Genesys

Genesys Rules System 8.1

Deployment Guide

About Genesys

Genesys is the world's leading provider of customer service and contact center software - with more than 4,000 customers in 80 countries. Drawing on its more than 20 years of customer service innovation and experience, Genesys is uniquely positioned to help companies bring their people, insights and customer channels together to effectively drive today's customer conversation. Genesys software directs more than 100 million interactions every day, maximizing the value of customer engagement and differentiating the experience by driving personalization and multi-channel customer service - and extending customer service across the enterprise to optimize processes and the performance of customer-facing employees. Go to 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

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

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. Before contacting technical support, please refer to the ***Genesys Care Program 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 Licensing Guide.***

Released by

Genesys Telecommunications Laboratories, Inc. www.genesyslab.com

Document Version: 81grs_dep_01-2013_v8.1.202.00

Table of Contents

List of Procedures

Preface

Welcome to the *Genesys Rules System 8.1 Deployment Guide*. This document describes how to install and configure Genesys Rules System.

This document is valid only for the 8.1 release of this product.

Note: For versions of this document created for other releases of this product, visit the Genesys Technical Support website, or request the Documentation Library DVD, which you can order by e-mail from Genesys Order Management at `orderman@genesyslab.com`.

This preface contains the following sections:

For information about related resources and about the conventions that are used in this document, see the supplementary material starting on page 117.

About Genesys Rules System

Genesys Rules System provides the ability to develop, author, and evaluate business rules. A business rule is a piece of logic that is defined by a business analyst. These rules are evaluated in a Rules Engine based on requests that are received from client applications.

Support for intelligent Customer Front Door (iCFD)

Genesys Rules System adds agility and control to the intelligent Customer Front Door (iCFD) solution by enabling customers to make dynamic decisions about how to treat their customers. For example, based on information about a customer collected through the Genesys Voice Platform and from Genesys

Conversation Manager, Genesys Rules System can help to determine the best message (such as a product upsell opportunity) to play to the customer.

Note: Support for hard-coded iCFD templates has been removed in release 8.1.2. See "Support for User-Defined Template Types" on page 10.

Support for intelligent Workload Distribution (iWD)

Genesys Rules System provides all the business rules functionality for the Genesys intelligent Workload Distribution (iWD) solution, a business application for dynamically prioritizing the distribution of work tasks to the people who are best suited to handle them. The Genesys Rules System enables business users to define priorities, SLAs, and other attributes of tasks.

Note: Starting with release 8.1.0 of iWD, the iWD solution no longer has its own embedded rules engine service, and rules development and authoring user interfaces are no longer integrated into iWD Manager. Instead, iWD now uses the Genesys Rules System to provide all of this functionality. iWD provides a Standard Rules Template for use with the Genesys Rules System, and the Genesys Rules Authoring Tool (GRAT) can be launched from iWD Manager without the need for separate user authentication.

Support for Web Engagement

Genesys Rules System release 8.1.2 implements a new template type (CEP—Complex Event Processing) for Web Engagement: This template type enables rule developers to build templates that rule authors then use to create rules and packages that use *event* fact types.

This selection determines how the Drools Rule Language (DRL) is eventually generated by the GRAT, and to which applications the rule package can be deployed.

Support for User-Defined Template Types

In release 8.1.2, hard-coding of template types for iCFD has been removed. Users can now define template types according to their own needs (including iCFD if required).

A template designer can assign a type to their templates, and then indicate whether or not that type supports events. GRAT now automatically displays the list of template types that have been published to it, and users can select these user-defined template types or define new ones.

Intended Audience

This document is intended primarily for IT staff who are responsible for the installation and configuration of Genesys Rules System, technical users who are responsible for rule template development, and business users who are responsible for rule authoring. It has been written with the assumption that you have a basic understanding of:

- Network design and operation.
- Your own network configurations.

Making Comments on This Document

If you especially like or dislike anything about this document, 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 scope of this document only and to the way in which the information is presented. Contact your Genesys Account Representative or 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.

Contacting Genesys Technical Support

If you have purchased support directly from Genesys, please contact Genesys Technical Support.

Before contacting technical support, please refer to the *Genesys Care Program Guide* for complete contact information and procedures.

Document Change History

This Section lists topics that are new or that have changed significantly since the first release of this document.

New in Document Version v8.1.202.00

he document has been updated to support localization of Genesys Rules System release 8.1.2. The following topics have been added or changed since the previous release of this document:

- Chapter 3, "Localization," on page 61.

New in Document Version v8.1.201.00

The document has been updated to support Genesys Rules System release 8.1.2. The following topics have been added or changed since the previous release of this document:

- "Support for Web Engagement" on page 10.
- "Support for User-Defined Template Types" on page 10.
- "New in Release 8.1.2" on page 16.
- "Creating an Application Cluster as a Rules Package Deployment Target" on page 37.
- "Rule Execution" on page 93.
- Appendix B, "DROOLS 5 Keywords," on page 99.

New in Document Version v8.1.101.00

The document has been updated to support Genesys Rules System release 8.1.1. The following topics have been added or changed since the previous release of this document:

- "Genesys Rules System Architecture" on page 15
- "Installation Prerequisites" on page 23
- "Installing the Genesys Rules Engine" on page 25
- "Installing the Genesys Rules Authoring Tool" on page 32
- "Installing the Genesys Rules Development Tool" on page 45
- "Rule Templates" on page 66
- "Workforce Management Parameters" on page 71
- "Rules" on page 81
- "Role Permissions" on page 90

Chapter

1 Genesys Rules System Overview

This chapter provides an overview of Genesys Rules System. It contains the following sections:

Genesys Rules System Features

Genesys Rules System provides the ability to develop, author, and evaluate business rules. A business rule is a piece of logic that is defined by a business analyst. These rules are evaluated in a Rules Engine based upon requests that are received from client applications. Some Genesys applications that can use the Rules Engine include VXML applications that are executed by the Genesys Voice Platform, SCXML applications that are executed by the Genesys Orchestration Server, and Genesys intelligent Workload Distribution (iWD) business processes that are executed by Genesys Interaction Server and Universal Routing Server.

Genesys Rules System includes three components:

- The Genesys Rules Development Tool (GRDT), which is an Eclipse plug-in that allows advanced users (business rules developers) to create templates that define the discrete rule conditions and actions that will comprise the rules. Each rule condition and action includes the plain-language label that the business rules author will see, as well as the rule language mapping that defines how the underlying data will be retrieved or updated.

- The Genesys Rules Authoring Tool (GRAT), which is a browser-based application that is used by business analysts to create and edit business rules based on the templates created in the Genesys Rules Development Tool.
- The Genesys Rules Engine (GRE), which evaluates the rule packages (groups of rules). Rule packages are deployed to the Rules Engine by the Rules Authoring Tool. When a rule package has been deployed, Genesys applications will be able to request the Rules Engine to evaluate the logic that is defined in this rule package.

Figure 1 illustrates the flow of a simple rule.

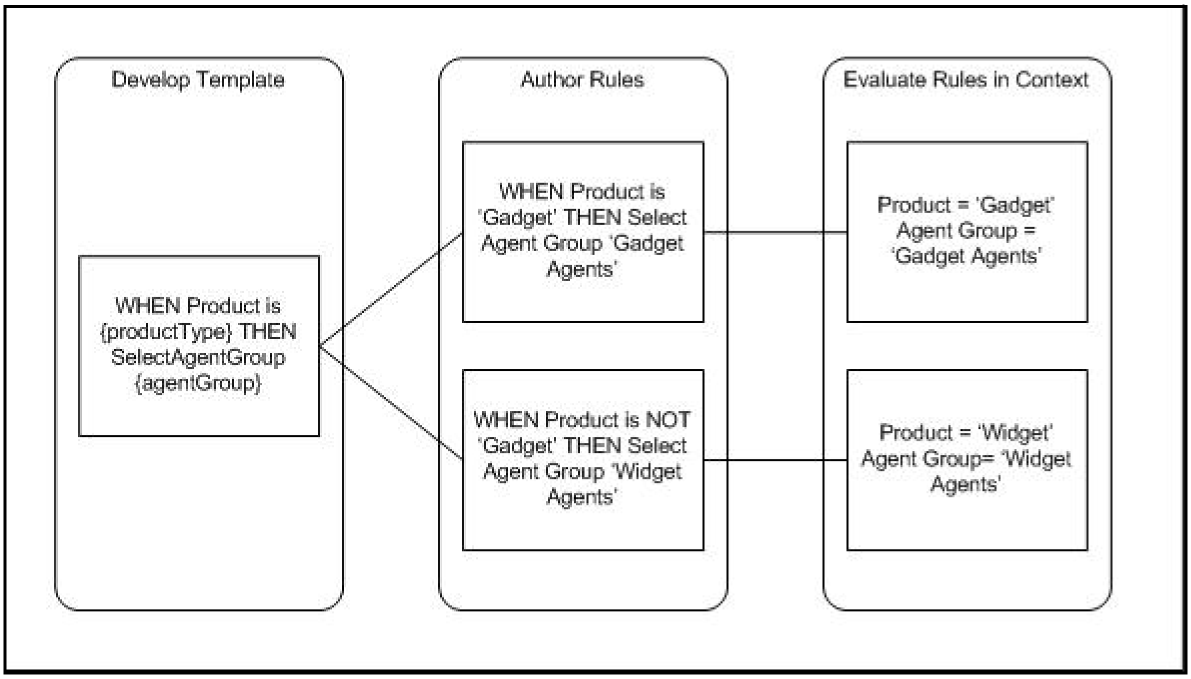

Develop Template
WHEN Product is {productType} THEN SelectAgentGroup {agentGroup}
Author Rules
WHEN Product is 'Gadget' THEN Select Agent Group 'Gadget Agents'
WHEN Product is NOT 'Gadget' THEN Select Agent Group 'Widget Agents'
Evaluate Rules in Context
Product = 'Gadget' Agent Group = 'Gadget Agents'
Product = 'Widget' Agent Group= 'Widget Agents'

Figure 1: Simple Rule

Genesys Rules System Architecture

Figure 2 illustrates the main Genesys Rules System building blocks, the relationships among them, and the external components that are involved.

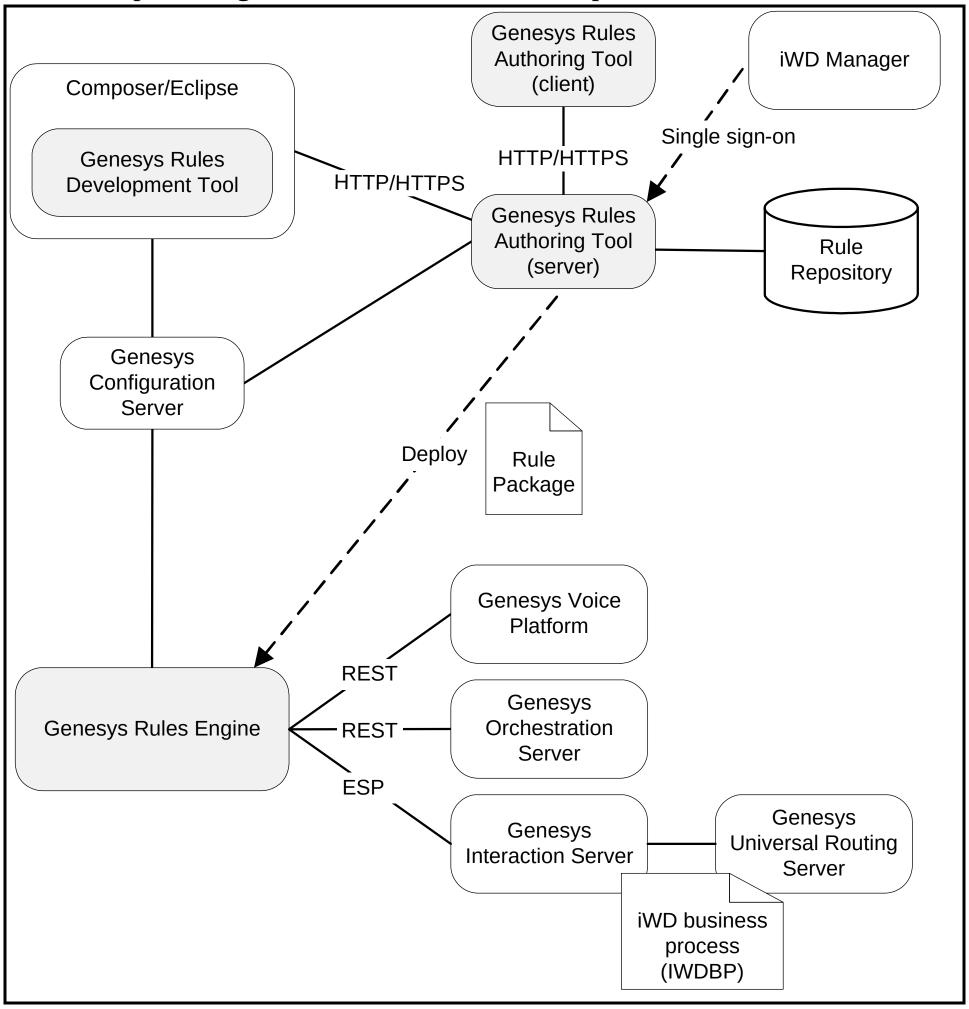

Genesys Rules Authoring Tool (client)
iWD Manager
Composer/Eclipse
Genesys Rules Development Tool
HTTP/HTTPS
HTTP/HTTPS
Single sign-on
Genesys Rules Authoring Tool (server)
Rule Repository
Genesys Configuration Server
Deploy
Rule Package
Genesys Voice Platform
REST
Genesys Rules Engine
REST
Genesys Orchestration Server
ESP
Genesys Interaction Server
Genesys Universal Routing Server
iWD business process (IWDBP)

