Interaction Concentrator 7.6

User's Guide

About Genesys

Genesys Telecommunications Laboratories, Inc., a subsidiary of Alcatel-Lucent, is 100% focused on software for call centers. Genesys recognizes that better interactions drive better business and build company reputations. Customer service solutions from Genesys deliver on this promise for Global 2000 enterprises, government organizations, and telecommunications service providers across 80 countries, directing more than 100 million customer interactions every day. Sophisticated routing and reporting across voice, e-mail, and Web channels ensure that customers are quickly connected to the best available resource—the first time. Genesys offers solutions for customer service, help desks, order desks, collections, outbound telesales and service, and workforce management. Visit www.genesyslab.com for more information.

Each product has its own documentation for online viewing at the Genesys Technical Support website or on the Documentation Library DVD, which is available from Genesys upon request. For more information, contact your sales representative.

Notice

Although reasonable effort is made to ensure that the information in this document is complete and accurate at the time of release, Genesys Telecommunications Laboratories, Inc., cannot assume responsibility for any existing errors. Changes and/or corrections to the information contained in this document may be incorporated in future versions.

Your Responsibility for Your System's Security

You are responsible for the security of your system. Product administration to prevent unauthorized use is your responsibility. Your system administrator should read all documents provided with this product to fully understand the features available that reduce your risk of incurring charges for unlicensed use of Genesys products.

Trademarks

Genesys, the Genesys logo, and T-Server are registered trademarks of Genesys Telecommunications Laboratories, Inc. All other trademarks and trade names referred to in this document are the property of other companies. The Crystal monospace font is used by permission of Software Renovation Corporation, www.SoftwareRenovation.com.

Technical Support from VARs

If you have purchased support from a value-added reseller (VAR), please contact the VAR for technical support.

Technical Support from Genesys

If you have purchased support directly from Genesys, please contact Genesys Technical Support at the following regional numbers:

Region	Telephone	E-Mail
North and Latin America	+888-369-5555 or +506-674-6767	support@genesyslab.com
Europe, Middle East, and Africa	+44-(0)-1276-45-7002	support@genesyslab.co.uk
Asia Pacific	+61-7-3368-6868	support@genesyslab.com.au
Japan	+81-3-6361-8950	support@genesyslab.co.jp

Prior to contacting technical support, please refer to the *Genesys Technical Support Guide* for complete contact information and procedures.

Ordering and Licensing Information

Complete information on ordering and licensing Genesys products can be found in the ***Genesys 7 Licensing Guide***.

Released by

Genesys Telecommunications Laboratories, Inc. www.genesyslab.com

Document Version: 76icon_us_10-2008_v7.6.101.00

Table of Contents

Preface

Welcome to the *Interaction Concentrator 7.6 User's Guide*. This document provides an overview of the basic Interaction Concentrator architecture and detailed information about Interaction Concentrator features and functionality. It also provides users with the information they need to know to perform ongoing maintenance and administration of Interaction Concentrator 7.6.

This document is valid only for the 7.6 release(s) of this product.

This preface provides an overview of this document, identifies the primary audience, introduces document conventions, and lists related reference information. It contains the following sections:

Interaction Concentrator collects and stores detailed data about the interactions and resources in customer interaction networks that use Genesys Framework (contact center, enterprise-wide, or multi-enterprise telephony and computer networks). Downstream reporting systems can access Interaction Concentrator data in near real time.

Intended Audience

This document, which is intended primarily for system administrators and database architects, assumes that you have a basic understanding of:

- Computer-telephony integration (CTI) concepts, processes, terminology, and applications.
- Network design and operation.
- Database design and operation.
- Your own network configurations.

You should also be familiar with:

- Genesys Framework architecture and functions.
- Genesys solutions deployed in your contact center.
- Your real-time and historical reporting objectives.

Chapter Summaries

To distinguish between information required to control Interaction Concentrator operations and conceptual information describing its features and functionality, this document is divided into two main parts.

Part One—Using Interaction Concentrator Data

Part One of this document, Using Interaction Concentrator Data, consists of Chapters 1 through 8. These chapters contain detailed information about Interaction Concentrator data, features, and functionality. They also provide a high-level overview of the basic architecture and components of Interaction Concentrator.

Although you must refer to Part One if you have never before maintained ICON or configured some of its features, you might also refer to the chapters in Part One even if you are already familiar with Interaction Concentrator to discover any changes to feature and functionality since you last used this product.

In addition to this preface, Part One of this guide contains the following chapters:

- Chapter 1, "Product Overview," on page 19, describes the basic architecture and components of Interaction Concentrator. It also provides a high-level overview of Interaction Concentrator functionality.
- Chapter 2, "Introducing IDB Schema," on page 27, provides a high-level description of Interaction Database (IDB) architecture. It also provides a summary of the precalculated call and party metrics that are available in the statistical tables.
- Chapter 3, "Integrating with Genesys Multimedia and Open Media," on page 37, describes how to make data about Multimedia and Open Media interactions available in IDB.
- Chapter 4, "Processing Attached Data," on page 47, describes how to make user data attached to voice calls and Multimedia interactions available in IDB. It provides the Extensible Markup Language (XML) schema definition for processing key-value pairs (KVPs) from the attached data that T-Server and Interaction Server provide with their event reporting. It also provides samples of the required attached data specification.
- Chapter 5, "Monitoring Virtual Queues and Routing Points," on page 63, describes how Interaction Concentrator monitors virtual queues and routing points, and stores this routing information in the IDB.

- Chapter 6, "Agent States and Login Sessions," on page 71, describes the agent state and login session models used in Interaction Concentrator. It also describes how to make agent state and login session data available in IDB.
- Chapter 7, "Integrating with Outbound Contact," on page 81, describes how to make data from the Genesys Outbound Contact solution available in IDB.
- Chapter 8, "Processing User Events and Custom-Defined States," on page 93, describes how to make data about customer-defined agent states, as well as common data attached to voice calls, available in IDB.

Part Two—Administering Interaction Concentrator

Part Two of this document, Administering Interaction Concentrator, consists of Chapters 9 through 15. It contains information all users need to know to perform ongoing maintenance and administration of Interaction Concentrator—for example, how to start, stop, and monitor ICON, and how to configure high availability (HA) and other features such as data filtering. It also contains need to know information about troubleshooting startup and runtime problems and how to resynchronize your IDB on demand.

Part Two of this guide contains the following chapters:

- Chapter 9, "Starting and Stopping Interaction Concentrator," on page 99, describes the prerequisites for Interaction Concentrator startup and provides instructions for starting and stopping ICON.
- Chapter 10, "Monitoring Interaction Concentrator," on page 109, describes how to access the HTTP Listener to monitor and report on Interaction Concentrator performance.
- Chapter 11, "Filtering IDB Data," on page 111, describes the data filtering capability of Interaction Concentrator 7.6 and how to configure it by setting options.
- Chapter 12, "Implementing HA in Interaction Concentrator," on page 119, describes the design and implementation of HA in Interaction Concentrator.
- Chapter 13, "Resynchronizing Configuration Changes," on page 131, describes the conditions under which you might consider invoking a forced resynchronization of IDB data and how to run that resynchronization.
- Chapter 14, "Using Special Stored Procedures," on page 145, describes the stored procedures that are of special relevance to ICON users, including instructions on how to set up and execute them.
- Chapter 15, "Troubleshooting ICON Installation and Deployments," on page 175, provides solutions for the most common Interaction Concentrator problems encountered during startup and runtime.

Document Conventions

This document uses certain stylistic and typographical conventions—introduced here—that serve as shorthands for particular kinds of information.

Document Version Number

A version number appears at the bottom of the inside front cover of this document. Version numbers change as new information is added to this document. Here is a sample version number:

76icon_dep_05-2008_v7.6.001.00

You will need this number when you are talking with Genesys Technical Support about this product.

Type Styles

Italic

In this document, italic is used for emphasis, for documents' titles, for definitions of (or first references to) unfamiliar terms, and for mathematical variables.

Examples:

- Please consult the *Genesys 7 Migration Guide* for more information.
- *A customary and usual practice* is one that is widely accepted and used within a particular industry or profession.
- Do *not* use this value for this option.
- The formula, $x + 1 = 7$ where x stands for . . .

Monospace Font

A monospace font, which looks like `teletype or typewriter text`, is used for all programming identifiers and GUI elements.

This convention includes the *names* of directories, files, folders, configuration objects, paths, scripts, dialog boxes, options, fields, text and list boxes, operational modes, all buttons (including radio buttons), check boxes, commands, tabs, CTI events, and error messages; the values of options; logical arguments and command syntax; and code samples.

Examples:

- Select the `Show variables on screen` check box.
- Click the `Summation` button.
- In the `Properties` dialog box, enter the value for the host server in your environment.
- In the `Operand` text box, enter your formula.
- Click `OK` to exit the `Properties` dialog box.

- The following table presents the complete set of error messages T-Server® distributes in `EventError` events.
- If you select true for the `inbound-bsns-calls` option, all established inbound calls on a local agent are considered business calls.

Monospace is also used for any text that users must manually enter during a configuration or installation procedure, or on a command line:

Example:

- Enter `exit` on the command line.

Screen Captures Used in This Document

Screen captures from the product GUI (graphical user interface), as used in this document, may sometimes contain a minor spelling, capitalization, or grammatical error. The text accompanying and explaining the screen captures corrects such errors *except* when such a correction would prevent you from installing, configuring, or successfully using the product. For example, if the name of an option contains a usage error, the name would be presented exactly as it appears in the product GUI; the error would not be corrected in any accompanying text.

Square Brackets

Square brackets indicate that a particular parameter or value is optional within a logical argument, a command, or some programming syntax. That is, the parameter's or value's presence is not required to resolve the argument, command, or block of code. The user decides whether to include this optional information. Here is a sample:

```
smcp_server -host [/flags]
```

Angle Brackets

Angle brackets indicate a placeholder for a value that the user must specify. This might be a DN or port number specific to your enterprise. Here is a sample:

```
smcp_server -host <confighost>
```

Related Resources

Consult the following additional resources as necessary:

- The *Interaction Concentrator 7.6 Deployment Guide,* which will help you install and configure Interaction Concentrator.
- The *Interaction Concentrator 7.6 Physical Data Model* for your relational database management system (RDBMS) type, which will help you learn about IDB tables.

- The *Interaction Concentrator 7.6 Database Size Estimator,* which will help you estimate the size of your IDB when you are planning your deployment. The estimator is a Microsoft Excel spreadsheet available from the Genesys Technical Support website.
- The *Genesys 7 Hardware Sizing Guide,* which contains information about recommended hardware architectures and additional information related to database size estimation.
- *Genesys 7.6 Combined Log Events Help,* which describes the log events generated by every Genesys server application, including Interaction Concentrator.
- The documentation set for Genesys Info Mart release 7.6, if you intend to use Interaction Concentrator as a source of data for Genesys Info Mart.
- The documentation set for Genesys Outbound Contact release 7.6, if you intend to store outbound-related data in IDB.
- The documentation set for Genesys Universal Routing release 7.6, if you intend to store information about virtual queue usage in interaction processing in IDB.
- The documentation set for Genesys Multimedia release 7.6, if you intend to store interaction-related and agent-related data about Multimedia interactions in IDB.
- The *Genesys Technical Publications Glossary,* which ships on the Genesys Documentation Library DVD and which provides a comprehensive list of the Genesys and CTI terminology and acronyms used in this document.
- The *Genesys 7 Migration Guide,* also on the Genesys Documentation Library DVD, which provides a documented migration strategy from Genesys product releases 6.x and higher to all Genesys 7.x releases. Contact Genesys Technical Support for additional information.
- The Release Notes and Product Advisories for this product, which are available on the Genesys Technical Support website at http://genesyslab.com/support.

Information about supported hardware and third-party software is available on the Genesys Technical Support website in the following documents:

- *Genesys 7 Supported Operating Systems and Databases*
- *Genesys 7 Supported Media Interfaces*

Genesys product documentation is available on the:

- Genesys Technical Support website at http://genesyslab.com/support.
- Genesys Documentation Library DVD, which you can order by e-mail from Genesys Order Management at orderman@genesyslab.com.

Making Comments on This Document

If you especially like or dislike anything about this document, please feel free to e-mail your comments to Techpubs.webadmin@genesyslab.com.

You can comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this document. Please limit your comments to the information in this document only and to the way in which the information is presented. Speak to Genesys Technical Support if you have suggestions about the product itself.

When you send us comments, you grant Genesys a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

Document Change History

This section lists topics that are new in the current release of this document, or that have changed significantly from the preceding release.

New in Release 7.6.1

This User's Guide includes the following new features for the 7.6.1 release of Interaction Concentrator:

- An enhancement to support a large number of concurrently active Multimedia interactions. See "Handling Active Multimedia Interactions" on page 40.
- A purging mechanism for Multimedia data stored in IDB. See "Purging Voice and Multimedia Interaction Data" on page 157.
- Improved performance of the purging procedure for voice data stored in IDB. See "Optimizes Performance" on page 158.
- The ability to filter out some Multimedia data that is not relevant for reporting in order to save IDB storage space and improve overall performance. See "Strategy Activity Data" on page 116.

Part

1 Part One: Using Interaction Concentrator Data

Part One of this document provides detailed information about Interaction Concentrator data, features, and functionality. It also provides a high-level overview of the basic architecture and components of Interaction Concentrator.

This information appears in the following chapters:

Chapter

1 Product Overview

This chapter describes the basic architecture and components of Interaction Concentrator. It also provides a high-level overview of Interaction Concentrator functionality. It contains the following sections:

Basic Architecture

Interaction Concentrator is a Genesys product that collects and stores detailed data from various sources in a contact center that utilizes Genesys software. Downstream reporting systems can access Interaction Concentrator data in near real time.

Operating on top of Genesys Framework, the Interaction Concentrator product consists of a server application called Interaction Concentrator (ICON) and a database called Interaction Database (IDB). The server receives data from the data sources such as Configuration Server, T-Server, or particular Genesys solutions; it then stores this data into IDB through Genesys DB Server.

Figure 1 depicts the basic ICON architecture, omitting the Framework components for the sake of simplicity.

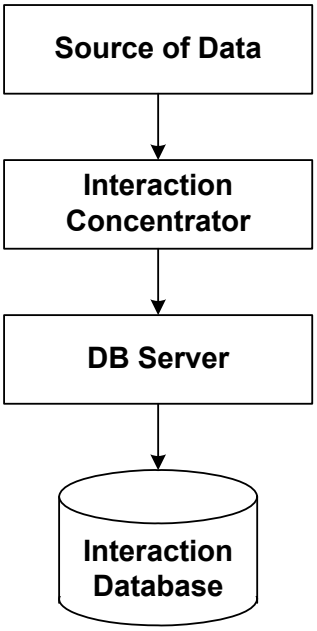

Figure 1: Basic Interaction Concentrator Architecture

Components and Functions

Interaction Concentrator consists of the following elements:

- ICON server
- Interaction Database

The following subsections describe each of these in turn.

ICON Server

The ICON server:

- Performs preprocessing of events received from Configuration Server, T-Server, Interaction Server, and Outbound Contact Server, according to the role configured for the ICON instance.

  Processing occurs in the in-memory queue (*accumulator*). You can configure the size of the in-memory queue or the interval at which data is written from it to the persistent queue.

  In ICON 7.6.1 you can also configure the maximum number of keep-in-memory interactions that concurrently reside in an interaction queue or interaction workbin by setting memory optimization configuration options on the ICON application. This functionality requires Interaction Server release 7.6.1 or higher.
- Prepares the data that will be stored in IDB.

- Writes the prepared data from the in-memory queue to the persistent queue. For more information about the persistent queue, see "Persistent Queue".
- Manages the data in the persistent queue.
- Writes data from the persistent queue into IDB.

For detailed information about the configuration options that determine ICON functionality and performance, see the *Interaction Concentrator 7.6 Deployment Guide.*

Persistent Queue

The persistent queue is a file that ICON creates and uses to store data before writing it to IDB. The persistent queue also stores information about database-writing requests to IDB. Data in the persistent queue survives a shutdown and restart of ICON.

Configuration Database Synchronization

In ICON 7.5, there is one persistent queue that stores data for all ICON instances and roles. In release 7.6, there is an additional persistent queue (`cfg-sync.db`) for the ICON instance that performs the `cfg` role. This queue plays an important role in maintaining IDB synchronization with the Configuration Database. ICON keeps a timestamp in the persistent queue for configuration data changes and, on startup, requests from Configuration Server all configuration changes that occurred after that timestamp. For more information about the role of the persistent queue in maintaining synchronization, see "How ICON Gathers Configuration Data" on page 132.

Persistent Queue Configuration Options

ICON configuration options enable you to specify:

- The file name of the persistent queue for all roles except the `cfg` role. The name of the persistent queue for the ICON instance with the `cfg` role is `cfg-sync.db` and it cannot be changed.
- The frequency (in terms of number of committed transactions) with which ICON clears data out of the persistent queue.
- Thresholds for environment failure alarms.
- Persistent queue behavior at startup.

The alarm thresholds can also be used to monitor ICON performance. For more information about the persistent queue configuration options, see the *Interaction Concentrator 7.6 Deployment Guide.*

Note: The size of the persistent queue is not formally limited by ICON, but the operating system may impose some limitations.

ICON Server Interfaces

The ICON server interfaces with:

- Solution Control Server (SCS) through Local Control Agent (LCA), to control when the ICON server starts and stops.
- Configuration Server, to read Interaction Concentrator application configuration options and other configuration objects and options that affect Interaction Concentrator functionality. (This interface is logically separate from ICON's connection to Configuration Server as a source of data about contact center resources—see "Sources of Data" on page 23.)
- Message Server, to log messages to the Central Logger.

Note: Interaction Concentrator does not support the use of the Transport Layer Security (TLS) protocol to secure data exchange between the components with which the ICON server interfaces.

Interaction Database

The Interaction Database stores data about contact center interactions and resources at a granular level of detail. IDB is a database optimized for storage (in other words, for inserting data). Interaction Concentrator itself does not provide a reporting facility. You can use IDB as a consistent and reliable data source for downstream reporting applications.

For a high-level description of the IDB architecture, see "Introducing IDB Schema" on page 27. For a complete table structure and descriptions of all IDB tables and fields, see the *Interaction Concentrator 7.6 Physical Data Model* document for your particular RDBMS.

Stored Procedures

Interaction Concentrator uses a number of stored procedures. Most of these are totally internal to Interaction Concentrator functioning, and therefore they are not relevant to end users. However, the following stored procedures do require user input or action:

Merge

- `gsysIRMerge` and `gsysIRMerge2`—The merge procedures that finalize data processing of closed single-site and multi-site interactions.

Purge

- Use either of the two sets of purge procedures:
  - `gsysPurgeIR`, `gsysPurgeUDH`, `gsysPurgeLS`, and `gsysPurgeOS` to safely purge voice interactions from IDB.
  - `gsysPurge76` and `gsysPurgeOS` to safely purge voice and multimedia interactions from IDB. Genesys recommends that new customers, customers purging large amounts of data, and customers concerned about performance, use this set of purge procedures.

Time-Setting

- `gsysInitTimeCode`—The stored procedure that populates the `G_TIMECODE` table, to enable time-interval reporting.

Custom Dispatchers

- `gudCustDisp1` and `gudCustDisp2`—The stored procedures that are used to customize attached data processing.

For more information about these stored procedures, see "Using Special Stored Procedures" on page 145.

Sources of Data

Table 1 summarizes the sources from which ICON collects data. For Interaction Concentrator 7.5 and 7.6, the range of types of data and sources is wider than for Interaction Concentrator 7.2.

Table 1: ICON Sources of Data

Type of Data	Source	Applicable Interaction Concentrator Release		
		7.2	7.5	7.6
Configuration data for your contact center resources	Configuration Server	Yes	Yes	Yes
Detailed CTI–related data about call activity in your contact center	T-Server	Yes	Yes	Yes
Detailed CTI-related data about Voice over IP (VOIP) interaction activity in your contact center	T-Server (SIP Server)	No	No	Yes
Detailed data about Open Media interaction activity (including e-mail and non-SIP chat) in your contact center	Interaction Server	No	Yes	Yes
Detailed data about SIP chat interaction activity in your contact center	SIP Server	No	Yes	Yes
Media types for Open (custom-defined) Media	Interaction Server	No	No	Yes
Detailed data about virtual queue usage in interaction processing	• For voice calls: T-Server (from Universal Routing Server [URS]) • For Multimedia interactions: Interaction Server (from URS)	No	Yes	Yes
Data specific to outbound calls and campaigns	OCS	Yes	Yes	Yes

Supported Features and Functionality

In addition to new functionality described in "New in Interaction Concentrator 7.6" on page 25, and depending on the role configured for the Interaction Concentrator instance, Interaction Concentrator provides the following features and functionality that support reporting about contact center activities:

- Captures and stores information about the current contact center configuration (objects and associations), and preserves information about deleted configuration objects and terminated associations.
- Captures and stores detailed information about active and completed voice interactions, including switch, DN, time, and routing information about calls and parties. Interaction Concentrator uses a globally unique call identifier.
- Captures and stores detailed information about Multimedia interactions (e-mail and chat). For more information, see Chapter 3 on page 37.
- Captures and stores detailed information about agent states and login sessions, for agents handling voice as well as Multimedia (e-mail and chat) interactions. For more information, see Chapter 6 on page 71.
- Supports custom agent states, for agents handling voice interactions. For more information, see Chapter 6 on page 71.
- For all types of interactions, captures and stores detailed information about virtual queue usage in interaction processing. For more information, see Chapter 5 on page 63.
- For voice interactions, captures and stores detailed information about Routing Point (RP) usage in call processing.
- Captures and stores detailed information about interactions that are generated in a network-based contact solution environment.
- Captures and stores detailed information about interactions that are generated in a network call parking environment.
- Stores attached data and captures the history of attached data changes for voice interactions as well as Multimedia (e-mail and chat) interactions. For more information, see Chapter 4 on page 47.
- Supports customized attached data processing for voice calls. For more information, see Chapter 4 on page 47.
- Captures and stores detailed information about outbound campaigns, including:
  - History of campaign processing
  - History of chain processing
  - Precalculated metrics provided by OCS

  For more information, see Chapter 7 on page 81.
- Supports real-time, intraday reporting by writing data to IDB as soon as the data is available (as opposed to after the interaction is completed).

- Provides a sophisticated recognition mechanism, utilizing Inter-Site Call Linkage (IS-Links), to process multi-site interactions. ICON receives information from T-Server regarding the relationship between a given interaction and an interaction at a different site. As a result, complete data is available for reporting across sites. Interaction Concentrator provides a stored procedure to merge the interaction records for multi-site interactions. For more information about the merge procedure, see Chapter 14, "Using Special Stored Procedures," on page 145.
- Supports multibyte character encoding.
- Stores time information in two formats:
  - Greenwich Mean Time (GMT)—As a `datetime` data type.
  - Coordinated Universal Time (UTC) seconds—As an `integer` data type.

  ICON obtains the time information from the timestamps of the data provider events (for example, T-Server TEvents), in the form of UTC seconds.

New in Interaction Concentrator 7.6

The 7.6.1 release of Interaction Concentrator includes the following new features:

- An enhancement to support a large number of concurrently active Multimedia interactions.
- A purging mechanism for Multimedia data stored in IDB.
- Improved performance for purging voice data that is stored in IDB using the new purging mechanism.
- The ability to filter out some Multimedia data that is not relevant for reporting in order to save IDB storage space and improve overall performance.

The 7.6 release of Interaction Concentrator provides the following additional or changed functionality:

- Provides the ability to resynchronize the configuration data in Interaction Database (IDB) with Configuration Database on demand. For more information, see Chapter 13 on page 131.
- Enables database size optimization by providing a filtering mechanism for certain types of data. For more information, see Chapter 11 on page 111.
- Supports high availability of data at the extraction, transformation and loading (ETL) level by providing data redundancy through the use of two or more IDBs. The applicable data includes configuration data, agent-related data, voice details data, and SIP chat data. For more information, see Chapter 12 on page 119.
- Provides an ability to report after-call work (ACW) for the first interaction associated with ACW. For more information, see Chapter 6 on page 71.

- Supports media types for Open Media in all the areas where e-mail and non-SIP data were previously supported. For more information, see Chapter 3 on page 37.
- Supports reporting on SIP chat, and the ability to identify if a chat session has the focus. For more information, see Chapter 3 on page 37 and Chapter 6, “Agent States and Login Sessions,” on page 71.
- Provides the ability to suppress the interruption of the after-call work (`ACW`) and `NotReady` agent states by interactions coming to, or produced by, the agent. For more information, see Chapter 6 on page 71.

Chapter

2 Introducing IDB Schema

The Interaction Database stores in its tables all reporting data that ICON provides. Logically, IDB tables can be divided into groups that correspond to the classes of information that ICON writes.

This chapter describes the logical groups of tables. This chapter also provides a summary of the precalculated call and party metrics that are available in the statistical tables. It contains the following sections:

Notes: For a complete table structure and description of all IDB tables and fields, see the *Interaction Concentrator 7.6 Physical Data Model* document for your particular RDBMS.

The data filtering feature limits the data stored to IDB. See "Filtering IDB Data" on page 111 for more information.

Operational Tables

There are six subgroups of tables that contain operational data:

- Call–related and Party–related tables
- Agent State–related and Login Session–related tables
- Custom State–related tables
- Attached Data–related tables
- Outbound–related tables
- Virtual Queue–related table

These groups of tables are described in detail in the following subsections.

Call-Related and Party-Related Tables

There are three subgroups of tables that contain data about calls and parties:

- Core tables
- History tables
- Statistical tables

The following subsections describe each of these subgroups in turn.

Core Tables

The core tables contain current (the most up-to-date) information about calls and parties as well as all related information.

The five tables in this subgroup are as follows:

- G_CALL—Contains information about telephone calls (both completed and active), SIP chat, and Open Media interactions
- G_PARTY—Contains information about call parties
- G_IR—Contains information about the data that is common to all of the calls in a particular scenario
- G_IS_LINK—Contains information about call links in a multi-site scenario
- G_ROUTE_RESULT—Contains information about the results of routing for the call

History Tables

The history tables contain intermediary (history) states of the data that was previously stored in the core tables. The history tables are named G_CALL_HISTORY, G_PARTY_HISTORY, G_IR_HISTORY, and G_IS_LINK_HISTORY. They correspond to the G_CALL, G_PARTY, G_IR, and G_IS_LINK core tables, respectively.

Statistical Tables

The two statistical tables, G_CALL_STAT and G_PARTY_STAT, contain statistical information about calls and parties, respectively.

Party Metrics

Table 2 lists the Interaction Concentrator metrics that are stored in the G_PARTY_STAT table, in the order of their appearance in that table. Table 2 also indicates the media type for which the metric is calculated.

Table 2: Available Party Metrics, by Media Type

Metric Name	Description	Media Type Supported		
		Voice	Multimedia	
			E-Mail	Chat
TT_ALERTING	Total time that a party spent in the ALERTING state	Yes	Yes	Yes
TT_CONNECTED	Total time that a party spent in the CONNECTED state. The states of other parties in the call do not affect this metric.	Yes	Yes	Yes
TT_HOLD	Total time that a party spent in the HOLD state	Yes	No	No
TT_QUEUED	Total time that a party spent in the QUEUED state	Yes	Yes	Yes
TT_ACW	Total time that a party spent in after-call work (ACW)	Yes	No	No
CNT_ALERTING	Number of times that a party changed its state to Alerting	Yes	Yes	Yes
CNT_CONNECTED	Number of times that a party changed its state to Connected (for example, from the Alerting or Hold state)	Yes	Yes	Yes
CNT_HOLD	Number of times that a party changed its state to Hold	Yes	No	No
CNT_QUEUED	Number of times that a party changed its state to Queued	Yes	Yes	Yes
CNT_ACW	Flag of ACW presence for this party	Yes	No	No
TT_ON_ALERT	Total time that another party in the interaction spent in the ALERTING state	Yes	Yes	Yes
TT_ON_HOLD	Total time that another party in the call spent in the HOLD state	Yes	No	No
TT_ON_QUEUE	Total time that another party in the interaction spent in the QUEUED state	Yes	Yes	Yes
TT_ON_CONNECTED	Total time that all parties in the call were simultaneously in the CONNECTED state	Yes	Yes	Yes

Table 2: Available Party Metrics, by Media Type (Continued)

Metric Name	Description	Media Type Supported		
		Voice	Multimedia	
			E-Mail	Chat
T_DURATION	Duration of a party's existence	Yes	Yes	Yes
PM_EXT_1–PM_EXT_10	Reserved for custom metrics calculation. You can perform calculations of custom metrics by using custom stored procedures.	Yes	Yes	Yes

Interaction Metrics

Table 3 lists the Interaction Concentrator metrics that are stored in the `G_CALL_STAT` table, in the order of their appearance in that table. Table 3 also indicates the media type for which the metric is calculated.

Table 3: Available Interaction Metrics, by Media Type

Metric Name	Description	Media Type Supported		
		Voice	Multimedia	
			E-Mail	Chat
F_CONN	Flag `Connected`	Yes	Yes	Yes
F_CONN_EXTN	Flag `Connected on Extension`	Yes	Yes	Yes
F_TE_ABND	Flag `Event Abandoned`	Yes	No	Yes
CNT_HOLD	Number of times that the call was placed on hold.	Yes	No	No
CNT_DIVERT	Number of times that the interaction was diverted (for example, from a Queue or Routing Point)	Yes	Yes	Yes
CNT_TRANSFER	Number of times that the interaction was transferred, given that the transfer was completed	Yes	Yes	Yes
CNT_TRANSFER_LGIN	Number of times that the interaction was transferred by a party associated with a device that had a logged-in agent	Yes	Yes	Yes
CNT_CONFERENCE	Number of times that the interaction was conferenced.	Yes	No	Yes
T_DURATION	Duration of the interaction	Yes	Yes	Yes

Table 3: Available Interaction Metrics, by Media Type (Continued)

Metric Name	Description	Media Type Supported		
		Voice	Multimedia	
			E-Mail	Chat
T_CONN	Time until `Connected` with the first party in the interaction. This metric's value can be considered the time to answer on any device (either by an agent of IVR).	Yes	Yes	Yes
T_CONN_EXTN	Time until `Connected on Extension`.	Yes	Yes	Yes
T_TE_ABND	Time until `Abandoned`.	Yes	No	Yes
TT_ALERTING	This is the sum of all the time interval durations if there was at least one internal party in a call in the `ALERTING` state.	Yes	Yes	Yes
TT_CONNECTED	This is the sum of all the time interval durations when all parties in a call were in a `CONNECTED` state.	Yes	Yes	Yes
TT_HOLD	This is the sum of all time interval durations when there was at least one internal party in a call in the `HOLD` state.	Yes	No	No
TT_QUEUED	This is the sum of all the time interval durations when there was at least one internal party in a call in the `QUEUED` state.	Yes	Yes	Yes
CM_EXT_1–CM_EXT_10	Reserved for custom metrics calculation. You can perform calculations of custom metrics by using custom stored procedures.	Yes	Yes	Yes

Agent State–Related and Login Session–Related Tables

There are three subgroups of tables that contain historical data about contact center agents.

- Core tables
- History tables
- Statistical tables

The following subsections describe each of these subgroups in turn.

Core Tables

The core tables contain information about agent states, login sessions, and the association of sessions with endpoints (DNs).

The three tables in this subgroup are as follows:

- G_LOGIN_SESSION—Contains information about agent login sessions.
- GX_SESSION_ENDPOINT—Contains information about the association between a login session and an endpoint (DN).
- G_AGENT_STATE_RC—Contains information about reason codes for agent states.

History Tables

The history tables contain historical data relating to agent states and login sessions at a DN.

The two tables in this subgroup are as follows:

- G_AGENT_STATE_HISTORY—Stores the history of agent states within a given login session.
- G_DND_HISTORY—Stores the history of DND (Do Not Disturb) feature activation and deactivation on a device (DN).

