- The **metadata** for **source** and **target** databases
- Descriptions and code for user-defined and DBMS **functions** and **transforms**

InfoArea

In **SAP BW**, an arbitrary grouping of **InfoCube**s.

InfoCube

In **SAP BW**, a multidimensional star schema that is limited to one fact table and 16 dimensions.

InfoCubes store only numeric data at the fact level.

SAP BW reserves three of the dimensions for currencies, time, and units of measure. Dimensions have to be linked back to master data and hierarchy tables to gain access to the information stored there.

When a user query is initiated, BW creates a temporary **OLAP** cube that stores a constrained subset of the permanent OLAP cube. The temporary InfoCube persists until the user disconnects from BW or changes the query.

InfoObject

In **SAP BW**, an InfoObject is equivalent to a data element and its associated metadata (for example, description or business rules). Can be a **Key Figure** or a Characteristic.

InfoPackage

In **SAP BW**, a schedule for running BW updates.

Information resource (IR)

A producer or consumer of enterprise information. An information resource can be a flat file, a **back-office application**, an EAI system, or a Web application.

InfoSource

In **SAP BW**, a logical construct that defines a set of **InfoObjects** and their mappings to source data elements. InfoSources do not store data, but rather serve as conduits via **Transfer Structures**.

Integration Rapid Mart

An **Data Integrator application** that provides point-to-point content mapping and data movement mechanisms between different **back-office applications**.

interactive debugger

A Designer feature that allows you to step through the data of a job one row at a time using filters and breakpoints on a line. Like executing a job, you can start the interactive debugger from the Debug menu when a job is active in the workspace. While in debug mode, all other Designer features are set to read-only.

interface

Data Integrator offers two types of interfaces:

An internal **Data Integrator interface**, allows you to create datastore connections to natively supported applications.

An external **Data Integrator interface** (or **adapter**), allows Data Integrator to communicate with information exchange technologies such as Web Services and MQ Series.

IR

See **Information resource (IR)**.

job

The unit of work that can be scheduled independently for execution by the **Administrator**. Jobs are special **work flows** that can be called by other work flows or jobs.

Job Server

A process that receives requests from the **Designer** and the **Administrator** to start and stop **jobs**. To start batch or real-time jobs, the Job Server triggers the **Data Integrator engine**. Engine processes run on the same computer as the Job Server process that triggers them.

join rank

A value given to or calculated for all data sets in a **data flow**. Data Integrator uses the join rank to determine which **source** to read first when assembling the **data set** in a join. Data Integrator uses the source with the lower join rank as the inner source of the join and uses the source with the higher join rank as the outer source of the join. If more than one of the sources is located in the same **DBMS**, the highest rank is used for all of the sources from that DBMS. When a source is processed through a **Query transform** or another **transform**, and then used in a join, the maximum join rank defined for the source is used as the join rank in the query.

Key Figure

In **SAP BW**, an **InfoObject** that represents a numeric fact.

legacy system

An information or transaction processing system used to store data such as bank balances, inventories, payroll, and manufacturing parts.

license-controlled feature

A Data Integrator feature that is enabled or disabled based on the product license. The product license controls which icons and settings are available in Data Integrator as an internal **Data Integrator interface**.

line

The line you use to connect objects in a workspace diagram. Lines represent a left-to-right flow path for data moving between two objects during job execution.

memory datastore

A datastore connection/container for memory tables.

memory table

Internal Data Integrator table used to store a data set in memory while a job runs. Use instead of staging tables to improve performance of a real-time job built with multiple data flows. Use a memory table to move a data set between data flows.

message

Represents hierarchical data, such as a header with line items, for document-oriented transactions, such as a purchase order.

metadata

In Data Integrator, information acquired and maintained to describe tables in **source** and **target** databases. This information includes the names of tables and their columns, and the data types of the columns. All the metadata for all user tables is imported at one time for a given **datastore**.

In general, metadata typically includes a description of data models, a description of the layouts used in database design, the definition of the system of record, the mapping of data from the system of record to other places in the environment, and specific database design definitions.

multiple data flow model

One of three data flow models you can use in real-time jobs. Use this model to develop real-time jobs in separate logical object segments (data flows) rather than in a single data flow. Allows for any number of data flows in a real-time processing loop under these conditions: The first data flow must have a single message source, the last data flow must have a single message target. Other data flows in the same real-time job cannot have a message source or message target. See also **put/acknowledge data flow model** and **single data flow model**.

nested data

Data in one **table** that is related to a single row of another table. A nested table appears in Data Integrator as a column in a parent table. Columns in the nested table can themselves contain tables.

object

Any item that you create in the **Designer**. Data Integrator distinguishes two classes of objects: **reusable objects** that are complete and can be reused in your projects (such as **data flows**) and **single-use objects** that only appear as components of other objects (such as a **try/catch block**). This distinction affects how you create and retrieve each type of object.

object definition

The options that describe the operation of an **object**. To view and modify an object definition, open the object so that its definition appears in the **workspace**.

object library

A tool in the **Designer** that gives you access to reusable objects.