Figure 2: Genesys Rules System Architecture

There are three main capabilities of Genesys Rules System. They are the following:

- Development interface—Supported primarily by the Genesys Rules Development Tool, which is an Eclipse plug-in that can be installed either on its own or within Genesys Composer. The Genesys Rules Development Tool (GRDT) is used by advanced users to create and modify rule templates. Rule templates provide a way to map the underlying data model to a more suitable business vocabulary that can be exposed to business users.

- Authoring interface—Supported primarily by the Genesys Rules Authoring Tool. This is an application that is used to create, edit, and deploy business rules. These rules use the templates that are developed in the development interface. The primary user of this application is a business user. In release 8.1.2, test scenarios can also be created, run and evaluated before rules are deployed.
- Rules evaluation—this is supported by the GRE. The Rules Engine responds to client applications that request rules evaluations.

Genesys Composer

You can use Genesys Composer to create applications for rules evaluation requests. Composer 8.1.0 is required, as it is the version in which the `Business Rule` block has been introduced. Refer to the Genesys Composer documentation for more information about how to use the `Business Rule` block. This function block is available on both the callflow diagram palette and the workflow diagram palette.

New in This Release

New in Release 8.1.2

This section contains a brief description of the new features in release 8.1.2.

- Genesys Rules Authoring Tool
  - Support for creating suites of test scenarios that can be used to validate the behavior of rules prior to deploying, and ensuring that existing rule functionality has not regressed.
  - Support for iWD and Genesys Web Engagement.
  - It is no longer necessary to create "initialize" conditions in order to create a reference to a fact (for example, `Initialize Customer`). Reference variables are now generated automatically.
  - Support for forced password change during user login.
  - For security, the user's last login time is displayed after login.
  - Support for mapping multiple instances of a rule parameter to a single parameter definition.
  - Support for duplicate template names across tenants.
  - Support for single-click grid editing.
  - Support for resizing of columns in linear rule grids. Any changes made by users are stored as preferences for that user and retained across logins.

    - Support for IP v6.
    - Support for localization by using Genesys Language Packs.
- Genesys Rules Development Tool
    - Support for either iWD, CEP or Stateless information for template types and package types.
    - Support for mapping multiple instances of a rule parameter to a single parameter definition.
    - Support for duplicate template names across tenants.
    - Support for multi-line actions and conditions, to enable longer expressions to be used.
    - Support for IP v6.
- Genesys Rules Engine
    - Support for Genesys TLS and TLS-FIPS security protocols.
    - In addition to returning to the calling application the action resulting from the evaluation of the rule conditions for an executed rule, GRE now also returns the following:
        - The name of each rule which evaluated as true
        - For decision tables, the name of each row of the decision table which evaluated as true
    - New Genesys branding is applied throughout the product's user interface and installers.
    - Support for IP v6.

New in Release 8.1.1

This section contains a brief description of the new features in release 8.1.1.

- The Genesys Rules Authoring Tool:
    - Enables a user to import and export all rules within a package to facilitate moving an entire rule package from one system to another (e.g., from a lab to production environment).
    - Includes enhanced rule search functionality that includes the ability to find a rule condition or action by parameter name and search for all rules pending deployment.
    - Provides a business user with a tooltip that describes the constraint to which the user must adhere for input value parameters. In the Genesys Rules Development Tool, a rule developer is given the ability to specify tooltip text that will display for a given parameter to the business user.
    - Enables a business user to view the name of the rule template from which each condition and action is derived.
    - Explorer Tree displays a visual indication if rules have changed in the package but have not yet been deployed.

    - Includes a Genesys Workforce Management (WFM) integration that enables the business user to select a value to be used in a rule action that is dynamically retrieved from a list of Multi-site Activities and Activities on the Genesys WFM Server.
- The Genesys Rules System (GRS):
    - Is now integrated with Intelligent Workload Distribution (iWD) 8.1.

New in Release 8.1.0

Genesys Rules System 8.1.0 includes the following features:

- The first release of the Genesys Rules Development Tool, which is an Eclipse plug-in that can be installed inside Genesys Composer or in a standalone version of the Eclipse platform. This initial release provides:
    - Editors for rule facts, conditions, actions, enumerations, and functions.
    - Authentication of the user with the Genesys Configuration Server to control access to Genesys configuration objects (for example, agent groups or skills) that can be referenced in rule templates.
    - The ability for a user to reserve a rule template so that no other users can modify it simultaneously.
- The first release of the Genesys Rules Authoring Tool, which is a browser-based user interface that is used primarily by business analysts to create and edit business rules. This initial release includes:
    - Support for both decision tables and linear rules.
    - On-demand and scheduled deployment of rule packages, to the GRE.
    - Version history of rules, allowing the user to revert a rule package with a previous version of a rule, or restore a previously deleted rule.
    - An audit trail of rule deployment history.
    - Creation and editing of business calendars that define the working hours and holidays for the business.
    - Role-based access control, using permissions that are defined in the Genesys Configuration Layer, to determine what parts of the application can be accessed by the user. Role-based access control requires Configuration Server 8.0.2 or higher, and Genesys Administrator 8.0.2 or higher.
    - Search capabilities to find instances of rules that match a specific Rule ID, that include specific words in the rule name, that were last modified by a specific user, that contain a certain business calendar, or that were created within a specific date range
    - Import and export of linear rules (in XML format)
    - Import and export of decision tables (in XML and XLS formats)
    - Contains the centralized, and versioned, repository that manages rules and templates

- The first release of the GRE, which is a rules evaluation component that runs inside a customer-provided application server. This initial release:
    - Hosts rule packages that are deployed by users through the Rules Authoring Tool
    - Evaluates rules, based on requests from client applications

Chapter

2 Installation

This chapter provides information about how to install Genesys Rules System. It contains the following sections:

Task Summary: Installing Genesys Rules System 8.1

The following table outlines the task flow for installation of Genesys Rules System 8.1. The procedures in this table provide instructions about installing Genesys Rules System components on Microsoft Windows. For information about how to install on UNIX-based operating systems, refer to “Installing Genesys Rules System on UNIX-Based Operating Systems” on page 48.

Task Summary: Installing Genesys Rules System 8.1

Objective	Related Procedures and Actions
1. Prepare for installation and review prerequisites.	• Ensure that your environment meets the prerequisites that are outlined in "Installation Prerequisites" on page 23. • Ensure that the required CD is available. See "CD Structure" on page 24.
2. Create the database for the Rules Repository.	Procedure: Creating the Rules Repository database, on page 25
3. Install the Genesys Rules Engine	Genesys Administrator: • Procedure: Deploying the Genesys Rules Engine in Genesys Administrator, on page 26 Configuration Manager: • Procedure: Creating the Genesys Rules Engine Application object in Configuration Manager, on page 29 • Procedure: Installing the Genesys Rules Engine, on page 31
4. Install the Genesys Rules Authoring Tool	Genesys Administrator: • Procedure: Deploying the Genesys Rules Authoring Tool in Genesys Administrator, on page 32 Configuration Manager: • Procedure: Creating the Genesys Rules Authoring Tool Application objects in Configuration Manager, on page 38 • Procedure: Installing the Genesys Rules Authoring Tool, on page 41
5. Deploy the `genesys-rules-authoring.war` and `genesys-rules-engine.war` files to your application server.	Procedure: Deploying the .war files, on page 44
6. Install the Genesys Rules Development Tool	Procedure: Installing the Genesys Rules Development Tool, on page 45
7. Define your business structure	See Chapter 5, "Business Structure," on page 85.

Task Summary: Installing Genesys Rules System 8.1 (Continued)

Objective	Related Procedures and Actions
8. Login to the Genesys Rules Authoring Tool	Procedure: Logging into the Genesys Rules Authoring Tool, on page 47
9. Review the Troubleshooting section for configuration tips and considerations	See “Troubleshooting” on page 52.

Preparing for Installation

This section describes how to prepare for the Genesys Rules System installation. It contains the following topics:

- “Installation Prerequisites”
- “CD Structure” on page 24

Note: Always stop the web application server (such as Tomcat or WebSphere) before you install or upgrade Genesys Rules System.

Installation Prerequisites

System support in this release includes:

- Virtual platforms:
  - IBM LPAR
  - Solaris 10 Containers
  - VMWare vSphere 4 Hypervisor
  - VMWare ESX 3.5
- Application servers:
  - IBM WebSphere Application Server 7.0
  - Apache Tomcat 6
- Databases (for the Genesys Rules Authoring Tool):
  - Oracle 10g, R2 & Oracle 11g, including RAC configurations
  - MS SQL Server 2005 & 2008
  - IBM DB2 9.5 & 9.7
- Web Browsers (for the Genesys Rules Authoring Tool):
  - Firefox Versions 4-16
  - Microsoft Internet Explorer 7.0, 8.0
- Other Platforms:
  - Citrix Xen App 5

Before you start the installation, ensure that the environment meets the following prerequisites:

GRE and GRAT:

- Web application server (IBM WebSphere Application Server 7.0 or Tomcat 6).
- JDK 6.0.
- In case multiple instances of Tomcat are deployed, ensure the `CATALINA_HOME` environment variable is set correctly for your application server. For example, for Tomcat, this might be set as `CATALINA_HOME=C:\GCTI\Tomcat` (depending on the root directory of your Tomcat installation).
- One of the supported databases must be installed and working.
- If you are using Genesys Composer on the same host on which you will be installing the application server for GRAT or GRE, ensure that Composer's embedded instance of Tomcat is not using the same port as the Genesys Rules System application server.

Genesys Rules Development Tool:

- Eclipse (minimum version 3.5.0) and/or Genesys Composer (minimum version 8.1.0) must be installed.

Notes: Refer to the *Genesys Supported Operating Environments Reference Manual* for a complete list of the application servers, operating systems, and databases that are supported.

Consult your application server documentation for information about memory and CPU.

CD Structure

Genesys Rules System software is packaged on one CD. The Genesys Rules System CD contains the installation packages for the following components:

- GRE—A server component that is invoked by Genesys client applications to evaluate application data against a set of deployed business rules.
- Genesys Rules Authoring Tool—A tool that consists of both a browser-based user interface (UI) and a server component. The UI component allows users to create, edit, deploy, and audit business rules that you can use with various Genesys solutions.
- Genesys Rules Development Tool—An Eclipse plug-in that you can use with a stand-alone Eclipse platform or with Genesys Composer, providing a UI for developing business rule templates.

Installing Genesys Rules System 8.1

This section contains the procedures that are used to install Genesys Rules System 8.1

Procedure:
Creating the Rules Repository database

Purpose: To create the database that will be used as the Rules Repository.