Statistical Tables

The two statistical tables contain several statistics related to agent states and login sessions.

The two tables in this subgroup are as follows:

- GS_AGENT_STAT—Stores durations of agent states.
- GS_AGENT_STAT_WM—Stores durations of work modes for agent states.

Custom State–Related Tables

Interaction Concentrator supports customer-defined states and attached data in UserEvents for compatibility with Call Concentrator.

The three tables related to custom states are as follows:

- G_CUSTOM_DATA_P—This is a flat table with key values delivered in UserEvents associated with voice calls.
- G_CUSTOM_DATA_S—This is a generic table. Information about key values are delivered in UserEvents associated with voice calls.
- G_CUSTOM_STATES—Stores detailed information about an agent's state changes during the agent login session.

Attached Data–Related Tables

The tables related to attached data (also sometimes referred to as `UserData`) contain data that T-Server and, if applicable, Interaction Server attach to an interaction. ICON selects the data from TEvents or Multimedia reporting events. The exact data that ICON selects depends on the way in which ICON has been configured. The data in these tables can include the current state of attached data or the history of attached data changes, if so configured.

There are three subgroups of attached data–related tables:

- Attached Data State tables for voice (*flat* tables)
- Attached Data History tables for voice (*generic* tables)
- Multimedia Attached Data tables

The following subsections describe each of these subgroups in turn. For more information, see Chapter 4, "Processing Attached Data," on page 47.

Attached Data State Tables

The Attached Data State (flat) tables store the current (latest received) state of the attached data attributes that are associated with calls.

The four tables in this subgroup are as follows:

- `G_CALL_USERDATA`—Stores predefined attached data attributes.
- `G_CALL_USERDATA_CUST`—Stores custom attached data attributes.
- `G_CALL_USERDATA_CUST1`—Stores custom attached data attributes.
- `G_CALL_USERDATA_CUST2`—Stores custom attached data attributes.

For more detailed information about stored Multimedia data, see "Database Schema Extensions for Multimedia Attached Data" on page 60.

Attached Data History Tables

The Attached Data History (generic) tables store historical information about attached data.

The two tables in this subgroup are as follows:

- `G_USERDATA_HISTORY`—Stores information about the attached data fields that require no security protection.
- `G_SECURE_USERDATA_HISTORY`—Stores information about the sensitive attached data fields that do require security protection (for example, a customer's Social Security number).

Multimedia Attached Data Tables

The Multimedia Attached Data tables store information about Multimedia attached data.

The two tables in this subgroup are as follows:

- GM_L_USERDATA—Stores the values of attached data keys for suggested and auto responses and acknowledgements, customer IDs, and reasons for stopping processing.
- GM_F_USERDATA—Stores metadata information about e-mail and chat interactions (for example, the sender's name and e-mail address, the subject, and the type).

Outbound-Related Tables

The outbound-related tables include either the GO_ or GOX_ prefix in their names. These tables store information about the entities and attributes that are related to the processing of outbound calls and campaigns as reported by Outbound Contact Server (OCS).

For more detailed information about stored outbound data, see "Available Outbound Data" on page 83.

Virtual Queue Table

The table that stores information related to interaction processing at virtual queues is called G_VIRTUAL_QUEUE. It stores the history of associations between interactions and virtual queues, as reported by T-Server, provided that Universal Routing Server (URS) provides this information to T-Server.

For more detailed information about stored virtual queue data, see "Monitoring Virtual Queues and Routing Points" on page 63.

Configuration Tables

There are two subgroups of tables that contain configuration data:

- Object tables
- Object links tables

The following subsections describe each of these subgroups in turn.

Object Tables

The object tables include the GC_ prefix in their names. These tables contain current (the most up-to-date) information about the configuration objects that ICON tracks. These tables also preserve information about the configuration objects that have been deleted from the Configuration Database.

Object Links Tables

The object links tables include the GCX_ prefix in their names. These tables contain information about the *associations* (links) between configuration objects in the Configuration Database. Examples of associations between configuration objects include assignments of Skills or Logins to Agents and assignments of Agents to Agent Groups. These tables also preserve information about terminated associations—for example, information about the fact that an Agent was removed from an Agent Group.

Service and Dictionary Tables

The service and dictionary tables are used for ICON internal purposes or describe fields in other tables.

The five tables in this group are as follows:

- G_DICTIONARY and G_DICT_TYPE—Contain dictionary information for certain enumerator fields in other tables (for example, the STATE, STATUS, and CAUSE fields).
- G_DB_PARAMETERS—Contains general information about the database schema (for example, the schema version).
- G_HA_CONTROL, G_SYNC_CONTROL, and G_PROV_CONTROL—Are used by ICON to implement internal transaction control.
- G_TIMECODE—Expands the timecode values that are referenced in other tables (for example, CREATED_TCODE and DELETED_TCODE) into specific time value entities such as month, day of the week, day of the month, and so on.
- GSYS_SYSPROCINFO, GSYS_DNPREMOTELOCATION, G_LOG_MESSAGES, G_LOG_ATTRS, and G_LOG_GETIDRANGEREQ—Are internal tables used by ICON's system procedures.

Chapter

3 Integrating with Genesys Multimedia and Open Media

Genesys Multimedia (formerly known as Multi-Channel Routing or MCR) refers to those parts of the Genesys Customer Interaction Management (CIM) platform that work together to manage interactions that involve nontraditional, non-voice media (for example, e-mail and chat) and open (custom-designed) media (for example, fax and web forms).

This chapter describes how to make data about Multimedia and Open Media interactions available in Interaction Database (IDB). It also discusses enhancements made in Interaction Concentrator (ICON) 7.6.1 to support the processing of a large number of active multimedia interactions over a sustained period without failure due to insufficient memory.

It contains the following sections:

Multimedia Objects

This section introduces the terminology and elements (objects) that pertain to Interaction Concentrator data about Multimedia activities.

This section contains information about the following:

- Endpoints
- DNs

Endpoints

ICON stores reporting data about the following logical endpoints:

- `Interaction Queue`—Configured in the Configuration Layer as a `Script` object (of type `Interaction Queue`).
- `Interaction Workbin`—Configured in the Configuration Layer as a `Script` object (of type `Interaction Work Bin`).
- `Routing Strategy`—Configured in the Configuration Layer as a `Script` object (of type `Simple Routing`).
- `Agent's Place`—Configured in the Configuration Layer as a `Place` object.

ICON stores configuration-related information about `Script` objects in the `GC_SCRIPT` table.

ICON stores configuration-related information about `Place` objects in the `GC_PLACE` table.

DNs

The Interaction Server uses a `Switch` configuration object of type `MultimediaSwitch`. ICON stores information about DNs that are configured under the Interaction Server switch in the same way that it stores information about DNs that are configured under any other type of switch.

ICON stores configuration-related information about `DN` configuration objects in the `GC_ENDPOINT` table.

ICON stores configuration-related information about the association between DNs and places in the `GCX_ENDPOINT_PLACE` table.

Multimedia Interaction Data Processing and Storage

This section contains information about the following:

- Multimedia Reporting Protocol events (see page 39)
- Multimedia interactions (see page 39)
- Supported scenarios (see page 40)

For information about how to capture information about agent states and login sessions for Multimedia interactions, see Chapter 4 on page 47.

For information about the way in which ICON handles attached data for Multimedia interactions, see “Attached Data Processing for Multimedia” on page 60.

For detailed information about the tables in IDB in which ICON stores Multimedia data, see the *Interaction Concentrator 7.6 Physical Data Model* document for your particular RDBMS.

Multimedia Reporting Protocol Events

ICON connects to Interaction Server, and it receives notifications, in the form of Multimedia Reporting Protocol events, about Multimedia e-mail and chat interaction processing.

ICON processes the following Multimedia Reporting Protocol events for interactions:

- `EventInteractionSubmitted`
- `EventProcessingStopped`
- `EventPlacedInQueue`
- `EventTakenFromQueue`
- `EventPlacedInWorkbin`
- `EventTakenFromWorkbin`
- `EventPartyAdded`
- `EventPartyRemoved`
- `EventPropertiesChanged`
- `EventAgentInvited`
- `EventRejected`
- `EventRevoked`
- Multimedia Reporting custom message (envelope for virtual queue–related TEvents)

For more information about Multimedia Reporting Protocol events, see the *Genesys 7 Events and Models Reference Manual*.

Multimedia Interactions

ICON stores the details about Multimedia interaction processing in the same tables in IDB in which it stores voice records:

Core Tables

- `G_CALL`—Each interaction with a media type of e-mail or chat is represented as a single record. ICON creates the record when it receives `EventInteractionSubmitted`. ICON updates the record (marks the record as terminated) when it receives `EventProcessingStopped`. ICON sets the value of the `GSYS_EXT_INT1` field to `1` when the record is for a Multimedia interaction (in other words, if the interaction record was created as a result of Interaction Server event processing).
- `G_PARTY`—Contains information about the parties who participate in a Multimedia interaction. ICON creates a new record when Interaction Server reports that a relationship has been established between an endpoint and a Multimedia interaction.

> **Note:** No records are stored for external parties who participate in a Multimedia interaction.

- G_IR—Contains information about the data that is common to all of the interactions in a particular scenario. ICON sets the value of the GSYS_EXT_INT1 field to 1 when the record is for a Multimedia interaction (in other words, if the interaction record was created as a result of Interaction Server event processing).

  A new record for a Multimedia interaction is created if either of the following conditions is met:
  - The interaction is not associated with an existing parent interaction.
  - The interaction is associated with an existing parent interaction, but the information about the associated G_IR parent record is not available. In this case, ICON stores the ID information that Interaction Server provides for the parent record in the GSYS_EXT_VCH1 field.
- G_ROUTE_RESULT—Refers to the record created in the G_PARTY table for the strategy and contains information about the results of routing for the interaction.

Note: No Multimedia interaction data is written to the G_IS_LINK table.

History Tables

- G_CALL_HISTORY, G_PARTY_HISTORY, and G_IR_HISTORY—Contain intermediary (history) states of the data that was previously stored in the core tables.

Statistical Tables

- G_CALL_STAT and G_PARTY_STAT—Contain metrics about Multimedia interactions and parties, respectively. For information about the available precalculated metrics, see Tables 2 and 3 on pages 29 and 30.

Supported Scenarios

Interaction Concentrator supports the following scenarios for Multimedia interactions:

- Interaction submission
- Interaction distribution
- Transfer
- Conference
- Auto-acknowledgement
- Autoresponse
- Abandonment without handling (for chat only)

Handling Active Multimedia Interactions

Enhancements made to ICON in release 7.6.1 enable it to handle millions of active multimedia interactions over a sustained period of time without a failure due to insufficient memory.

To make this possible, ICON 7.6.1 does the following:

- Removes interactions from operational memory. See "Removing Interactions From Memory".
- Reconstructs interaction data later for further processing. See "Reconstructing Interaction Data".
- Tracks user data changes. See "Tracking User Data Changes" on page 61.
- Filters out strategy data not required for reporting. See "Strategy Activity Data" on page 116.

Removing Interactions From Memory

When ICON 7.6.1 receives an `EventPlacedInQueue` and/or `EventPlacedInWorkbin` reporting event from Interaction Server, it considers this the signal to remove the corresponding Multimedia interaction from memory provided that certain user-defined options have been set:

- The global `om-memory-optimization` option must be set to `true` to allow ICON to optimize memory according to the user-defined values of the other memory options. If this option is set to `false`, no memory optimization will occur regardless of the values of the other options.
- The `om-max-in-memory` option defines the maximum number of keep-in-memory interactions that concurrently reside in an interaction queue or interaction workbin. When this maximum is reached, ICON removes the oldest interaction.
- If the `om-memory-clean` option is set to `true` (on `script` objects of type `interaction queue`), ICON immediately removes interactions from operational memory as soon as they arrive; it does not wait until the value of the `om-max-in-memory` option is reached before removing interactions.

For more information about these and other options, see the chapter about configuration options in the *Interaction Concentrator 7.6 Deployment Guide.*

Reconstructing Interaction Data

When ICON 7.6.1 receives an `EventTakenFromQueue` and/or `EventTakenFromWorkbin` reporting event from Interaction Server, it considers this the signal to reconstruct the corresponding multimedia interaction if the interaction had been previously removed from memory.

To reconstruct the interaction, ICON restores the following key data:

- `CALLID` of the interaction
- `PARTYID` of the party for the last interaction in the queue and/or the last interaction in the workbin
- User data

Once the interaction is reconstructed, ICON processes it as a regular multimedia interaction using the rules implemented in release 7.6. See

"Multimedia Interaction Data Processing and Storage" on page 38 for more information.

Open Media Interaction Processing

ICON stores detailed information about open media interaction processing and agent activities related to this processing.

Each interaction received from Interaction Server contains a media type name defined in Configuration Server. When ICON processes reporting events, such as `EventInteractionSubmitted`, it extracts the media type name from the interaction and maps this string to an integer value.

ICON stores the details about Open Media interactions in the same tables in IDB in which it stores voice and multimedia records:

Core Tables

- `G_CALL`
- `GX_SESSION_ENDPOINT`
- `G_AGENT_STATE_RC`
- `GS_AGENT_STAT`
- `G_AGENT_STATE_HISTORY`

In each of the core tables (above), ICON sets the following values for Open Media interactions:

- `GSYS_EXT_INT1` field =`1000`;
- `GSYS_EXT_VCH1` field = a string value containing the name of the media type.

In addition, ICON populates the `MEDIATYPE` field (= `1000`) for the `G_CALL` table.

Note: For a description of these tables, see "Multimedia Interaction Data Processing and Storage" on page 38.

isOnline Chat Attribute

The `isOnline` attribute is introduced in release 7.6 to support chat interactions. This attribute allows Genesys Chat Server to notify Interaction Server when a chat session is active by giving the `isOnline` attribute a value of `on`. When a chat session is terminated—that is, the last person leaves the chat session—the value of the `isOnline` attribute changes to `off` and Chat Server sends notification of this attribute change to Interaction Server.

Chat Server is responsible for monitoring and changing the `isOnline` attribute. If Chat Server disconnects or shuts down, Interaction Server does not send any notification regarding the status of the chat interaction to Interaction Concentrator. Figure 2 illustrates the possible transmissions and states of the `isOnline` chat attribute.

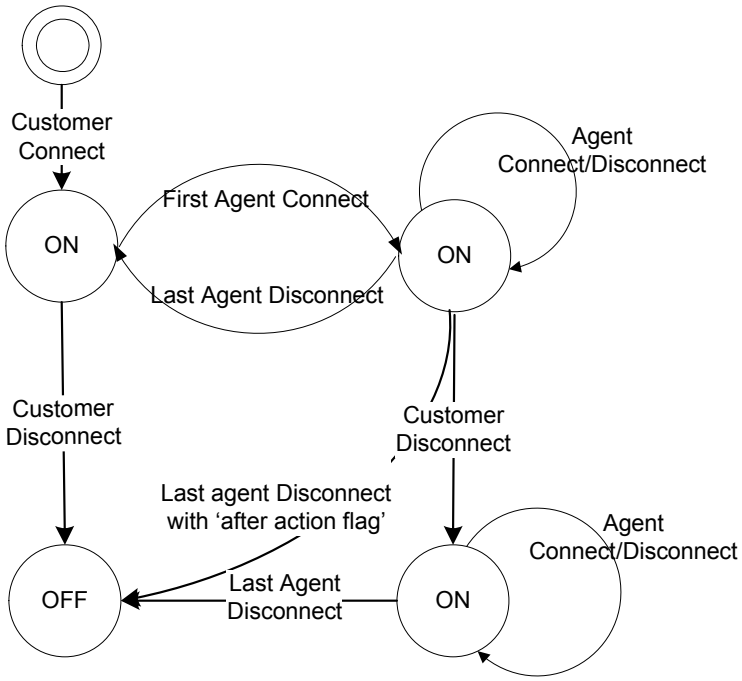

Figure 2: isOnline Attribute of Chat Interactions

Processing the isOnline Attribute

ICON monitors changes in the `isOnline` attribute in all interaction reporting events (`EventInteractionSubmitted`, `EventPropertiesChanged`, and `EventProcessingStopped`). Using the same mechanism to store the `isOnline` attribute as it does to process attached data, ICON creates a new record in the `G_USERDATA_HISTORY` table to store the value of the `isOnline` attribute. Table 4 describes the key fields in the `G_USERDATA_HISTORY` table and the possible values.

Table 4: Key Fields in G_USERDATA_HISTORY Table

Field Name	Value	Comment
`KeyName`	`_attr_is_online`	This value does not change.
`ChangeType`	`_created`	Value stored when ICON receives `EventInteractionSubmitted`
`ChangeType` (continued)	`_updated`	Value stored when ICON receives `EventPropertiesChanged`
	`_terminated`	Created when `EventProcessingStopped` is received

Table 4: Key Fields in G_USERDATA_HISTORY Table (Continued)

Field Name	Value	Comment
Type	Source_Attributes	The type of the data source—extensions, reasons, or attached data (userdata)—represented by these values: 1—userdata 2 —reasons 3 —extensions 4 —attributes (reserved for future use) 5 —mcr_workbin
KeyID	9995	This is a hard-coded value.
Value	0 or 1 (value is taken from event)	1—chat session is active 0 —chat session is stopped
Added	Timestamp	This is the time corresponding to when the record was modified.

Configuration Recommendations

In order to store Multimedia interaction, agent state, and login session data in IDB for reporting purposes, certain configuration settings are required in the Genesys Configuration Layer. This section describes the configuration settings that are required on the ICON Application object.

ICON Application

To enable ICON to receive Multimedia data and store it in IDB, you must configure ICON connections to appropriate Interaction Server instances.

Connections

ICON will not connect directly to an Application object of type Interaction Server. Therefore, in an environment with a single Interaction Server, create an application of type T-Server with connection parameters to the Interaction Server. Add a connection to this T-Server Application object to the Connections tab of the ICON Application object.

In an environment with multiple Interaction Servers, decide on your deployment topology—that is, decide whether a single ICON instance will handle the data from all or a subset of Interaction Servers, or whether each Interaction Server will have a dedicated ICON instance. Based on your deployment decision, create one or more applications of type T-Server with connection parameters to the Interaction Servers. Add the required T-Server

`Application` objects to the `Connections` tab of the ICON `Application` objects that must store data from those Interaction Servers.

Configuring for Multimedia

There are no special requirements for other ICON `Application` object configuration options. The type of data that ICON captures for Multimedia objects and interactions depends on the `role` configuration option that you configure for the ICON instance. For more information on configuration objects, refer to the *Interaction Concentrator 7.6 Deployment Guide.*

Configuring for Open Media

To enable ICON to store information about Open Media interactions in IDB, you must configure the `mcr-om-processing` configuration option in the `callconcentrator` section of the ICON `Application` object. For information about setting this option, refer to the *Interaction Concentrator 7.6 Deployment Guide*.

Chapter

4 Processing Attached Data

This chapter describes how to make user data that is attached to voice calls, Multimedia, and Open Media interactions available in Interaction Database (IDB). It contains the following sections:

Note: The processing and storage of attached data is resource intensive and expensive. Genesys recommends that you carefully consider your reporting and troubleshooting requirements in order to limit the amount of attached data that you configure Interaction Concentrator to capture.

Attached Data Processing for Voice Calls

This section briefly describes how Interaction Concentrator (ICON) processes user data that is attached to voice calls.

T-Server Interactions

When processing data from T-Server, ICON checks all TEvents for changes to attached data. When attached data changes for a particular interaction, ICON analyzes the change and stores the data in IDB, according to either its application configuration or the attached data specification.

Call-Specific Data

Call-specific data is attached data that is associated only with a call. This data is stored in IDB when the call is cleared after a specified timeout. For information about setting the `call-deletion-timeout` configuration option to configure the timeout interval, see the configuration chapter in the *Interaction Concentrator 7.6 Deployment Guid*e.

Historical Data

Historical data is associated with the attached data. Historical data can have the following associations:

- Party—When a third-party DN represents a party in `EventAttachedDataChanged`.
- Endpoint—When a third-party DN is specified in `EventAttachedDataChanged`.
- Agent—When an agent is associated with a specific device at the moment when the data is modified.

The stored change type reflects the change type for records with a history type of `all`. For more information about the history types, see “History Types” on page 54.

Table 5 defines the values for types of attached data changes.

Table 5: Attached Data Change Types

Type of Change	Numeric Equivalent	Description
created	1	Call was created with the specified key.
added	2	Key was added into the existing attached data.
updated	3	Existing key was updated.
deleted	4	Existing key was deleted.
terminated	5	Call was terminated with an existing key.

Database Schema Extensions for Voice Attached Data

IDB contains two types of tables for `UserData` collection:

- So-called *flat* tables are used when data that corresponds to a particular key name is stored into a particular field in a table.
- *Generic,* or *historical,* tables are used when each attached data value is stored in a separate row with the corresponding context.

Table 6 lists IDB tables that are used to store user data that is attached to voice calls.

Note: The attached data storage tables store user data about each call in a separate record (one record per call).

Table 6: Attached Data Storage Tables for Voice

Table Name	Type of Data	Description
G_CALL_USERDATA	Call	Designed for predefined `UserData` that is collected during the entire duration of a call.
G_CALL_USERDATA_CUST	Call	Designed for custom `UserData` that is collected during the entire duration of a call.
G_CALL_USERDATA_CUST1	Call	Designed for custom `UserData` that is collected during the entire duration of a call.
G_CALL_USERDATA_CUST2	Call	Designed for custom `UserData` that is collected during the entire duration of a call.
G_USERDATA_HISTORY	Historical	Designed to store historical changes to `UserData`.
G_SECURE_USERDATA_ HISTORY	Historical	Designed to store historical changes to `UserData`. Permissions for this table must be set at your particular site.

Attached Data Security

By providing a number of separate tables in which to store attached data, Interaction Concentrator provides a mechanism to secure sensitive user data. The Database Administrator (DBA) can assign user privileges in order to restrict access to the secure user data history table. Similarly, the DBA can restrict access to one or more of the flat tables.

Multi-Site and Distributed Environments

In a multi-site environment or a geographically distributed environment, when attached data is propagated between two T-Servers, each of these T-Servers stores the attached data in its own IDB. As a result, attached data is duplicated in IDBs across the sites.

Customized Attached Data Processing

You can create a custom stored procedure, or *custom dispatcher*, to handle user data that is attached to voice calls and to store the attached data in custom tables in IDB. ICON calls the custom dispatcher when the call ends.

Data Types

ICON can process two types of attached data values:

- String
- Integer

Key Groups

The values of the key names, specified in the same group, are provided by the same call to the custom dispatcher stored procedure. Within the attached data configuration file, you can specify groups of keys, with each group containing a maximum of 17 string key-value pairs (KVPs) and 17 integer KVPs. You can configure the maximum number of key groups that ICON will process (see the description of the `gud-cust-disp-groups` configuration option in the *Interaction Concentrator 7.6 Deployment Guide*).

For an example of the XML specification for customized attached data processing, see "Sample Specification for Customized Attached Data" on page 59.

Custom Dispatchers

The IDB initialization scripts create two custom dispatcher stored procedures:

- `gudCustDisp1`
- `gudCustDisp2`

The `gud-cust-disp` configuration option in the ICON `Application` object specifies which custom dispatcher ICON calls (see the option description in the configuration option chapter in the *Interaction Concentrator 7.6 Deployment Guide*).

While ICON is running, you can switch from one dispatcher to the other. This enables you to make changes to the custom dispatcher configuration without interrupting the processing of attached data.

The default custom dispatchers do nothing. You must modify the scripts in order to create the custom dispatchers that store the attached data that you require. You must also create scripts that, in turn, create the required custom tables in IDB.

Sample Scripts

In addition to the IDB initialization scripts, the Interaction Concentrator installation package contains the following sample scripts:

- `sample_gcc_<db_type>_custdisp_schema.sql`—A sample script that illustrates how you can create a custom attached data storage table (`G_SAMPLE_CUST_ADATA`).

- `sample_gcc_<db_type>_custdisp_api.sql`—A sample script that illustrates how you can modify the `gudCustDisp1` stored procedure. The modified stored procedure stores arguments in the `G_SAMPLE_CUST_ADATA` table.

Notes: Genesys provides the sample scripts to help you understand how you can modify the custom dispatcher stored procedures. Genesys does not recommend that you execute the sample scripts during installation.

Carefully verify the syntax and operation of your modified `gudCustDisp1` or `gudCustDisp2` stored procedure. Any types of errors or RDBMS violations that the custom dispatcher stored procedure produces can affect ICON processing of all other attached data for voice calls and Multimedia interactions.

Configuration Recommendations

Configuring Interaction Concentrator to store attached data in IDB is a two-part process:

1. Specify the attached data key configuration file, which maps the KVPs in reporting event attributes to IDB tables and fields. For more information, see "Attached Data Specification File" on page 52.
2. Specify the attached data configuration settings in the Genesys Configuration Layer.

This section describes the configuration settings that are available on the ICON `Application` object.

ICON Role

For every ICON instance that must store attached data, make sure that the `role` option on the `Options` tab of the ICON `Application` object includes `gud` in the list of values. If you deploy a single ICON instance for the entire contact center, you can keep the default value (`all`). For more information, see the description of the `role` configuration option in the *Interaction Concentrator 7.6 Deployment Guide*.

Attached Data Specification File

The following ICON configuration option enables you to point ICON to a different attached data specification file:

- `adata-spec-name`

Attached Data Configuration Options

The following ICON configuration options enable you to specify what attached data ICON should store, and in what manner:

- `adata-default-storage`
- `adata-extensions-history`
- `adata-reasons-history`
- `adata-userdata-history`

Review the descriptions and values for the attached data configuration options in the *Interaction Concentrator 7.6 Deployment Guide.* Select the appropriate values for your environment, and make related configuration changes on the `Options` tab of the ICON `Application` object.

Custom Dispatcher Configuration Options

The following ICON configuration options enable you to specify how the custom dispatcher will process attached data:

- `gud-cust-disp`
- `gud-cust-disp-groups`

Review the descriptions and values for the custom dispatcher configuration options in the *Interaction Concentrator 7.6 Deployment Guide*. Select the appropriate values for your environment, and make related configuration changes on the `Options` tab of the ICON `Application` object.

Attached Data Specification File

This section presents the XML schema definition for processing KVPs from the attached data that T-Server provides with TEvents. If you require ICON to store attached data in IDB, create an attached data specification for ICON to use, based on the definition in this section.

The attached data specification is an XML file stored in the installation directory that you specified when you installed the Interaction Concentrator application (see the chapter about installing and configuring ICON in the *Interaction Concentrator 7.6 Deployment Guide*).

For a sample attached data specification, see “Sample Basic Attached Data Specification” on page 58.

Parser Limitations

The ICON XML parser imposes the following limitations:

- ICON ignores unknown attributes if they are present in the specification. When parsing the XML specification, ICON checks only for missing attributes.
- The ICON XML parser does not support namespaces.
- ICON ignores duplicate keys. Only the first occurrence of a key name will be used to update the specified field in the database table (see Note and Example on page 58).

Schema Definition

The following is the XML schema definition for your attached data specification.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema targetNamespace="http://www.genesyslab.com/standards/icon/ed1"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:icon="http://www.genesyslab.com/standards/icon/ed1" elementFormDefault="qualified"
    attributeFormDefault="unqualified">
<xsd:annotation>
  <xsd:documentation>Attached data configuration specification</xsd:documentation>
</xsd:annotation>

<xsd:simpleType name="AdataHistoryType">
<xsd:restriction base="xsd:string">
      <xsd:enumeration value="none" />
<xsd:enumeration value="first" />
<xsd:enumeration value="last" />
<xsd:enumeration value="all" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="AdataCallHistoryType">
<xsd:restriction base="xsd:string">
<xsd:enumeration value="first" />
<xsd:enumeration value="last" />
   </xsd:restriction>
 </xsd:simpleType>

<xsd:simpleType name="AdataCallFieldType">
<xsd:restriction base="xsd:string">
      <xsd:enumeration value="customer-segment" />
<xsd:enumeration value="service-type" />
      <xsd:enumeration value="service-subtype" />
      <xsd:enumeration value="busines-result" />
      <xsd:enumeration value="customer-id" />
      <xsd:enumeration value="transaction-id" />
      <xsd:enumeration value="cause-id" />
      <xsd:enumeration value="account-id" />
      <xsd:enumeration value="destination-id" />
      <xsd:enumeration value="target-id" />
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="AdataCallCustFieldType">
<xsd:restriction base="xsd:string">
      <xsd:enumeration value="cust-data-1" />
      <xsd:enumeration value="cust-data-2" />
      <xsd:enumeration value="cust-data-3" />
      <xsd:enumeration value="cust-data-4" />
      <xsd:enumeration value="cust-data-5" />
      <xsd:enumeration value="cust-data-6" />
      <xsd:enumeration value="cust-data-7" />
      <xsd:enumeration value="cust-data-8" />
      <xsd:enumeration value="cust-data-8" />
      <xsd:enumeration value="cust-data-10" />
```

```
      <xsd:enumeration value="cust-data-11" />
      <xsd:enumeration value="cust-data-12" />
      <xsd:enumeration value="cust-data-13" />
      <xsd:enumeration value="cust-data-14" />
      <xsd:enumeration value="cust-data-15" />
      <xsd:enumeration value="cust-data-16" />
      <xsd:enumeration value="cust-data-17" />
      <xsd:enumeration value="cust-data-18" />
      <xsd:enumeration value="cust-data-19" />
  </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="AdataHistoryRecord">
   <xsd:attribute name="name" type="xsd:string" />
   <xsd:attribute name="source" type="icon:AdataSource" />
   <xsd:attribute name="history" type="icon:AdataHistoryType" />
</xsd:complexType>
<xsd:complexType name="AdataCallRecord">
   <xsd:attribute name="name" type="xsd:string" />
<xsd:attribute name="source" type="icon:AdataSource" />
   <xsd:attribute name="history" type="icon:AdataCallHistoryType" />
   <xsd:attribute name="field" type="icon:AdataCallFieldType" />
</xsd:complexType>
<xsd:complexType name="AdataCallCustRecord">
   <xsd:attribute name="name" type="xsd:string" />
   <xsd:attribute name="source" type="icon:AdataSource" />
   <xsd:attribute name="history" type="icon:AdataCallHistoryType" />
   <xsd:attribute name="field" type="icon:AdataCallCustFieldType" />
</xsd:complexType>
<xsd:element name="adata_spec">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="public" type="icon:AdataHistoryRecord" />
      <xsd:element name="secure" type="icon:AdataHistoryRecord" />
      <xsd:element name="call" type="icon:AdataCallRecord" />
      <xsd:element name="call-cust" type="icon:AdataCallCustRecord" />
      <xsd:element name="call-cust1" type="icon:AdataCallCustRecord" />
      <xsd:element name="call-cust2" type="icon:AdataCallCustRecord" />
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
  </xsd:schema>
```

Attribute Values

This section describes the attributes that are used in the XML schema definition.

History Types

The following values can be used as history types:

`none`	No value for a given key is recorded in IDB.
`first`	Only the first value for a given key is recorded in IDB.

`last`	Only the last value for a given key is recorded in IDB.
`all`	Every change in value for a given key is recorded in IDB. This value applies only to keys that are configured to be stored in the history tables.

Storage Types Table 7 shows the database table in which each attribute is stored.

Table 7: Attached Data Storage Types

Attribute Name	IDB Table Name
public	G_USERDATA_HISTORY
secure	G_SECURE_USERDATA_HISTORY
call	G_CALL_USERDATA
call-cust	G_CALL_USERDATA_CUST
call-cust1	G_CALL_USERDATA_CUST1
call-cust2	G_CALL_USERDATA_CUST2

Data Source Types Table 8 shows the TEvent attribute from which each attribute is derived.

Table 8: Attached Data Attribute Source Types

Attribute Name	TEvent Attribute Name
reasons	AttributeReasons
extensions	AttributeExtensions
userdata	AttributeUserData

Predefined Fields Table 9 shows the predefined IDB field in which each attribute is stored.

Table 9: Predefined Attached Data Fields

Attribute Name	G_CALL_USERDATA Field Name
customer-segment	G_CUSTOMER_SEGMENT
service-type	G_SERVICE_TYPE
service-subtype	G_SERVICE_SUBTYPE
business-result	G_BUSINESS_RESULT
customer-id	CUSTOMER_ID
transaction-id	TRANSACTION_ID

Table 9: Predefined Attached Data Fields (Continued)

Attribute Name	G_CALL_USERDATA Field Name
cause-id	CAUSE_ID
account-id	ACCOUNT_ID
destination-id	DESTINATION_ID
target-id	TARGET_ID

Custom Fields

Table 10 shows the IDB fields that ICON creates for the custom attributes that you might use in your attached data specification.

Table 10: Predefined Attached Data Fields

Attribute Name	G_CALL_USERDATA_CUST* Field Name
cust-data-1	CUST_DATA_1
cust-data-2	CUST_DATA_2
cust-data-3	CUST_DATA_3
cust-data-4	CUST_DATA_4
cust-data-5	CUST_DATA_5
cust-data-6	CUST_DATA_6
cust-data-7	CUST_DATA_7
cust-data-8	CUST_DATA_8
cust-data-9	CUST_DATA_9
cust-data-10	CUST_DATA_10
cust-data-11	CUST_DATA_11
cust-data-12	CUST_DATA_12
cust-data-13	CUST_DATA_13
cust-data-14	CUST_DATA_14
cust-data-15	CUST_DATA_15
cust-data-16	CUST_DATA_16
cust-data-17	CUST_DATA_17

Table 10: Predefined Attached Data Fields (Continued)

Attribute Name	G_CALL_USERDATA_CUST* Field Name
cust-data-18	CUST_DATA_18
cust-data-19	CUST_DATA_19

Preconfigured Keys

The history type `none` is preconfigured in ICON for the following attached data keys:

- `RTargetRuleSelected`
- `RTargetObjectSelected`
- `RTargetTypeSelected`
- `RTargetAgentSelected`
- `RTargetPlaceSelected`
- `RStrategyName`
- `RRequestedSkillCombination`
- `RTenant`

By default, ICON does not store values for these keys in the IDB history tables. If you require some or all of this data to be stored, explicitly define the respective keys in your attached data specification.

Note: When storage for the `RTargetRuleSelected` key is enabled, its values are stored in a separate table, `G_ROUTE_RESULT`.

Sample Basic Attached Data Specification

The following is an example of a basic adata_spec.xml file for voice calls.