ODBC

Open database connectivity. A standard for **database** access initiated by the SQL Access Group consortium and now owned by Microsoft.

ODS

Operational data store. An OLAP-designed relational **database** that an enterprise has designated as the operational database of record (for example, a finance department might use an ODS to close its books).

OLAP

Online analytical processing. A relational **database** design that is optimized for random access. OLAP systems are used in a query environment, such as for a **front-office application**.

OLTP

Online transaction processing. A relational database design optimized for operational use. OLTP systems are used in an operational environment, such as for a **back-office application**.

operation code

A flag associated with a row in a **data set** that indicates the status of the data in the row. The operation codes are INSERT, UPDATE, DELETE, and NORMAL.

operational statistics

A category of reports within the metadata reporting tool that provides you with job and data flow performance information over a given period of time or number of executions.

parameter

A value passed to a **work flow** or **data flow** when that flow is called.

partition

To divide table data into sets based on a criteria such as a range or list of values in each row. You can configure Data Integrator to read and write partitioned table data in parallel threads. Designing jobs with partitioned table data can improve job performance if a Job Server's computer memory and number of CPUs supports the job's parallel-processing configuration settings.

project

The collection of **jobs** available in the **Designer** at a given time. A project provides a way to organize the **objects** you create.

property

Detailed descriptive information about **objects**.

put/acknowledge data flow model

One of three data flow models you can use in real-time jobs. Similar to the multiple data flow model, this model does not require a data flow with a target message. Data Integrator creates a target message to automatically acknowledge a request message. Designed for use with IDoc sources in SAP R/3 systems. See also **multiple data flow model** and **single data flow model**.

Query transform

A **data transformation object** that you can use to map columns from a **source** to a **target** schema, add new columns to the target schema, determine the data to extract, and perform operations on the data. Like a SQL SELECT statement, a query creates a **data set** that satisfies the conditions you specify.

Rapid Mart

A prepackaged **Data Integrator application** that delivers data for a specific business purpose. Easily deployed and customized, BusinessObjects™ Rapid Marts include the extraction, transformation, and data loading logic to support analysis or **e-commerce**. There are three types of Rapid Mart:

- **Analytic Rapid Mart**
- **Commerce Rapid Mart**
- **Integration Rapid Mart**

R/3 data flow

A **data flow** that extracts data from an **SAP R/3** source table. Data Integrator translates steps you define in an R/3 data flow into **ABAP** and then passes the ABAP program back to your SAP R/3 system for execution. The resulting **table** or **file** resides on the SAP R/3 system to be used as a **source** in the parent data flow.

real-time job

A group of objects (data flows, work flows, conditionals, scripts, and so forth) that execute on-demand as a "request-response" system. You design real-time jobs in the **Designer**, then configure them as real-time services and associate them with an **Access Server** in the **Administrator**, where they are started, managed and monitored. When a real-time service receives a request from a caller, it processes the request and returns a reply. This type of job is recommended for operations on small amounts of data.

relational data

A **data set** in which data in each column contains a scalar value. Data Integrator can process relational data; it can also process **nested data**.

repository

See **Data Integrator repository**.

reusable object

An **object** that can be defined, stored, and reused independent of other objects. Create reusable objects from the **object library**.

RFC server

Remote Function Call server. The Data Integrator RFC server allows third-party programs, including SAP R/3, to schedule and initiate Data Integrator jobs and return the results to Data Integrator.

SAP BW

Business Information Warehouse, an SAP product.

SAP R/3

An **ERP system**.

script

A step in a **work flow** that allows you to calculate values to pass to other parts of the work flow. The script can call **function**s, execute if-then-else statements, and assign values to **variable**s. Write a script in the Data Integrator scripting language.

segment

Format with which the data records of **IDocs** are interpreted.

server group

A defined collection of Job Servers on different computers. A server group automatically measures resource availability on each Job Server in the group and distributes scheduled batch jobs to the Job Server with the lightest load at run-time. Use the Server Groups node in the Administrator's navigation tree to group Job Servers that are associated with the same repository into a server group.

service request

Any message sent from a Web client that requires processing by a **real-time job**.

session

Associated with the **Adapter Properties** tab in an adapter datastore, a session is created by a stream-oriented adapter to group adapter components and maintain state for interactivity between Data Integrator and the adapter.

single data flow model

One of three data flow models you can use in real-time jobs. Allows for one data flow in a real-time processing loop. This data flow must have a single message source and a single message target. See also **multiple data flow model** and **put/acknowledge data flow model**.

single-use object

A step in a **work flow** or **data flow** that cannot be saved independently of the flow. Create single-use objects from the tool palette.

smart editor

A flexible editing tool in Data Integrator used for creating scripts, expressions, and custom functions without having to type the names of existing elements like column, function, and variable names.