Start of procedure

1. Create a new database. The database will be populated by the Genesys Rules Authoring Tool.
2. Create a database user. This user must have full access, including `CREATE` and `INDEX`.
3. Most database distributions include the jdbc connector that is needed; otherwise, you must download it from the vendor's site. Genesys does not provide the jdbc connector.

   Note: Genesys Rules System 8.1 uses Java 6. The connector should be tested according to the database vendor's installation instructions to ensure that it can connect to the configured database.

4. (For all databases) Copy the JDBC connector to the application server common library directory (for example, `<Tomcat installation directory\Lib>`). The installation script prompts for the correct information.

End of procedure

Installing the Genesys Rules Engine

The GRE can be configured by using either Genesys Administrator or Configuration Manager. If you use Genesys Administrator, you can deploy the installation package from within Genesys Administrator. If you use Configuration Manager, you will have to create the application first and then run the installation package manually.

Procedure: Deploying the Genesys Rules Engine in Genesys Administrator

Purpose: To configure the GRE application and deploy the GRE installation package (IP) by using Genesys Administrator.

Genesys Administrator needs to be at Release 8.1.0 or later to install the GRE on 8.1.0 or later Configuration Servers.

Start of procedure

1. Import the installation package into Genesys Administrator:
   a. On the `Deployment` tab of Genesys Administrator, select `Import`.
   b. Select `Installation CD-ROM`.
   c. Click `Next`.
   d. Browse to the `MediaInfo.xml` file on the CD or the CD image location on the network (the path must be in UNC format).
   e. Click `Next`.
   f. Select GRE for your operating system as well as the appropriate type in the list in order to import the installation package.
      - For Management Framework 8.1, the type is `Business Rules Execution Server`.
      - For Management Framework 8.0 and earlier, the type is `Genesys Generic Server`.
   g. Select `Next` to start the import.
   h. Click `Finish` when the import is complete.
2. Install the GRE IP. Select the `Deployment` tab in Genesys Administrator. The list of installation packages will now show the GRE. Right-click and select `Install Package` for the IP for your operating system and type (see Step 1). Click `Next` to start the installation wizard. The following parameters must be defined/selected:
   a. Application Name for the GRE application
   b. Target Host—The host to which the `.war` file will be copied during the installation procedure
   c. Working Directory—The directory in which the `.war` file will be created
   d. Client Side IP Address (optional)
   e. Client Side Port (optional)
   f. Configuration Server hostname

g. Configuration Server port.

> Note: For a secure connection, the Configuration Server port should be of type `Auto Detect (Upgrade)`.

h. Connection delay time in seconds

i. Reconnect Attempts.

> Note: Items **a** through **i** will be written to the `bootstrapconfing.xml` file in the `.War` file. Any subsequent updates to the parameters will have to be made in that file.

On the next screen, enter Connection ID and Connection Port for the GRE.

Edit the Connection port for the genesys-rules-engine connection. The Connection Port is the connector port of the servlet container. For example, on Tomcat the default listening port is `8080`.

The Connection Protocol can be set in the configuration part under Provisioning.

Verify the previously-defined installation parameters on the Deployment Summary screen.

3. Configure the Rules Engine application:

   a. In the `Server Info` section, verify the default listening port, as well as the connector port on which the Rules Engine Servlet receives requests:
      - The `ID` value is the name of the Rules Engine web application. The default name of this application is “`genesys-rules-engine`”.
      - The `Listening port` is the connector port of the servlet container. For example, on Tomcat the default listening port is `8080`.
      - The `Connection Protocol` must be `http`.

   b. On the `Tenants` tab, add the Tenants that will be available to the Rules Engine.

   c. On the `Connections` tab, add a connection to Message Server if you want to use network logging.

d. On the `Options` tab, configure options. In addition to the standard logging options that you can configure, you can configure an option named `fileEncoding` in the `logging` section.

`fileEncoding` specifies the encoding that is to be used during creation of the log file—for example, `UTF-8`. This value is optional. If you do not specify this option, the server's locale information will determine the log file encoding.

This option is available for both the GRE and the Genesys Rules Authoring Tool. Also, the `log4j.properties` file that is included in both components supports a similar option, `log4j.appender.runtime.Encoding`. The `log4j.properties` file is used for initial log configuration prior to the reading of the log configuration from the Configuration Server database.

e. There are several optional configuration options in the `settings` section:

- `verify-deploy-address` indicates whether to verify that the TCP address of the application that is deploying rules is that of an associated Genesys Rules Authoring Tool. The default value is `true`.
- `sequential-mode` specifies whether to run the GRE in sequential mode. In sequential mode, no additional data can be inserted or modified after the initial data set, simplifying the GRE operations. The default value is `false`.
- `max-number-rule-executions` specifies the maximum number of rules to be executed during a request. This option is used to detect unwanted recursions when `sequential-mode` is `false`. Once the maximum is reached, an error is reported. The default value is `10000`.
- `deployed-rules-directory` (which must be manually added if you want to assign it a value) specifies the directory in which to keep the working copy of deployed rule packages. When a package is deployed, a copy of the deployed package is placed here. When the GRE is restarted, all packages that are defined in this directory are loaded and made available for execution. Specifying a `deployed-rules-directory` is recommended. If `deployed-rules-directory` is not assigned a value, the rule packages will be placed in the `WEB-INF\config` sub-directory within the `genesys-rules-engine` web application directory. At this location the deployed rule packages may get deleted when an updated `.war` file is deployed.
- `esp-worker-threads` (new in release 8.1.2), specifies the maximum number of worker threads available when using the ESP interface to execute rules. The default for this value is 5 threads.

f. `Save` your changes.

End of procedure

Next Steps

- Deploy the `genesys-rules-engine.war` file to your application server. See Procedure: Deploying the .war files, on page 44.

Procedure: Creating the Genesys Rules Engine Application object in Configuration Manager

Purpose: To create the `Application` object in Configuration Manager that will link the GRE with Configuration Server.

Start of procedure

1. In Configuration Manager, navigate to the `Application Templates` folder.
2. Import the application template that is to be used for the Rules Engine application:
   a. Right-click the `Application Templates` folder and select `Import Application Template`.
   b. Browse to the `templates` folder of the installation CD, and select the appropriate template for your version of Management Framework:
      - For Management Framework 8.1, select `Genesys_Rules_Engine.apd`.
      - For Management Framework 8.0 and earlier, select `Genesys_Rules_Engine_Generic_Server.apd`.
   c. Click `OK` to save the template.
3. Configure the application:
   a. Right-click the `Applications` folder and select `New > Application`.
   b. Select the template that you imported in the previous step.
   c. On the `General` tab, enter a name for the application, such as `Rules_Engine`.
   d. On the `Tenants` tab, add the Tenants that will be available to the Rules Engine.
   e. On the `Server Info` tab, select the `Host` on which the application will be installed.
   f. Add a default listening port.
   g. Add an additional port. This port is the connector port on which the Rules Engine Servlet receives requests:

- The `ID` value is the name of the Rules Engine web application. The default name of this application is “`genesys-rules-engine`”.
- The `Listening port` is the connector port of the Servlet Container. For example, on Tomcat the default listening port is 8080.
- The `Connection Protocol` must be `http`.

h. On the `Start Info` tab, enter `x` for each field. These fields are not used, but you must enter some text there in order to save the configuration.

i. On the `Options` tab, configure options. Logging options can be configured, and there are several optional configuration options in the `settings` section:

- `verify-deploy-address` indicates whether to verify that the TCP address of the application that is deploying rules is that of an associated Genesys Rules Authoring Tool. The default value is `true`.
- `sequential-mode` specifies whether to run the GRE in sequential mode. In sequential mode, no additional data can be inserted or modified after the initial data set, simplifying the GRE operations. The default value is `false`.
- `max-number-rule-executions` specifies the maximum number of rules to be executed during a request. This option is used to detect unwanted recursions when `sequential-mode` is `false`. Once the maximum is reached, an error is reported. The default value is `10000`.
- `deployed-rules-directory` (which must be manually added if you want to assign it a value) specifies the directory in which to keep the working copy of deployed rule packages. When a package is deployed, a copy of the deployed package is placed here. When the GRE is restarted, all packages that are defined in this directory are loaded and made available for execution. Specifying a `deployed-rules-directory` is recommended. If a value is not assigned to the `deployed-rules-directory`, the rule packages are placed in the `WEB-INF\config` sub-directory within the `genesys-rules-engine` web application directory. At this location the deployed rule packages may be deleted when an updated `.war` file is deployed.
- `esp-worker-threads` (new in release 8.1.2), specifies the maximum number of worker threads available when using the ESP interface to execute rules. The default for this value is 5 threads.

j. Click `Save`.

End of procedure

Next Steps

- Procedure: Installing the Genesys Rules Engine

Procedure: Installing the Genesys Rules Engine

Purpose: To run the installation package for the GRE, after the application has been created in Configuration Manager.

Prerequisites

- Procedure: Creating the Genesys Rules Engine Application object in Configuration Manager, on page 29

Start of procedure

1. From the host on which the GRE is to be installed, locate and double-click `Setup.exe` in the `rulesengine` folder of the Genesys Rules System CD.
2. Click `Next` on the `Welcome` screen of the installation wizard.
3. Enter the connection parameters to connect to Configuration Server (`Host`, `Port`, `User name`, and `Password`).
4. On the `Client Side Port Configuration` screen, if you do not want to configure client-side port parameters, leave the checkbox empty and click `Next`. If you do want to configure these settings, select the checkbox to display to additional options: `Port` and `IP Address`. Enter values for these options and click `Next`.
5. Select the Rules Engine application that you created in Procedure: Creating the Genesys Rules Engine Application object in Configuration Manager, on page 29. Click `Next`.
6. Specify the destination directory for the installation, or accept the default location, and click `Next`.
7. Enter the host and port of the optional backup Configuration Server and click `Next`.
8. Enter the number of times that the Rules Engine application should attempt to reconnect to Configuration Server (`Attempts`) before switching to the backup Configuration Server, and the amount of time (`Delay`) between attempts. Click `Next`.

   Note: After the specified number of attempts to connect to the primary Configuration Server all fail, then connection to the backup Configuration Server is attempted. If these attempts to the backup Configuration Server fail, then once again connection to the Primary Configuration Server is attempted. If no backup Configuration Server is configured, there is no limit on the number of connection attempts.

9. Click `Install`.
10. Click `Finish`.

End of procedure

Next Steps

- Deploy the `genesys-rules-engine.war` file to your application server. See Procedure: Deploying the .war files, on page 44.

Installing the Genesys Rules Authoring Tool

Genesys recommends that you configure the GRAT by using Genesys Administrator. You can configure the GRAT by using Configuration Manager if you are using an older version of Configuration Server, prior to 8.0.200, where Roles are not supported. If you use Genesys Administrator, you can deploy the installation package from within Genesys Administrator. If you use Configuration Manager, you will have to create the applications first, and then run the setup program manually.

Note: When operating the GRAT in a non-English environment, you will need to configure the `URIEncoding` option to properly operate and integrate with the Genesys Framework environment. By default, Tomcat uses ISO-8859-1 character encoding when decoding URLs received from a browser. If you wish to use characters not included in this character set, you will need to set the URIEncoding option to UTF-8 in the server.xml file on the Connector that is used for the Genesys Rules Authoring Tool.

For example:

```
<Connector connectionTimeout="20000" port="8080"
protocol="HTTP/1.1" redirectPort="8443" URIEncoding="UTF-8"
useBodyEncodingForURI="true"/>
```

Procedure:
Deploying the Genesys Rules Authoring Tool in Genesys Administrator

Purpose: To configure the GRAT applications and deploy the GRAT installation package using Genesys Administrator.

Genesys Administrator needs to be at release 8.1.1 or later to install the GRAT on 8.1.1 or later Configuration Servers.