```
<?xml version="1.0" encoding="utf-8" ?>
<adata_spec>
<public>
      <key name = "u_key1" source="userdata"    history ="all"/>
</public>
<secure>
      <key name = "u_key2" source="userdata"    history ="all"/>
</secure>
<call>
      <key name = "customer-segment" source="userdata" history ="first" field="customer-segment"/>
      <key name = "svc_class_cd" source="userdata" history ="first" field="service-type"/>
      <key name = "CCTP_CALLTYPE" source="userdata" history ="first" field="service-subtype"/>
      <key name = "cid" source="userdata" history ="first" field="customer-id"/>
      <key name = "transact_tn_final" source="userdata" history ="first" field="transaction-id"/>
</call>
<call-cust>
      <key name = "customer-segment0" source="userdata"  history ="first" field="cust-data-2"/>
      <key name = "STATE" source="userdata"  history ="last" field="cust-data-1"/>
</call-cust>
<call-cust1>
      <key name = "customer-segment1" source="userdata" history="first" field="cust-data-1"/>
      <key name = "PegTD" source="userdata"  history ="last" field="cust-data-2"/>
</call-cust1>
<call-cust2>
      <key name = "customer-segment2" source="userdata" history="first" field="cust-data-4"/>
      <key name = "vrapp_ctl_lang" source="userdata"  history ="last" field="cust-data-3"/>
</call-cust2>
</adata_spec>
```

Important! **Note:** ICON ignores duplicate keys. Only the first occurrence of a key name will be used to update the specified database table.

In the following example, only the cust_data_2 field in the G_CALL_USERDATA_CUST table will be populated by the value corresponding to key name = customer-segment (if it is present in userdata). The field cust_data_2 in table G_CALL_USERDATA_CUST1 will not be updated.

Example

```
<call-cust>
<key name = "customer-segment" source="userdata" history ="first" field="cust-data-2"/>
<key name = "STATE" source="userdata" history ="last" field="cust-data-1"/>
</call-cust>
```

```
<call-cust1>
<key name = "customer-segment" source="userdata" history ="first"
field="cust-data-1"/>
<key name = "PegTD" source="userdata" history ="last" field="cust-data-
2"/>
</call-cust1>
```

Sample Specification for Customized Attached Data

The following is an example of an `adata_spec.xml` file that has been modified for customized attached data processing.

```
<?xml version="1.0" encoding="utf-8" ?>
<adata_spec>
    <cust-disp-group-1>
        <key name = "name1_1" source="userdata" history ="first" field="cust-int-1"></key>
        <key name = "name2_1" source="userdata" history ="last" field="cust-int-2"></key>
        ...
        <key name = "name17_1" source="userdata" history ="last" field="cust-int-17"></key>
        <key name = "name18_1" source="userdata" history ="last" field="cust-str-1"></key>
        <key name = "name19-1" source="userdata" history ="last" field="cust-str-2"></key>
        ...
        <key name = "name34_1" source="userdata" history ="last" field="cust-str-17"></key>
    </cust-disp-group-1>
    <cust-disp-group-2>
        <key name = "name1_2" source="userdata" history ="first" field="cust-int-1"></key>
        <key name = "name2_2" source="userdata" history ="last" field="cust-int-2"></key>
        ...
        <key name = "name17_2" source="userdata" history ="last" field="cust-int-17"></key>
        <key name = "name18_2" source="userdata" history ="last" field="cust-str-1"></key>
        <key name = "name19_2" source="userdata" history ="last" field="cust-str-2"></key>
        ...
        <key name = "name34_2" source="userdata" history ="last" field="cust-str-17"></key>
    </cust-disp-group-2>
    ...
    <cust-disp-group-16>
        <key name = "name1_3" source="userdata" history ="first" field="cust-int-1"></key>
        <key name = "name2_3" source="userdata" history ="last" field="cust-int-2"></key>
        ...
        <key name = "name17_3" source="userdata" history ="last" field="cust-int-17"></key>
        <key name = "name18_3" source="userdata" history ="last" field="cust-str-1"></key>
        <key name = "name19_3" source="userdata" history ="last" field="cust-str-2"></key>
        ...
        <key name = "name34_3" source="userdata" history ="last" field="cust-str-17"></key>
    </cust-disp-group-16>
</adata_spec>
```

Attached Data Processing for Multimedia

This section briefly describes how ICON processes user data that is attached to Multimedia interactions (e-mail and chat).

Interaction Server Interactions

When processing data from Interaction Server, ICON checks all Multimedia Reporting Protocol events for changes to attached data. When attached data changes for a particular interaction, ICON analyzes the change and stores the data in IDB, according to its application configuration and the attached data specification.

Multimedia Interaction–Specific Data

Multimedia interaction–specific data is attached data that is associated only with a Multimedia interaction. ICON stores this data in predefined fields in separate Multimedia attached data tables in IDB.

Note: By default, Interaction Server does not automatically attach the keys that are required in order to report all the Multimedia attached data that Interaction Concentrator can support. You might need to modify your routing strategies so that Interaction Server attaches data for the required keys (for example, Suggested Response Name).

Database Schema Extensions for Multimedia Attached Data

IDB contains two tables for Multimedia-specific user data collection:

- `GM_L_USERDATA`—Stores the values of attached data keys for suggested and auto responses and acknowledgements, customer IDs, and reasons for stopping processing. ICON writes information to this table when Interaction Server reports that the interaction has finished.
  - ICON captures the names of the responses and acknowledgements from the value that the applicable keys had when ICON first received the report about the interaction from Interaction Server.
  - ICON captures the customer IDs and reason information from the final value of the applicable keys when Interaction Server reports that the interaction has ended.
- `GM_F_USERDATA`—Stores the following information about e-mail and, where applicable, chat interactions:
  - Sender (“From” name and e-mail address)
  - Called back
  - Subject

- Type and subtype
- Origination source (webform or e-mail)
- Time downloaded by e-mail server

ICON writes information to this table when Interaction Server first sends reporting events about the interaction to ICON. Therefore, the values that are stored by ICON are the first values that are presented for the applicable keys.

Tracking User Data Changes

The new memory optimization feature available in ICON 7.6.1 provides ICON with the capability to process a large number of active multimedia interactions. To do so, it removes, from operational memory, interactions that are not in the active stage of processing, and later reconstructs the interaction and attached user data for further processing.

With this feature enabled, ICON 7.6.1 tracks changes to multimedia user data in the following ways:

- Reconstructs the user data attached to interactions that were removed from operational memory. As part of the reconstruction of an interaction, ICON also reconstructs the user data content from the `EventTakenFromQueue` reporting event that was received from Interaction Server. Any changes in user data are then processed using the same rules implemented in release 7.6.
- Processes user data for interactions in the Interaction queue. While an interaction is in the Interaction Queue, Interaction Server sends `EventPropertiesChanged` reporting events to ICON for storage in IDB. Because ICON does not know whether the user data key is new or changed, ICON stores this information in IDB with an attribute `changed`.

Attached Data Specification File for Multimedia

The following is an example of an `adata_spec.xml` file that has been modified for Multimedia interactions.

```
<?xml version="1.0" encoding="utf-8" ?>
<adata_spec>
<mcr-f-data>

          <key name = " FromPersonal" source="userdata"  field="from-name"/>
          <key name = "CalledBack" source="userdata"  field="called-back"/>
          <key name = "Subject"  source="userdata"  field="email-subject"/>
          <key name = "Origination_Source" source="userdata"  field="origination-source"/>
          <key name = " FromAddress" source="userdata"  field="from-address"/>
          <key name = "attr_itx_subtype" source="attributes"  field="sub-type"/>
          <key name = "attr_itx_received_at" source="attributes"  field="received-at"/>
```

```
</ mcr-f-data>
<mcr-l-data>

          <key name = "SuggestedResponseID" source="userdata" history ="first" field="suggested-
          response"/>
          <key name = "AutoResponseID" source="userdata" history ="first" field="auto-response"/>
          <key name = " AutoAcknowledgementID" source="userdata" history ="first" field="auto-
          ask"/>
          <key name = "attr_reason_system_name" source="attributes" history ="last" field="stop-
          reason"/>
          <key name = "ContactId" source=" userdata"  history ="last" field="Multimedia-contact-
          id"/>