SNMP

System Network Management Protocol helps network administrators manage network routing hardware. The protocol can manage a variety of hardware and software devices.

source

An **object** from which Data Integrator reads data.

SQL

Structured Query Language. A query language for accessing relational, **ODBC**, DRDA, or non-relational **database** systems.

SQL query tool

An end-user tool that accepts **SQL** to be processed against one or more relational databases.

star schema

A **database** design you can use to format data in a **data mart**. This design is based on a single fact table to which any number of dimensional tables may be joined. This type of database design supports multi-dimensional database analysis.

step

An **object** that is part of the definition of a **work flow** or **data flow**. Each step is represented by an icon in the diagram of the flow and is connected to other steps to indicate the flow of data through the data flow or the order of execution in the work flow.

table

A **database** table that Data Integrator reads data from or loads data into. The path and mechanisms for reading and loading data and apportioning the data among rows and columns are defined in the **datastore** that the table is associated with. Writing a **data set** to a database table means sending a combination of rows with appropriate **operation code** to the database table.

target

An object in which Data Integrator loads extracted and transformed data in a **data flow**. Data Integrator only loads rows flagged as INSERT or UPDATE.

TCP/IP

Transmission Control Protocol/Internet Protocol. The basic communication protocol of the internet, and often intranets and extranets. A computer having direct access to the internet contains a copy of the TCP/IP program. TCP/IP makes it possible for computers to communicate with each other.

Tdpid

(Teradata Director Program ID) The server name Data Integrator uses when loading with the bulk loader option. Data Integrator uses tdpid as a Teradata Warehouse Builder operator attribute.

Transfer Structure

In **SAP BW**, a data structure that defines a set of **InfoObjects** to be moved into BW. To an ETL tool like Data Integrator, a Transfer Structure looks like a table.

transform

A step in a **data flow** that acts on a **data set**. Data Integrator transforms are available through the **object library**. You can also import user-defined transforms and transforms from your **DBMS**.

Transformation Rule (Transfer Rule)

In **SAP BW**, a data manipulation rule between a **Transfer Structure** and a **Communication Structure**. The results of these transformation are globally available to all **InfoCubes**.

try/catch block

A combination of a try object and one or more catch objects that define alternate execution paths in case an error occurs during the execution of a **job**. You can tune try/catch blocks to trap specific errors and to provide general alerts or messages if an error occurs.

Update Rule

In **SAP BW**, a data manipulation rule between a **Communication Structure** and an **InfoCube**. The result of the data manipulation is specific to one InfoCube.

variable

A symbolic placeholder for a value. Data Integrator lets you define local variables and global variables.

Local variables pertain to the **work flow** in which they are defined. You can pass the value into another **work flow** or **data flow** using a **parameter**.

Global variables pertain to the **job** in which they are defined. With global variables, there is no need to define parameters between **objects** in the job. Global variables can also be selected at execution time. This eliminates the need to open the Designer to set global variable values.

Web Services

A standard platform for integrating applications. Web Services allow different programs, constructed in different languages, on different platforms to communicate with each other. DATA INTEGRATOR supports three XML web service technologies: SOAP, WSDL, and **XML Schema**.

work flow

A **reusable object** containing **steps** to define the order of **job** execution. Work flows call **data flow**s, but cannot manipulate data themselves.

You call work flows from inside other work flows or **job**s. You pass information into or out of work flows using **parameter**s. You reuse work flows by dragging existing work flows from the **object library**.

workspace

The window inside the **Designer** in which you define, display, and modify **objects**. The workspace for a **data flow** contains an area to build a **diagram** representing the data flow definition. The workspace for a **transform** contains an editor for modifying the transform options.

WSDL

Web Services Definition Language. Web services are self-contained, modular business process applications based on open Internet standards.

XML

Extensible Markup Language. This markup language is like HTML (Hypertext Markup Language) in that it specifies a standard with which you can define your own markup languages with their own sets of tags. XML allows you to define various tags with various rules, such as tags that represent business rules, tags that represent data description, or tags that represent data relationships.

XML Schema

The XML format used by Data Integrator to support message processing that includes **Web Services**. XML Schemas describe the data structure of an **XML** file or message. Data flows can read and write data to messages or files based on a specified XML Schema format. You can use the same XML Schema to describe multiple XML sources or targets. XML Schema properties include: Name, Description, Imported from, Root element name, and Namespace.

Index

Business Objects, Americas
3030 Orchard Parkway
San Jose, CA 95134
USA

info@businessobjects.com
www.businessobjects.com

Master Index

B

C

D

E

F

G

H

I

J

K

L

Q

R

T

W

X

Y

Z

Business Objects, Americas
3030 Orchard Parkway
San Jose, CA 95134
USA

info@businessobjects.com
www.businessobjects.com