Start of procedure

1. Import the installation package into Genesys Administrator:
2. On the `Deployment` tab of GA select the `Import` button.
   a. Select the `Installation CD-ROM` radio button.
   b. Click `Next`.
   c. Browse to the `MediaInfo.xml` file on the CD or the CD image location on the network (the path must be in UNC format).
   d. Click `Next`.
   e. Select GRAT for your operating system as well as the appropriate type in the list in order to import the installation package.
      - For Management Framework 8.1.1, the type is `Business Rules Application Server`.
      - For Management Framework 8.1 and earlier, the type is `Genesys Generic Server`.
   f. Select `Next` to start the import.
   g. Click `Finish` when the import is complete.
3. Install the GRAT IP. Select the `Deployment` tab in Genesys Administrator. The list of installation packages will now show the Genesys Rules Authoring Tool. Right-click and select `Install Package` for the IP for your operating system and type (see Step 2). Click `Next` to start the installation wizard. The following parameters must be defined/selected:
   a. `Application Name` for the Genesys Authoring Tool *server* application
   b. `Target Host`—The host to which the `.war` file will be copied during the installation procedure.
   c. `Working Directory`—The directory in which the `.war` file will be created.
   d. `Client Side IP Address` (optional).
   e. `Client Side Port` (optional).
   f. Backup Configuration Server hostname.
   g. Backup Configuration Server port.
   h. Connection delay time in seconds.
   i. `Reconnect Attempts`.

Note: After the specified number of attempts to connect to the primary Configuration Server all fail, then connection to the backup Configuration Server is attempted. If these attempts to the backup Configuration Server fail, then once again connection to the Primary Configuration Server is attempted. If no backup Configuration Server is configured, there is no limit on the number of connection attempts.

j. Client application name—The name of the GRAT client application, which will be configured in Step 5.

> Note: Items **a** through **i** will be written to the `bootstrapconfing.xml` file in the `.War` file. Any subsequent updates to the parameters will have to be made in that file.

k. `Database Engine` (such as MSSQL or Oracle)

l. `DBMS User`—The user must have write permissions on the database that you created in Procedure: Creating the Rules Repository database, on page 25.

m. `DBMS Password`

> Note: The values provided for the following properties (Connector Class and Database URL) are **examples only**. Consult the database vendor's jdbc documentation for detailed information specific to your database type. See for example values for the supported database types.

n. `Connector Class`—The connector class will differ depending on the database type. See Table 1 on page 44 for examples.

o. `Database URL`—The database URL will depend on the database type. See Table 1 on page 44 for examples.

On the next screen, enter the `Connection ID` and `Connection Port` for the Genesys Rules Authoring Server.

Specify the connections for the Rules Authoring Server on the next screen (select the GRE application). You can also add this connection later under the Configuration for the application.

Verify the previously-defined installation parameters on the `Deployment Summary` screen.

4. Configure the GRAT server application:

a. On the `Tenants` tab, add all tenants that should be visible in the GRAT interface.

a. In the `Server Info` section, configure a default listening port.

b. On the `Connections` tab, add a connection to the Rules Engine application.

c. On the `Options` tab, configure log options. In addition to the standard logging options that you can configure, you can configure an option named `fileEncoding` in the logging section.

`fileEncoding` specifies the encoding to be used when creating the log file. For example, `UTF-8`. This value is optional. If you do not specify this option, the server's locale information will determine the log file encoding.

This option is available for both the GRE and the Genesys Rules Authoring Tool. Also, the `log4j.properties` file that is included in both components supports a similar option, `log4j.appender.runtime.Encoding`. The `log4j.properties` file is used for initial log configuration prior to the reading of the log configuration from the Configuration Server database.

d. In the `settings` section, the following options can be configured:

— `session-timeout`—Specifies the amount of time (in minutes) that a client (browser) session can have no communication with the Rules Authoring Server before timing out. If no value is specified, the timeout (if any) defined by the application server applies. If the value is less than or equal to `0`, the session will not timeout. The default value is `30`.

— `session-timeout-alert-interval`—The amount of time (in minutes) prior to an expected timeout for a user to be warned of a pending timeout. If no value is specified, or if the value is less than or equal to `0`, the default warning period of `1` minute will be used. For example, if you set the value of this option to `3`, the user will be warned 3 minutes prior to an expected timeout. This warning dialog box will prompt the user to extend the session. If the session is not extended, the user will be logged out and the login dialog box will be displayed. Any unsaved changes that the user made during their session will be lost.

— `verify-deploy-address`—Indicates whether to verify that the TCP address of the application that is requesting the deployment of a package is that of an associated GRE. The default value is `true`.

— `encoding`—Activates Unicode support for the conversion of data between the local character set that is used by Configuration Manager and the UTF-8 encoding that is used by the Rules Authoring Server. By default, code page conversion is disabled. To activate this functionality, set this option to the name of a converter that can translate the local character set to UTF format. The converter that is suitable for a particular deployment can be found by using the ICU Converter Explorer. There is no default value for this option. For valid values, see the ICU Home > Converter Explorer pages (`http://demo.icu-project.org/icu-bin/convexp`). Changes take effect after the Genesys Rules Authoring Server restarts.

— `group-by-level`—The default value is `true`. When this option is `true`, rules are grouped by business level:
All global rules belong to agenda group “level0”.
Department rules belong to agenda group “level1”.
Process rules belong to agenda group “level2”.
When a rule package is executed, “level0” rules are executed first, facts are marked as updated, and “level1” rules are executed. This process is repeated for each level. Note that when `group-by-level` is `true`, the GRE option `sequential-mode` must be set to `false`.

— `strict-mode`—Specifies whether the GRAT enables “strict mode” in the DROOLS rule compiler. Enabling this option causes the compiler to catch common mistakes when the rule author attempts to validate or save a rule. The default value is `true`.

— `max-connections`—Specifies the maximum number of different users that can be connected to the server. Multiple connections from the same user id are only counted once. The default value is `99`.

e. Give the application Read, Create, and Change permissions on the `Scripts` folder for each Tenant that you add. (One approach is to create a user called `GRAT_Application_Proxy` and add that user to the `SYSTEM` access group. Then, on the `Security` tab of the application, in the `Log On As` section, select `This account` and add the `GRAT_Application_Proxy` user.) The `Security` tab is available only in Genesys Administrator 8.1.0 and later. Therefore, if you are **not** using Genesys Administrator 8.1.0 or higher, you must perform this step through Genesys Configuration Manager.

5. Create the GRAT *client* application by first importing the `Genesys_Rules_Authoring_Generic_Client_810.apd` to create the application template. From the application template, create the GRAT client application. The name of this application was specified during the installation of the IP. You just need to create the application and save it. You are not required to fill in any of the configuration properties.

End of procedure

Next Steps

- Optionally create an application cluster as a deployment target. See “Creating an Application Cluster as a Rules Package Deployment Target” on page 37.
- Deploy the `genesys-rules-authoring.war` file to your application server. See Procedure: Deploying the .war files, on page 44.
- Create the business structure (hierarchy) for each tenant. See Chapter 5, “Business Structure,” on page 85 for information about how to configure the business structure.

Creating an Application Cluster as a Rules Package Deployment Target

A Configuration Server or Genesys Administrator application of type `Application Cluster` can be used to define a group of Genesys Rules Engine (GRE) or Genesys Web Engagement engines. Engines in the group must be all of the same type—either all GRE engines or all Genesys Web Engagement engines.

When deploying a package in GRAT, the deployment target list may also contain cluster application names. When deployed to a cluster, the package is deployed to every engine in the cluster.

If deployment to any of the engines fail, details of the failure(s) are shown to the GRAT user and logged in the GRAT log. A deployed package is placed in service only after the deployment to all engines in the cluster is successful.

Procedure:
Creating an application cluster in Configuration Manager

Purpose: To create an application cluster in Configuration Manager to which rules packages can be deployed.

Start of procedure

1. Create an application template of type `Application Cluster`, if one does not already exist in your environment.
2. Create a Configuration Server application of type `Application Cluster`.
3. Add as connections to this cluster application the engine applications you wish to treat as a cluster. For each connection be sure to select the Port ID for the Rules Engine Web Application (either GRE or Genesys Web Engagement).
4. Add the cluster application as a connection to the GRAT application.
5. Save the changes.

End of procedure

This cluster application will now appear in the `Location` drop-down list in the `Deploy` window of GRAT and rules authors can select it as a deployment target.

Procedure: Creating an application cluster In Genesys Administrator

Purpose: To create an application cluster in Genesys Administrator to which rules packages can be deployed.

Start of procedure

1. Create an application template of type `Application Cluster`, if one does not already exist in your environment.
2. Create a Genesys Administrator application of type `Application Cluster`. Go to `Provisioning > Environment > Applications.`
3. If required, navigate to the folder in which you want to store the new `Application` object.
4. Open the `Tasks` panel, if necessary, and click `Create Application` in the `Create` section.
5. Follow the steps in the `Create New Application` wizard.
6. Add as connections to this cluster application the engine applications you wish to treat as a cluster. For each connection be sure to select the Port ID for the Rules Engine Web Application (either GRE or Genesys Web Engagement).
7. Add the cluster application as a connection to the GRAT application.
8. Save the changes.

End of procedure

Procedure: Creating the Genesys Rules Authoring Tool Application objects in Configuration Manager

Purpose: To create the `Application` objects in Configuration Manager that will link the GRAT with Configuration Server. The GRAT requires two applications in Configuration Server: a server application and a client application.

Start of procedure

1. In Configuration Manager, navigate to the `Application Templates` folder.
2. Import the application template that is to be used for the *server* application:
   a. Right-click the `Application Templates` folder, and select `Import Application Template`.
   b. Browse to the `templates` folder of the installation CD, and select the appropriate template for your version of Management Framework.
      - For Management Framework 8.1.1, select `Genesys_Rules_Authoring_Server_811.apd`.
      - For Management Framework 8.1 and earlier, select `Genesys_Rules_Authoring_Generic_Server_811.apd`.

   c. Click `OK` to save the template.
3. Import the template that is to be used for the *client* application:
   a. Right-click the `Application Templates` folder and select `Import Application Template`.
   b. Browse to the `templates` folder of the installation CD, and select `Genesys_Rules_Authoring_Generic_Client_810.apd`.
   c. Click `OK` to save the template.
4. Configure the server application:
   a. Right-click the `Applications` folder and select `New > Application`.
   b. Select the `Genesys_Rules_Authoring_Generic_Server` template.
   c. On the `General` tab, enter a name for the application, such as `Rules_Authoring_Server`.
   d. On the `Tenants` tab, add the Tenants that will be visible in the GRAT interface.
   e. On the `Server Info` tab, select the `Host` on which the application will be installed, and configure a default listening port.
   f. On the `Start Info` tab, enter `x` for each field. This is required in order to save the configuration.
   g. On the `Connections` tab, add a connection to the Rules Engine application (multiple Rules Engine applications can be added).

Note: The `Port ID` selected for a Rules Engine connection should be the name of the Rules Engine Web application.

Optionally, a connection to an application cluster of Rule Engines may be added.

   h. On the `Options` tab, configure log options.
   i. In the `settings` section, the following options can be configured:

- `session-timeout`—Specifies the amount of time (in minutes) a client session can have no communication with the Rules Authoring Server before timing out. If no value is specified, the timeout (if any) defined by the application server applies. If the value is less than or equal to `0`, the session will not timeout. The default value is `30`.
- `session-timeout-alert-interval`—The amount of time (in minutes) prior to an expected timeout for a user to be warned of a pending timeout. If no value is specified, or if the value is less than or equal to `0`, the default warning period of `1` minute will be used. For example, if you set the value of this option to `3`, the user will be warned 3 minutes prior to an expected timeout. This warning dialog will prompt the user to extend the session. If the session is not extended, the user will be logged out and the login dialog will be displayed. Any unsaved changes that the user made during their session will be lost.
- `verify-deploy-address`—Indicates whether to verify that the TCP address of the application that is requesting the deployment of a package is that of an associated GRE. The default value is `true`.
- `group-by-level`—The default value is `true`. When this option is `true`, rules are grouped by business level:
  All global rules belong to agenda group “level0”.
  Department rules belong to agenda group “level1”.
  Process rules belong to agenda group “level2”.
  When a rule package is executed, “level0” rules are executed first, facts are marked as updated, and “level1” rules are executed. This process is repeated for each level. Note that when `group-by-level` is `true`, the GRE option `sequential-mode` must be set to `false`.
- `strict-mode`—Specifies whether the GRAT enables “strict mode” in the DROOLS rule compiler. Enabling this option causes the compiler to catch common mistakes when the rule author attempts to validate or save a rule. The default value is `true`.
- `max-connections`—Specifies the maximum number of different users that can be connected to the server. Multiple connections from the same user id are only counted once. The default value is `99`.

j. Give the application Read, Create, and Change permissions on the `Scripts` folder for each Tenant that you add. (One approach is to create a user called `GRAT_Application_Proxy` and add that user to the `SYSTEM` access group. Then, on the `Security` tab of the application, in the `Log On As` section, select `This account` and add the `GRAT_Application_Proxy` user.)

k. Click `Save`.