</ mcr-l-data>
</adata_spec>
```

Attached Data Processing for Open Media

This section briefly describes how ICON processes user data that is attached to Open Media interactions (for example, fax and web forms).

Open Media Interaction–Specific Data

Open Media interaction–specific data is attached data that is associated only with an Open Media interaction. ICON stores information about Open Media attached data in the `GM_L_USERDATA` and `GM_F_USERDATA` tables (see page 60 for more information about these tables). For more information about these or other IDB tables, see the *Interaction Concentrator 7.6 Physical Data Model* for your relational database management system (RDBMS) type.

Chapter

5 Monitoring Virtual Queues and Routing Points

This chapter describes how Interaction Concentrator monitors virtual queues and routing points, and stores this routing information in the Interaction Database (IDB). It discusses the two modes that ICON supports for virtual queue data storage, and provides recommendations about the configuration settings in the Genesys Configuration Layer that are related to virtual queue functionality.

This chapter contains the following sections:

Monitoring Route Results on Virtual Queues

Interaction Concentrator (ICON) is capable of:

- Monitoring virtual queue objects that are configured in the contact center and that are used for routing purposes. The related data is provided to Interaction Concentrator by Universal Routing Server (URS) through T-Server.
- Storing, as separate records in a special table in IDB, associations between virtual queues and interactions that are being queued.

This section describes how Interaction Concentrator processes TEvents that pertain to a virtual queue, and also what virtual queue data Interaction Concentrator stores, and how.

Note: For detailed information about virtual queue data that is available in IDB, see the *Interaction Concentrator 7.6 Physical Data Model* document for your particular RDBMS.

Storage Modes

Interaction Concentrator supports two modes for virtual queue data storage:

- One-step storage
- Two-step storage

One-Step Storage

The one-step storage mode, which is the default, forces Interaction Concentrator to combine, into a single database transaction, all data that otherwise would be stored during separate steps for record creation and record update. In this mode, ICON creates a record in the `G_VIRTUAL_QUEUE` IDB table when the association between a virtual queue and an interaction ends—that is, after ICON receives either the `EventDiverted` or the `EventAbandoned` TEvent. This is the recommended storage mode, because it minimizes the number of inserts to IDB, thus improving performance. Keep in mind, however, that interactions must be promptly delivered from a virtual queue to the agents' DNs.

Two-Step Storage

In the two-step storage mode, ICON first creates a record after receiving `EventQueued`, and the ICON updates the record after receiving either `EventDiverted` or `EventAbandoned`. This storage mode is particularly useful in an environment where interactions remain in a virtual queue for long periods of time.

Data Processing Steps

For the purpose of explaining the details of virtual queue data processing, this section describes the two-step storage mode.

Step One—Record Creation

When an `EventQueued` TEvent arrives that pertains to a particular virtual queue that is configured to be monitored by Interaction Concentrator, a new row is

inserted into the G_VIRTUAL_QUEUE IDB table. The stored data includes information, taken from both the TEvent and Configuration Database, about:

- The interaction, in the form of a T-Server–reported CallUUID, which later identifies the original interaction for reporting purposes.
- The switch through which the interaction arrived, in the form of a database identifier that Configuration Server assigned to the corresponding Switch object, if available.
- The virtual queue at which the interaction is being queued, in the form of both the number reported in the ThisDN attribute of EventQueued and the database identifier that Configuration Server assigned to the DN object corresponding to this virtual queue.

In addition, Interaction Concentrator stores:

- The status of the association. The value is 8, which signifies queued.
- The cause of the change in the virtual queue state. The value is 1, which signifies normal.
- The time at which the association was created. The value is the time at which EventQueued arrived.

Step Two—Record Update

When either the EventDiverted, or the EventAbandoned TEvent arrives and it pertains to the same virtual queue and to the same interaction for which a record has already been created, Interaction Concentrator updates that record in the G_VIRTUAL_QUEUE IDB table.

ICON updates the following data in the G_VIRTUAL_QUEUE IDB table:

- The status of the association, stored in the STATUS field, which changes to one of the following:
  - 13—Signifies diverted, if EventDiverted arrived with CallState=0, indicating that the call was diverted from this virtual queue.
  - 1—Signifies connection cleared, if either EventDiverted arrived with CallState=22, indicating that the call was diverted from another virtual queue, or EventAbandoned arrived for this virtual queue.
- The cause of the change in the virtual queue state, stored in the CAUSE field, which in the case of EventDiverted, is one of the following:
  - 1—Signifies normal. The interaction was routed to the target destination defined by the target selection object in the strategy.
  - 3—Signifies stuck. The record was processed after the interaction was stuck in a virtual queue.
  - 101—The interaction was routed in a parallel virtual queue to the target destination (see Note).
  - 102—The interaction was routed by URS to the default destination as defined by the URS 7.6 configuration options (see Note).

- 103—The interaction was routed by the switch to the default destination (see Note).
- 104—The interaction was cleared from the virtual queue by the URS strategy ClearTarget function (see Note).
- 105—Signifies other (not classified) causes reported by URS as others (see Note).
- 133—The routing interaction timeout, configured on Interaction Server, expired (see Note).
- 134— The interaction was removed (pulled out) from the strategy by Interaction Server (see Note).

Note: The extended-route-result configuration option must be set to 1 on the ICON 7.6 Application object to store this value in the CAUSE field. Universal Routing Server (URS) release 7.6 and Interaction Server release 7.6.000.18 (or higher) are also required.

- The cause of the change in the virtual queue state, stored in the CAUSE field, which in the case of EventAbandoned, changes to:
  - 2— Signifies abandoned.

ICON also adds the following data to the record:

- The identifier of the interaction that has been either diverted or abandoned, in the form of a T-Server–reported CallUUID that the interaction has on a physical device at the moment of distribution or abandonment, if available. This value later identifies the distributed or abandoned interaction for reporting purposes.
- The switch to which the interaction has been delivered or at which it was abandoned, in the form of the database identifier that Configuration Server assigned to the corresponding Switch object, if available.
- The DN to which the interaction is being distributed, in the form of a number reported in the ThirdPartyDN attribute of EventDiverted, if the interaction is diverted from this virtual queue and if the information about that DN is available.
- The time at which the association ended, which is equal to the time at which either EventQueued or EventAbandoned arrived.
  - Information about the original interaction, the switch through which it arrived, the virtual queue, and the start time of the association remain unchanged at the time of update.

Monitoring Route Results on Routing Points

If configured to do so, URS 7.6 distinguishes the routing results from interactions that are distributed from routing points or routing queues. Interaction Concentrator 7.6 can then store these "extended" routing results, that are received from URS through T-Server 7.6, in the `RESULT` field of IDB table `ROUTE_RESULT` (see Table 11).

Note: To support this feature, URS and ICON configuration options must be set as described on page 69.

Table 11: Summary of Values Stored in G_ROUTE_RESULT

Reporting Event	Value of RESULT Field	Description of Stored Results
When **`extended-route-result = 0`** (ICON release 7.5 functionality)		
EventRouteUsed	1 (normal, `ROUTE_RESULT_SUCCESS`)	The call/interaction was routed.
EventAbandoned or EventPartyRemoved	2 (abandoned, `ROUTE_RESULT_FAIL`)	The call was abandoned or the interaction was removed.
When **`extended-route-result = 1`** (ICON release 7.6 functionality)		
EventRouteUsed	1 (normal, `ROUTE_RESULT_SUCCESS`)	The call/interaction was routed by the URS strategy.
	102 (timeout)	The call was either routed to the default destination after the timeout expired, or function `TRoute[DN]` was called in the strategy to route the call to a specific DN.
	103 (routed by switch)	The call was routed by the switch to the default location.
EventAbandoned	2 (abandoned, `ROUTE_RESULT_FAIL`)	The call was abandoned.
EventPartyRemoved	1 (normal, `ROUTE_RESULT_SUCCESS`)	The interaction was routed.
	2 (abandoned, `ROUTE_RESULT_FAIL`)	The interaction was stopped
	134	The interaction was pulled out by Interaction Server.

Table 11: Summary of Values Stored in G_ROUTE_RESULT (Continued)

Reporting Event	Value of RESULT Field	Description of Stored Results
EventPartyRemoved (continued)	133	Routing timeout, defined on Interaction Server, expired.
	105	While URS attempted to route interaction, the connection between Interaction Server and URS was lost.

Note: For more information about the `extended-route-result` configuration option, see the *Interaction Concentrator 7.6 Deployment Guide*. For more information about IDB tables, see the *Interaction Concentrator 7.6 Physical Data Model* for your relational database management system (RDBMS) type.

Reliability Flag

ICON uses a reliability flag stored in the `GSYS_EXT_INT1` field in the `G_ROUTE_RESULT` table. This flag indicates the reliability of the virtual queue ID information written in the `GSYS_EXT_VCH1` field in the `G_ROUTE_RESULT` table (see Table 12).

Table 12: Reliability Flag Values

Value	What Value Indicates
Voice Calls/Interactions	
`1` (ok)	The virtual queue ID (VQID), stored in the `GSYS_EXT_VCH1` field, is valid.
`2` (valid in past)	The virtual queue ID (VQID), stored in the `GSYS_EXT_VCH1` field, is valid in the past.
`0` (unknown)	There is no virtual queue ID information in routing related notifications.

Configuration Recommendations

This section provides information about the configuration settings in the Genesys Configuration Layer that are related to virtual queue functionality.

The default configuration settings enable the storage of virtual queue data, provided that your releases of both Interaction Concentrator and URS support virtual queue functionality.

Configuration settings on the ICON `Application` object, the virtual queue `DN` object, and the `Switch` object enable you to manipulate virtual queue monitoring in the following ways:

- Change the storage mode of Interaction Concentrator.
- Disable monitoring and data storage for a particular virtual queue.
- Disable monitoring and data storage at the switch level—that is, for all virtual queues that belong to a particular switch.

Universal Routing Server

Although a URS release that supports virtual queue functionality is necessary in order to enable virtual queue monitoring in Interaction Concentrator, no special configuration is required on the URS side.

Beginning in release 7.6, URS can now provide additional information to ICON regarding the reason for routing an interaction using the `AttributeReason` of routing requests. URS can also attach information to interactions about the targets for which it is waiting (see "Monitoring Route Results on Routing Points" on page 67). To make this information available to Interaction Concentrator for downstream reporting purposes, set the following configuration options to `true` on the URS `Application` object:

- `report_reasons`
- `report_targets`

For more information about these URS configuration options, see the *Universal Routing 7.6 Reference Manual.*

ICON Application

The default settings enable ICON to receive virtual queue data and store it in IDB.

Connections Although a URS release that supports virtual queue functionality is necessary in order to enable virtual queue monitoring in Interaction Concentrator, ICON receives the data from T-Server. Therefore, no connection to URS is required.

vq-write-mode The `vq-write-mode` configuration option enables you to switch the storage mode for virtual queue data, if necessary. For descriptions of the storage modes, see "Storage Modes" on page 64. You configure the `vq-write-mode` option in the `callconcentrator` section on the `Options` tab of the ICON `Application` object.

extended-route-result The `extended-route-result` configuration option specifies whether ICON stores extended routing results (from URS) in IDB. You configure

`extended-route-result` in the `callconcentrator` section on the `Options` tab of the ICON `Application` object.

Note: To store extended route results in IDB, ICON requires Universal Routing Server (URS) release 7.6 and Interaction Server release 7.6.000.18 (or higher).

For more information about these options, see the configuration options chapter in the *Interaction Concentrator 7.6 Deployment Guide.*

Virtual Queue DN

Unless you need to disable monitoring and data storage for a particular virtual queue, no configuration is necessary for the `DN` object that represents this virtual queue in the Configuration Layer.

monitor The `monitor` configuration option enables you to turn off the ICON monitoring and data storage for a particular virtual queue, if necessary. If the option is set to `0`, ICON does not register with T-Server to receive events that pertain to this virtual queue. You configure the `monitor` option in the `gts` section on the `Annex` tab of the `DN` object that is configured for this virtual queue. For more information, see the option description in the *Interaction Concentrator 7.6 Deployment Guide.*

Switch

Unless you need to disable virtual queue monitoring and data storage for a particular switch, no configuration is necessary for the corresponding `Switch` object in the Configuration Layer.

support-dn-type-5 The `support-dn-type-5` configuration option enables you to turn off Interaction Concentrator monitoring and data storage for all virtual queues that belong to a particular switch, if necessary. If the option is set to `0`, ICON does not register with T-Server to receive events that pertain to virtual queue DNs that belong to this switch. In this case, ICON does not process or store virtual queue–related TEvents, even if the `monitor` option is set to `1` for any of the virtual queues that belong to the switch. You configure the `support-dn-type-5` option in the `gts` section on the `Annex` tab of the `Switch` object that is configured for this switch. For more information, see the option description in the *Interaction Concentrator 7.6 Deployment Guide.*

Chapter

6 Agent States and Login Sessions

Agent state data in Interaction Concentrator provides detailed information about agent activity, within the context of agent login sessions. This chapter describes how to make data about agent states and login sessions available in IDB.

This chapter contains the following sections:

Agent and Login Session Models

This section introduces the terminology, elements (objects), and models that pertain to Interaction Concentrator (ICON) data about agent activities.

This section contains information about the following:

Agent and Login Session Objects

The following terms refer to the agent and login session objects and elements about which ICON stores reporting data:

- *ACDQ*—An Automatic Call Distribution (ACD) queue object (ACD device or ACD group).

- *Agent state*—The state that an agent may take in relation to the interactions that are associated with the agent. For voice interactions, it is also the state that an agent may take in relation to an ACDQ.

  ICON obtains information about agent state changes through agent state event reporting. ICON tracks agent states against different media types independently.
- *Login session*—The period of time during which the agent was logged in to an ACDQ or other endpoint on a particular switch.
- *Media type*—The communication medium applicable for the agent's interactions (for example, e-mail or chat).
- *Pending agent state*—The agent state to which the agent will transition after the Busy *state.*
  - When the agent transitions from any state to Busy, ICON keeps the pre-transition state as the pending state.
  - If ICON receives information about an agent state change while the agent state is Busy, ICON keeps the new state as the pending state.
  - When the Busy state ends, ICON restores the agent's state from the pending state.
- *Reason*—The reason for the agent state change.

  For voice calls, ICON obtains reasons for agent states from T-Server reporting, from any of the following sources:
  - The workmode parameter
  - TEvent Attribute Extensions (known as hardware reasons), from the value of predefined ReasonCode keys
  - TEvent Attribute Reason (known as software reasons), from the values of one or more key-value pairs

  For Multimedia, the Interaction Server Multimedia Reporting events may include attributes that contain information about the reason for the agent state change. ICON stores this reason code information in the same way that it stores hardware reasons. There can be only one reason for each agent state–related event that Interaction Server reports.

Agent State Model—Voice and Multimedia

Agent-Orientated Agent State Model

Interaction Concentrator supports an agent-orientated agent state model. In this model, the switching function maintains a single state for the agent for each media type (voice, e-mail, or chat) within the agent login session, regardless of the number of endpoints or ACDQ objects with which the agent is associated. The agent can have a different state in relation to each media type.

Agent States

Table 13 lists the agent states, described in relation to voice calls. The ACDQ is not applicable to Multimedia, but the agent states for Multimedia media types are equivalent. Table 13 also provides the T-Server or Interaction Server (MM) reporting events that represent entry into each state.

Table 13: Agent States and State Transition Events

Agent State	Description / State Entry Event
Null	The agent is not logged on to the ACDQ at a particular device. Logging on to the ACDQ causes a transition from this state. Similarly, logging off from the ACDQ causes a transition to this state. TEvent • EventAgentLogout—Reports the agent state change for all ACDQ objects to which the agent was logged on. • EventQueueLogout—Reports the agent state change for a particular ACDQ. MM Event • EventAgentLogout—Reports the agent state change for all endpoints and media types at the place to which the agent was logged on. • EventMediaRemoved—Reports that the agent is no longer logged on for the specified media type.
Login	The agent is logged on to the ACDQ at a particular device and is ready to contribute to the activities of the ACD queue. The state does not indicate that the agent is ready to accept ACD calls (see Ready). TEvent EventAgentLogin **Note:** The presence or absence of the ThisQueue parameter in the TEvent specifies whether the agent has logged on to an ACD queue. MM Event EventAgentLogin **Note:** If the event includes information about media types, a separate record is created for each media type, and the appropriate agent state for each media type is set.
NotReady	The agent is logged on to the ACDQ at a particular device but is not ready to handle interactions distributed from the ACD queue. An agent in this state can receive calls that are not ACD calls. TEvent EventAgentNotReady, with the workmode parameter not equal to AgentAfterCallWork MM Event • EventNotReadyForMedia • EventMediaAdded—Provides information about a media type that was added to the agent's login session. **Note:** ICON obtains information about the agent state from the event attribute. In Multimedia, the agent state specified for the EventMediaAdded event is always NotReady.

Table 13: Agent States and State Transition Events (Continued)

Agent State	Description / State Entry Event
Ready	The agent is logged on to the `ACDQ` at a particular device and is ready to handle ACD interactions, even though the agent might be involved in non-ACD calls. TEvent `EventAgentReady` MM Event `EventReadyForMedia`
Busy	The agent is logged on to the `ACDQ` at a particular device and is involved in an existing call at the device. The call may be on hold at the device. There is no specific entry event for this agent state. This state covers the period during which the agent is involved with the interaction, until the agent is disconnected. In addition to ACD interactions, calls between agents, calls between supervisors and agents, and private calls can also cause transition to `Busy` state.
ACW	The agent is no longer connected to a call but is still occupied with work related to the previous call (for example, the agent might be performing administrative duties, such as updating a business order form). In this state, the agent cannot receive ACD calls but may be able to receive non-ACD calls. **Note:** ICON cannot obtain call information from the T-Server event. ICON keeps information about the last call represented on the agent's device before the transition from `Busy` state to any other state. In cases where `ACW` state follows `Busy` state, ICON assigns the last call on the agent's device as related to `ACW` state. TEvent `EventAgentNotReady`, with the `workmode` parameter equal to `AgentAfterCallWork` MM Event Not applicable
Unknown	An agent's login session exists, but ICON has no information about the agent state (for example, because of disconnection from T-Server).

Note: Interaction Concentrator release 7.5 and higher also supports custom agent states. For more information, see Chapter 8 on page 93.

Agent State FSM

Figure 3 illustrates the finite state machine (FSM) for the agent state for voice calls, within a login session. Except for the `AfterCallWork` state and the names of the state transition events, the FSM for the agent state for Multimedia is identical.

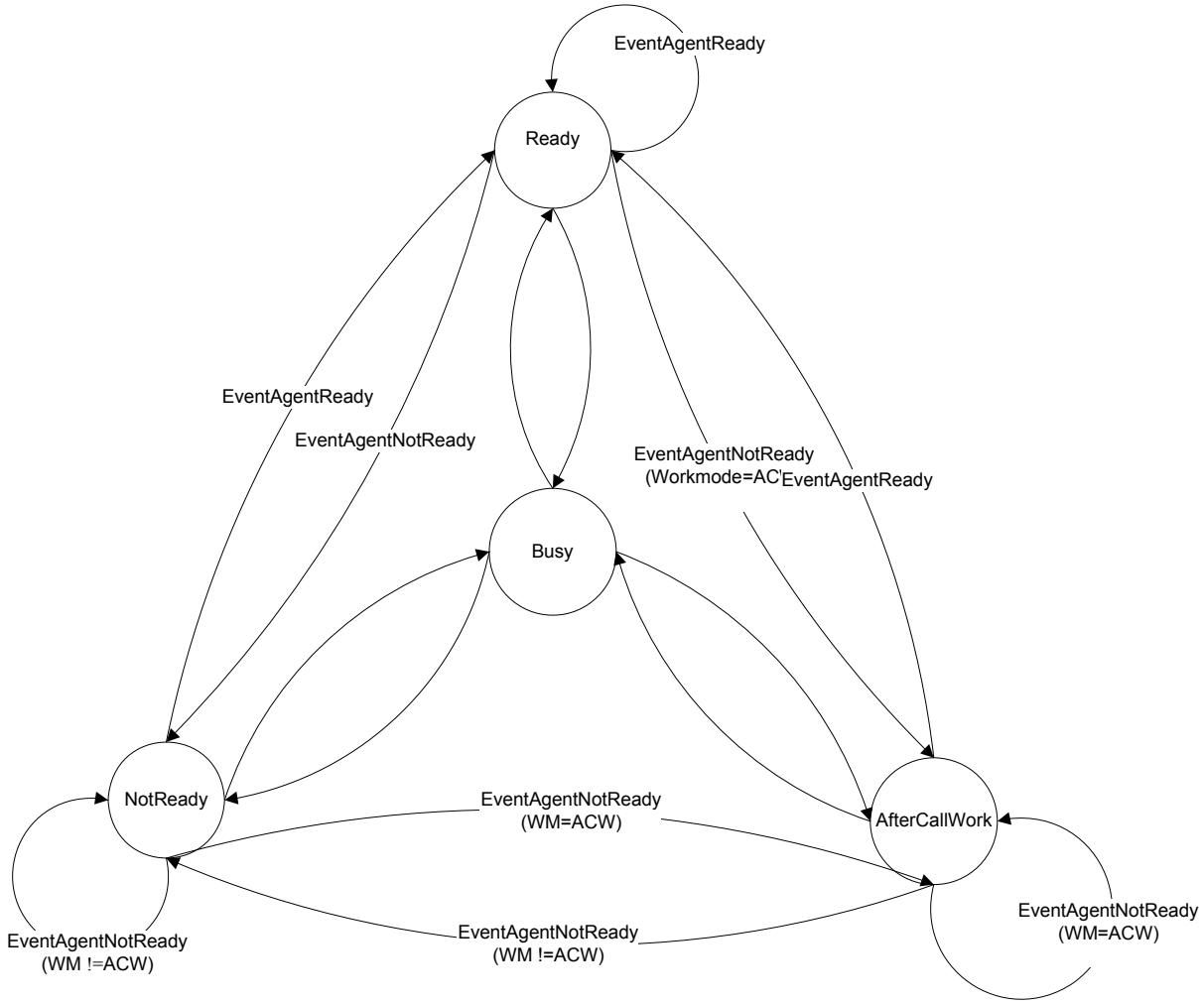

Figure 3: Agent State FSM

Agent State Restoration

On startup or reconnection, when ICON registers with T-Server to start monitoring agent states, the `EventRegistered` TEvent provides ICON with a snapshot of the current agent state for voice calls, as well as a list of the calls that were distributed by T-Server and presented on the agent's desktop.

In a Multimedia solution, when ICON registers with Interaction Server to start monitoring agent states, the *EventPlaceAgentState* reporting event similarly provides ICON with a snapshot of the current agent states for the various media, as well as a list of the interactions that were distributed by Interaction Server and presented on the agent's desktop.

Agent State Model—SIP Chat

Interaction Concentrator processes the `chat` media type in agent state-related events received from SIP Server. Data extracted from these events is stored in

existing IDB tables. Within these tables, the media type is stored as an integer code in the `GSYS_EXT_INT1` field.

State-Related Events

Interaction Concentrator supports a simple agent model within each media type. An agent on a DN can have a different state for each media type (for example, the agent state can be `Busy` for voice but `Ready` for chat).

SIP Server does not provide information about the media type in the agent's state–related events (`EventAgentLogin, EventAgentReady, EventAgentNotReady, EventAgentLogout`). As result, ICON cannot determine the media type from an agent's state related event and therefore processes information from received events as applicable to *all* media types associated with agent's login session.

Party State Changes

Interaction Concentrator can determine the media type from the calls related to the call party. ICON will process party state changes against agent's state created for this media type only. Agent's state for any other media types is not affected by this party state change.

DND State Changes

SIP Server does not provide information about the media type in the `EventDNDOn` and `EventDNDOff` events. Interaction Concentrator processes and stores information about changes in DND states regardless of the media type. If more than one media type is configured on a DN, and the agent state is different for each media type, the agent state in the related record will be `LoggedIn`.

ThisQueue

Interaction Concentrator processes and stores information about the `ThisQueue` attribute for voice media interactions. If the voice media type is disabled on a DN, this attribute will be processed for the chat media type instead.

Login Sessions Model

An agent's login session starts when ICON receives the first `EventAgentLogin` event for that agent, either after a period of time during which the agent was not logged on to the switch at all, or after a configurable period of time during which information about the agent was not available (see the description of the `gls-max-inactivity` configuration option in the *Interaction Concentrator 7.6 Deployment Guide*).

If an agent is already logged on to the switch and then either logs on to additional `ACDQ` or endpoint objects, or a media type is added or removed, ICON continues the existing login session.

An agent's login session ends when the agent is no longer logged on to the `ACDQ`, or if there has been no agent-related activity within the configured maximum inactivity interval.

Figure 4 represents the login session FSM.

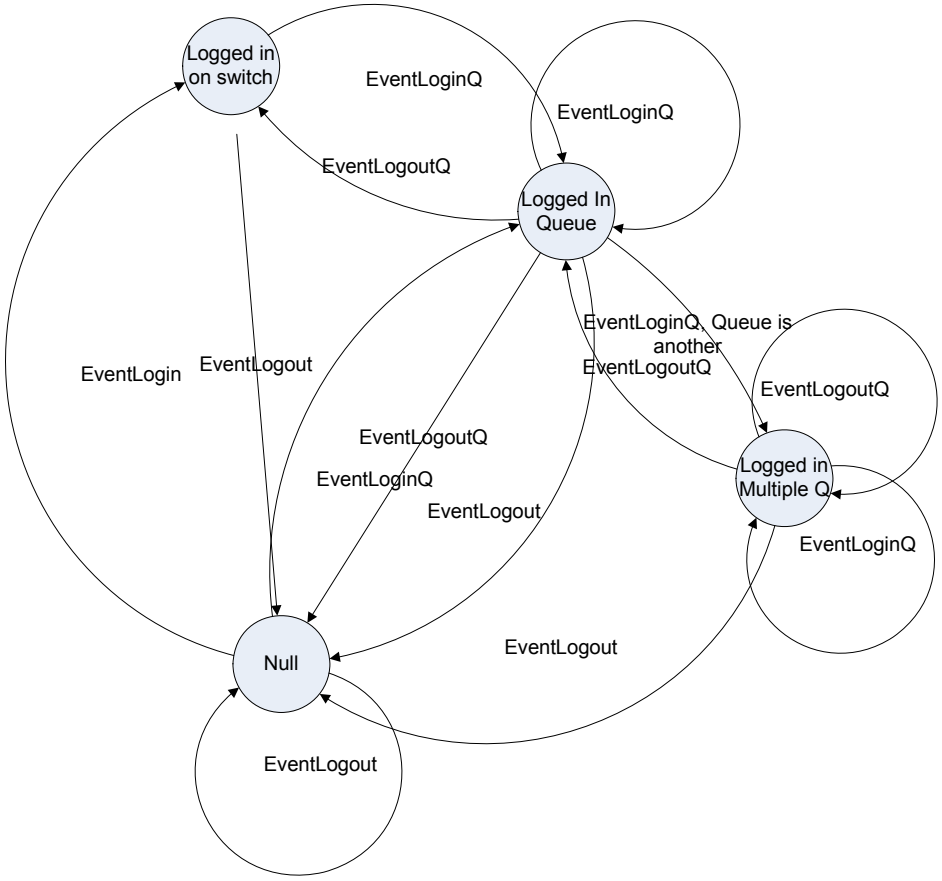

Figure 4: Login Session Finite State Machine

Available Agent State and Login Session Data

This section describes the kinds of agent state and login session data that ICON captures, provided that ICON has been configured to perform this role.

For a high-level summary of the IDB tables in which ICON stores data about agent states and login sessions, see "Agent State–Related and Login Session–Related Tables" on page 31. For more detailed information, see the *Interaction Concentrator 7.6 Physical Data Model* document for your particular RDBMS.

Agent State and Login Session Data

ICON stores both actual and historical data about login sessions and agent state changes, as well as the associations between agent states, login sessions, and endpoints. ICON tracks changes against different media types independently.

The following data is available:

- Current and historical information about agent login sessions.

- Current and historical information about the associations between login sessions and endpoints.

  Note: Multimedia Reporting events include information about the agent place, but not about agent endpoint DNs. Therefore, ICON records in the agent activity–related tables do not include meaningful endpoint information for Multimedia.

- Detailed information about agent state changes during the agent login session, including:
  - Changes to the agent's state, pending state, workmode, and hardware reason code.
  - Time of the state change.
  - Other party connections and disconnections.
- Detailed information about the hardware and software reason code changes during the agent login session.

  Note: If Multimedia Reporting events include information about reason codes for agent state changes, ICON stores the reason code in the same way that it stores hardware reason codes that are provided by T-Server reporting for voice. Hardware reason codes do not apply to Multimedia.

- Detailed information about usage of the DND feature within the agent login session. For voice, DND can be activated and deactivated for individual DNs. For Multimedia, DND can be activated and deactivated only for an entire place.

Precalculated Agent State Metrics

In addition to raw object data, ICON stores the following agent state–related statistics:

- Duration of agent state:
  - `Duration_ready`
  - `Duration_notready`
  - `DurationACW` (not applicable to Multimedia)
  - `DurationBusy`
- Duration of agent workmode (not applicable to Multimedia):
  - `Duration_UNKNOWN`
  - `Duration_AUX`
  - `Duration_LegalGuard`
  - `Duration_GoAway`
  - `Duration_ReturnBack`

After-Call Work and Not-Ready Agent States

After-call work (ACW) is the work that is required of an agent immediately following an inbound call, such as entering data, or making internal calls to complete the transaction. The agent is considered unavailable to receive another inbound call while in this mode. This section describes the functionality of the after-call work and not-ready agent states in Interaction Concentrator.

Uninterrupted ACW or Not-Ready Duration

Interaction Concentrator provides the capability to continue the after-call work or `NotReady` agent state without any interruption due to the agent placing or receiving calls during the after-call work or not-ready period. ICON recognizes completion of after-call work or the `NotReady` agent state when any of the following occur:

- The agent logs out.
- The agent places himself/herself in `Ready` mode.
- The agent goes `NotReady` for any reason other than to perform after-call work. (This includes indirect work mode changes such as when the agent walks away from his desk for a period of time.)

To enable this uninterrupted `ACW` or `NotReady` agent state functionality, the `gls-enable-acw-busy` configuration option must be set to `false` on the switch `Annex` tab.

Interrupted ACW or Not-Ready Duration

Interaction Concentrator also provides the capability to interrupt the after-call work or not-ready agent state when the agent places or receives calls during the `ACW` or `NotReady` period. By default, ICON interrupts `ACW` or `NotReady` agent states while an agent is handling another call. You can also set the `enable-acw-busy` configuration option to `true` on the switch `Annex` tab to enable this behavior. See the configuring options chapter in the *Interaction Concentrator 7.6 Deployment Guide* for more information on setting this option.

Associating ACW with an Interaction

Interaction Concentrator can associate after-call work with the voice interaction that immediately precedes the *start* of the after-call work (the first voice interaction), or with the last interaction, which is the default behavior.

In the first case, subsequent voice interactions that occur during the period of after-call work are considered as related to ACW processing and do not interrupt measurement of ACW-related metrics.

To support this functionality, the `gls-acw-first` configuration option can be set either at the switch level or on the ICON `Application`. ICON uses the value set on the `Application` for all switches except the SIP switch. For SIP switches, the option must be specified on the switch level. See the configuring options chapter in the *Interaction Concentrator 7.6 Deployment Guide* for more information.

Chapter

7 Integrating with Outbound Contact

This chapter describes how to make data from the Genesys Outbound Contact solution available in the Interaction Database.

This chapter contains the following sections:

Outbound Objects and Models

This section introduces the terminology, elements (objects), and models that pertain to Genesys Outbound Contact.

Outbound Contact Objects

The following terms refer to the Outbound Contact objects and elements about which Interaction Concentrator (ICON) stores reporting data:

- *Campaign*—A master plan for managing customer data, generating calls, and monitoring and handling call results. Outbound Contact uses campaigns to automate outbound dialing.
- *Campaign Group*—A runtime association among a campaign, a calling list, and a group of resources (such as agents) that is assigned to handle the campaign activity.
- *Calling List*—A database table made up of records that contain customer data, which could have dialing filters applied.

- *Record*—The basic unit of customer data. Each record represents one customer phone number.
- *Field*—A single piece of data (for example, a phone number) within a record. A calling list includes mandatory fields, which are required in order for the Outbound Contact solution to operate. A calling list can also include custom fields.
- *Chain*—Records that are logically united, usually representing the same customer contact information. For example, three records with the home, business, and cell phone numbers for the same customer comprise a chain of records.
- *Format*—A template that organizes the data in a calling list. When calling lists are created and populated with records, the customer data in the records fills the fields and conforms to the field properties that are established in the format, according to either the system default setting or a user-defined setting.

Note: For more detailed information about Outbound Contact objects, see the *Outbound Contact 7.6 Deployment Guide.*

Campaign Model

Figure 5 shows the finite state machine (FSM) for a campaign group.

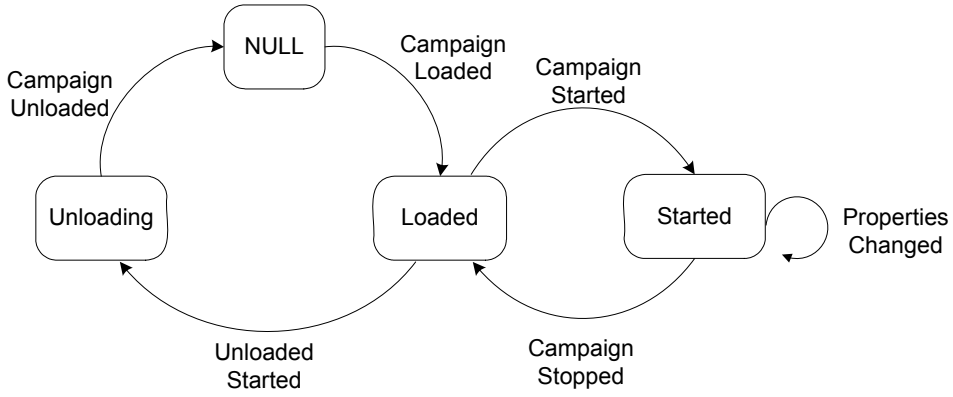

Figure 5: Finite State Machine for a Campaign Group

Chain Model

Figure 6 shows the FSM for a chain.

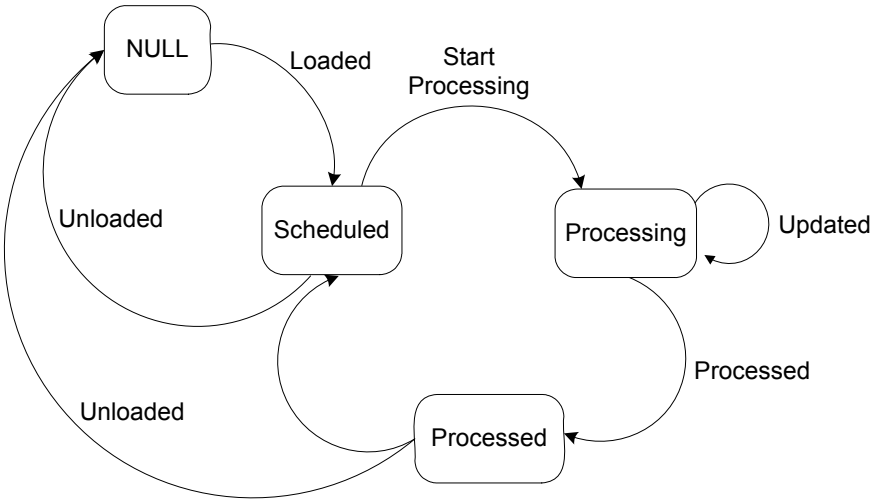

Figure 6: Finite State Machine for a Chain

Chain States

In Figure 6, the chain states are as follows:

- *NULL*—The initial state, in which no records for this chain have yet been selected from the database. This is also the final state, in which all records in the chain are either finalized or unloaded, and updated in the database.
- *Processing*—The state in which the processing of a record from the chain has started, and resources for this processing have been allocated.
- *Processed*—The state in which a record is processed. Depending on the actual processing result, the record(s) in the chain are finalized or rescheduled.
- *Scheduled*—The state in which a record in the chain is scheduled for processing, either because the chain has been loaded from the calling list or because of the previous processing attempt. The chain can be scheduled for processing either at a specified date and time or *for now*—that is, as soon as resources for processing the chain are available to Outbound Contact Server (OCS).

Available Outbound Data

This section describes how ICON communicates with OCS and what kind of data OCS can send to ICON, provided that both OCS and ICON are properly configured.

ICON and OCS Communications

ICON connects to OCS at the port that OCS opens for regular client connections. Unlike other OCS clients, ICON uses a special protocol to receive reporting-related data. The OCS data includes runtime information about OCS objects, as well as certain statistics that OCS calculates specifically for reporting purposes.

If two or more OCS instances are running simultaneously in a given contact center, a single ICON instance can process and store data from either a single OCS instance or multiple OCS instances.

Outbound Objects Data

ICON stores campaign and chain information, including both their history within a given campaign session and the associations between calls and parties, when this data is available.

Specifically, you can integrate ICON with OCS so that the following data is stored in IDB for outbound reporting purposes:

- History of changes in campaign configuration properties, and the target values of optimization parameters.
- History and results of chain processing.
- History of changes that OCS makes in calling list records during chain processing, and that ICON is configured to track.
- Results of each attempt to generate an outbound call, whether successful or not.
- Call progress detection (CPD) call results.
- Call results assigned by an agent, which are stored independently from the CPD call results, and which do not overwrite them.

Note: For the full list and descriptions of IDB tables that store outbound data, see the *Interaction Concentrator 7.6 Physical Data Model* document for your RDBMS.

Precalculated OCS Metrics

In addition to raw object data, you can configure OCS to calculate, and ICON to store, a set of outbound-related statistics, which are also referred to as *precalculated metrics.*

Table 14 lists the precalculated metrics that OCS provides to ICON for storage in the `GO_METRICS` IDB table.

Table 14: Precalculated OCS Metrics

Metric	Description
OCS calculates the following subset of metrics for all currently active campaign groups, and for all calling lists that participate in the active campaign groups. OCS delivers these metrics to ICON in a single data packet every 10 minutes.	
Total number of records per calling list	Number of physical records in the calling list database view (for example, records from the database table and, possibly, the `WHERE` clause from the dialing filter).
Total number of records per campaign group	Number of physical records in all database views whose calling lists are assigned to the campaign that participates in the campaign group.
Total number of chains per calling list	Number of logical chains in the calling list database view (for example, records from the database table and, possibly, the `WHERE` clause from the dialing filter). A *logical chain* is defined as one or more database table records with the same `Chain ID` value.
Total number of chains per campaign group	Number of logical chains in all database views, whose calling lists are assigned to the campaign that participates in the campaign group.
Current number of records not processed per calling list	Number of physical records in the calling list database view (for example, records from the database table and, possibly, the `WHERE` clause from the dialing filter) that have the type `GENERAL` and the status `READY`.
Current number of records not processed per campaign group	Number of physical records in all database views that have the type `GENERAL` and the status `READY`, and whose calling lists are assigned to the campaign that participates in the campaign group.
Current number of chains not processed per calling list	Number of logical chains in the calling list database view (for example, records from the database table and, possibly, the `WHERE` clause from the dialing filter), in which at least one record of the chain has the type `GENERAL` and the status `READY`.
Current number of chains not processed per campaign group	Number of logical chains in all database views, in which at least one record of the chain has the type `GENERAL` and the status `READY`, and whose calling lists are assigned to the campaign that participates in the campaign group.
Current number of busy dialer ports per campaign group	Current number of busy (that is, occupied with outbound calls that are in the process of being placed) CPD Server ports or Switch call classifier ports.

Table 14: Precalculated OCS Metrics (Continued)

Metric	Description
Current number of engaged (busy) ports per campaign group	Current number of ports used by engaged calls that CPD Server already placed. **Note:** OCS calculates this metric only for campaign groups that are activated in Active Switching Matrix (ASM) dialing modes with CPD Server.
Outbound Calls Overdialed	Indicates that Outbound Dialing Engine is to consider a given call overdialed. For information about overdialed calls, see the Outbound Contact documentation. **Note:** OCS calculates this metric for all currently active campaign groups. When a call is overdialed, OCS delivers this metric to ICON. OCS does not deliver this metric at a preset interval, but instead calculates it and passes it to ICON on a *per-occurrence* basis (that is, for each overdialed call).
OCS calculates the following subset of metrics for all currently active campaign groups. As a result of each successful dial attempt, OCS delivers these metrics to ICON in the form of a single snapshot event. OCS does not deliver these metrics at a preset interval, but instead calculates them and passes them to ICON on a *per-occurrence* basis (that is, for each successfully dialed outbound call).	
Outbound Call Dialing Time	Duration of dialing, in milliseconds.
Outbound Call Transfer Time	Duration of a call transfer, in milliseconds.
CPD Time	Duration of post-connect CPD, in milliseconds.

Outbound Contact Deployment Scenarios

Interaction Concentrator supports all types of Outbound Contact deployments and, most importantly, stores outbound data consistently, regardless of the deployment scenario.

This section discusses two main deployment scenarios:

- Single-site deployment in which one ICON instance is dedicated to handling outbound data.
- Multi-site deployment in which a single OCS instance serves all sites, and in which there is a single ICON instance for each site.

Diagram Conventions

To simplify the deployment diagrams in this section:

- DB Server, which enables a connection between ICON and IDB, is omitted from the diagrams, although it is required in actual deployments (see Figure 1 on page 20).
- Storage of configuration data is not shown, although it is required in actual deployments.

Single-Site Deployment Example

The simplest deployment scenario, which is suitable for single-site contact centers, consists of a single ICON instance that is dedicated to storing all outbound data to a single IDB instance (see Figure 7).

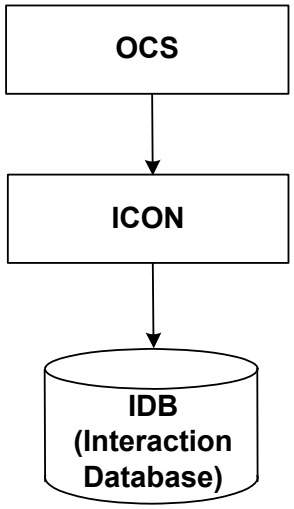

Figure 7: Single-Site Deployment: A Single OCS Instance and a Dedicated ICON Instance

In this scenario, ICON is dedicated to storing outbound data, including both object information and precalculated metrics. This ICON instance does not have a connection to T-Server or, in a Multimedia solution, to Interaction Server, nor does it gather interaction data from T-Server or Interaction Server. It is assumed that another ICON instance stores CTI–related data and interaction data to either the same IDB or a separate IDB.

Multi-Site Deployment Example

The simplest multi-site deployment scenario consists of ICON instances that run at various contact center sites. All of these ICON instances store outbound data that they receive from the same OCS instance, and each of them also stores CTI data that it receives from its particular T-Server or Interaction Server.

Figure 8 shows a simple multi-site deployment scenario. To simplify the diagram, the scenario shows a deployment for voice calls only, and it omits the

IDB deployment. The relationship between the OCS instance and the Interaction Server in a Multimedia solution is the same as the relationship between the OCS instance and the T-Server in a voice solution.

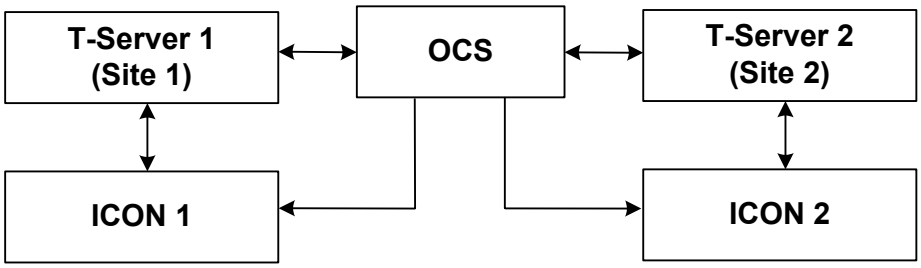

Figure 8: Multi-Site Deployment: A Single OCS Instance and Multiple ICON Instances

In this scenario:

- ICON 1 stores:
  - Data related to outbound activity generated by OCS at Site 1 only.
  - CTI data related to call activity reported by T-Server 1, which is serving Site 1.
- ICON 2 stores:
  - Data related to outbound activity generated by OCS at Site 2 only.
  - CTI data related to call activity reported by T-Server 2, which is serving Site 2.

Note that if a call originates at Site 1 but it is being handled by an outbound agent at Site 2, ICON 1 reports the outbound-specific data, whereas ICON 2 reports the voice (CTI) activity that is associated with this call.

Matching Outbound and CTI Data in IDB

To match CTI or interaction data (reported by T-Server or Interaction Server) and outbound data (reported by OCS) for the same interaction, use the call attempt identifier (`CallAttId`) that ICON stores in the `GOX_Chain_Call` table.

Note: This functionality is not available with Outbound Contact Server (OCS) release 7.2.

OCS Reporting Limitation

OCS is not able to track certain types of outbound calls when the Campaign Group is running in the following dialing modes:

- `Preview` mode—Calls that are dialed by the Agent Desktop
- `Power GVP` mode—Calls that are dialed by GVP Dialer

- `Push Preview` mode—Interactions that are pushed to the Agent Desktop from Interaction Server

Chain-related events for these types of calls do not contain the unique identifier (Call GUID) of the outbound call(s) dialed during chain processing. As a result, ICON does not receive the information it needs to create a record in the `GOX_Chain_Call` table.

Workaround As a workaround, you can use the call attempt GUID to bridge data about outbound chain activity with data about telephony activity. OCS reports the call attempt GUID in chain-related events, and T-Server and Interaction Server include it in the user data of outbound calls.

Configuration Recommendations

In order to store outbound data in IDB for reporting purposes, certain configuration settings are required in the Genesys Configuration Layer, both for certain outbound-related configuration objects and for the ICON `Application` object.

Outbound Contact

Special configuration is required in order to enable OCS to process and send data to ICON about the content of the fields in calling list records.

Field Object

For every `Field` configuration object that describes a single piece of data (for example, a phone number) within a record, you must configure the `icon_attribute` option if you want that data to be stored in IDB.

To configure this option:

1. Go to the Configuration Manager main window.
2. Open the `Properties` dialog box for a particular `Field` configuration object.
3. Click the `Annex` tab.
4. Create a new section named `default`, if it does not already exist.
5. Within this section, create a new option named `icon_attribute`.
6. Set the option to one of the following values:
   - 1—To store OCS mandatory fields in the `GO_RECORD` table, custom defined fields in the `GO_CUSTOM_FIELDS` table, and history of field changes in `GO_FIELDHIST` table.
   - 2—To store data as a secured field in the special `GO_SECURE_FIELDS` and `GO_SEC_FIELDHIST` IDB tables.

If you do not configure this option, or if you set its value to `0` (zero), OCS does not process data that is related to this `Field` object and ICON receives no such data.

For more information, see the Outbound Contact 7.6 documentation.

OCS Application Object

For every `OCS Application` object that is required to select and send data from the fields in a calling list mentioned in the preceding section, configure the `send_attribute` option. For more information, see the section about field-level options in the Outbound Contact configuration options chapter in the *Outbound Contact 7.6 Deployment Guide.*

Campaign Group Object

To enable reporting for all the activity associated with a Campaign Group, including chain activities, ensure that the `Campaign Group` object's configuration properties specify a valid Voice Transfer Destination DN. The DN must be located on the switch served by the T-Server to which OCS is connected, and the T-Server must have a CTI link connected with the switch.

ICON Application

To enable ICON to receive OCS data and store it in IDB, you must configure ICON connections to appropriate OCS instances, and you must set relevant configuration options.

Connections

- In an environment with a single OCS instance, add the `OCS Application` object to the `Connections` tab of the `ICON Application` object.
- In an environment with multiple OCS instances, decide on your deployment topology—that is, decide whether a single ICON instance will handle the data from all or a subset of OCS instances, or whether each OCS will have a dedicated ICON instance. Based on your deployment decision, add one or more `OCS Application` objects to the `Connections` tab of the `ICON Application` object that must store data from those OCS instances.

Outbound-Related Options

Role Configuration Option

For every ICON instance that must store outbound data, make sure that the `role` option on the `Options` tab of the ICON `Application` object includes `gos` in the list of values. If you deploy a single ICON instance for the entire contact center, you can keep the default value (`all`). For more information, see the

description of the role option in the *Interaction Concentrator 7.6 Deployment Guide.*

If you store different types of data to different IDBs, make sure that gos is also specified for the role option on the Options tab of the appropriate Database Access Point (DAP). Configure this option on the Options tab of the Application object for the DAP that your ICON instance uses to store outbound data to IDB. For more information, see the description of the role option in the *Interaction Concentrator 7.6 Deployment Guide*.

OCS-Specific Options

The following ICON configuration options enable you to specify what outbound data ICON should store, and in what manner:

- gos-write-duplicate-metrics
- gos-write-metrics
- gos-write-metrics-only

Review the descriptions and values for the gos-write configuration options in the *Interaction Concentrator 7.6 Deployment Guide.* Select the appropriate values for your environment, and make related configuration changes on the Options tab of the ICON Application object.

Multi-Tenant Environment

For multi-tenant deployments, add the following configuration options to the Annex tab of the Environment tenant.

- icon_attribute—Depending on whether you want to store data in secure or unsecure tables, set this option to 1 or 2. (See also "Field Object" on page 89 for a description of all possible values.)
- send_attribute

Note: If the Environment tenant does not appear on the ICON Application object, you must add it before you can configure tenant-specific options.

By default, the tenant's Annex tab is not displayed in Configuration Manager. To display the tenant's Annex tab:

1. Select View > Options.
2. Select Show Annex tab in Object Properties check box.
3. Click OK.

Chapter

8 Processing User Events and Custom-Defined States

Interaction Concentrator support for customer-defined states is designed to provide compatibility with Call Concentrator functionality in legacy deployments. This feature enables customers to use existing investments in customized desktop applications.

This chapter describes how to make data about custom agent states and common data that is attached to voice calls available in Interaction Concentrator.

This chapter contains the following sections:

Custom States in Interaction Concentrator

The term *custom states* refers to customer-defined states of endpoints. If it has been configured to do so, Interaction Concentrator (ICON) supports the processing of data from the T-Server `EventUserEvent`, in order to store associations between:

- Common data and the voice call (or the call party, when applicable).
- Custom states and the voice call (or the call party, when applicable).

Both an active call and the last call (the call that ended immediately before the start of the custom state or the arrival of `EventUserEvent`) on the device can participate in the association.

Storing Data from EventUserEvent

ICON processes data from EventUserEvent separately from call user data.

Common Data

If it has been configured to support custom states, ICON stores common data from EventUserEvent in two tables that are created by the ICON installation script: G_CUSTOM_DATA_S and G_CUSTOM_DATA_P. You can configure unique keys to store values of customer-defined keys. Duplicate key names in the attached data is not supported.

Note: Interaction Concentrator release 7.5 and 7.6 do not support storage of common data in customer-created tables or additional customer-created fields.

Custom States

ICON stores data related to custom states from EventUserEvent in the G_CUSTOM_STATES table, which is created by the ICON installation script.

ICON writes information to IDB when the custom state is finished. To avoid problems arising from stuck custom states, ICON clears all active custom states when the agent's login session is terminated.

For more information about the custom data and custom state tables, see the *Interaction Concentrator 7.6 Physical Data Model* document for your particular RDBMS.

Using Custom States

In order for ICON to store information to support reporting about custom states and common data, you must do the following:

- Set the appropriate ICON application configuration options:
  - AgentRecordUserTypes
  - AgentUserFields
  - EventData
  - GlobalData
  - store-event-data

  The AgentRecordUserTypes option defines the custom agent states.
- Configure the agent desktop application to send the applicable key-value pairs (KVPs) to T-Server, so that they can be included in the UserData of EventUserEvent (see "Agent Desktop Application Configuration").

Agent Desktop Application Configuration

ICON records the beginning and end of a custom state, based on information that it receives in the UserData of an EventUserEvent from T-Server. You must

configure your agent desktop application to send T-Server the appropriate KVP information for the `EventUserEvent UserData`.

Starting Recording of a Custom State

In order for ICON to start recording a custom state, the desktop application must send the following KVP:

Key = "<StateKeyName>", Value = "+"

Example

```
"AfterCallWork", "+"
```

Sending Custom State Data

In order for ICON to store additional information about an active custom state, the desktop application must send the following KVP:

Key = "<CommentKey>", Value = "<StateCode>, <Comment>"

You can configure more than one comment key for the same custom state. However, for each comment key, ICON can store only one value. If multiple KVPs are sent for the same comment key, ICON stores only the last value.

Example

```
"Comment", "207, This is data about the state"
"Explanation", "207, This is more data about the state"
"Explanation", "207, This is more, changed data about the state"
```

In this example, ICON will store the following values:

- In the `Comment` field for state `207`: `This is data about the state`
- In the `Explanation` field for state `207`: `This is more, changed data about the state`

Stopping Recording of a Custom State

In order for ICON to stop recording a custom state, the desktop application must send the following KVP:

Key = "<StateKeyName>", Value = "-"

Example