5. Configure the client application:

a. Right-click the `Applications` folder and select `New > Application`.

b. Select the `Genesys_Rules_Authoring_Generic_Client` template.
c. On the `General` tab, enter a name for the application, such as `Rules_Authoring_Client`.
d. Click `Save`.

End of procedure

Next Steps

- Procedure: Installing the Genesys Rules Authoring Tool

Procedure: Installing the Genesys Rules Authoring Tool

Purpose: To run the the installation package for the GRAT, after the applications are configured in Configuration Manager.

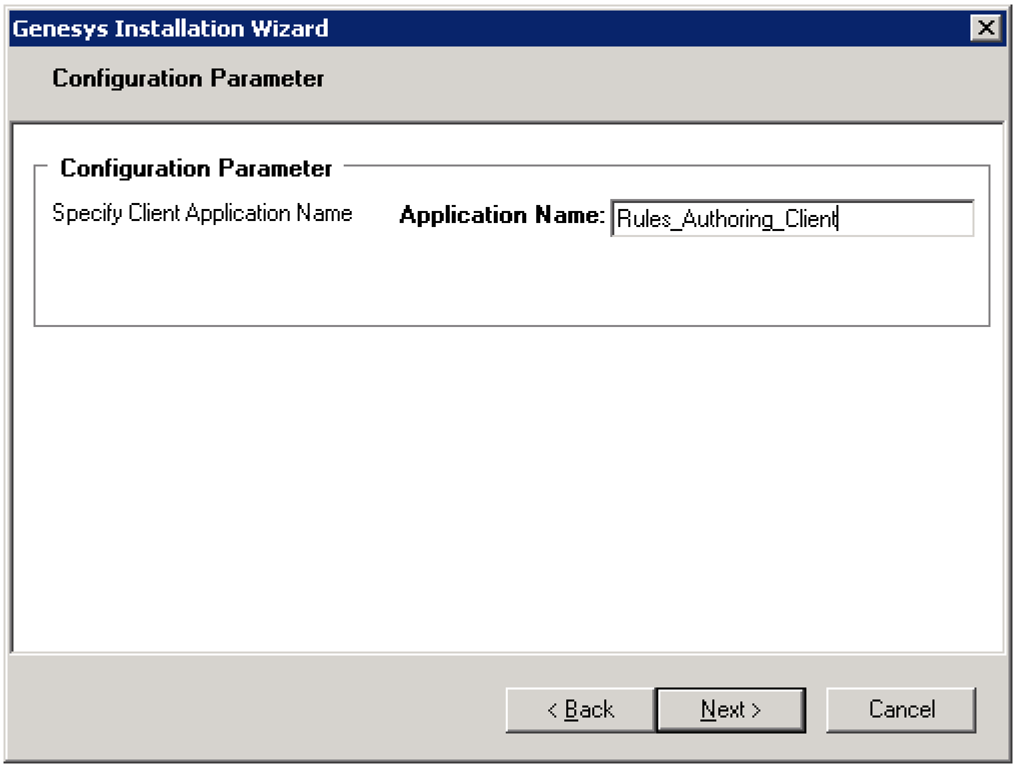
Prerequisites

- Procedure: Creating the Rules Repository database, on page 25
- Procedure: Creating the Genesys Rules Authoring Tool Application objects in Configuration Manager, on page 38

Start of procedure

1. From the host on which the GRAT is to be installed, locate and double-click `Setup.exe` in the `rulesauthoring` folder of the Genesys Rules System CD.
2. Click `Next` on the `Welcome` screen of the installation wizard.
3. Enter the connection parameters to connect to Configuration Server (`Host`, `Port`, `User name`, and `Password`).
4. On the `Client Side Port Configuration` screen, if you do not want to configure client-side port parameters, leave the checkbox empty and click `Next`. If you do want to configure these settings, select the checkbox to display to additional options: `Port` and `IP Address`. Enter values for these options and click `Next`.
5. Select the GRAT application that you created in Procedure: Creating the Genesys Rules Authoring Tool Application objects in Configuration Manager, on page 38. Click `Next`.
6. Specify the destination directory for the installation, or accept the default location, and click `Next`.
7. Enter the host and port of the optional backup Configuration Server and click `Next`.

8. Enter the number of times that the GRAT Server application should attempt to reconnect to Configuration Server (`Attempts`) and the amount of time (`Delay`) between attempts. Click `Next`.
9. On the screen that is shown in Figure 3, enter the name of the Rules Authoring *client* application that you created in Procedure: Creating the Genesys Rules Authoring Tool Application objects in Configuration Manager, on page 38 and click `Next`.


Genesys Installation Wizard

Configuration Parameter

Configuration Parameter

Specify Client Application Name

Application Name: Rules_Authoring_Client

Figure 3: Specify the Rules Authoring Client Application Name

10. Select the database engine that you used to create the Rules Repository database in Procedure: Creating the Rules Repository database, on page 25, and click `Next`.
11. Enter the following connection details and then click `Next`:

Note: The values provided for the following properties (Connector Class and Database URL) are **examples only**. Consult the database vendor's jdbc documentation for detailed information specific to your database type. See Table 1 on page 44 for example values for the supported database types.

- Connector class—Connector class will differ depending on the database type. See Table 1 on page 44 for examples.

- Database URL—The database URL will depend on the database type. See Table 1 on page 44 for examples.

Note: Consult the database vendor's jdbc documentation for detailed information specific to your database type.

- User Name—The user name that is used to connect to the database. The user must have write permissions on the database created in Procedure: Creating the Rules Repository database, on page 25.
- Password—The password that is used to connect to the database

12. Click `Install`.

13. Click `Finish`.

End of procedure

Next Steps

- Before using the GRAT, you will need to set up users and roles. See "Role Task Permissions" on page 91 and "Configuring a User" on page 92 for more information.

Example Values for Database Connection Parameters

During the installation of the Genesys Rules Authoring Tool, you will be prompted to enter various connection parameters for the database you are using as the Rules Repository (created in Procedure: Creating the Rules Repository database, on page 25). Table 1 provides some example values for the three supported database types (MSSQL, Oracle, and DB2). Note that these values are **examples only**, and you must consult your database vendor's documentation for specific information. The last column in the table lists the JDBC drivers that you must to copy to the `lib` directory of your application server.

Table 1: Example database connection parameter values

Database Type	Example Connector Class	Example Database URL	JDBC Driver to be Copied
MSSQL	`com.microsoft.sqlserver.jdbc.SQLServerDriver`	`jdbc:sqlserver://chair2.us.int.genesyslab.com:1433;databaseName=GRS_db27` where `GRS_db27` is the database name	`sqljdbc.jar`

Oracle	`oracle.jdbc.driver.OracleDriver`	`jdbc:oracle:thin:@//gplus.us.int.host.com:1521/orcl`
		where `gplus.us.int.host.com` is the Oracle database host, and `orcl` is the database instance (SID)
			`ojdbc14.jar`
			`ojdbc14_g.jar`
			`ojdbc14dms_g.jar`
			`ojdbc6_g.jar`

DB2	`com.ibm.db2.jcc.DB2Driver`	`jdbc:db2://135.225.59.38:50000/GRSDB2_4` where `GRSDB2_4` is the database name	`db2jcc.jar` `db2jcc_license_cu.jar`

Deploying the .war Files

The GRAT and GRE `.war` files must be copied or deployed to your web container. When the .war files have been deployed, you will be able to launch the GRE and GRAT.

Procedure: Deploying the .war files

Purpose: To deploy the `genesys-rules-authoring.war` and `genesys-rules-engine.war` files.

Start of procedure

1. The `.war` files can be found in the destination folder that you specified when you installed the IPs.
2. If you are using Tomcat, copy the files and paste them into the Tomcat `webapps` folder.
3. If you are using WebSphere, deploy the `.war` files by using WebSphere Administrative Console.

4. Refer to the documentation for your web application server (Tomcat or WebSphere) for specific deployment instructions.

End of procedure

Installing the Genesys Rules Development Tool

The following procedure outlines how to install the Genesys Rules Development Tool.

Procedure: Installing the Genesys Rules Development Tool

Purpose: To install the Genesys Rules Development Tool (GRDT). The GRDT is an Eclipse plug-in that can be installed either into a stand-alone Eclipse instance or into Genesys Composer.

Prerequisites

- Genesys Composer or Eclipse must be installed. If you want to install the GRDT Eclipse plug-in into a stand-alone Eclipse IDE platform (not Composer), and do not already have Eclipse, you can download it from the following location:
  `http://www.eclipse.org/downloads/`
- Ensure your version of Eclipse is up-to-date. In Eclipse, select Help > Check for Updates.
- Before installing GRDT in Composer, enable the Galileo update site in Composer. This is found in `Windows/Preferences`, under the `Install/Updates/Available Software Sites` node. Find or add the entry for `http://download.eclipse.org/releases/galileo` and enable it.

Start of procedure

1. Locate the GRDT installation zip file on the Genesys Rules CD (in the `rulesdevelopment` folder) and save it locally.
2. Start up Eclipse or Composer.
3. In Eclipse or Composer, select `Help > Install New Software`.
4. Browse to the GRDT installation zip file and drag it onto the `Available Software` dialog box. This action adds the location as a “site”. When it has been added, it will appear in the drop-down list. It does not have to be added each time. If you get an error when you drag and drop the file, open the drop-down list to see if the site already exists, and select it from the list.
5. Check `Genesys Rules System` in the list of software and click `Next`.

6. Check `Template Development Tool`, accept the license terms, and click `Finish`.

> Warning! If you do not check the checkbox, and click `Next`, you will get an error.

7. Change the perspective so that you can view the GRDT interface. Navigate to `Window > Open Perspective > Other > Template Development`. You will be prompted to restart Eclipse or Composer in order for the new Template Development perspective to be enabled.

8. Click on `Server Preferences`, and edit the following information (you can also access these preferences by directly navigating to `Window/Preferences/Genesys Rules System/Repository Server)`:
   - `Name`—The name of the server on which the web container is running that is hosting the GRAT server.
   - `Port`—The listening port for your web container (such as 8080).
   - `servlet-path`: `genesys-rules-authoring`.
   - In the `Authentication` section, enter the user name and password for a user who is defined in Configuration Server. The user entered here—or an access group to which the user belongs—must have, at a minimum, Read and Execute permissions to the Genesys Rules Authoring client application (in Configuration Server) in order to access the Rules Repository through the GRDT. That is, the user whose name and password is provided here must have Read and Execute permissions—or must belong to an access group that has those permissions—to the GRAT client `Application` object. Refer to Chapter 6, "Role-Based Access Control," on page 89 for more information about roles.

> Note: Even after configuring the connection parameters to the GRS repository server as described in Step 8, you will not see a connection to the GRS repository in the GRS Server Explorer view of the Rules Development Tool until you start your application server, so that the GRAT web application is deployed and running.

9. While still in the `Preferences` dialog, select `Genesys Rules System/Configuration Server`, and edit the following information:
   - `Name`—The name of the server on which the Genesys Configuration Server is running.
   - `Port`—The listening port for the Genesys Configuration Server (normally `2020`).

- `Application`—The name, as configured in Genesys Configuration Server, of the GRAT client application that you created, as described earlier in Procedure: Installing the Genesys Rules Authoring Tool, on page 41.
- `User name`—The name of a Configuration Server user. Note that this user's access control determines which objects can be accessed from the Genesys Rules Development Tool, such as Business Attributes and Transaction objects.
- `Password`—The password of the Configuration Server user.

10. If you have a sample to import, navigate to `File > Import > General > Existing Projects into Workspace`, and click `Next`.

11. Browse to the sample, check in the list of projects, and click `Finish`.

End of procedure

Note: If you are working with Genesys Technical Support, you will need to supply the exact version of the GRDT you are using. Refer to "Genesys Rules Development Tool Version Number" on page 57 for information about how to find the version number.

Next Steps

- Before using the GRDT, you will need to set up users and various script parameters. See "Template Script Objects" on page 92 and "Configuring a User" on page 92 for more information.

Testing the Installation

Test the installation by logging in a user to the GRAT.

Procedure: Logging into the Genesys Rules Authoring Tool

Purpose: To verify your Genesys Rules System installation by logging in a user to the GRAT. See also Procedure: Configuring a user for the Genesys Rules Authoring Tool, on page 92, to verify that the user has the correct permissions.

Start of procedure

1. Start your web application container (Tomcat or WebSphere) on the server(s) that are hosting the GRAT and the GRE.
2. Open a web browser and enter the URL for the GRAT—for example:

   `http://<host>:<port>/genesys-rules-authoring/login.jsp`

   Where `<host>` is the name of the server on which the web container is running that is hosting the GRAT server, and `<port>` is the listening port for your web container (such as 8080).

   Note: These are the same host and port that you entered in Step 8 of Procedure: Installing the Genesys Rules Development Tool, on page 45. The default name is `genesys-rules-authoring` is the default, but you can override this name during deployment.

3. On the login screen, enter the credentials for a user to login to the GRAT. Users who log into the GRAT *must* have access to one or more tenants in a multi-tenant environment, with, at minimum, Read permission to the tenant(s). In addition, users or access groups must have, at a minimum, Read and Execute permissions to this GRAT client `Application` object in Configuration Server, in order to log in to the GRAT.

End of procedure

Installing Genesys Rules System on UNIX-Based Operating Systems

For the supported UNIX versions, please consult the *Genesys Supported Operating Environment Reference Manual*.

To install the GRE or the GRAT on UNIX systems, first create the `Application` objects in Configuration Manager or Genesys Administrator (for reference, see Procedure: Creating the Genesys Rules Engine Application object in Configuration Manager, on page 29 and Procedure: Creating the Genesys Rules Authoring Tool Application objects in Configuration Manager, on page 38). Next, locate and run the `install.sh` scripts for each component (found in their respective directories on the CD).

The following is an example of the command terminal from an installation of the GRE on a Linux host. This example includes the script's prompts, as well as the user's input (in **bold**).

```
bash-3.2$ ./install.sh
**************************************************
* Welcome to the Genesys 8.1 Installation Script *
**************************************************

Installing Genesys Rules Engine, version 8.1.xxx.xx

Please enter the hostname or press enter for "rh5x64-vm1" => <ENTER>
was selected

Unable to find configuration information.
Either you have not used configuration wizards and the
GCTISetup.ini file was not created or the file is corrupted.

Please enter the following information about your Configuration
Server:

Configuration Server Hostname =>host1
Network port =>2020
User name =>default
Password => the password was entered

Client Side Port Configuration
Select the option below to use a Client Side Port. If you select
this option, the application can use Client Side Port number for
initial connection to Configuration Server.
Do you want to use Client Side Port option (y/n)?y
Client Side Port port =>8888
Client Side IP Address (optional), the following values can be used
135.xxx.xx.xxx
=><ENTER> was selected
Backup Configuration Server Hostname =>host2
Backup Network port =>2020

Please choose which application to install:
1 : GRE8100025_rh5x64-vm1
=>1

Press ENTER to confirm "0" as
the Number of attempts to reconnect to primary Configuration Server
or enter a new one =>6

Press ENTER to confirm "0" as
the Delay in seconds between reconnect attempts or enter a new one
=>3

Please enter full path of the destination directory for installation
=>/home/GRS/GRE/8100025/linux

The target install directory /home/GRS/GRE/8.1.xxx.xx/linux
has files in it. Please select an action to perform:
```

```
1. Back up all files in the directory
2. Overwrite only the files contained in this package
3. Wipe the directory clean
1, 2, or 3 =>2

Extracting tarfile: data.tar.gz to directory:
/home/user/GRS/GRE/8.1.xxx.xx/linux
..

Installation of Genesys Rules Engine, version 8.1.xxx.xx has
completed successfully.
```

The following is an example of the command terminal from an installation of the GRAT on a Linux host. This example includes the script's prompts, as well as the user's input (in **bold**).

```
bash-3.2$ ./install.sh
**************************************************
* Welcome to the Genesys 8.1 Installation Script *
**************************************************

Installing Genesys Rules Authoring Tool, version 8.1.xxx.xx

Please enter the hostname or press enter for "rh5x64-vm1" =><ENTER>
was selected

Unable to find configuration information.
Either you have not used configuration wizards and the
GCTISetup.ini file was not created or the file is corrupted.

Please enter the following information about your Configuration
Server:

Configuration Server Hostname =>host1
Network port =>2020
User name =>default
Password => the password was entered

Client Side Port Configuration
Select the option below to use a Client Side Port. If you select
this option,
the application can use Client Side Port number for initial
connection to
Configuration Server.

Do you want to use Client Side Port option (y/n)?y
Client Side Port port =>9999
Client Side IP Address (optional), the following values can be used
135.xxx.xx.xxx
 =><ENTER> was selected
Backup Configuration Server Hostname =>host2
Backup Network port =>2020

Please choose which application to install:
1 : GRAT8100037_rh5x64-vm1
2 : GRE8100025_rh5x64-vm1
=>1

Press ENTER to confirm "0" as
the Number of attempts to reconnect to primary Configuration Server
or enter a new one =>3

Press ENTER to confirm "0" as
the Delay in seconds between reconnect attempts or enter a new one
=>6

Client connection application =>GRSRuleClient

Please specify the database type for:
1) Oracle
2) DB2
3) MSSQL
=>1

JDBC Connector Class =>oracle.jdbc.driver.OracleDriver
Database URL =>jdbc:oracle:thin:@//hostname:1521/orcl
Database user =>SA
Database user password => the database user password was entered


Please enter full path of the destination directory for installation
=>/home/GRS/GRAT/8.1.xxx.xx/linux

The target install directory /home/GRS/GRAT/8.1.xxx.xx/linux
has files in it. Please select an action to perform:
1. Back up all files in the directory
2. Overwrite only the files contained in this package
3. Wipe the directory clean
1, 2, or 3 =>2

Extracting tarfile: data.tar.gz to directory:
/home/user/GRS/GRAT/8.1.xxx.xx/linux
..

Installation of Genesys Rules Authoring Tool, version 8.1.xxx.xx has
completed successfully.
```

Troubleshooting

This section provides some information about troubleshooting your installation of Genesys Rules System.

Configuration Considerations

This section contains some considerations that you should keep in mind when you are configuring your Genesys Rules System environment.

Genesys Rules Authoring Tool (Server)

- This application must have a connection to at least one GRE application, Genesys Web Engagement Engine application, or application cluster.
- A default listening port must be specified in the configuration.
- On the `Security` tab, under `Log On As`, you must provide the username of a user who has `Read`, `Change`, and `Create` permissions to the `Scripts` folder.

Note: The `Security` tab is available only in Genesys Administrator 8.1.0 or later. Otherwise, you must perform this part of the configuration through Configuration Manager.

Genesys Rules Authoring Tool (Client)

- Users or access groups must have, at a minimum, Read and Execute permissions to this application, in order to log in to the Genesys Rules Authoring Tool.
- Users or access groups must have, at a minimum, Read and Execute permissions to this application, in order to access the Repository through the Rules Development Tool. That is, on the Repository Server preferences screen in the Genesys Rules Development Tool, the user whose name and password is provided must have Read and Execute permission—or must belong to an access group that has those permissions—to the GRAT client application object.

Genesys Rules Engine

- Tenants that may use this Rules Engine must be specified.
- A default listening port must be specified in the configuration.
- A second port must be specified in the configuration:
  - ID: `genesys-rules-engine` (the name of the Rules Engine web application; can be changed by the installer)

- Port: `[port being used by Tomcat or WebSphere]`
- Protocol: `http`

Access Groups

- No access groups are created out of the box for Genesys Rules System.
- Suggested access groups to create, at a minimum, are the following:
  - Rule Authors
  - Rule Developers

Roles

- Requires Configuration Server and Genesys Administrator 8.0.2 or later.
- No roles are created out of the box for Genesys Rules System.
- Suggested roles to create, at a minimum, are the following:
  - Rules Administrator (all privileges)
  - Rules Author (relevant privileges in the Rule Authoring and Business Calendar groups)
  - Rules Developer (all privileges in the Rule Templates group)
- Users may be assigned individually to these roles, and/or access groups to which the users belong may be assigned to these roles.
- Role changes take effect immediately.

Note: See Chapter 6, "Role-Based Access Control," on page 89 for more information about roles and role-based access control.

Users/Persons

- No users are created out of the box for Genesys Rules System.
- Genesys Rules System users can be agents or non-agents.
- Users who log in to the GRAT must have access to one or more tenants, in a multi-tenant environment, with at least Read permission to the tenant(s).
- The user who is specified in the GRDT preferences must have access to one or more tenants, in a multi-tenant environment, with at least Read permission to the tenant(s).
- In addition to the users for the GRAT and the user(s) for the Rules Development Tool, you must create one non-agent user (for example, `GRAT_Application_Proxy`) who has Read and Change permissions to the `Scripts` folder.

Business Structure

- No business structure is created out of the box for Genesys Rules System.
- If you are using the Genesys Rules System with intelligent Workload Distribution, the business structure is created in iWD Manager and is then synchronized with Configuration Server, after which it becomes available for use by the Genesys Rules System.
- A top-level folder must be created, of type `Business Unit` (called `Configuration Unit` in Configuration Manager) or `Site`, with the exact name of `Business Structure`.
- Within the `Business Structure` folder, at least one more Business Unit or Site must be created (it does not matter which one). This will represent the solution. Within that folder, additional levels of hierarchy may be created, as needed, using either Business Units or Sites. Those levels of hierarchy beneath the Solution level will represent the *business context*.
- Multiple solutions may be created by creating additional Business Units or Sites directly beneath the `Business Structure` folder.
- `Business Structure` gets created under Resources for single tenant Configuration Server or under a Tenant for a multi-tenant Configuration Server.
- Read permission to the `Business Structure` folder must be provided to the users and/or access groups that you want to use the Rules Authoring Tool. Normally, if the user or access group has permission to the `Tenant` object, this will be propagated automatically. If you do not want a user or access group to have permission to see all nodes of the business structure, you can control this by not giving that user or access group(s) Read permission to those folders.

Note: See Chapter 5, "Business Structure," on page 85 for more information.

Scripts

- A user (such as `GRAT_Application_Proxy`) on whose behalf the GRAT server will update the `Scripts` folder must have Read, Create, and Change permissions to this folder.
- Individual Rules Development Tool users, or one or more access group(s) to which they belong, must have Read permissions to the individual `Script` objects that represent the rule templates to which they should have access. Alternatively, you might decide to grant permission to the entire `Template Access Control` scripts folder to individual users or an access group such as Rule Developers, and allow that permission to propagate to all scripts that might be created in the future.

- Individual GRAT users, or one or more access group(s) to which they belong, must have read permissions to individual `Script` objects that represent the rule templates that rule authors should be able to add to a rule package when creating it.
- Users need Read access to parameter scripts. These scripts are maintained via Genesys Administrator Extension.

Configuration Diagrams

Figure 4 on page 56 and Figure 5 on page 57 provide visual representations of the configuration.