```
"AfterCallWork", "-"
```

Using Multiple Custom States Simultaneously

For each type of custom state, only one state can be active for a DN at any one time. However, ICON can simultaneously handle multiple different states independently. For example, two different states can be active on one DN, with different data corresponding to each. ICON does not support duplicate key names in attached data; KVPs with the same key name should not be sent in one `EventUserEvent (see` “Example”).

Example

```
"AfterCallWork", "+"
"Break", "+"
"Comment", "207, This is data about the call"
"Comment", "208, This is data about the break"
"Break", "-"
"AfterCallWork", "-"
```

Note: In the example above, ICON will store the key value only for the custom state "207, This is data about the call".

Part

2 Part Two: Administering Interaction Concentrator

Part Two of this *Interaction Concentrator User's Guide* contains information that is required to perform ongoing maintenance and administration of Interaction Concentrator, as well as to troubleshoot startup and runtime problems. It also contains detailed information about Interaction Concentrator features and functionality.

This information appears in the following chapters:

Chapter

9 Starting and Stopping Interaction Concentrator

This chapter describes the prerequisites for Interaction Concentrator startup, and it provides instructions for starting and stopping Interaction Concentrator (ICON). It contains the following sections:

Overview

You can start and shut down Interaction Concentrator components by using the Management Layer, a startup file, a manual procedure, or Services Manager.

All of these methods usually require command-line parameters for a server application as well as an executable file name. The next section describes the command-line parameters that are common to most Genesys server applications. Subsequent sections describe the startup and shutdown procedures.

Note: For information about using the Management Layer, startup files, and Services Manager for startup, see the *Framework 7.6 Deployment Guide.*

Command-Line Parameters

The following startup command-line parameters are supported by Interaction Concentrator:

`-host`	The name of the host on which Configuration Server is running.
`-port`	The communication port that client applications must use to connect to Configuration Server.
`-app`	The exact name of an application as configured in the Configuration Database.
`-V`	The version of a component. Note that specifying this parameter does not start an application, but instead returns its version number. You can use either an uppercase letter (`V`) or lowercase letter (`v`).
`-lmspath`	The full path to the log messages files that an application uses to generate log events. (These files are the common file named `common.lms` and the application-specific file with the extension `*.lms`.) Use this parameter when the common and application-specific log message files are located in a directory other than the application's working directory—for example, when the application's working directory differs from the directory to which the application was originally installed. Note that if the full path to the executable file is specified in the startup command line (for instance, `c:\gcti\multiserver.exe`), the path that is specified for the executable file is used to locate the `*.lms` files, and the value of the `lmspath` parameter is ignored.

Warning! An application that does not locate its `*.lms` file at startup cannot generate application-specific log events and send them to Message Server.

Starting ICON

This section provides manual startup instructions for ICON server. You can start ICON in any of the following ways:

- From SCI (see page 59).
- Manually on UNIX (see page 60).
- Manually on Windows (see page 61).
- As a Windows Service on Windows (see page 62).

Starting ICON with Solution Control Interface

Complete the following procedure to start ICON with Solution Control Interface (SCI).

Procedure: Starting ICON with SCI

Prerequisites

Genesys recommends that the following applications be running before you start ICON:

- The DB Server that provides access to IDB.
- The relational database management system.
- T-Server.
- Outbound Contact Server, if ICON is configured to collect data from OCS.
- Interaction Server, if ICON is configured to collect data from Multimedia.

If you have configured ICON to store attached data, ensure that there is a proper attached data specification file in ICON's working directory. (By default, ICON uses the `ccon_adata_spec.xml` file.)

For a short period of time after starting or restarting, ICON may produce `[cp:...]` or FSM errors in the log. These errors occur when ICON encounters elements of interactions that it cannot resolve because the interactions were already in progress when ICON was started or restarted. You can safely ignore these errors.

For detailed instructions about starting the Genesys components on which Interaction Concentrator depends, see:

- *Framework 7.6 Deployment Guide*
- *Framework 7.6 T-Server Deployment Guide* for your particular T-Server type
- *Framework 7.6 DB Server User's Guide*
- *Outbound Contact 7.6 Deployment Guide*
- *Multimedia 7.6 Deployment Guide*

Start of procedure

1. On the list pane in the SCI `Applications` view, select your ICON `Application` object.
2. Do one of the following:
   - On the toolbar, click the `Start` button.
   - From the `Action` menu, select `Start`.

- Right-click the Application object to access the shortcut menu, and then select Start.

3. In the confirmation box that appears, click Yes.

   SCI starts your Interaction Concentrator application.

End of procedure

Next Steps

- You have completed all the steps necessary to start ICON using SCI.

Starting ICON Manually

Complete the following procedure to start ICON manually on UNIX.

Procedure: Starting ICON manually on UNIX

Start of procedure

1. Go to the directory to which you have installed ICON.
2. Enter the name of the ICON executable, followed by the appropriate command-line parameters, using the following syntax:

   ```
   ./icon -host <hostname> -port <portno> -app <application>
   ```

   Where:

   - *hostname* is the name of the host on which Configuration Server is running.
   - *portno* is the communication port that client applications must use to connect to Configuration Server.
   - *application* is the name of the Interaction Concentrator Application object, as defined to Configuration Server.

Note: If the host name or application name contains spaces or hyphens (-), enclose them in double quotation marks.

For example, to start ICON with command-line parameters that specify the host as cs-host, the port as 2020, and the name as ICON 03, enter the following:

```
./icon -host "cs-host" -port 2020 -app "ICON 03"
```

End of procedure

Next Steps

- You have completed all the steps necessary to start ICON manually on UNIX.

Starting ICON on Windows

Complete the following procedure to start ICON on Windows.

Procedure: Starting ICON on Windows

Purpose: To start ICON from the `Start > Programs` menu, or from the console window.

Start of procedure

1. Open a console window.
2. Go to the directory to which you installed Interaction Concentrator.
3. Enter the following command line:

   `icon.exe -host <hostname> -port <portno> -app <application>`

   Where:
   - *`hostname`* is the name of the host on which Configuration Server is running.
   - *`portno`* is the communication port that client applications must use to connect to Configuration Server.
   - *`application`* is the name of the Interaction Concentrator `Application` object, as defined to Configuration Server.

Note: If the host name or application name contains spaces or hyphens (-), enclose them in double quotation marks.

For example, to start ICON with command-line parameters that specify the host as `cs-host`, the port as `2020`, and the name as `ICON 03`, enter the following:

```
icon.exe -host "cs-host" -port 2020 -app "ICON 03"
```

End of procedure

Next Steps

- You have completed all the steps necessary to start ICON on Windows.

Starting ICON as a Windows Service

On Microsoft Windows platforms, by default, the installation process installs Interaction Concentrator as a Windows Service. If you stopped ICON from running as a Windows Service and need to start it again as a Windows Service, complete the following procedure.

Procedure: Starting ICON as a Windows service

Start of procedure

1. Open the Windows `Control Panel`, and then double-click the `Services` icon. The `Services` dialog box opens.
2. In the `Services` list box, select your ICON service, and then click `Start`. (If you disabled Interaction Concentrator from operating as a Windows Service, the `Start` option for this application will not be available.)

Note: You can install the Local Control Agent (LCA) as a Windows Service with the user interface disabled. In this case, all servers that are started through SCI are started without a console, unless you specifically select the `Allow Service to Interact with Desktop` check box for both LCA and ICON.

End of procedure

Next Steps

- You have completed all the steps necessary to start ICON as a Windows service.

Stopping ICON

You can stop ICON in any of the following ways:

- From SCI (see page 63). (This is the recommended method.)
- Manually on UNIX (see page 63).
- Manually on Windows (see page 64).
- As a Windows Service on Windows (see page 65).

Note: To prevent ICON from self-starting, make sure that you clear the `autorestart` property in the ICON `Application` object in Configuration Manager.

Stopping ICON with Solution Control Interface

If you are using LCA and SCS, complete the following procedure to stop ICON with SCI.

Procedure: Stopping ICON using SCI

Start of procedure

1. On the list pane in the SCI `Applications` view, select your ICON `Application` object.
2. Do one of the following:
   - On the toolbar, click `Stop`.
   - From the `Action` menu, select `Stop`.
   - Right-click the `Application` object to access the shortcut menu, and then select `Stop`.
3. In the confirmation box that appears, click `Yes`.

   SCI stops your Interaction Concentrator application.

End of procedure

Next Steps

- You have completed all the steps to stop ICON using SCI.

Stopping ICON on UNIX

Stop ICON on UNIX by using one of the following procedures.

Procedure: Stopping ICON on UNIX from the command line

Start of procedure

- On the command line, enter the following:

  `kill -SIGTERM <processid>`

  Where `<processid>` is the application's UNIX process ID.

End of procedure

Next Steps

- You have completed all the steps to stop ICON from the command line.

Procedure: Stopping ICON on UNIX from the console window

Start of procedure

- From the active console window, press `CTRL+C`.

End of procedure

Next Steps

- You have completed all the steps to stop ICON from the console window.

Note: If you are using LCA and SCS, you can also use SCI to stop ICON (see “Stopping ICON with Solution Control Interface” on page 63).

Stopping ICON on Windows

If ICON is running as an application—not as a Windows Service—stop it using the following procedure.

Procedure: Stopping ICON on Windows from the console window

Start of procedure

- From the application’s console window, press `CTRL+C`.

End of procedure

Next Steps

- You have completed all the steps to stop ICON from the console window.

Note: If you are running ICON as a Windows Service, you should stop it only from the Services Control Manager (see “Stopping ICON as a Windows Service”).

Stopping ICON as a Windows Service

To stop Interaction Concentrator running as a Windows Service, use the following procedure.

Procedure: Stopping ICON running as a Windows service

Start of procedure

1. Open the `Control Panel`, and then double-click the `Services` icon. The `Services` dialog box opens.
2. In the `Services` list box, select your ICON service, and then click `Stop`.

End of procedure

Next Steps

- You have completed all the steps to stop ICON running as a Windows Service.

Chapter

10 Monitoring Interaction Concentrator

This chapter describes how to access the HTTP Listener to monitor and report on Interaction Concentrator performance. It contains the following section:

Accessing the Performance Counter

You can configure an HTTP Listener to access a performance counter page from your browser. You can use this performance counter to report on ICON performance in real time and to identify performance and usage patterns which you can then use to fine-tune your ICON configuration. You can also access the performance counter pages as needed for troubleshooting purposes.

Note: Accessing the performance counter pages can significantly impact ICON performance. Therefore, Genesys recommends that you do not routinely access these pages.

To configure an HTTP Listener, create a `listeners` section on the ICON `Application` object that describes the HTTP listening port. Once the HTTP Listener is configured, you can access the performance counter pages when ICON is running, as follows:

- Open a browser window and go to the following address:

  `http://<host>:<port>`

  where:

  - `host` is the name of the ICON server host
  - `port` is the value configured for the `port` option in the `<http-connection>` section on the `Options` tab of the ICON `Application` object.

For more information about configuring the HTTP Listener, see the chapter about configuring options in the *Interaction Concentrator 7.6 Deployment Guide.*

Performance Counter Pages

The performance counter pages provide the following types of information:

- Configuration object statistics—for example, the number of `DN`, `Person`, `Place`, and `Agent Login` objects read by ICON.
- CTI statistics—for example, detailed information about active calls, parties, and agent sessions; information about active T-Server connections; statistics about CTI events (such as the number of call processing errors and the average volume of calls and events); and accumulated statistics about agent login sessions.

  Similar information is provided for Multimedia interactions as well.
- ICON statistics—information about memory allocation for current activity involving key ICON entities, such as calls and agent sessions.
- Database-writing statistics—information and statistics about database-writing activity for the ICON instance as a whole as well as by DAP role.

Figure 9 shows a sample performance counter page—the main `Database Writer` page.

Note: Figure 9 is only an example of the web page. Your actual screen might look quite different from this one.

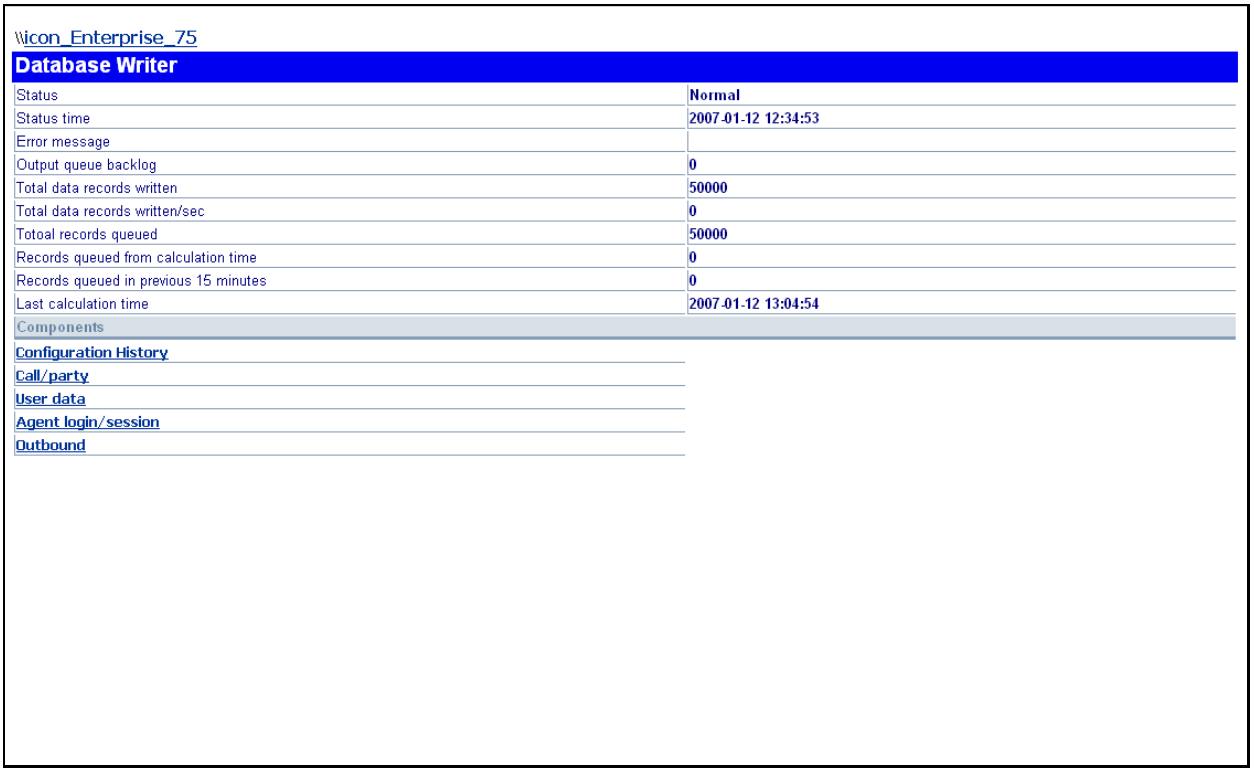

Figure 9: Main Database Writer Page

Chapter

11 Filtering IDB Data

This chapter describes the data filtering capability of Interaction Concentrator 7.6 and how to configure it by setting options on the ICON `Application` and `script` objects.

The data filtering feature provides users with the ability to configure what data will *not* be stored in IDB in order to maintain a smaller IDB and improve database performance.

This chapter contains the following sections:

Overview

The data filtering capability of ICON allows you to control what type of data is stored in IDB based on your reporting requirements. Excluding certain types of data from IDB storage may be helpful in your environment, by:

- Saving database space (maintaining a smaller IDB)
- Improving the performance of your IDB
- Improving the performance of downstream reporting applications (such as Genesys Info Mart)

This feature is implemented by setting configuration options in the `filter-data` section of your Interaction Concentrator `Application` object. Because setting these options can cause ICON to cease writing data to the corresponding IDB tables, carefully evaluate whether your reports require each type of data described in the following sections before setting any options. Refer to Chapter 2, “Introducing IDB Schema,” on page 27 for an overview of IDB schema and its tables.

ICON 7.6.1 also supports the ability to filter out Multimedia interaction activity related to strategies, where such activity data is not required for

reporting purposes. To implement this feature, set configuration options in the `callconcentrator` section of your Interaction Concentrator `Application` and `script` configuration objects.

Note: By default, the filtering configuration options are set to `false` and all available data is stored in the IDB.

What Data Can Be Filtered?

The data described in this section can be filtered— included or excluded from storage in IDB—by setting various configuration options. It includes the following sections:

Note: In addition to the information contained here, refer also to the *Interaction Concentrator 7.6 Deployment Guide* for information about setting configuration options in ICON, and the *Interaction Concentrator 7.6 Physical Data Model* for your RDBMS type, for details about data stored in the IDB tables mentioned.

Party History Data

The `G_PARTY_HISTORY` table contains call party information related to when the history of a party state changed. For distribution devices only, this table contains information about queuing on a device and distribution from a device only (for example, it is not possible to have a ringing or hold state on a distribution device).

For SIP and voice interactions only, you can choose not to store party history information about distribution devices such as ACD queues, routing points, and virtual routing points, in IDB. Setting the `acd-party-history` configuration option to `true` on the ICON `Application` will prevent ICON from writing party–related information to the `G_PARTY_HISTORY` table.

Party Metrics

ICON collects precalculated party metrics for distribution devices, such as ACD queues, routing points, and virtual routing points, and stores this information in the G_PARTY_STAT table.

For SIP and voice interactions only, you can choose not to store party metrics data for distribution devices in IDB by setting the acd-party-metrics configuration option to true on the ICON Application object.

External Parties

ICON collects information about external parties (for example, interaction participants outside a given switch domain) and stores this information in the following IDB tables:

- G_PARTY
- G_PARTY_HISTORY
- G_PARTY_STAT

You can choose not to store external-party data from IDB storage by setting the external-party configuration option to true on the ICON Application object.

User Data History

When ICON is configured in a way that it should store an entire history of UserData values for certain keys, ICON collects data about every change in value for those keys and, at interaction termination, stores this information in the following IDB tables:

- G_USERDATA_HISTORY
- G_SECURE_USERDATA_HISTORY

By setting the udata-history-terminated configuration option to true on the ICON Application object, ICON will not store the call–termination value of UserData keys to IDB. ICON will, however, continue to store the history of UserData changes—the creation, addition, and removal of key-value pairs—to these tables as changes occur during the call's life time. Table 15 describes the information stored in these IDB tables.

Table 15: Change Information Types Stored in IDB

Value	Column Change Type	Description
1	created	1 indicates that the value of the key was attached to the call at the moment the call was created. For chat interactions, ICON assigns a change type of 1 upon receipt of an eventInteractionSubmitted event with an isOnline attribute of true.
2	added	2 indicates that the value of the key has just been added.

Table 15: Change Information Types Stored in IDB (Continued)

Value	Column Change Type	Description
3	updated	3 indicates that the value of the key has changed. For chat interactions, ICON assigns a change type of 3 upon receipt of an eventPropertiesChanged event with an isOnline attribute setting of either true or false.
4	deleted	4 indicates that the key has been deleted from UserData.
5	terminated	5 indicates that ICON records the value of the key upon call termination.

Call Metrics

ICON stores information about call metrics in the G_CALL_STAT table. To exclude call metrics from IDB storage, set the call-metrics configuration option to true on the ICON Application object.

Call History

ICON stores call history information in the G_CALL_HISTORY table. To exclude call history information from IDB storage, set the call-history configuration option to true on the ICON Application object.

Interaction Record History

ICON collects interaction record history and stores this information in the G_IR_HISTORY table. To exclude data about the interaction record history from IDB storage, set the ir-history configuration option to true on the ICON Application object.

Agent Activity Data

ICON collects information about agent activity, such as login sessions and agent states. Unless certain types of data are configured to be excluded (see the subsections below), ICON stores this information in the following IDB tables:

- G_LOGIN_SESSION
- GX_SESSION_ENDPOINT
- G_AGENT_STATE_HISTORY
- G_AGENT_STATE_RC
- G_DND_HISTORY
- GS_AGENT_STAT
- GS_AGENT_STAT_WM

You can choose not to store any agent activity information in IDB by setting the gls-all configuration option to true on the ICON Application object. ICON, however, continues writing each agent's ID to the G_PARTY table.

Agent Metrics

ICON collects and stores agent state information in the following IDB tables:

- GS_AGENT_STAT
- GS_AGENT_STAT_WM

You can choose to exclude agent state information by setting the gls-metrics configuration option to true on the ICON Application object.

Agent Activity From IVR Devices

ICON collects data about agent activity when agent login sessions are initiated from IVR endpoints, and stores this information in the following IDB tables:

- G_LOGIN_SESSION
- GX_SESSION_ENDPOINT
- G_AGENT_STATE_HISTORY
- G_AGENT_STATE_RC
- G_DND_HISTORY
- GS_AGENT_STAT
- GS_AGENT_STAT_WM

You can choose to exclude data about agent activity at IVR endpoints, from IDB storage, by setting the gls-ivr configuration option to true on the ICON Application object. ICON verifies whether the DN at which an agent logs in is an IVR device, and in this case, does not store information about this agent's activity to IDB. Furthermore, for parties associated with an IVR device, ICON does not record the agent's ID in the G_PARTY table.

Agent Activity For Persons Not Configured

ICON collects data about all agent activity and stores this information in IDB tables (see "Agent Activity Data" on page 114). You can choose to exclude data from IDB storage about agent activity for any agent whose login ID is not associated with any Person configuration object by setting the gls-no-person configuration object to true on the ICON Application object. If so configured, ICON verifies whether the LoginID that was reported in events regarding agent states is assigned to any Person object configured in the Configuration Database. If this is not the case, ICON does not store information about this agent's activity to these tables.

Agent Work Mode

ICON collects and stores data about agents' work mode and changes in agents' work modes in the following IDB tables:

- G_AGENT_STATE_HISTORY
- G_AGENT_STATE_RC

- GS_AGENT_STAT_WM

You can choose to exclude data about changes in agent work mode that do not coincide with changes in agent state by setting the gls-wm configuration option to true on the ICON Application object. ICON instead records a value of unknown in the IDB tables.

Note: This option does not affect ICON's ability to track after-call work.

Agent Queue

ICON collects and stores information about agents' queue(s) in the following IDB tables:

- G_AGENT_STATE_HISTORY
- G_AGENT_STATE_RC
- GS_AGENT_STAT
- GS_AGENT_STAT_WM
- GX_SESSION_ENDPOINT

You can filter out information about the queues where agents are logged in, by setting the gls-queue configuration option to true on the ICON Application object. In this case, ICON ceases writing queue-related data to the first four tables (above). ICON will, however, continue writing information to the GX_SESSION_ENDPOINT table about the queues where agents are logged in.

Service Observer Data

ICON collects data related to parties with role observer on a call and stores this information in the following IDB tables:

- G_PARTY
- GS_PARTY_STAT

You can choose not to store data about the party with the role observer in IDB, by setting the observer-party configuration option to true on the ICON Application object.

Strategy Activity Data

ICON 7.6.1 supports the ability to filter out activity related to a particular strategy—information such as parties or user data changes made by the strategy. Such data may not be required for reporting purposes, and therefore you can choose to exclude it from storage in IDB.

To filter out strategy activity data, set the following configuration options to the values specified:

- `om-activity-report`— Set this option to `false` on the `Annex` tab of `script` configuration objects of type `simple routing` (for a routing strategy). ICON will not store to IDB any data that is related to any Multimedia interaction handled by this strategy, provided that the `om-check-filter-flag` option is also set to `1`.
- `om-check-filter-flag`—Set this option to `1` on the ICON `Application` object (`callconcentrator` section) to allow ICON to store strategy activity according to the value of the `om-activity-report` option. If set to `0`, ICON continues to store all strategy activity regardless of the value of `om-activity-report`.

Chapter

12 Implementing HA in Interaction Concentrator

This chapter describes the design and implementation of high availability (HA) in Interaction Concentrator. It discusses the extraction of real-time voice, agent-specific, Outbound, and virtual queue interaction data in an HA pair of IDBs. In addition, the consequences of dropped server connections as they relate to inconsistent data are discussed. Finally, configuration considerations for the ICON Application and IDB in an HA environment are presented here.

Note: This chapter is intended for users who have developed their own ETL engine or who use Genesys Info Mart to create reports based on data extracted from IDB.

This chapter contains the following sections:

Overview

You can deploy Interaction Concentrator in a high availability configuration to ensure redundancy of your reporting data for voice interaction, attached data, agent state, and virtual queue details.

In an HA configuration, if a component or network outage prevents one of the ICON processes from storing any voice interaction, attached data, or virtual queue data in its IDB, data is not lost as long as the other ICON process is able to store those interactions in its IDB.

Furthermore, a downstream reporting application such as Genesys Info Mart can continue to extract, transform, and load voice interaction, attached data, and virtual queue data, as long as it can connect to at least one of the IDBs in the HA pair.

ICON HA Model

The High Availability model used in Interaction Concentrator differs significantly from the Genesys standard HA model implemented in a majority of Genesys servers.

Different from Standard Genesys HA Model

Unlike a typical Genesys server (for example, T-Server), two Interaction Concentrators that are deployed as a redundant (HA) pair, do not operate in either primary or backup mode, nor does their mode switch from backup to primary when the other server fails. Rather, both Interaction Concentrators in the HA pair operate in parallel as stand-alone servers and process incoming data independently.

A redundant pair of Interaction Concentrators receives events from the same T-Server (either a stand-alone T-Server or the primary T-Server from a redundant pair) and stores data to two independent IDBs. The downstream reporting applications are responsible for de-duplication of redundant data retrieved from both IDBs. Figure 10 illustrates the major components of an ICON HA solution.

Note: For simplicity, Configuration and DB Servers are not included in Figure 10.

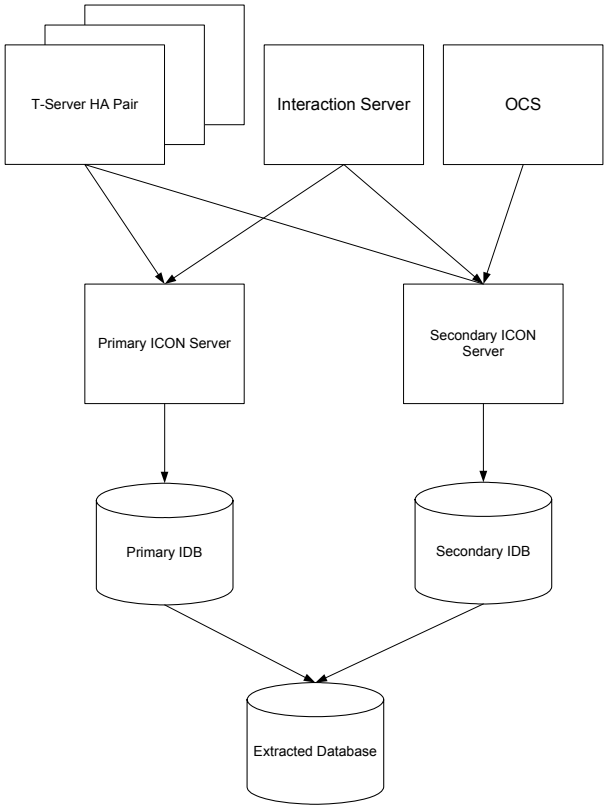

Figure 10: ICON HA Model

Data Extraction from HA IDB Pair

This section describes several types of data extraction from the HA pair of IDBs. Recommendations are based on assumptions that the chosen method of data extraction takes into consideration the type of data and the complexity of the environment.

Real-Time Interaction Data

Real-time interaction data, such as voice-specific interactions, is stored in the following IDB tables:

G_IR	G_CALL	G_CALL_USERDATA
G_IR_HISTORY	G_CALL_HISTORY	G_CALL-USERDATA-CUST
G_IS_LINK	G_CALL_STAT	G_CALL_USERDATA_CUST1
G_IS_LINK_HISTORY	G_PARTY	G_USERDATA_HISTORY
G_ROUTE_RESULT	G_PARTY_STAT	G_SECURE_USERDATA_HISTORY

Extraction Procedure

An extraction procedure should extract `G_IR` records completely from both the primary and backup IDBs:

- Completed records are populated with `MERGESTATE 3` in table `G_IR`.
- For incremental extractions, ICON populates the `GSYS_MSEQ` field.

When data is incomplete, the decision to process the available data is made by Interaction Concentrator using the stuck call resolution procedure. This procedure is based on a timeout mechanism and allows for continued data processing in the event of partial data loss. Within IDB, Interaction Concentrator marks the records it detects to be incomplete as having low reliability. For more information about such records, see the `G_IR.GSYS_EXT_VCH2` field description in the *Interaction Concentrator 7.6 Physical Data Model* document for your particular RDBMS.

Note: See "Merge Stored Procedure" on page 145 for a description of the `gsysIRMerge` stored procedure for voice data merge.

Figure 11 illustrates Interaction Concentrator's data extraction model.

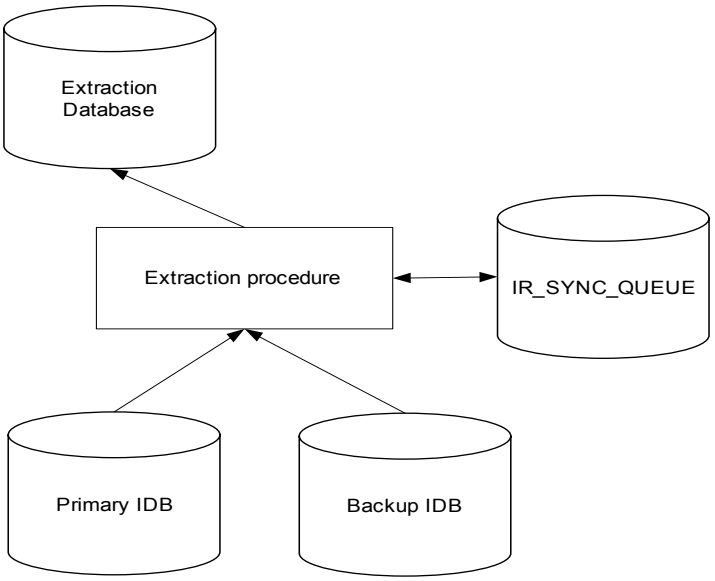

Figure 11: Data Extraction in IDB—Logical Model

Records extracted from an IDB instance are processed as described in the following scenarios.

Scenario 1: The record is not found in `IR_SYNC_QUEUE`.

- If the record's termination time is *greater than* the timeout specified:
  - The record is ignored, or

  - The existence of the record is checked against the Extraction database. If no record exists, it is processed as a record with a termination time less than the timeout specified.
- If the record's termination time is *less than* the timeout specified:
  - The record is extracted to the Extraction database, and
  - The record ID (IRID) is added to `IR_SYNC_QUEUE` (see Table 16).

Scenario 2 A record is found in `IR_SYNC_QUEUE`.

- The record is removed.

Scenario 3 The `IR_SYNC_TIMEOUT` expires.

- The IR cache maintenance procedure removes the IR record from the cache.

> **Note:** The time specified in `IR_SYNC_TIMEOUT` should be more than the maximum delay of data available from the secondary database.

Table 16: Structure of the IR_SYNC_QUEUE Table

Field	Type	Description
ID	numeric	This is the primary key generated by the database.
IRID	varchar (50)	The is the value of the `IRID` field from the `G_IR` extracted record.
Extracted	datetime	The GMT data and time when the IR record was extracted. This value is used by the IR cache maintenance procedure.

Assumptions

Record extraction from the HA pair of IDBs is based on the following assumptions:

- All data related to the interaction is stored in IDB.
- Attached data is stored in the same IDB instance as the interaction data.
- Inconsistencies in data related to dropped or failed server connections have been handled by IDB.

Record Processing

Interaction Concentrator processes records from the following tables in the order specified:

1. G_IR_HISTORY
2. G_CALL
3. G_CALL_HISTORY
4. G_CALL_STAT
5. G_PARTY
6. G_PARTY_HISTORY
7. G_PARTY_STAT
8. G_IS_LINK
9. G_IS_LINK_HISTORY
10. G_ROUTE_RESULT
11. G_CALL_USERDATA
12. G_CALL_USERDATA_CUST
13. G_CALL_USERDATA_CUST1
14. G_CALL_USERDATA_CUST2
15. G_CALL_USERDATA_HISTORY
16. G_SECURE_USERDATA_HITSTORY

Records in the G_IR table serve as the base records for data extraction. The value of the IRID field is used as an identifier to determine which record in the G_IR_HISTORY and G_CALL tables to extract.

After data extraction from the G_CALL table, the value of the CALLID field is used as an identifier to determine which records to extract in the following tables.

G_CALL_HISTORY	G_CALL_USERDATA_CUST
G_PARTY	G_CALL_USERDATA_CUST1
G_CALL_STAT	G_CALL_USERDATA_CUST2
G_USERDATA_HISTORY	G_IS_LINK
G_SECURE_USERDATA_HISTORY	G_IS_LINK_HISTORY

After data extraction from G_PARTY, the value of the PARTYID field is used as an identifier to determine the records in the G_PARTY_HISTORY and G_PARTY_STAT tables.

Stuck Records

Stuck records can result when ICON restarts if, during the time that ICON is shut down:

- A change occurs in agent session data (for example, an agent logs in or out).
- Outbound campaign processing completes.
- A call is distributed from a virtual queue.
- A call or party is deleted.

Stuck calls or parties can also result from stuck calls or parties on T-Server's side.

Note: The disconnection of ICON from T-Server does not in itself result in stuck calls. ICON can retrieve a snapshot of the active calls and compare this to the calls in memory.

Virtual Queue-Specific Data

Virtual queue records have the same unique ID in both IDB instances (the `VQID` field in the `G_VIRTUAL_QUEUE` table). The extraction procedure for virtual queue data is the same as for real-time interaction data. Virtual queue records have a status associated with them that can be used for data validation. For example, a virtual queue record with a value of `stuck` (integer value = `2`) in the `Status` field can be ignored in favour of a virtual queue record with a `normal` or `abandoned` status.

Outbound Data

The procedure used to extract Outbound data is similar to that used for voice data interaction. It uses records from the `GO_CAMPAIGN` and `GO_CHAIN` tables as the base records. The records from these two tables can be extracted independently.

Note: Only terminated records are considered for extraction.

The extraction procedure for Outbound data also uses a combination of the values in the following fields as the unique record identifier:

`SessID`	`OCSID`
`CallingListID`	`RECORDHANDLE`
`CHAINGUID`	`TYPE`
`CALLATTID`	`ADDED_TS`

Extraction of Campaign-Level Data

Data extraction of terminated records from the `GO_CAMPAIGN` table uses the value of the `SessID` field as a unique identifier (the same value of the `SessID` field is present in both IDB instances). It then uses this value to extract data from the `GO_CAMPAIGNHISTORY` and `GO_CAMPPROP_HIST` tables in the same IDB instance.

Extraction of Chain-Level Data

Data extraction of terminated records from the `GO_CHAIN` table uses the value of the `CHAINGUID` field as a unique identifier (the same value is present in both IDB instances). It then uses the value of the `CHAINGUID` field to extract data

from the GO_CHAINREC_HIST, GO_RECORD, GO_CUSTOM_FIELDS, GO_FIELDHIST, GO_SECURE_FIELDS, GO_SEC_FIELDHIST and GO_CHAIN_CALL tables in the same IDB instance.

Extraction of Precalculated Metrics

OCS precalculates some metrics and sends this data to Interaction Concentrator. Although OCS does not provide a unique identifier for these metrics, the object's identifier (for the metric calculated) and the metric's timestamp are the same in both IDB instances.

HA of Configuration Data

To identify configuration objects stored in IDB, Interaction Concentrator uses database identifiers (DBIDs) assigned by Configuration Server. With this approach, the DBID values stored in two HA IDBs must be identical for a given configuration object. In addition, ICON 7.6 flags the reliability of configuration data in the GSYS_EXT_INT1 field of the GC_* and GCX_* tables. When the reliability value of the same configuration record differs between two HA IDBs, the data should be extracted from the more reliable record. For more information, see "Reliability Flag" on page 136.

HA of Agent-Specific Data

ICON 7.6 supports high availability of agent-specific data (login sessions and after-call work) through its use of a redundant pair of ICON Servers. To support high availability of agent data, primary T-server creates AgentSessionIDs from the initial agent login and timestamp. It does not propagate AgentSessionID to backup T-Server. Instead, when backup T-Server becomes primary (after a switchover), new AgentSessionIDs are assigned to all known agent sessions. ICON processes only AgentSessionIDs received from the primary T-Server 7.6.

ICON 7.6 first stores data about login sessions in a persistent cache file—apstorage.db—to prevent any duplicate login sessions from storage in IDB, and to prevent stuck login sessions. If the reported AgentSessionID exists in IDB or ICON memory, the login session is considered to be a duplicate. In other cases, the existing login session will be marked as closed (for example, reason is stuck) and a new session is created. ICON stores the HA agent data in the G_LOGIN_SESSION IDB table. It also stores unique agent-specific data (for example, after-call work) and information about the reliability of the login session and the restored agent state in IDB.

Note: You can change the name of the `apstorage.dbs` persistent cache file using the `agent-pstorage-name` configuration option. For more details, see the chapter about configuring options in the *Interaction Concentrator 7.6 Deployment Guide.*

Dropped Server Connections

This section describes the consequences of dropped server connections in an ICON HA configuration. The consequences of these server disconnections as they relate to data inconsistencies will also be described.

Dropped T-Server Connection

If ICON loses connection to T-Server, the following data inconsistencies can occur in IDB:

- Incorrect party transitions from ICON's point of view resulting in missing transition or party error records
- Calls deleted at unknown times
- Missing information about:
  - Agent states
  - Attached data
  - Changes in attached data
  - Parties
  - Call linkages

Dropped CTI-Link

A dropped CTI-link can cause the following data inconsistencies in IDB:

- Stuck calls
- Incorrect transitions in call- and agent- related events
- Stuck parties—parties which ICON reports as active (that is, not deleted) when the associated call interaction has in fact been deleted

Dropped Active T-Server in HA Pair

The failure of the active T-Server in an HA pair can cause the following data inconsistencies in IDB:

- Incorrect transitions in TEvents
- Stuck parties
- Missing attached data

- Other inconsistencies due to incomplete eventflows

Dropped Interaction Server Connection

An unplanned ICON disconnection from Interaction Server can cause the following data inconsistencies in IDB:

- Incorrect party transitions from ICON's point of view resulting in missing transition or party error records
- Interactions deleted at unknown times
- Stuck interactions
- Missing information about:
  - Agent state
  - Attached data

Dropped OCS Connection

An unplanned ICON disconnection from Outbound Contact Server may cause the following data inconsistencies in IDB:

- Missing information about:
  - Campaign processing
  - Changes in calling lists
  - Linkages between OCS and voice call data
- Missing OCS pre-calculated metrics

ICON Server Failure

A failure of the ICON Server will cause data loss for the particular ICON instance and possible failure in restoring queued data from the persistent queue. As a result, the following data inconsistencies may occur in IDB:

- Stuck calls
- Stuck parties
- Missing records

Configuration Considerations

In an HA deployment, consider the following:

- You must set configuration options in both Interaction Concentrator `Application` objects exactly the same. Because this is not a typical redundant pair from Genesys perspective, Configuration Server does not synchronize the configuration options for two Interaction Concentrators automatically.

  For example, to configure your redundant ICON applications to store voice interaction data in a pair of HA IDBs:

  - In both ICON `Application` objects, set the `role` option so that it contains `gcc` and `gud`. This enables both ICON applications to store call-related and attached data.
  - For any configuration options that affect the data populated by those roles, set the same option values in both ICON applications. For example, the two applications must use the same ICON configuration options for virtual queue monitoring, storage of attached data, and so on.

    For more information about setting configuration options, refer to the *Interaction Concentrator 7.6 Deployment Guide.*
- You must configure a connection to the same T-Server in both Interaction Concentrator Application objects.
- You must create two identical IDBs. Genesys recommends using two databases located on different hosts, but having the same RDBMS type and version number, to host the HA pair of IDBs.
- You must configure a Database Access Points (DAPs) for each ICON to access its IDB.

For more information about configuring applications and connections in Configuration Manager, see the *Framework 7.6 Deployment Guide.*

Chapter

13 Resynchronizing Configuration Changes

If configured to do so, ICON gathers information about configuration objects from the Configuration Server upon initial startup and keeps track of changes made to these objects throughout its operation by monitoring dynamic real-time notifications from Configuration Server. ICON stores Configuration Server information about the addition of new objects and the deletion or update of existing objects into tables prefixed with `GC_` in IDB; ICON stores Configuration Server information about object relationships in `GCX_` IDB tables.

Under certain conditions, IDB data, can fall out of synchronization with Configuration Server data. Operating ICON and downstream applications that rely on ICON data under these circumstances can lead to results that are difficult to interpret. Under such conditions, you might consider resynchronizing IDB with the current configuration data.

Note: If you configure ICON to store configuration changes, it is important to ensure that ICON runs continually (without stopping). This will limit the chance of IDB losing synchronization with Configuration Server data.

This chapter describes the conditions under which you might consider invoking a forced resynchronization of IDB data and the steps required to resynchronize your IDB. This chapter also describes how to restore Configuration Database should you need to recover its data from a crash. It includes the following sections:

- Recommendations for Resynchronization, page 140
- Restoring the Configuration Database from Backup, page 141

How ICON Gathers Configuration Data

The value of ICON's `role` configuration option determines whether ICON collects and stores configuration-related data. If this option includes the value of `cfg`, then ICON will collect the data:

- Upon startup—This is the first deployment of ICON when the IDB is empty.
- When the persistent queue is unavailable.
- Through real-time object change notifications.
- Upon receipt of the Configuration Server's history log file.
- Upon user-request for resynchronization.

Reading the Configuration Database Upon Startup

Upon initial startup, or if the local cache file is not available (see "Persistent Cache is Not Available"), ICON queries Configuration Server for all active configuration objects and active relationships. ICON loads a local file—a *persistent cache*—with the information it gathers about these objects and relationships. (The name of this local persistent cache file is `cfg-sync.db`.) It then submits the updated transactions to its persistent queue for updating the following configuration-related tables in IDB:

- GCX_AGENT_PLACE
- GCX_CAMPGROUP_INFO
- GCX_CAMPLIST_INFO
- GCX_ENDPOINT_PLACE
- GCX_FORMAT_FIELD
- GCX_GROUP_AGENT
- GCX_GROUP_ENDPOINT
- GCX_GROUP_PLACE
- GCX_GROUP_ROUTEDN
- GCX_LIST_TREATMENT
- GCX_LOGIN_INFO
- GCX_OBJTABLE_RECORD
- GCX_SKILL_LEVEL
- GCX_SUBCODE
- GC_ACTION_CODE
- GC_AGENT
- GC_APPLICATION
- GC_ATTR_VALUE
- GC_BUS_ATTRIBUTE
- GC_CALLING_LIST
- GC_CAMPAIGN
- GC_ENDPOINT
- GC_FIELD
- GC_FILTER
- GC_FOLDER
- GC_FORMAT
- GC_GROUP
- GC_IVR
- GC_IVRPORT
- GC_LOGIN
- GC_OBJ_TABLE
- GC_PLACE
- GC_SCRIPT
- GC_SKILL
- GC_SWITCH
- GC_TABLE_ACCESS
- GC_TENANT
- GC_TIME_ZONE
- GC_TREATMENT
- GC_VOICE_PROMPT

Note: ICON stores information in `GC_APPLICATION` about the following application types only—Outbound Contact Server, CPD Server, T-Server, and Agent Desktop.

Persistent Cache is Not Available

If the persistent cache is not available during a *routine* startup (that is to say, not the initial startup when the IDB is empty), ICON performs a resynchronization of IDB automatically. Upon startup, ICON verifies the content of the IDB using the last processed real-time notification from Configuration Server. If there is no information about the last notification because the persistent queue is not available, ICON requests all configuration-related information from Configuration Server and recovers the persistent cache.

ICON Receives Dynamic Notifications

ICON is a client of Configuration Server. Whenever both applications are operating and changes are made to configuration objects or their relationships to other objects within Configuration Manager, Configuration Server immediately notifies its clients of the change. (Genesys does not support such notification if objects are changed directly within the Configuration Database.) The persistent cache is designed to always be in sync with Configuration Database.When ICON receives the notification, it immediately sends the information to its persistent memory cache, and records it in the appropriate IDB table using the actual timestamps when the object was changed in Configuration Server.

ICON Reads the Configuration History Log

Configuration Server maintains a history log for the purpose of enabling clients to restore a session that was terminated by a service interruption and to request any changes to configuration objects that occurred during that the interruption. Dynamic changes made to configuration objects are reported directly by Configuration Server. ICON requests this information from Configuration Server every time it connects to it.

With earlier releases of Configuration Server (prior to 7.6), you must ensure that the Configuration Server's `history-log-active` option is set to `true`. If set to `false`, configuration changes are not recorded in the history log file. Therefore, if ICON lose connection to Configuration Server during this time, it cannot later retrieve information about configuration changes during the time of the disconnect. Setting this option to `false`, however, does not prevent resynchronization.

In release 7.6, you can set the following Configuration Server 7.6 `[history-log]` configuration options to control the history log functionality:

- `all`
- `expiration`
- `client-expiration`
- `max-records`
- `active`
- `failsafe-store-processing`

Note: If `failsafe-store-processing` is set to `false`, the history log database may not be wholly preserved.

Refer to the configuration history log section in the *Framework 7.6 Deployment Guide* and the history log section in the *Framework 7.6 Configuration Options Reference Manual* for more information about these options.

User Request For Resynchronization

On demand resynchronization occurs when a customer manually runs the resynchronization procedure (see “How to Resynchronize Configuration Data” on page 139 for complete step-by-step instructions). When instructed to start resynchronization, ICON requests all configuration data from Configuration Server and stores it in its persistent cache. At the same time, all other activity—such as dynamic notifications—between ICON and Configuration Server is disabled. ICON then transfers configuration data in the persistent cache to the IDB and begins to monitor real-time notifications from Configuration Server again.

How Resynchronization Works

ICON stores configuration data in two types of tables, one for configuration objects and the other for configuration object relationships. Each configuration data record is therefore defined by a pair of attributes: configuration object type and database identifier (DBID).

During resynchronization, ICON requests all configuration data from Configuration Server and stores it to the local cache. For each pair of attributes (object type and DBID), ICON checks whether it exists in the IDB:

- If it does *not* exist, a new record is added to the IDB.
- If it does exist, ICON checks if any stored properties are different for this object, and if so, the IDB record is updated.

Note: Any active objects that exist in the IDB but not in the Configuration Database, are marked for deletion.

During resynchronization, ICON suspends receipt of real-time notifications about new configuration object updates from Configuration Server. Configuration Server queues the notifications and delivers them after synchronization completes.

Table 17 shows some of the information that ICON records in IDB for new, updated, and deleted configuration objects when resynchronization occurs:

- The `Field` column indicates the column in the targeted `GC_*` IDB table that ICON will write to when ICON transfers data to IDB.
- Values in the `GSYS_EXT_INT1` field indicate the reliability of the timestamps flag (see Table 19 on page 136 for a description of the values).

Table 17: IDB Stores Object Changes

Field	New Object	Updated Object	Deleted Object
STATUS	1 (for active)	1 (for active)	2 (for inactive)
CREATED	sync time		
DELETED	null	null	sync time
LASTCHANGED	sync time	sync time	sync time
GSYS_EXT_INT1	1		2 or 3

Where no value is provided in the table above, ICON makes no change to the value that pre-exists for the associated field.

Table 18 shows some information that ICON records in IDB for new, updated, and deleted object relationships when resynchronization is run. `LASTCHANGED` information does not pertain to relationships and thus is not included in the table.

Table 18: IDB Stores Object Relationship Changes

STATUS	New Object Relationship	Updated Object Relationship	Deleted Object Relationship
CREATED	1 (for active)	1 (for active)	2 (for inactive)
DELETED	sync time		
LASTCHANGED	null	null	sync time
GSYS_EXT_INT1	1		2 or 3

Reliability Flag

ICON uses a reliability flag to provide downstream reporting applications (such as Genesys Info Mart) the ability to extract the most reliable data from two separate instances of IDB. The `GSYS_EXT_INT1` field stores a value, in all IDB configuration records, which indicates the reliability of the data (see Table 19).

Table 19: Reliability Flag Values

Value	What Value Indicates
0	The creation or deletion timestamp is reliable. The timestamp was provided by either real-time Configuration Server notifications or the configuration history log.
1	The creation timestamp is not reliable. The timestamp corresponds to the initial data load or resynchronization time.
2	The deletion timestamp is not reliable. The timestamp corresponds to the initial data load or resynchronization time.
3	Neither the creation nor the deletion timestamp is reliable. Both timestamps correspond to the time of the initial data load or resynchronization.

How Long Does Resynchronization Take?

The amount of time required for the synchronization process to complete depends on a number of factors including:

- Performance of Configuration Server
- Network performance
- Size of the Configuration Database
- Performance of the database that hosts the IDB.

Under normal conditions, synchronization should take several minutes.

When to Resynchronize IDB

In some cases, ICON recognizes that the configuration data in the IDB is suspect and produces a special log message (`09-25131`) to that effect. In most cases, however, no such message will be generated and the decision to resynchronize configuration data is left to your discretion.

Note: If ICON generates this special log message, it will stop monitoring configuration changes, and move to a waiting state where it will remain until it receives the user command to start resynchronization. To avoid any loss of configuration data changes, Genesys strongly recommends setting an alarm condition (see "Setting an Alarm Condition"). ICON immediately resumes monitoring configuration changes upon completion of resynchronization.

The following *exceptional* circumstances are guidelines you can use to determine if resynchronization is required:

- The RDBMS for the Configuration Database has crashed and you must recover it from backup. See "Restoring the Configuration Database from Backup" on page 141.
- Unavoidable events have prevented ICON from running for a period of time, while Configuration Server was operating and changes were made to configuration objects or their relationships to other objects.
- Configuration tracking for ICON was mistakenly turned off for a period of time while changes were made to configuration-related data.
- ICON ran configuration tracking against the wrong Configuration Database. (This is a user-driven error.)
- ICON detects an inconsistency in configuration data and logs a message accordingly. (The first message of inconsistency logged following IDB upgrade is normal and should be ignored.)
- Your downstream reporting application (such as Genesys Info Mart), detects missing or inconsistent configuration data in IDB. For Genesys Info Mart users, refer to the Genesys Info Mart documentation for more information on this exceptional circumstance.

Setting an Alarm Condition

ICON generates the following Standard-level log event if it suspects that the configuration data in the IDB is inconsistent:

```
09-25131: Configuration data inconsistency is detected; reason:
[reason]. Waiting for customer command...
```

It will then stop monitoring configuration changes, and move to a waiting state where it will remain until it receives the user command to start resynchronization.

To avoid any loss of data, Genesys *strongly* recommends that you set an alarm condition using the Solution Control Interface based on this triggering log event. *Framework 7.6 Solution Control Interface Help* explains how to create alarm conditions using the Alarm Condition Wizard.

When you arrive at the `Alarm Detection` page in the wizard, specify a `Detect` log event as shown in Figure 12.

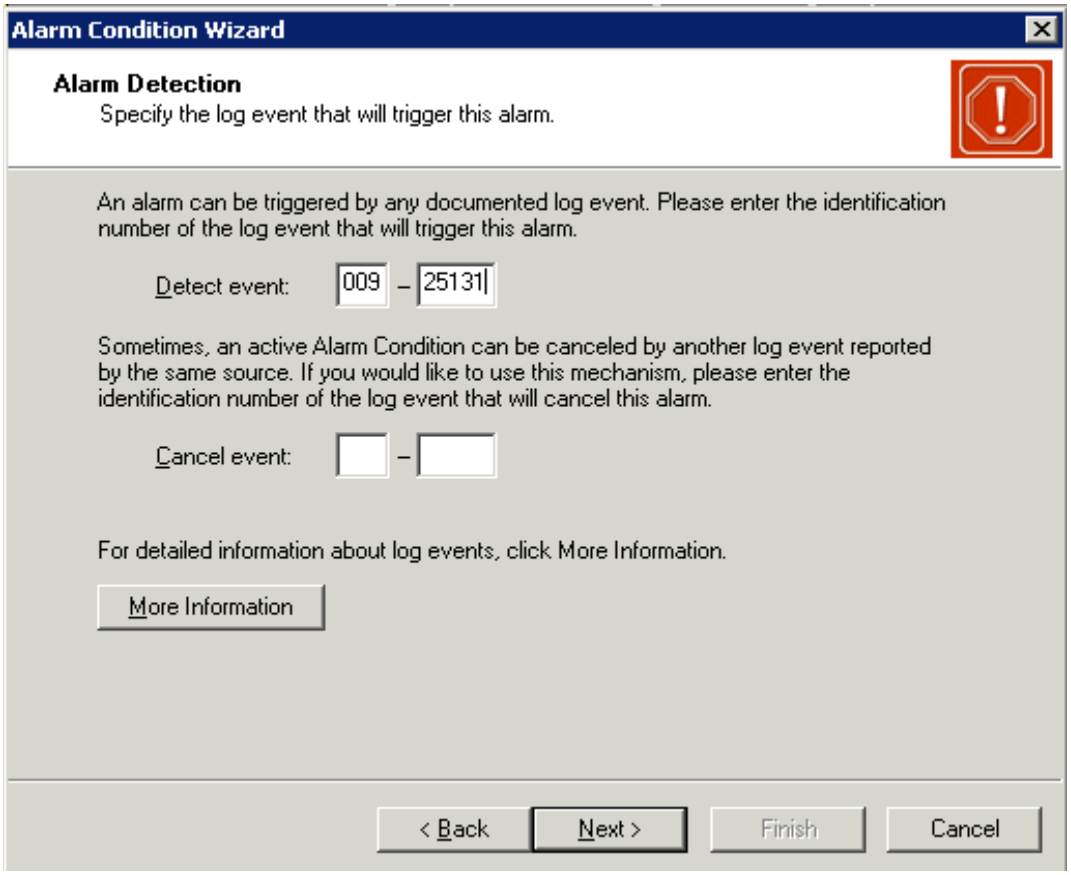

Figure 12: Specifying Alarm Detection Event for ICON Within SCI

Although there is no corresponding `Cancel` (or clearing) event, ICON generates log event `09-25017`, when all the data necessary for resynchronization has been retrieved. This log event can be used as a clearance event (see "Recommendations for Resynchronization" on page 140). You can also set an alarm `Clearance Timeout` which will clear the alarm condition after the specified time (in hours) has expired (see Figure 13).

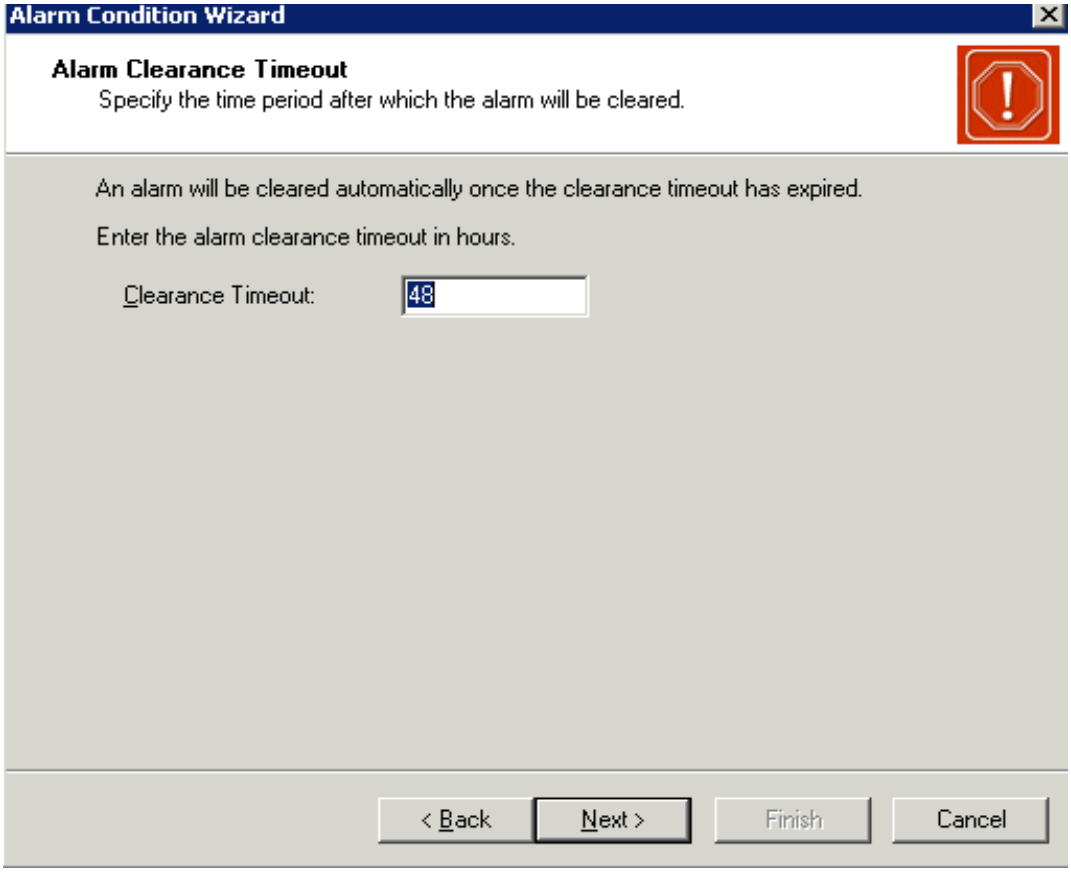

Figure 13: Specifying Clearance Timeout for ICON Within SCI

How to Resynchronize Configuration Data

If you must resynchronize your configuration data in IDB, do the following while ICON is running:

1. Verify that the ICON application, with the `role` configuration option set to `cfg`, is started.
2. In Configuration Manager, set the `start-cfg-sync` option value to `0` in the `Application Properties` window of this ICON application.
3. Click `OK` in the option properties window.
4. Click `Apply` in the ICON `Application Properties` window to enable this setting.
5. Change the option value to `1`.
6. Click `OK` in the `Option Properties` window.
7. Click `Apply` in the ICON `Application Properties` window to enable this new setting.

This action prompts Interaction Concentrator to start the resynchronization process. Once started, the resynchronization process continues until completion regardless of any subsequent changes to the option value. ICON logs a message when synchronization begins and when it completes.

Note: It does not suffice for the `start-cfg-sync` option to be preset to `1` or for it to be set to any other nonzero value and then to `1`. ICON starts synchronization only when it detects the change from `0` to `1`.

Recommendations for Resynchronization

- Consider changing the Configuration Server operation mode to `Read-only` for the time period during which data resynchronization is performed.

  When ICON retrieves from Configuration Server all the data necessary for resynchronization, ICON generates the following Standard-level log event:

  ```
  09-25017 Configuration objects are reloaded in IDB
  ```

  At this point, you can change the Configuration Server operation mode back to normal. Consider using this log event as a clearance event when configuring the alarm condition for resynchronization (see “Setting an Alarm Condition” on page 137).
- If you are using downstream reporting applications that use Interaction Concentrator as their data source, do the following to verify that the data in IDB is ready before you start your ETL engine for the first time after the resynchronization process:
  - Check ICON logs for log event: `09-25017, Configuration objects are reloaded in IDB`
  - Execute the following SQL statement against IDB:

    ```
    select eventid from G_SYNC_CONTROL where providertag = 5
    ```

  If the above statement returns no records or `eventID = 0`, the resynchronization is still in progress.

  If the above statement returns a non-zero value of `eventID`, the resynchronization is completed, and it is safe to run your ETL engine.

If you are restoring data from a backup Configuration Database (after your primary Configuration Database is corrupted, for example), perform the following steps:

a. Restore the Configuration Database from a backup copy (see “Restoring the Configuration Database from Backup”).

b. Restart Interaction Concentrator.

c. Perform manual resynchronization of configuration data in IDB as described in “How to Resynchronize Configuration Data” on page 139.

Restoring the Configuration Database from Backup

To restore your Configuration Database from backup, do the following:

1. Stop Configuration Server.
2. Stop the Interaction Concentrator instance tracking configuration data changes (`role = cfg`).
3. Restore the Configuration database from backup.
4. Make modifications in the Configuration Database to prevent Configuration Server from reusing the same DBID previously reported to ICON (see "Modifying the Configuration Database").
5. Start Configuration Server.

Modifying the Configuration Database

To prevent Configuration Server from reusing the same (already reported) DBIBs, make the following modifications to the Configuration Database:

Note: These steps must be repeated for each type of configuration object stored by ICON (see Table 20).

1. Check the last object DBID reported to ICON by Configuration Server by executing the following SQL statement against the IDB (and saving the result):

   select `max(id)` from `<IDB_TABLE_NAME>`

   Note: Replace `<IDB_TABLE_NAME>` in the SQL statement with the correct table name (see Table 20).

2. If the result of the statement executed in step 1 is *not* `Null`—that is to say, some records exist in the IDB table—go to Step 3. If the result is `Null`, repeat Step 1 for the next type of configuration object in Table 20.
3. Check the maximum value of the last object DBID by executing the following SQL statement against the Configuration Database:

   select `MAX_DBID` from `CFG_MAX_DBID` where `object_type = <cfg_object_type>`

   Note: Replace `<cfg_object_type>` in the SQL statement with the integer code of the Configuration Server object type (see Table 20).

4. If the result of the statement executed in Step 3 is any value—that is to say, *not* `Null`— go to Step 6. If the result is `Null`, go to Step 5.

5. Insert a record in the `CFG_MAX_DBID` Configuration Database table by executing the following SQL statement against the Configuration Database:

   insert into `CFG_MAX_DBID (object_type,max_dbid)` values `(<cfg_object_type>, <max_id> + 1)`

   Notes: Replace `<cfg_object_type>` in the SQL statement with the integer code of the Configuration Server object type (see Table 20).

   Replace `<max_id>` in the SQL statement with the value of `max(id)` from Step 1.

6. Update the record in the `CFG_MAX_DBID` Configuration Database table by executing the following SQL statement against the Configuration Database:

   update `CFG_MAX_DBID` set `max_dbid = <max_id> + 1` where `object_type = <cfg_object_type>`

   Notes: Replace `<cfg_object_type>` in the SQL statement with the integer code of the Configuration Server object type (see Table 20).

   Replace `<max_id>` in the SQL statement with the value of `max(id)` from Step 1.

7. Repeat Steps 1 through 6 for each configuration object type stored by ICON.

Table 20: Configuration Server Object Types and IDB Tables

Object Type	IDB Table Name: <IDB_TABLE_NAME>	Config Server Object Type: <cfg_object_type>
Switch	GC_SWITCH	1
Endpoint (DN)	GC_ENDPOINT	2
Person (agent)	GC_AGENT	3
Place	GC_PLACE	4
Groups	GC_GROUP	5
Tenant	GC_TENANT	7
Application	GC_APPLICATION	9
Scripts	GC_SCRIPT	12
Skills	GC_SKILL	13

Table 20: Configuration Server Object Types and IDB Tables (Continued)

Object Type	IDB Table Name: <IDB_TABLE_NAME>	Config Server Object Type: <cfg_object_type>
Action codes	GC_ACTION_CODE	14
Agent Login	GC_LOGIN	15
Folder	GC_FOLDER	22
Table Field	GC_FIELD	23
Table Formats	GC_FORMAT	24
Table access	GC_TABLE_ACCESS	25
Calling List	GC_CALLING_LIST	26
Campaigns	GC_CAMPAIGN	27
Treatments	GC_TREATMENT	28
Filters	GC_FILTER	29
Time Zones	GC_TIME_ZONE	30
Voice Prompts	GC_VOICE_PROMPT	31
IVR Ports	GC_IVRPORT	32
IVRs	GC_IVR	33
Business Attributes	GC_BUS_ATTRIBUTE	35
Attribute Values	GC_ATTR_VALUE	36
Objective Table	GC_OBJ_TABLE	37

Chapter

14 Using Special Stored Procedures

Aside from the stored procedures that Interaction Concentrator (ICON) uses internally to perform its functions, Interaction Concentrator provides a number of stored procedures that are relevant for users.

This chapter describes the stored procedures that are of special relevance to ICON users, including instructions on how to set up and execute them. Examples for each supported RDBMS are also provided.

This chapter contains the following sections:

Warning! SQL-like statements in this chapter are not true SQL statements; they are intended only to demonstrate the logic of the described stored procedures.

Merge Stored Procedure

The merge stored procedure merges voice interaction records in the Interaction Database (IDB) in order to finalize data processing of closed single-site and multi-site interactions. The procedure merges voice interaction records that are stored within a single IDB (*intra-IDB merge*). The procedure does not merge records that are distributed across more than one IDB (*inter-IDB merge* or *multi-IDB merge*).

The merge procedure can run in the background while instances of the ICON process are actively writing data to IDB.

Scheduling It is the responsibility of the customer (usually the Database Administrator) to run the merge procedure as required. Genesys recommends that you schedule the merge stored procedure to run on a regular basis. If you do not have this stored procedure scheduled to run regularly (not less than every 15 minutes), ensure that you run it before any data is extracted from IDB.

Note: If you are using Genesys Info Mart, run this procedure every five minutes.

If you run the merge procedure occasionally, on an ad hoc basis, execution can take significantly longer than if you run it regularly. Also, in this case, ICON performance can be adversely affected while the procedure is executing.

This section contains the following information about the merge procedure:

- How the `gsysIRMerge` and `gsysIRMerge2` stored procedures function.
- Setting up the merge procedure (see page 150).
- Merge procedure parameters (see page 152).
- Executing the merge procedure (see page 153).

gsysIRMerge and gsysIRMerge2

Starting with release 7.5, Interaction Concentrator supports two stored procedures for voice data merge:

- `gsysIRMerge`—A wrapper for `gsysIRMerge2`.
- `gsysIRMerge2`—The stored procedure that actually performs the merge.

`gsysIRMerge` has been provided to ensure backward compatibility with Interaction Concentrator 7.2. Calls to `gsysIRMerge` 7.2 are compatible with calls to `gsysIRMerge` 7.5 and higher.

Tables Involved in the Procedure

Figure 14 shows the tables that are involved in the merge procedure.

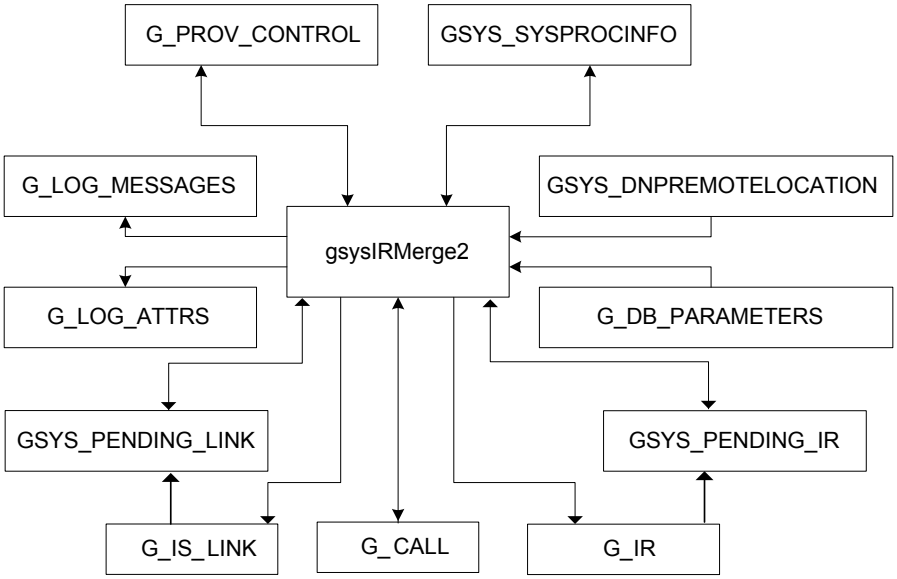

Figure 14: Tables Involved in the Merge Procedure

The following subsections describe each of these tables, and how they are used in the merge procedure.

G_LOG_MESSAGES and G_LOG_ATTRS

G_LOG_MESSAGES

The `G_LOG_MESSAGES` table contains information about the start (`MESSAGE_ID = 5020`) and end (`MESSAGE_ID = 5030`) of the merge procedure.

G_LOG_ATTRS

The `G_LOG_ATTRS` table contains the error descriptions (code and message), and detailed information about the processed data.

You can read information from these two tables by using Genesys Solution Control Interface (SCI), which is the graphical user interface (GUI) component of the Genesys Management Layer.

G_PROV_CONTROL and GSYS_SYSPROCINFO

G_PROV_CONTROL

From the `G_PROV_CONTROL` table, the merge procedure reads information about the ICON instances that are writing data to the database, and the corresponding number of the latest transaction.

The procedure executes the following query against the `G_PROV_CONTROL` table:

```
SELECT PRIMARYID SYS_ID          /* ICON instance identity*/
   SEQCURRENT                    /* number of latest transaction*/
FROM G_PROV_CONTROL
   WHERE PROVIDERTAG = 1 AND SEQCURRENT IS NOT NULL
```

In addition, the merge procedure reads from this table the number of its own latest transaction, using the following query:

```
SELECT SEQCOUNTER FROM G_PROV_CONTROL
   WHERE DOMAINID = 0
   AND PRIMARYID = 99
   AND PROVIDERTAG = 0
```

After this, the merge procedure starts data processing. After data processing is complete, the merge procedure updates the number of its own transaction to a new value.

GSYS_ SYSPROCINFO

From the `GSYS_SYSPROCINFO` table, the merge procedure reads information about the number of the latest ICON transaction that was processed, using the following queries for each ICON instance:

```
SELECT SEQPROCESSED FROM GSYS_SYSPROCINFO
   WHERE PROVIDERTAG = 1 AND PROCNAME = 'gsysirmerge'
```

After data processing is complete, the merge procedure updates the number of the transaction to the processed one.

GSYS_DNPREMOTELOCATION and G_DB_PARAMETERS

The data from the `GSYS_DNPREMOTELOCATION` and `G_DB_PARAMETERS` tables is used to process *stuck* inter-server (IS) links (that is, links for which there is no information about a corresponding IS-Link at another site).

The normal situation for the merge procedure is to merge interactions when IDB contains complete information about all parts of the interaction. However, for interactions that cross multiple sites, if all the sites are not monitored by ICON instances writing to the same IDB, the IDB will never have enough information to conclude that the interaction has ended and to identify the constituent parts of the interaction. The IS-Link is stuck.

GSYS_ DNPREMOTE LOCATION

From the `GSYS_DNPREMOTELOCATION` table, the merge procedure reads the names of the switches that are not monitored by any ICON instance that writes to this particular instance of IDB. From the point of view of this database, unmonitored switches are considered to be remote locations.

Records in the `GSYS_DNPREMOTELOCATION` table indicate to the merge procedure that IS-Links that point to remote locations are unpaired within this instance of IDB. Therefore, the merge procedure does not expect any further information about the interaction on the other end of the IS-Link. In this way, data from the `GSYS_DNPREMOTELOCATION` table enables the merge procedure to minimize the amount of time required in order to process stuck IS-Links, because the procedure will not wait for the IS-Link timeout to expire. (For more information about the IS-Link timeout, see “G_DB_ PARAMETERS”. See also `stuckthreshold` on page 150.

G_DB_ PARAMETERS

From the `G_DB_PARAMETERS` table, the merge procedure reads the time interval, in seconds, at which information about the IS-Link at another site is expected to be reported, using the following query:

```
SELECT VAL FROM G_DB_PARAMETERS
```

```
WHERE SECT = 'merge' AND
      OPT = 'stuckthreshold'
```

The merge procedure also uses other parameters from the G_DB_PARAMETERS table. For more information, see “IS-Link Timeout and Other Parameters” on page 150.

GSYS_PENDING_IR and GSYS_PENDING_LINK

The GSYS_PENDING_IR and GSYS_PENDING_LINK tables track the merge state of the interaction records and IS-Links while they are being processed. The merge procedure populates the GSYS_PENDING_IR and GSYS_PENDING_LINK tables with temporary data.

G_IS_LINK, G_IR, and G_CALL

G_IS_LINK

The merge procedure uses the G_IS_LINK table to determine multi-site interactions. A selected multi-site interaction is considered to be completed when one of the following occurs:

- All IS-Links have a corresponding IS-Link from another site.
- An IS-Link has been opened to an unmonitored switch.
- The timeout for an IS-Link to arrive from another site has expired, and all corresponding IRs are closed.

When a multi-site interaction is completed, the procedure makes the following updates:

```
UPDATE G_IS_LINK SET MERGESTATE = 3
  WHERE LINKID in (ALL IS LINK IN INTERACTION)

UPDATE G_IR SET MERGESTATE = 3
   ROOTIRID = (FIRST IR IN INTERACTION)
   GSYS_MSEQ = (the procedure's current TRANSACTION ID)
   GSYS_MSEQ_TS = (time of TRANSACTION)
WHERE ROOTIRID in (ALL ROOT IRs IN INTERACTION)

UPDATE G_CALL
   SET ROOTIRID = (FIRST IR IN INTERACTION)
WHERE ROOTIRID in (ALL ROOT IRs IN INTERACTION)
```

For a single-site interaction, if all corresponding IRs are closed, the interaction is completed, and the procedure makes the following update:

```
UPDATE G_IR SET MERGESTATE = 3
   GSYS_MSEQ = (the procedure's current TRANSACTION ID)
   GSYS_MSEQ_TS = (time of TRANSACTION)
WHERE IRID in (IRs IN INTERACTION)
```

Setting Up the Merge Procedure

To set up the merge procedure, ensure that the following information in IDB is correct for your purposes:

Unmonitored Switches

- The `GSYS_DNPREMOTELOCATION` table contains the names of the switches that are not monitored by any ICON instance that writes to this IDB. If necessary, manually update the `GSYS_DNPREMOTELOCATION` table, using a standard `INSERT` statement. The `REMOTELOCATION` column in the `GSYS_DNPREMOTELOCATION` table stores the names of unmonitored switches.

Example

For example, suppose that you have two switches, each of which is monitored by a separate ICON that writes to its own IDB:

- `ICON1` monitors switch `SITE1_sw1` and writes to `IDB1`.
- `ICON2` monitors switch `SITE2_sw2` and writes to `IDB2`.

For optimal performance of the merge stored procedure, add the following records to the respective IDBs:

- In `IDB1`, set `GSYS_DNPREMOTELOCATION.REMOTELOCATION='SITE2_sw2'`
- In `IDB2`, set `GSYS_DNPREMOTELOCATION.REMOTELOCATION='SITE1_sw1'`

IS-Link Timeout and Other Parameters

- The `G_DB_PARAMETERS` table stores parameters that the merge procedure uses to control its operation. Table 21 lists the parameters and their default values that the merge procedure uses.

Table 21: Merge Parameters in the G_DB_PARAMETERS Table

OPT Column	VAL Column			Description
	Oracle	Microsoft SQL	DB2	
nodnrl	0	0	0	Specifies whether the merge procedure disregards remote locations. Valid values: `1` Disregard `REMOTELOCATION`. `0` Do not disregard `REMOTELOCATION`.
stuckthreshold	28860	28860	28860	Specifies the time interval, in seconds, at which IS-Link information is expected from another site. After this time period expires, the IS-Link is considered to be stuck.
Note: All entries have column `SETID` = `0` and column `SECT` = `'merge'`.				

Table 21: Merge Parameters in the G_DB_PARAMETERS Table (Continued)

OPT Column	VAL Column			Description
	Oracle	Microsoft SQL	DB2	
step	75	75	25	Specifies the number of transactions that the merge procedure selects at the same time (when it is reading in new IRs and new links). A low value limits exposure to locks on core tables. However, a very low value can result in too many iterations. For additional information about large-scale deployments using Microsoft SQL, see "Note for Large-Scale Deployments Using Microsoft SQL".
limit	1000	2000	1000	Specifies the number of root interactions that the merge procedure considers for closure at the same time (when it updates the `G_IS_LINK`, `G_IR`, and `G_CALL` tables). This setting effectively limits the number of IRs per `MSEQ`. A very low value can result in too many iterations. For additional information about large-scale deployments using Microsoft SQL, see "Note for Large-Scale Deployments Using Microsoft SQL".
limit2	10000	10000	10000	Specifies the number of new links and interaction records that the merge procedure reads before it begins the closure phase (when it updates the `G_IS_LINK`, `G_IR`, and `G_CALL` tables). A very high value can result in suboptimal performance.
Note: All entries have column `SETID` = `0` and column `SECT` = `'merge'`.				

Note for Large-Scale Deployments Using Microsoft SQL

With Microsoft SQL, the default values of the `step` and `limit` parameters are not optimal for IDBs with large amounts of data (in the order of millions of interactions). For better performance, Genesys recommends the following as a first step:

- Increase the value of the `step` parameter to `200`.
- Increase the value of the `limit` parameter to `3000`.

However, you will likely need to experiment to find the optimal values for your large-scale deployment.

Updating the G_DB_PARAMETERS Table

If necessary, update the `G_DB_PARAMETERS` table. Interaction Concentrator provides a stored procedure, `svcUpdateDBParameters`, to perform this function. The stored procedure requires you to specify values for `SECT` (always `'merge'`), `OPT` (see the `OPT Column` in Table 21 on page 150), and `VAL`.

For example, to write the IS-Link timeout to the database, execute the following statement (the exact syntax depends on the RDBMS):

```
EXEC svcUpdateDBParameters
0,
'merge',
'stuckthreshold',
'<TIMEOUT>'
```

gsysIRMerge2 Parameters

The `gsysIRMerge2` stored procedure accepts five parameters: two input parameters and three output parameters.

When the `gsysIRMerge2` stored procedure is invoked by a call to the `gsysIRMerge` stored procedure, `gsysIRMerge` supplies the required parameters.

Table 22 lists the `gsysIRMerge2` input and output parameters, as well as the values that are used for the input parameters when `gsysIRMerge2` is called from `gsysIRMerge`.

Table 22: Merge Procedure Parameters

Parameter	Data Type	Description
Input		
OVERRIDE	int	Specifies the lock override setting. Valid values: `0` Do not override lock. `1` Override if `PREVCALLER==CALLER`. `2` Always override. `gsysIRMerge` value: `0`

Table 22: Merge Procedure Parameters (Continued)

Parameter	Data Type	Description
CALLER	varchar	Specifies the name of the caller of the stored procedure. Valid values: Any string (for example, `Infomart_DBID_1001`, `DCA6`) `gsysIRMerge` value: `'singleIDBMerge'`
Output		
RESULT	int	Specifies the result of the stored procedure call. Valid values: `0` Success. `2` IDB locked. `1` Other error. `gsysIRMerge` value: `RESULT`
PREVCALLER	varchar	If `RESULT==2`, specifies the name of the previous caller. `gsysIRMerge` value: `PREVCALLER`
PREVAGE	int	If `RESULT==2`, specifies the number of seconds since the previous caller obtained the lock. `gsysIRMerge` value: `PREVAGE`

The `RESULT` parameter provides information about the status of execution of the procedure. The other parameters are used for concurrency control, to ensure that no more than one instance of the procedure is running at any given time.

Note: The `gsysIRMerge` stored procedure discards the output parameters (`RESULT`, `PREVCALLER`, and `PREVAGE`).

Executing the Merge Procedure

Genesys recommends that you execute the merge procedure periodically, in order to reduce the amount of time that is required for data processing and for the delivery of correct data to other applications—for example, downstream reporting systems.

You can call the merge stored procedure in two ways:

- By calling `gsysIRMerge`
- By calling `gsysIRMerge2`

The following subsections discuss each of these methods in turn.

Note: The merge procedure can be executed on voice interaction data only.

Calling gsysIRMerge

To execute the merge procedure, use the following statement (the exact syntax depends on the RDBMS):

```
EXEC gsysIRMerge
```

The `gsysIRMerge` stored procedure then, in turn, calls `gsysIRMerge2`, using the following parameters: `1`, `'singleIDBMerge'`, `RESULT`, `PREVCALLER`, and `PREVAGE`.

Calling gsysIRMerge2

Invoking the `gsysIRMerge2` stored procedure directly enables you to specify your own values for the input and output parameters. You must supply the required parameters when you call the procedure. (For more information about the input and output parameters, see “gsysIRMerge2 Parameters” on page 152.)

For each supported RDBMS, there are different syntax requirements for the script that invokes the `gsysIRMerge2` stored procedure directly. This section provides the following examples:

- Example for Oracle
- Example for Microsoft SQL
- Example for DB2

Example for Oracle

```
declare
  OVERRIDE$ int := 1;
  CALLER$ varchar2(40) := 'singleIDBMerge';
  RESULT$ int;
  PREVCALLER$ varchar2(40);
  PREVAGE$ int;
begin
  gsysIRMerge2( OVERRIDE$, CALLER$, RESULT$,
     PREVCALLER$, PREVAGE$ );
  dbms_output.put_line('='||RESULT$);
end;
```

Example for Microsoft SQL

```
declare @OVERRIDE        int
declare @CALLER          varchar(40)
declare @RESULT          int
declare @PREVCALLER      varchar(40)
```

```
declare @PREVAGE         int
set @OVERRIDE  = 1
set @CALLER = 'singleIDBMerge'
exec gsysIRMerge2 @OVERRIDE, @CALLER, @RESULT, @PREVCALLER, @PREVAGE
```

Example for DB2

```
create procedure gsysIRMergeTest
DYNAMIC RESULT SETS 1
language SQL
begin
    declare OVERRIDE        int default 1;
    declare CALLER          varchar(40) default 'singleIDBMerge';
    declare RESULT          int;
    declare PREVCALLER      varchar(40);
    declare PREVAGE         int;

    call gsysIRMerge2( OVERRIDE, CALLER, RESULT,
        PREVCALLER, PREVAGE );
    begin
        declare c_cur cursor with return for
           select RESULT, PREVCALLER, PREVAGE from sysibm.sysdummy1;
        open c_cur;
    end;

end;
call gsysIRMergeTest;
drop procedure gsysIRMergeTest;
```

Improving Merge Procedure Performance

Database configuration, database settings, and merge procedure scheduling can significantly impact merge procedure performance.

To minimize the occurrence of deadlocks and to optimize merge procedure performance, carefully review the information in "Setting Up the Merge Procedure" on page 150 and in "Merge Procedure Performance Is Slow or Unstable" on page 183.

Computing Statistics

Periodically gathering statistics is a generic requirement for good database performance. For optimal performance of the merge procedure, ensure that the available database statistics are representative (in other words, relatively fresh).

Troubleshooting the Merge Procedure

For information about merge procedure execution problems and their solutions, see "Merge Procedure Problems" on page 180.

Note: To minimize the occurrence of deadlocks and to optimize merge procedure performance, carefully review the information in "Setting Up the Merge Procedure" on page 150 and in "Merge Procedure Performance Is Slow or Unstable" on page 183. However, deadlocks are inevitable, and the downstream reporting application must be capable of handling them.

If the procedure raises any exceptions, issue a rollback statement before performing any other actions on the connection.

Purge Stored Procedures

The size of IDB is the single biggest factor that affects Interaction Concentrator performance. To maintain a manageable IDB, you must implement a suitable purging strategy.

Interaction Concentrator provides two sets of stored procedures to purge IDB voice and Multimedia data. The purge procedures provide the following functionality:

- Safely purge voice and multimedia interaction data from IDB. All logically related information is deleted at the same time.
- Do not affect ICON performance while operating.
- Accept input parameters to limit the range of deleted data.

Note: Interaction Concentrator 7.5 and 7.6 does not support purging of data by Tenant ID.

This section contains the following information about the purge procedures:

- Descriptions of the purge procedures (see "Purge Procedures")
- Logging and error handling (see page 159 and page 166)
- Scheduling the purge procedures (see page 167)
- Setting up the purge procedures (see page 169)
- Executing the purge procedures (see page 170)

Purge Procedures

ICON 7.6.1 provides two sets of purge procedures to purge data safely from IDB:

- `gsysPurge76` and `gsysPurgeOS`
- `gsysPurgeIR`, `gsysPurgeUDH`, `gsysPurgeLS`, and `gsysPurgeOS`

Note: Genesys recommends using the `gsysPurge76` and `gsysPurgeOS` set of purge procedures if one of the following applies:

- You are a new ICON 7.6.1 customer.
- You need to purge large amounts of data.
- You are concerned about performance.

These stored procedures purge the following data from IDB:

- `gsysPurge76`—Purges voice, multimedia, open media, attached data, and agent login sessions in ICON release 7.6.1 (see "Purging Voice and Multimedia Interaction Data").
- `gsysPurgeIR`—Purges voice interaction records (IRs) (See "Purging Voice and Logically-Related Interaction Data" on page 164).
- `gsysPurgeUDH`—Purges User Data History (UDH) data (see "Purging User Data History" on page 165).
- `gsysPurgeLS`—Purges Agent Login Session (ALS) data (see "Purging Agent Login Sessions" on page 166).
- `gsysPurgeOS`—Purges Outbound data (see "Purging Outbound Sessions Data" on page 163).

The purge stored procedures are protected from concurrent use. (In other words, you can execute only one instance of a particular primary procedure at any one time).

The purge procedures can run in the background while instances of the ICON process are actively writing data to the IDB.

Retention Period

All the purge procedures accept an input parameter, `<number of days>` or `<count days>`, which specifies the *retention period* (the number of days, including the current date, for which data will be left in the database after the purge). Data for days preceding the retention period is eligible for purging. The input parameter must be a positive integer (minimum value = `1`, no maximum).

Purge Type

The `gsysPurge76` purge procedure accepts a second input parameter, `<deleteAllFlag>`, which further defines which data to purge in IDB.

Purging Voice and Multimedia Interaction Data

ICON 7.6.1 provides a new stored procedure—`gsysPurge76`—to purge voice, multimedia, open media, attached data, and agent login sessions from IDB. Genesys recommends that you use this procedure as your primary purge procedure if you are purging large amounts of data or if you are concerned about ICON performance. The `gsysPurge76` procedure uses two user-specified input parameters, `<number of days>` and `<deleteAllFlag>`, to purge IDB data.

First Input Parameter

The first input parameter, `<number of days>`, is used to calculate the retention period for data. It specifies the number of days, including the current one, for

which data will be retained. For example, if you run this stored procedure with <number of days> = 1, and <deleteAllFlag> = 1 (delete all records), then all voice, multimedia, open media, attached data, and agent login session interaction data older than one day will be purged from IDB. The valid values for this parameter are positive integers greater than, or equal to, 1.

Second Input Parameter

The second input parameter, <deleteAllFlag>, further defines which data to purge in IDB. The only valid values for this parameter are 0 and 1, where:

- <deleteAllFlag> = 0—ICON deletes only terminated records older than <number of days> day(s). Non-terminated records—records that ICON may update in the future—are retained in IDB.
- <deleteAllFlag> = 1— ICON deletes all records (voice and multimedia) older than <number of days> day(s).

Optimizes Performance

The new gsysPurge76 stored procedure differs from the gsysPurgeIR procedure in that it does not consider merged interactions when purging data. Instead, gsysPurge76 collects information about existing records in IDB, and then uses this data to optimize its performance in subsequent purges. This data may take considerable time to acquire the first time the purge procedure is executed.

The new purge procedure also provides the option to purge only terminated records using the <deleteAllFlag> parameter. This will affect the time it takes to perform the purge operation, because ICON first conditionally deletes terminated records in IDB and then rechecks all records not deleted on the first pass in case previously non-terminated records are now terminated.

Tables Involved in the gsysPurge76 Procedure

The gsysPurge76 purge procedure affects data in the following tables:

- GS_AGENT_STAT
- GS_AGENT_STS_WM
- G_LOGIN_SESSION
- G_AGENT_STATE_RC
- G_AGENT_STATE_HISTORY
- G_DND_HISTORY
- G_CUSTOM_DATA_P
- G_CUSTOM_DATA_S
- G_CUSTOM_STATES
- G_SECURE_USERDATA_HISTORY
- G_USERDATA_HISTORY
- GM_F_USERDATA
- GM_L_USERDATA
- G_IR
- G_IR_HISTORY
- G_VIRTUAL_QUEUE
- G_ROUTE_RESULT
- G_IS_LINK
- G_IS_LINK_HISTORY
- GX_SESSION_ENDPOINT
- G_PARTY
- G_PARTY_HISTORY
- G_PARTY_STAT
- G_CALL
- G_CALL_HISTORY
- G_CALL_STAT
- G_CALL_USERDATA
- G_CALL_USERDATA_CUST
- G_CALL_USERDATA_CUST1
- G_CALL_USERDATA_CUST2

gsysPurge76 Logging and Error Handling

The gsysPurge76 purge procedure provides logging information about the state and the results of the purge procedure, including the number of records that are flagged for deletion in a partition and the actual number of purged records in a partition.

The gsysPurge76 purge procedure stores information about the purge process in two IDB tables:

- G_LOG_MESSAGES— Stores log messages in the MESSAGETEXT and APPNAME fields.
- G_LOG_ATTRS—Stores parameters of log messages as attribute pairs in the ATTR_NAME and ATTR_VALUE fields.

G_LOG_MESSAGES

The MESSAGETEXT field in G_LOG_MESSAGES contains information about the action performed by the gsysPurge76 procedure, including:

- The start of the purge procedure (MESSAGE_ID=25922).
- The end of the purge procedure (MESSAGE_ID=25927).
- The start of the next purge procedure action (MESSAGE_ID=25920).
- The end of the purge procedure action (MESSAGE_ID=25930).
- Any execution errors (MESSAGE_ID=25925).

The APPNAME field in G_LOG_MESSAGES contains information about the name of the stored procedure that created the record in G_LOG_MESSAGES. It has the format: ICON DB:gsysPurge76

G_LOG_ATTRS

The G_LOG_ATTRS table stores parameters of log messages, generated by the gsysPurge76 procedure, as attribute pairs in the ATTR_NAME and ATTR_VALUE fields of the G_LOG_ATTRS table. The primary key (ID) of the G_LOG_MESSAGES table is used to identify related records in the G_LOG_ATTRS table, where LRID is a foreign key. For example, a record in G_LOG_MESSAGES with an ID = <id_value>, relates to the corresponding records in G_LOG_ATTR that have the same LRID value: LRID = <id_value>.

Table 23 shows all possible logging messages (MESAGES_TEXT) and attributes (ATTR_NAME, ATTR_VALUE) that the gsysPurge76 purge procedure (APPNAME) generates and stores in the G_LOG_MESSAGES and G_LOG_ATTRS tables.

Note: You can read information from the G_LOG_MESSAGES and G_LOG_ATTRS tables in their entirety by using Genesys Solution Control Interface (SCI), which is the graphical user interface (GUI) component of the Genesys Management Layer.

Table 23: Messages and Attributes Generated by gsysPurge76

G_LOG_MESSAGES Table		G_LOG_ATTRS Table	
MESSAGETEXT	**APPNAME**	**ATTR_NAME**	**ATTR_VALUE**
Purge procedure <purge procedure version> started	ICON DB: GSYSPurge76	MAX_PARTITION_TO_PURGE	<yyyymmdd>
		NUMBER_OF_DAYS_LEFT	<positive non-zero integer>
		DELETE_ALL	<0 or 1>
Purge initialization completed	ICON DB: GSYSPurge76	PartitionID	<0>
		Record_Count	<0>
GSYS_MARK_PARTITION mark partition started...	ICON DB: GSYSPurge76	PARTITION_TO_MARK From	<yyyymmdd>
		PARTITION_TO_MARK To	<yyyymmdd>
purging in progress...	ICON DB: GSYSPurge76	MAX_PARTITION_TO_PURGE	<yyyymmdd>
		PARTITION_COUNT	<the number of terminated and non-terminated records in the partition>
		DELETE_ALL	<0 or 1>
GSYS_PURGE_PARTITION purge started...	ICON DB: GSYSPurge76	PartitionID	<yyyymmdd>
		Record_Count	<0>
IDB:Purge-table:<table name> initiated...	ICON DB: GSYSPurge76	PartitionID	<yyyymmdd>
		Record_Count	<the number of terminated and non-terminated records in the table that belongs to the partition>
IDB:Purge-table:<table name> completed...	ICON DB: GSYSPurge76	PartitionID	<yyyymmdd>
		Record_Count	<the actual number of deleted records>
GSYS_PURGE_PARTITION purge completed	ICON DB: GSYSPurge76	PartitionID	<yyyymmdd>

Table 23: Messages and Attributes Generated by gsysPurge76 (Continued)

G_LOG_MESSAGES Table		G_LOG_ATTRS Table	
MESSAGETEXT	**APPNAME**	**ATTR_NAME**	**ATTR_VALUE**
GSYSPurge76 purge completed	ICON DB: GSYSPurge76	MAX_PARTITION_TO_PURGE	<yyyymmdd>
		NUMBER_OF_DAYS_LEFT	<positive non-zero integer>
		DELETE_ALL	<0 or 1>

Consider the following examples:

Example 1 The G_LOG_ATTRS table contains two attribute pairs, (PartitionID/20080627) and (Record_Count/2673), corresponding to the log message, "GSYS_PURGE_PARTITION purge started..." in the G_LOG_MESSAGES table.

In this example:

- The first attribute pair, (PartitionID/20080627), provides the ID of the partition to be purged.
- The second set of attributes, (Record_Count/2673), returns the total number of records found in the partition. In this case, 2673 records were found in partition 20080627. The actual number of records purged depends on the value of the <deleteAllFlag>.

Example 2 The following is a sample SQL statement which you might use to extract information from G_LOG_MESSAGES and G_LOG_ATTRS tables:

```
SELECT
G_LOG_MESSAGES.id,G_LOG_MESSAGES.TIMEWRITTEN,G_LOG_MESSAGES.MESSAGE
TEXT,G_LOG_ATTRS.attr_name,G_LOG_ATTRS.attr_value
FROM G_LOG_MESSAGES,G_LOG_ATTRS
WHERE G_LOG_MESSAGES.ID=G_LOG_ATTRS.lrid and G_LOG_MESSAGES.id>0
order by G_LOG_MESSAGES.id;
```

Scheduling gsysPurge76

It is the responsibility of the customer (usually the Database Administrator) to run the gsysPurge76 procedure as required. Genesys recommends that you run the purge procedure at a time when contact center activity is low.

If you run the purge procedure occasionally, on an ad hoc basis, execution can take significantly longer than if you run it regularly. Also, in this case, ICON performance can be adversely affected while the procedure is executing.

There are no specific restrictions about the order in which you must run the gsysPurge76 and gsysPurgeOS purge procedures.

If you are using both the gsysPurge76 and the gsysPurgeOS procedures to purge data, and if you are running the purge procedures sequentially, note the following recommendation:

- You may need to adjust the <number of days> or <count days> input parameter for purge procedures that execute later in the sequence, because the procedures may start after midnight.

  For example, if the procedure to purge interaction data starts at 05/05/2007 20:00 (08:00 PM) and the <number of days> input parameter is 1, set the input parameter for all procedures that will start after 05/05/2007 23:59:59 (11:59:59 PM) to 2.

Setting up gsysPurge76

The G_DB_PARAMETERS table stores parameters that the purge procedures use to control their operation. To set up the gsysPurge76 procedure, ensure that the parameter settings in IDB are suitable for your deployment.

Table 24 lists the parameters and their default values.

Table 24: gsysPurge76 Parameters in the G_DB_PARAMETERS Table

Applicable Procedure	G_DB_PARAMETERS Table		Description
	OPT Column	VAL Column	
gsysPurge76	rowspertransaction	0	gsysPurge76 will purge all records in a single table with the same partition ID in one transaction. This is the default value. **Note:** Values from 1 to 100,000 are considered invalid and will be treated the same as a value of 0.
		>100000	Specifies the maximum size of one transaction.
Note: All entries have column SETID = 0, and column SECT = 'purge76'.			

Updating the G_DB_PARAMETERS Table

If necessary, update the G_DB_PARAMETERS table. Interaction Concentrator provides a stored procedure, svcUpdateDBParameters, to perform this function. The stored procedure requires you to specify values for SECT (always 'purge76'), OPT (always 'rowspertransaction'), and VAL.

Executing gsysPurge76

Genesys recommends that you execute the gsysPurge76 procedure on a daily basis, in order to reduce the amount of time that is required for data processing and for the delivery of correct data to other applications—for example,

downstream reporting systems. For additional scheduling considerations, see "Scheduling gsysPurge76".

Calling gsysPurge76

You must supply the <number of days> and <deleteAllFlag> input parameters when you call the gsysPurge76 purge procedure (see "Purging Voice and Multimedia Interaction Data" on page 157).

To execute the gsysPurge76 stored procedure, use the statement that corresponds to your RDBMS:

On Microsoft SQL