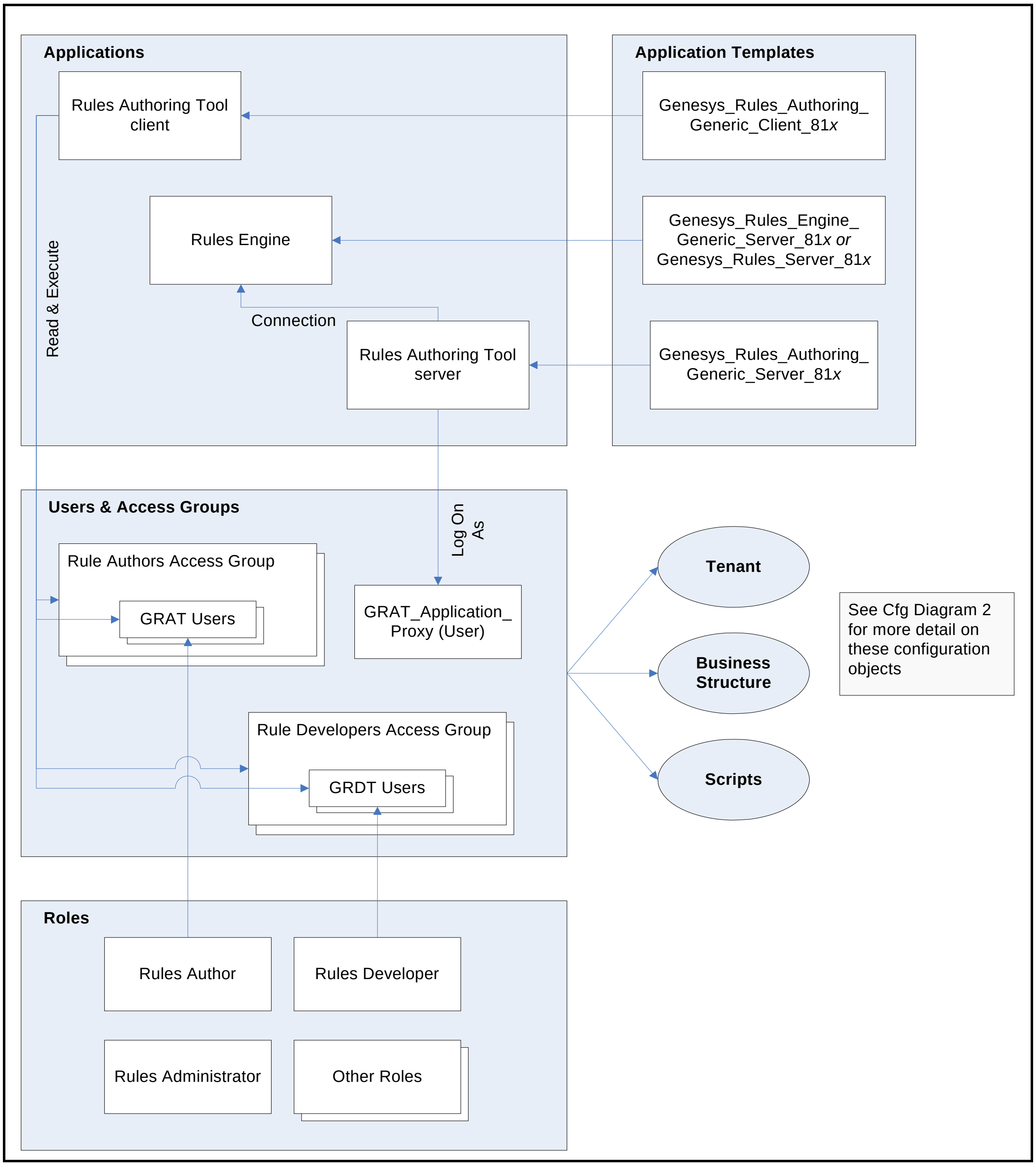

Applications
Rules Authoring Tool client
Rules Engine
Connection
Rules Authoring Tool server
Read & Execute
Application Templates
Genesys_Rules_Authoring_Generic_Client_81x
Genesys_Rules_Engine_Generic_Server_81x or Genesys_Rules_Server_81x
Genesys_Rules_Authoring_Generic_Server_81x
Log On As
Users & Access Groups
Rule Authors Access Group
GRAT Users
GRAT_Application_Proxy (User)
Rule Developers Access Group
GRDT Users
Tenant
Business Structure
Scripts
See Cfg Diagram 2 for more detail on these configuration objects
Roles
Rules Author
Rules Developer
Rules Administrator
Other Roles

Figure 4: Configuration Diagram 1

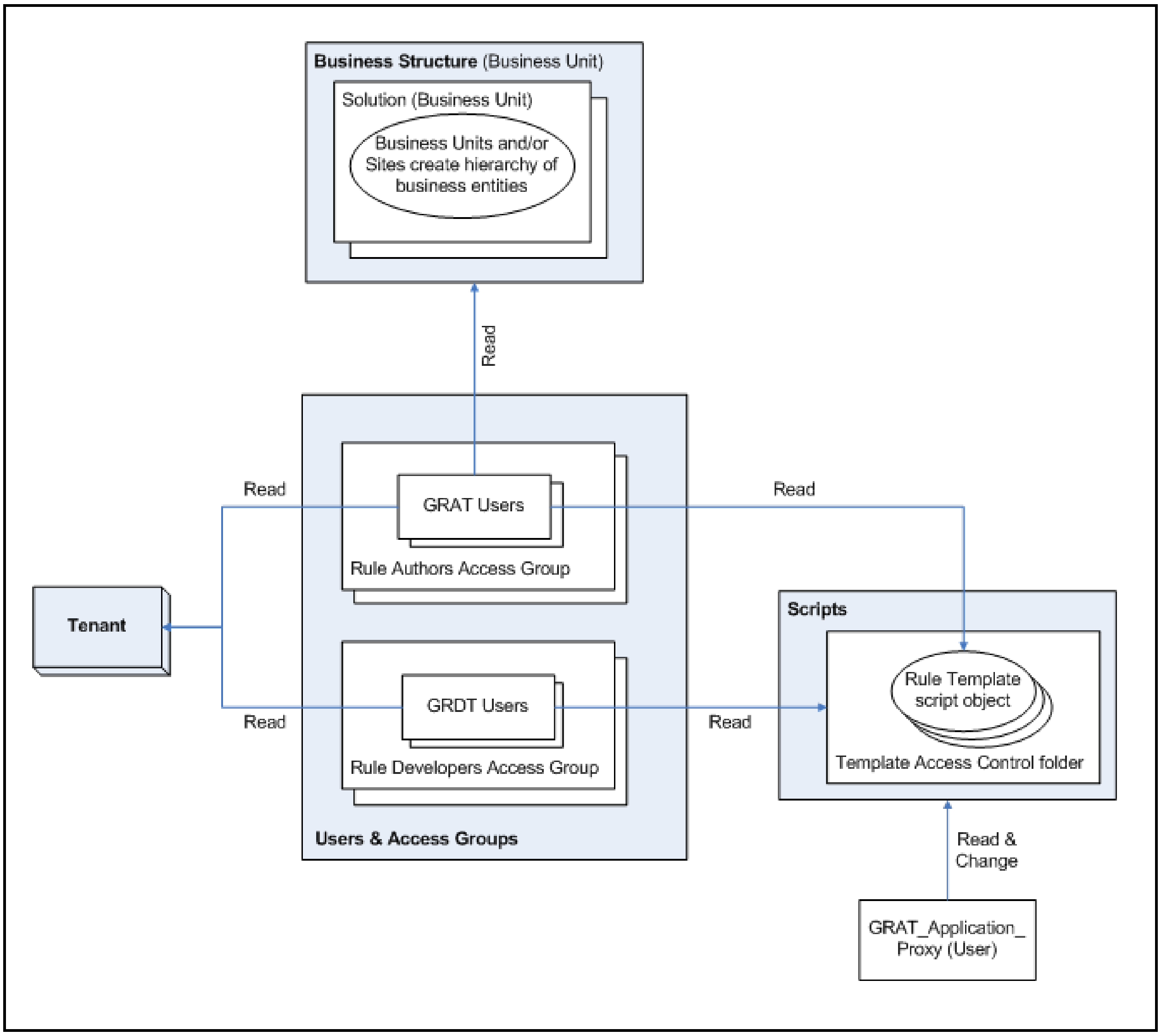

Business Structure (Business Unit)
Solution (Business Unit)
Business Units and/or Sites create hierarchy of business entities
Read
Read
GRAT Users
Rule Authors Access Group
Read
Tenant
Scripts
Rule Template script object
Template Access Control folder
GRDT Users
Read
Rule Developers Access Group
Read
Users & Access Groups
Read & Change
GRAT_Application_Proxy (User)

Figure 5: Configuration diagram 2

Genesys Rules Development Tool Version Number

If you are working with Genesys Technical Support, you will need to supply the exact version of the GRDT you are using. The following procedure outlines how to locate the version number.

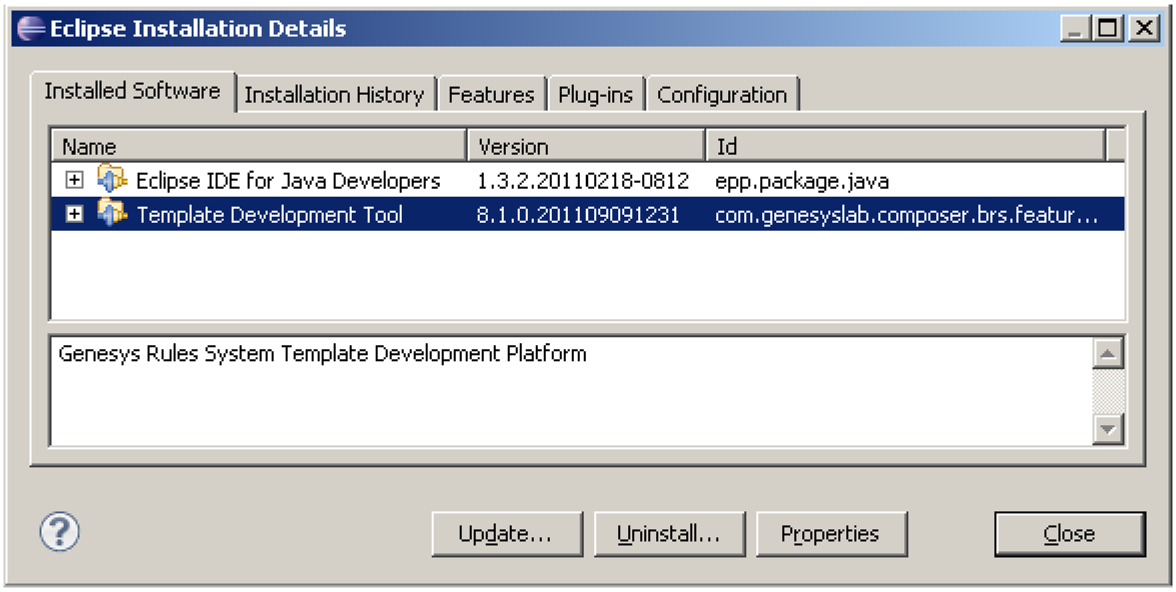
Procedure: Locating the Genesys Rules Development Tool version number

Purpose: Because the GRDT is an Eclipse plug-in, it has a specific version number format that is not easily located. This procedure guides you in identifying the version number, in case you need to provide it to Genesys Technical Support.

Start of procedure

1. In Composer, go to `Help > About Composer`. If you are using Eclipse, go to `Help > About Eclipse`.
2. Click on `Installation Details`.
3. On the `Installed Software` tab, you will see an entry for `Template Development Tool`. In the `Version` column, you will see the version number (in the format `8.1.0.xxxxxxxxxxxx`, as shown in Figure 6).

Note: You will not be able to select this version through the Web form when creating a Service Request, so you will need to select `Unspecified`. Include the full version number in your Service Request details.


Eclipse Installation Details

Installed Software | Installation History | Features | Plug-ins | Configuration

Name	Version	Id
Eclipse IDE for Java Developers	1.3.2.20110218-0812	epp.package.java
Template Development Tool	8.1.0.201109091231	com.genesyslab.composer.brs.featur...

Genesys Rules System Template Development Platform

Figure 6: GRDT Installation Details

End of procedure

The log4j.properties file

The `log4j.properties` file is used to configure initial logging for the Rules Engine and for the Genesys Rules Authoring Tool. Once the Rules Engine and GRAT are initialized, logging is done through the configured `Application` options. The `log4j.properties` file contains logging attributes that are used during the startup of the application, before the configured log settings are read by Configuration Server. In general, you should not have to modify this file and you can accept the default values. But should you need to change the defaults, perform the steps in the following procedure.