```
EXEC GSYSPurge76 <number of days>, <deleteAllFlag>
```

On Oracle

```
EXEC GSYSPurge76 (<number of days>, <deleteAllFlag>);
commit
```

On DB2

```
CALL GSYSPurge76 (<number of days>, <deleteAllFlag>);
```

Examples

The following examples illustrate the syntax required to purge terminated multimedia interaction records older than 30 days:

Microsoft SQL

```
EXEC GSYSPurge76 30,0
```

Oracle

```
EXEC GSYSPurge76 (30,0);
commit
```

DB2

```
CALL GSYSPurge76 (30,0)
```

Note: In these examples, gsysPurge76 retains any non-terminated records older than 30 days.

Purging Outbound Sessions Data

The gsysPurgeOS, which is used to purge outbound-session (OS) data, calls the following stored procedure:

- gsysPurge_GOS—Deletes data related to outbound sessions.

Input Parameter

The gsysPurgeOS input parameter, <count days>, is used to calculate the retention period for outbound-session data. The purge procedure deletes all session-related data for closed outbound campaign sessions that have a termination date earlier than the current date minus <count days>. The current date starts at midnight, and the time segment of the current date is ignored.

gsysPurgeOS deletes outbound session–related data for closed campaign sessions only. If the parent campaign session is not closed, all corresponding chains will not be deleted.

Tables Involved in the OS Purge Procedure

The following tables contain OS-related data:

- GO_CAMPAIGN
- GO_CAMPPROP_HIST
- GO_METRICS
- GO_CHAIN
- GO_CAMPAIGNHISTORY
- GOX_CHAIN_CALL
- GO_RECORD
- GO_CHAINREC_HIST
- GO_CUSTOM_FIELDS
- GO_SECURE_FIELDS
- GO_SEC_FIELDHIST
- GO_FIELDHIST

After the OS purge procedure is completed, there are no references to non-existent data in the tables, except for possible references to non-existent interaction records.

Purging Voice and Logically-Related Interaction Data

Note: Genesys recommends using the new gsysPurge76 purge procedure if you are purging large amounts of data, or if you are concerned about ICON's performance. See "Purging Voice and Multimedia Interaction Data" on page 157.

Purging Voice Interaction Data

The gsysPurgeIR procedure, which is used to purge IRs, calls the following stored procedures:

- gsysPurge_GCC—Deletes IR data.
- gsysPurge_GUD—Deletes user data.

gsysPurgeIR deletes merged IRs only.

Input Parameter

The gsysPurgeIR input parameter, <count days>, is used to calculate the retention period for IR-related data. The purge procedure deletes all IR-related data for merged IRs that have a merge date earlier than the current date minus <count days>. The current date starts at midnight, and the time segment of the current date is ignored.

For example, if the date and time at which the stored procedure is invoked is 05/05/2007 13:15 and the input parameter is 1, all IRs that have a merge date earlier than 05/04/2007 00:00 will be deleted.

Tables Involved in the gsysPurgeIR Purge Procedure

The following tables contain IR-related data. Depending on the deployment, all the tables may not be populated:

- G_IR
- G_IR_HISTORY
- G_CALL
- G_CALL_HISTORY
- G_PARTY
- G_PARTY_HISTORY
- G_IS_LINK
- G_IS_LINK_HISTORY
- G_ROUTE_RESULT
- G_VIRTUAL_QUEUE
- G_CALL_STAT
- G_PARTY_STAT
- G_CALL_USERDATA
- G_CALL_USERDATA_CUST
- G_CALL_USERDATA_CUST1
- G_CALL_USERDATA_CUST2

After the IR purge procedure is completed, there are no references to non-existent data in the tables. For example, the G_PARTY_STAT table will not contain references to a party that has been purged from the G_PARTY table.

Purging User Data History

Note: Genesys recommends using the new gsysPurge76 purge procedure if you are purging large amounts of data, or if you are concerned about ICON's performance. See "Purging Voice and Multimedia Interaction Data" on page 157.

The gsysPurgeUDH procedure is used to purge User Data History (UDH) records. The procedure deletes records that are related to merged IRs only.

Input Parameter

The gsysPurgeUDH input parameter, <count days>, is used to calculate the retention period for UDH data. The purge procedure deletes all UDH records that satisfy one of the following conditions:

- The merge date of the corresponding IR is earlier than the current date minus <count days>.
- No corresponding IR exists, and the date the history record was added is earlier than the current date minus <count days>.

The current date starts at midnight, and the time segment of the current date is ignored.

Tables Involved in the UDH Purge Procedure

The following tables contain UDH data. Depending on the roles and the attached data specification configured for the ICON application(s) writing to IDB, the tables may not be populated.

- G_SECURE_USERDATA_HISTORY
- G_USERDATA_HISTORY

After the UDH purge procedure is completed, there are no references to non-existent data in the tables.

Purging Agent Login Sessions

Note: Genesys recommends using the new gsysPurge76 purge procedure if you are purging large amounts of data, or if you are concerned about ICON's performance. See "Purging Voice and Multimedia Interaction Data" on page 157.

The gsysPurgeLS procedure, which is used to purge Agent Login Session (ALS) data, calls the following stored procedure:

- gsysPurge_GLS—Deletes sessions.

gsysPurgeLS deletes closed sessions only.

Input Parameter

The gsysPurgeLS input parameter, <count days>, is used to calculate the retention period for ALS-related data. The purge procedure deletes all session-related data for closed agent login sessions that have a termination date earlier than the current date minus <count days. The current date starts at midnight, and the time segment of the current date is ignored.

Tables Involved in the ALS Purge Procedure

The following tables contain ALS-related data:

- G_LOGIN_SESSION
- G_AGENT_STATE_HISTORY
- G_AGENT_STATE_RC
- G_DND_HISTORY
- GS_AGENT_STAT
- GS_AGENT_STAT_WM
- GX_SESSION_ENDPOINT

After the ALS purge procedure is completed, there are no references to non-existent data in the tables, except for possible references to non-existent IRs.

Logging and Error Handling

G_LOG_MESSAGES

The G_LOG_MESSAGES table logs the following information about the state of a purge procedure operation:

- purge stored procedures Started—The primary procedure has started.
- ERROR—A database error has occurred. (The G_LOG_ATTRS table provides a description of the error.)
- Completed [OK|ERROR|Locked]—The primary procedure has ended or stopped. Completed: Locked means that the primary procedure was terminated because a prior instance of the procedure was still running.

G_LOG_ATTRS

The G_LOG_ATTRS table contains the following information about a purge procedure operation:

- Error descriptions (code and message).
- The value of the <count days> input parameter.
- Information about the processed data, including the number of records purged from the main table (for each primary purge procedure) and the number of records purged from all tables.

You can read information from the G_LOG_MESSAGES and G_LOG_ATTRS tables by using Genesys Solution Control Interface (SCI), which is the graphical user interface (GUI) component of the Genesys Management Layer.

Auto-Recovery

The purge procedures use a number of temporary tables and also update additional indexes during execution. If an error occurs during execution of a purge procedure, the procedure is terminated, but no special action is required to reset the procedure. The next time the procedure is started, the entry point is calculated from minimum merge sequence or timestamp values in the applicable IDB tables.

Timeout Interval

A maximum transaction time parameter sets a timeout interval for the procedure lock. If a purge procedure terminates because of an unhandled error, the lock will be overridden if the next instance of the procedure starts after the timeout interval has expired. For more information about the maximum transaction time parameter, see "Setting Up the Purge Procedures" on page 169.

Scheduling the Purge Procedures

Note: To schedule the new gsysPurge76 purge procedure, refer to "Scheduling gsysPurge76" on page 161.

It is the responsibility of the customer (usually the Database Administrator) to run the purge procedures as required. Genesys recommends that you run the purge procedures at a time when contact center activity is low.

If you run the purge procedure occasionally, on an ad hoc basis, execution can take significantly longer than if you run it regularly. Also, in this case, ICON performance can be adversely affected while the procedure is executing.

There are no specific restrictions about the order in which you must run the purge procedures. However, for best performance and to maintain data consistency throughout the process, Genesys recommends that you run the purge procedures sequentially, in the following order:

- gsysPurgeIR (purge IRs)
- gsysPurgeUDH (purgeUDH data)

- gsysPurgeLS (purge ALS data)
- gsysPurgeOS (purge OS data)

In particular, if you are purging more than one day's worth of data, execute the purge procedures sequentially.

Note the following additional considerations and recommendations:

- If you are running the purge procedures sequentially, you may need to adjust the <count days> input parameter for purge procedures that execute later in the sequence, because the procedures may start after midnight.

  For example, if the procedure to purge IRs starts at 05/05/2007 20:00 (08:00 PM) and the <count days> input parameter is 1, set the input parameter for all procedures that will start after 05/05/2007 23:59:59 (11:59:59 PM) to 2.
- Because the gsysPurgeIR purge procedures purge merged interactions only, consider your merge procedure schedule when scheduling the purge procedures. In particular, note that, even in a single-site deployment, you must run the merge procedure before you will be able to purge any records from IDB.
- Consider the extraction, transformation, and loading (ETL) cycle of your downstream reporting application, to ensure that you do not delete data before your downstream reporting application has successfully extracted the data.

  In general, schedule the purge procedures to run after your downstream reporting application has completed its regular ETL cycle. Keep an appropriately large sliding window of data, and delete only the oldest ETL cycle's worth of data each time you run the purge procedures.

  Your downstream reporting application may provide functionality that spreads the extraction of a backlog of data across multiple ETL cycles (for example, the limit-extract-data configuration option in Genesys Info Mart). If you are exercising this functionality, be aware that, at the end of a particular ETL cycle, your downstream reporting application may not have extracted all the data that was available at the time of extraction.

  Note: With Genesys Info Mart as your downstream reporting application, it is possible to run the purge procedures in parallel with the Genesys Info Mart Job_AggregateGIM and Job_MaintainGIM jobs.

- If your downstream reporting application does not successfully complete an ETL job cycle, ensure that you stop invoking the purge procedures until the problem is fixed and the ETL job cycle starts running successfully again. Otherwise, you may delete IDB source data before the downstream reporting application has extracted it.

Setting Up the Purge Procedures

Note: To set up the new `gsysPurge76` purge procedure, refer to "Setting up gsysPurge76" on page 162.

The `G_DB_PARAMETERS` table stores parameters that the purge procedures use to control their operation. To set up the purge procedures, ensure that the parameter settings in IDB are suitable for your deployment.

Table 21 on page 150 lists the parameters and their default values.

Table 25: Purge Parameters in the G_DB_PARAMETERS Table

Applicable Procedure	G_DB_PARAMETERS Table		Description
	OPT Column	VAL Column	
gsysPurgeIR	IR_seq_step	75	Specifies the chunk size, in terms of number of ICON transactions, that will be used for data lookup.
gsysPurgeUDH	UDH_time_step	300	Specifies the chunk size, in seconds, that will be used to calculate the number of rows to purge in a single database transaction.
gsysPurgeLS	LS_time_step	500	Specifies the chunk size, in seconds, that will be used to calculate the number of rows to purge in a single database transaction.
gsysPurgeOS	OS_time_step	500	Specifies the chunk size, in seconds, that will be used to calculate the number of rows to purge in a single database transaction.
gsysPurgeIR	IR_max_tran_time	600	Specifies the timeout interval for the procedure lock, in seconds. The maximum transaction time is the amount of time between the most recent purge transaction and the current time, after which a new instance of the procedure will be allowed to execute.
gsysPurgeUDH	UDH_ max_tran_time		
gsysPurgeLS	LS_ max_tran_time		
gsysPurgeOS	OS_ max_tran_time		
Note: All entries have column `SETID` = `0` and column `SECT` = `'purge'`.			

Updating the G_DB_PARAMETERS Table

If necessary, update the `G_DB_PARAMETERS` table. Interaction Concentrator provides a stored procedure, `svcUpdateDBParameters`, to perform this function. The stored procedure requires you to specify values for `SECT` (always `'purge'`), `OPT` (see the `OPT Column` in Table 21 on page 150), and `VAL`.

For example, to change the value of the procedure lock timeout for the IR purge procedure, execute the following statement (the exact syntax depends on the RDBMS):

```
EXEC svcUpdateDBParameters
0,
'purge',
'IR_max_tran_time',
'<TIMEOUT>'
```

Executing the Purge Procedures

Note: To execute the new `gsysPurge76` purge procedure, refer to "Executing gsysPurge76" on page 162.

Calling Purge Stored Procedures

You must supply the `<count days>` input parameter when you call the `gsysPurgeIR`, `gsysPurgeUDH`, `gsysPurgeLS`, and `gsysPurgeOS` purge procedures.

For each supported RDBMS, there are different syntax requirements for the script that invokes a purge procedure. This section provides the following examples:

- Example for Oracle (see page 170)
- Example for Microsoft SQL (see page 170)
- Example for DB2 (see page 171)

In these examples, the IR purge and ALS purge procedures will be executed sequentially in the same session, and will use different values for the retention period.

Example for Oracle

```
declare
  COUNT_DAYS$ int := 14;
begin
  gsysPurgeIR( COUNT_DAYS$ );
end;

declare
  COUNT_DAYS1$ int := 15;
begin
  gsysPurgeLS( COUNT_DAYS1$ );
end;
```

Example for Microsoft SQL

```
declare @COUNT_DAYS        int
set @COUNT_DAYS = 14
```

```
exec gsysPurgeIR @COUNT_DAYS

declare @COUNT_DAYS1       int
set @COUNT_DAYS1 = 15
exec gsysPurgeLS @COUNT_DAYS1
```

Example for DB2

```
language SQL
begin
    declare COUNT_DAYS      int default 14;

    call gsysPurgeIR( COUNT_DAYS );

end;

language SQL
begin
    declare COUNT_DAYS1     int default 15;

    call gsysPurgeLS( COUNT_DAYS1 );

end;
```

Stored Procedure gsysInitTimeCode

ICON uses the `gsysInitTimeCode` stored procedure to populate the `G_TIMECODE` table. Figure 15 shows the three tables that are involved in the time-setting procedure.

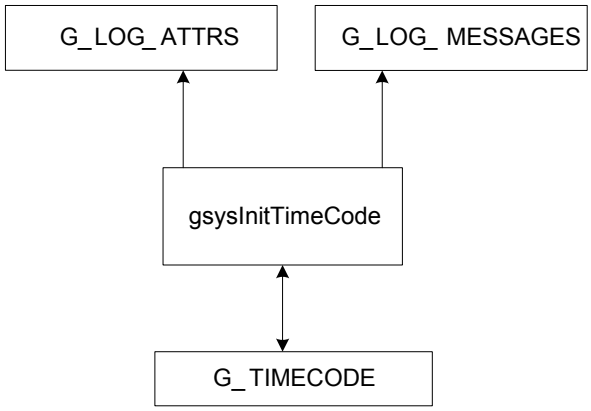

Figure 15: Tables Involved in the Time-Setting Procedure

The following subsections describe each of these tables, and how they are used in the time-setting procedure.

G_LOG_MESSAGES and G_LOG_ATTRS

G_LOG_MESSAGES The G_LOG_MESSAGES table contains information about the start (MESSAGE_ID = 5040) and end (MESSAGE_ID = 5050) of the time-setting procedure, as well as information about any errors (MESSAGE_ID = 5045) that occurred while the procedure was running.

G_LOG_ATTRS The G_LOG_ATTRS table contains the error descriptions (code and message) and detailed information about the processed data.

You can read information from these two tables by using Genesys SCI.

G_TIMECODE

After the time-setting procedure is executed, the G_TIMECODE table is filled, based on the input parameters for the procedure. You can use the G_TIMECODE table to create time-interval reports. The Field ID in this table is related to the *_TCODE fields in other tables, and it represents the amount of time, in seconds, counted in five-minute intervals, since January 1, 1970.

Setting Up the Time-Setting Procedure

No setup is required in order to execute the time-setting procedure.

Executing the Time-Setting Procedure

Execute the time-setting procedure as often as required, using the following input parameters:

- BEGIN_DATE—The date of the first interval.
- END_DATE—The date of the last interval.

To execute the time-setting procedure, use the following statement (the exact syntax depends on the RDBMS):

```
EXEC gsysInitTimeCode
getdate(),
getdate()+365
```

Custom Dispatchers

To support customized attached data processing, IDB initialization scripts create two custom dispatcher stored procedures:

- gudCustDisp1
- gudCustDisp2

You must modify the scripts in order to create custom dispatchers that store the attached data that you require.

For more information about the custom dispatchers, see “Custom Dispatchers” on page 50.

Chapter

15 Troubleshooting ICON Installation and Deployments

This chapter describes problems that you might encounter when starting or running your Interaction Concentrator (ICON) application, and how to resolve them. It contains the following sections:

Startup Problems

The following are the most common startup problems:

- ICON does not connect to the Configuration Server (see "No Connection to the Configuration Server").
- ICON closes at startup (see "ICON Exits at Startup" on page 120).

No Connection to the Configuration Server

Possible causes of this problem are as follows:

- Command-line parameters on the ICON `Application` object's `Server Info` tab incorrectly specify the Configuration Server host and port.

  Solution: Correct the command-line parameters and restart the application. For more information about the command-line parameters, see "Command-Line Parameters" on page 58.

- Configuration Server is not running, or it is inaccessible over the network.

  Solution: Start Configuration Server or re-establish the network connection.

ICON Exits at Startup

See the ICON log file for the reasons for the startup failure. Possible reasons include:

- The application name specified in the ICON startup command line does not correspond to any existing `Application` object in the Configuration Layer.

  Solution: Create the `Application` object. For more information about creating and configuring the ICON `Application` object, see the installation and configuration chapter in the *Interaction Concentrator 7.6 Deployment Guide.*

- The application name specified in the ICON startup command line refers to an `Application` object that is not of the `Call Concentrator` application type.

  Solution: Remove the `Application` object of the incorrect type, and then use the correct template to create a new `Application` object of the `Call Concentrator` type. For more information about creating and configuring the ICON `Application` object, see the installation and configuration chapter in the *Interaction Concentrator 7.6 Deployment Guide.*

- There is no assignment to a Database Access Point (DAP) `Application` object on the `Connections` tab of the ICON `Application` object.

  Solution: Add to the ICON `Application` object's `Connections` tab any DAP `Application` objects through which this ICON instance will access Interaction Databases (IDBs).

- The DAP `Application` object assigned on the ICON `Application` object's `Connections` tab does not have an associated DB Server application.

  Solution: Associate a DB Server with the DAP `Application` object. For more information, see the installation and configuration chapter in the *Interaction Concentrator 7.6 Deployment Guide.*

- The ICON instance has been configured to process call attached data (`role` = `gud`), but ICON cannot open the file specified in the `adata-spec-name` configuration option. The following error message in the log file indicates the existence of this condition:

  ```
  Std 02016 Unable to open attached data file '<attached data
  specification file name>', error code XXX
  ```

  Solution: Verify the following and correct as required.

  - The file specified in the `adata-spec-name` configuration option exists. If the file does not exist, create a new one or use the default attached data specification file (`ccon_adata_spec.xml`) provided in the Interaction Concentrator installation package.

  - The Interaction Concentrator user (the account under which ICON has been started) has the required permissions to read the attached data specification file.
- The persistent queue file has become corrupted.

  Solution: Force ICON to create a new persistent queue file by doing one of the following:
  - Using operating system commands, move or rename the corrupted `.pq` file. On restart, ICON will create a new `.pq` file with the original file name in the original location.
  - Reset the `pq-dbname` configuration option in the ICON `Application` object. On restart, ICON will create a new `.pq` file with the new file name in the specified location. For more information about the `pq-dbname` configuration option, see the configuration chapter in the *Interaction Concentrator 7.6 Deployment Guide.*

  In both cases, all unprocessed data in the old `.pq` file will be lost to ICON and IDB.
- There is no free disk space on the disk where the `apstorage.db` file resides.

  Solution: Free up memory on the disk or add more disk memory. For more information about the `apstorage.db` file, see "HA of Agent-Specific Data" on page 126.

Runtime Problems

The following are the most common runtime problems:

- ICON does not connect to T-Server or Interaction Server (see "No Connection to T-Server or Interaction Server" on page 122).
- ICON does not receive call-related events from T-Server (see "ICON Does Not Receive Call-Related Events from T-Server" on page 122).
- ICON does not write information to the database (see "ICON Does Not Write Information to the Database" on page 123).
- ICON has lost synchronization with the Configuration Database (see "ICON Has Lost Synchronization with the Configuration Database" on page 124).

No Connection to T-Server or Interaction Server

Possible causes of this problem are as follows:

- There is no assignment to the T-Server `Application` object or the Interaction Server `Application` object on the ICON `Application` object's `Connections` tab.

  Solution: Add to the ICON `Application` object's `Connections` tab any T-Server or Interaction Server `Application` objects from which this ICON instance will receive interaction-related information.

- The T-Server or Interaction Server application is not running, or it is not accessible over the network.

  Solution: Start the application or re-establish the network connection.

- The T-Server or Interaction Server `Application` object cannot connect to its `Switch` link.

  Solution: See the applicable troubleshooting guide for your particular T-Server or Multimedia Interaction Server.

- The release of the T-Server or Interaction Server `Application` object is not compatible with Interaction Concentrator. T-Server release 7.2 is the minimum version required by any release of Interaction Concentrator. Multimedia Interaction Server release 7.5 is the minimum version required for Interaction Concentrator support of Multimedia. For more information about Interaction Concentrator compatibility and interoperability with other Genesys components, see the first chapter in the Interaction Concentrator section of the *Genesys 7 Migration Guide*.

  Solution: Upgrade the T-Server or Interaction Server `Application` object to a compatible release.

- The `Switch` object associated with the T-Server `Application` object does not have all the necessary `DN` objects configured.

  Solution: Create the `DN` objects. For more information, see the *Deployment Guide* for your particular T-Server.

ICON Does Not Receive Call-Related Events from T-Server

Possible causes of this problem are as follows:

- ICON was not restarted after changes were made on the ICON `Application` object's `Connections` tab.

  Solution: Stop ICON, then restart. For more information, see "Starting and Stopping Interaction Concentrator" on page 57.

- There is no connection between the ICON `Application` object and T-Server. See "No Connection to T-Server or Interaction Server".

ICON Does Not Write Information to the Database

Possible causes of this problem are as follows:

- The database parameters are incorrectly specified on the DAP `Application` object. These parameters include the user name and password.

  Solution: Specify the correct values on the DAP `Application` object's `DB Info` tab, then restart ICON. For more information, see the configuration and installation chapter in the *Interaction Concentrator 7.6 Deployment Guide.*

- DB Server is not running, or it is inaccessible over the network.

  Solution: Start DB Server or re-establish the network connection.

- The RDBMS server is not available, or the IDB to which DB Server is trying to connect is not available.

  Solution: Take the necessary steps to make the database server and database available.

- The DAP `Application` object has been configured for a role that prevents it from writing certain classes of information to the database.

  Solution: Reconfigure the `role` option for the DAP `Application` object. Restart ICON. For more information about configuring a DAP see the configuration and installation chapter in the *Interaction Concentrator 7.6 Deployment Guide.* For a description of the `role` option, refer to the chapter about configuration options in the *Interaction Concentrator 7.6 Deployment Guide.*

- IDB has not been initialized by the Interaction Concentrator initialization scripts.

  Solution: Run the Interaction Concentrator initialization scripts. For more information, see the configuration and installation chapter in the *Interaction Concentrator 7.6 Deployment Guide.*

- ICON was not restarted after changes were made on the ICON `Application` object's `Connections` tab.

  Solution: Stop ICON, then restart. For more information, see "Starting and Stopping Interaction Concentrator" on page 57.

- Records are accumulating in the in-memory queue and are not being written to IDB.

  Solution: This might not be a problem. Configuration options control whether a size threshold or timeout triggers the transfer of records from the in-memory queue to the persistent queue, from which the records are then written to IDB. Wait for the event that triggers the transfer, and re-evaluate your configuration as necessary. For more information about `In-memory queue` configuration options, refer to the chapter about configuration options in the *Interaction Concentrator 7.6 Deployment Guide.*

- The program logic consistently produces an error because of incorrect RDBMS settings. For example, there may be insufficient free space available on the RDBMS for data storage, or the rollback segment may be too small.

  Solution: Review the error messages reported in the ICON log file. If you have configured an HTTP Listener, you can also view the error messages on the Database Writer performance counter web page (for more information, refer to "Monitoring Interaction Concentrator" on page 109). Provide the appropriate fix on the RDBMS side. For example, if the error messages cite no free space available for data storage, increase the table space.

  If the error was entirely related to the RDBMS problem, you do not need to restart ICON or perform any manipulation of the persistent queue (`.pq` file). However, if the `.pq` file has become corrupted and there are additional errors in the program logic, you must replace the `.pq` file (see the problem about a corrupted persistent queue file on page 121).

ICON Has Lost Synchronization with the Configuration Database

There are a number of reasons why ICON might lose synchronization with the Configuration Database, especially following a shutdown of ICON.

Loss of synchronization has the following impact on IDB:

- ICON fails to capture data about configuration objects created while ICON was stopped.
- ICON does not mark configuration data as deleted in cases where the applicable configuration objects were deleted while ICON was stopped.
- ICON fails to capture changes in associations between objects (while it is stopped).

Solution: If you suspect that your configuration data in IDB is inconsistent with Configuration Database, perform a manual resynchronization. See "How to Resynchronize Configuration Data" on page 139 for detailed instructions.

Merge Procedure Problems

The most common problems encountered in executing the merge procedure (`gsysIRMerge` or `gsysIRMerge2`) are as follows:

- The procedure fails to complete (see "Merge Procedure Does Not Complete Successfully").
- The procedure does not execute (see "Merge Procedure Does Not Execute" on page 126).

- Merge procedure performance is slow or constantly aborts (see "Merge Procedure Performance Is Slow or Unstable" on page 127).

For more information about the merge procedure, see "Merge Stored Procedure" on page 145.

Merge Procedure Does Not Complete Successfully

There are a number of possible causes of this problem, and describing them is beyond the scope of this section.

If the merge procedure does not proceed to the completed state, in addition to the steps you must take to resolve the specific cause of the merge procedure failure, you might also have to reset the merge procedure so that it recovers from its failed state (see "Merge Procedure Recovery").

Note: Under some circumstances, merge procedure recovery is not required. For example, the merge procedure may fail to complete successfully as a result of a deadlock condition. In this case, no special action is required other than to run the procedure again. However, if an error is discovered in the merge procedure, or more than one instance of ICON is running concurrently and the procedure state becomes corrupted, then the merge procedure recovery should be performed.

The following tables store information about the state of the merge procedure:

- `GSYS_PENDING_IR`
- `GSYS_PENDING_LINK`
- `GSYS_SYSPROCINFO`

Procedure: Merge Procedure Recovery

Purpose: To reset the merge procedure to recover from a failed state.

Start of procedure

1. Ensure that the merge procedure (`gsysIRMerge` or `gsysIRMerge2`) is not active.
2. Truncate the `GSYS_PENDING_IR` and `GSYS_PENDING_LINK` tables.
3. In the `GSYS_SYSPROCINFO` table, delete the rows in which `PROCNAME = 'gsysirmerge'`.
4. Reset the database lock if necessary (see Resetting the Database Lock Manually).

End of procedure

Merge Procedure Does Not Execute

Possible causes of this problem are as follows:

- The stored procedure was called incorrectly.

  Solution: Verify the syntax of the call to execute `gsysIRMerge` or `gsysIRMerge2`, and correct as required. For more information, see“Executing the Merge Procedure” on page 153.

- The database lock was not released after a previous execution (or attempted execution) of the merge procedure.

  The merge procedure sets a lock on IDB to prevent concurrent execution of the procedure and resulting data corruption. Steps to identify and correct the problem of an unreleased lock depend on whether the procedure was invoked by a call to `gsysIRMerge` or directly to `gsysIRMerge2`.

  - `gsysIRMerge` was called—`gsysIRMerge` runs `gsysIRMerge2` with the lock override setting at its most conservative (never override). If the previous lock was not released, the merge procedure aborts after doing the following:
    - Creates a record in the `G_LOG_MESSAGES` table with `MESSAGE_ID` = `5020` and `MESSAGETEXT` = `'LOCKED'`.
    - Creates a corresponding record in the `G_LOG_ATTRS` table with attributes `'PREVCALLER'` (the name of the previous caller) and `'PREVAGE'` (the number of seconds since the lock was obtained by the previous caller).

    Solution: Release the lock, then call `gsysIRMerge` again. To release the lock, see Resetting the Database Lock Manually.
  - `gsysIRMerge2` was called directly—`gsysIRMerge2` provides a `RESULT` output parameter to assist you in diagnosing the problem. If `RESULT==2`, the procedure failed to execute because the database was locked.

    Solution: Call `gsysIRMerge2` again, specifying an appropriate value for the `OVERRIDE` input parameter, and other input parameters as applicable, to allow this execution of the stored procedure to override the database lock. For more information about `gsysIRMerge2` parameters, see “gsysIRMerge and gsysIRMerge2” on page 146.

Procedure: Resetting the Database Lock Manually

Purpose: To manually release the lock on IDB introduced by the `gsysIRMerge` stored procedure.

Start of procedure

1. Ensure that the stored procedure is not running. The steps to verify this depend on your deployment and type of RDBMS.
2. Issue the following statement (the exact syntax depends on the RDBMS):

   ```
   update GSYS_SYSPROCINFO
   set TRANSACTION_ID = 0
   where DOMAINID = 0
   and  PRIMARYID = 99
   and  PROVIDERTAG = 0
   and  PROCNAME = '<CALLER>'
   ```

   <CALLER> is the value of the CALLER parameter that was provided to gsysIRMerge2 in the procedure call that resulted in the lock. If gsysIRMerge2 was called by gsysIRMerge, the CALLER was 'singleIDBMerge'.
3. Commit the statement issued in Step 2.

End of procedure

Merge Procedure Performance Is Slow or Unstable

Possible causes of this problem are as follows:

- On a DB2 platform, default values of certain database parameters result in an excessive number of deadlocks.

  Solution: Significantly increase the values of the LOCKLIST and MAXLOCKS database configuration parameters.

  Alternatively, execute ALTER TABLE...LOCKSIZE TABLE statements against the G_IR, G_CALL, and G_IS_LINK tables.
- On a Microsoft SQL platform, in large-scale deployments, default values of certain merge procedure parameters are not optimal.

  Solution: Significantly increase the values of the step and limit parameters in the G_DB_PARAMETERS table. For more information, see “Note for Large-Scale Deployments Using Microsoft SQL” on page 151.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X