Procedure: Updating the Rules Engine or Rules Authoring Tool log4j.properties file

Purpose: To configure logging attributes in the `log4j.properties` file.

Start of procedure

1. Locate the `log4j.properties` file. This file can be found in the `.war` file, which is located in the installation directory. Extract the `.war` file by using WinZip or a similar tool for extraction. (For the Rules Engine and the Rules Authoring Tool, the .war files are named `genesys-rules-engine.war` and `genesys-rules-authoring.war`, respectively).
2. Open the file in a text editor, and update any logging parameters.
3. Save the file.
4. Add the modified `log4j.properties` file back into the original `.war` file by using WinZip (or a similar tool). Be very careful to preserve the “path” of that file during this step.

End of procedure

Chapter

3 Localization

This chapter provides information about how to install and deploy localization resources for the Genesys Rules Authoring Tool component of the Genesys Rules System. It contains the following sections:

Overview

The Genesys Rules Authoring Tool (GRAT) can be localized by installing one or more Genesys Rules Authoring Tool Language Packs (GRAT LP) on top of the base installation.

Every time a Language Pack is installed, the `.war` file that is in the installation directory is modified to insert the localized resources, such as the text strings that appear on the screen, and the online help.

You can install more than one language pack; for example, one for each language that you anticipate your users will use. Each user can select their preferred language in their browser's `Options` screen (see "Selecting a preferred language in Internet Explorer" on page 64 and "Selecting a preferred language in Firefox" on page 64).

As each user logs in, GRAT attempts to render the screens in the user's preferred language. If the language is not available, it will default to English.

Deploying Localization Resources

This section describes how to install and configure localization resources.

Installing Language Packs

Procedure: Installing a language pack on Windows

Purpose: To install a language pack onto a base GRAT installation on Windows.

Start of procedure

1. Locate the machine where the base GRAT product is installed.
2. Run setup on the language pack you want to install.
3. When prompted, choose the correct installation of the base GRAT product (if GRAT is installed in more than one location).
4. When you confirm the correct location of the base GRAT product, the installation program updates the `.war` file.
5. Repeat Steps 2 through 4 for each language pack you want to install.
6. When all required language packs are installed, re-deploy the `.war` file. See "Deploying the .war files" on page 44.

Notes: 1. If you update your base GRAT product with a newer version, such as a hot fix, you will need to re-install the language packs by using this procedure.
2. You can install a newer version of the GRAT Language Pack, by following this procedure. The newer resource files will overwrite the older ones in the target `.war` file.

End of procedure

Next Steps

- "Selecting a preferred language in Internet Explorer" on page 64.
- "Selecting a preferred language in Firefox" on page 64.

Procedure: Installing a language pack on UNIX

Purpose: To install a language pack onto a base GRAT installation on UNIX.

Start of procedure

1. Locate the machine where the base GRAT product is installed.
2. Locate the Language Packs folder.
3. Add the following execute flag to the `install.sh`:
   - `[root@host ip]# chmod +x install.sh`
4. Run the install script:
   - `[root@host ip]# ./install.sh`
5. Provide the full path of the destination directory for installation:
   - `/root/GRS/GRAT/`
6. Repeat Steps 2 through 5 for each language pack you want to install.
7. When all required language packs are installed, re-deploy the `.war` file. See "Deploying the .war files" on page 44.

Notes: 1. If you update your base GRAT product with a newer version, such as a hot fix, you will need to re-install the language packs by using this procedure.

2. You can install a newer version of the GRAT Language Pack, by following this procedure. The newer resource files will overwrite the older ones in the target `.war` file.

End of procedure

Next Steps

- "Selecting a preferred language in Internet Explorer" on page 64.
- "Selecting a preferred language in Firefox" on page 64.

Procedure:
Selecting a preferred language in Internet Explorer

Note: Browsers change over time and you may need to consult your browser's documentation for up-to-date information.

Start of procedure

1. Locate `Tools > Internet Options > Languages`.
2. Add the preferred language and move it to the top of the list.
3. Refresh the screen where GRAT is running.

End of procedure

Procedure:
Selecting a preferred language in Firefox

Note: Browsers change over time and you may need to consult your browser's documentation for up-to-date information.

Start of procedure

1. Locate `Tools > Options`.
2. Select the `Content` tab.
3. Add the preferred language and move it to the top of the list.
4. Refresh the screen where GRAT is running.

End of procedure

Uninstalling Language Packs

When you uninstall any GRAT Language Pack it is removed from the system. However, the localized resource files are not removed from the target `.war` file. To remove them, you must re-install the base GRAT product (see "Installing the Genesys Rules Authoring Tool" on page 41).

Genesys

Chapter

4 Rule Templates and Rules

This chapter provides information about business rules and rule templates. It contains the following sections:

Note: For specific information about how business rules and templates are used in the Genesys intelligent Workload Distribution (iWD) solution, refer to the section about iWD and the Genesys Rules System in the *iWD 8.1 Deployment Guide*.

Rule Life Cycle

Figure 7 shows the rule life cycle.

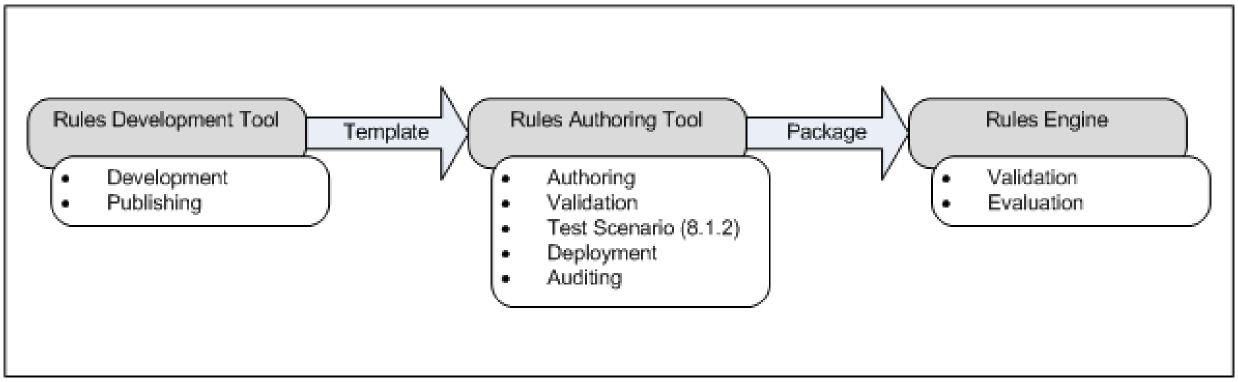


Figure 7: Rule Life Cycle

Rule Templates

Rule templates are developed in the Genesys Rules Development Tool. Each time a rule template is published, it will create a new version in the repository. The rule author will be able to select the latest version of the template when creating a rule package. Once a rule package is created, it will always use the same version of the rule template, even if newer versions are published.

The rule author can choose to upgrade to a newer version of the rule template at any time, but this will not happen automatically.

The rule *developer* should communicate to the rule *author* if a new version of the Rule Template is available and if they are advised to upgrade.

`Note:` When you are publishing newer versions of the rule template, be aware that certain changes could affect rules that already have been created using the earlier version of the template. Be careful not to make changes that could void existing rules, unless these changes are communicated to the rule author.

For example, if Rule Template version 1 contains a condition that is removed later in version 2, then if a rule were already built using that condition, it will no longer compile if the rule author upgrades to Rule Template version 2.

Refer to the *Genesys Rules Development Tool Help* for more information about rule templates and how to create them.

There are a number of components that can be created in a rule template:

- Actions
- Conditions
- Parameters
- Enumerations (Enums)
- Fact Models and, in release 8.1.2, Events
- Functions

Actions and Conditions

Actions and *conditions* define `WHEN/THEN` scenarios, such as `WHEN a customer is a Gold customer, THEN target the GoldAgentGroup`. The `WHEN` statement is the condition, and the `THEN` statement is the action. A rule may have zero or more conditions, and one or more actions. This example also includes parameters:

the status of the customer (`Gold`) and the name of the Agent Group (`GoldAgentGroup`).

Note: Whenever a condition contains a rule language mapping that begins with `eval(...)`, you must enclose the entire expression in parenthesis, as follows:

```
( eval(.... ) )
```

This will ensure it will compile properly when used with the `NOT` operator.

Enumerations

Enumerations are used to define lists of possible choices that will be displayed to the business rule author, when the author is creating rules that are based on the rule template.

Note: In some cases, the list of possible choices will be selected dynamically from Genesys Configuration Server objects or from external data sources. For WFM Activities and Multi-Site Activities, the list of possible choices is retrieved dynamically from the Genesys WFM Server.

Thus, enumerations are used during definition of a discrete list of choices that will not change dynamically.

Fact Models

A *fact model* structures basic knowledge about business operations from a business perspective. A fact model focuses on logical connections (called facts) between core concepts of the business. It indicates what you need to know about the business operations in order to support (or actually do) those operations.

A good fact model tells you how to structure your basic thinking (or knowledge) about the business process based on a standard vocabulary. By using standard, business-focused vocabulary, it ensures that the business rules themselves can be well-understood by key stakeholders such as business analysts.

For example, in your business you may have a Fact that represents a Customer, and another Fact that represents an Order.

The Customer could have fields such as `name`, `age`, `location`, `credit rating`, and `preferred language`.

The Order may have fields such as `order amount` and `order date`.

A rule could be constructed using these values such as:

```
When Customer is at least 21 years old and his order is > 100.00 then
invite customer to participate in survey.
```

Events

In release 8.1.2, in order to support Complex Event Processing, template developers need to be able to designate certain facts as *events*, and rules authors need to change the way that the DRL is generated when a *fact* is designated as an *event*.

So the fact model has been enhanced in release 8.1.2 to include events, and the fact model dialog now includes a `Create Event` button.

An event has the following fields:

- `Name`
- `Description`
- An optional list of `Properties`.
- User-defined expiration metadata for the event

In GRAT, the `@role` meta-data tag determines whether we are dealing with a fact or an event. The `@role` meta-data tag can accept two possible values:

- `fact`—Assigning the fact role declares the type is to be handled as a regular fact. Fact is the default role.
- `event`—Assigning the event role declares the type is to be handled as an event.

Functions

Functions are used to define elements other than Conditions and Actions. The Functions editor enables you to write specific Java functions for different purposes for use in rule templates. The specified functions may then be used in the rule language mappings (see “Rule Language Mapping” on page 72).

When the rule templates are created, the rule developer publishes them to the Rule Repository, making them available in the GRAT for business users to create rules.

Actions and conditions can contain *parameters*. Various types of parameters are supported. Refer to the *Genesys Rules Development Tool Help* for detailed information about creating parameters in the Genesys Rules Development Tool, including examples of parameters.

Certain dynamic parameter types that refer to external data sources require a Profile to be selected. The Profile is defined as a `Script` object of `Data Collection` type, and it provides connection information that enables the GRAT to retrieve this dynamic data from the external data source. The next sections describe how to configure Profiles for database, Web Service, and Workforce Management parameters.

Database Parameters

In Configuration Server, Database Scripts must have a section called `database`. Table 2 lists the properties that you can specify for database parameters.

Table 2: Database Parameter Properties

Property	Mandator/optional	Description
driver	Mandatory	The name of the jdbc driver to be used. For example, `com.mysql.jdbc.Driver`
url	Mandatory	The url for the database in the correct format for the jdbc driver to be used.
username	Mandatory	A valid username to connect to the database.
password	Mandatory	The password needed for the user to connect to the database.
initSize	Optional	The initial size of the connection pool. The default is `5`.
maxSize	Optional	The maximum size of the connection pool. The default is `30`.
waitTime	Optional	The maximum time (in milliseconds) to wait for obtaining a connection. The default is `5000`.

In general, the optional values do not need to be set or changed.

In the Genesys Rules Development Tool, you can only configure database parameters with an SQL `SELECT` statement. Any other type of statement will fail when configured.

Web Service Parameters

In Configuration Server, Web Service Scripts must have a section called `webservice`. Table 3 lists the properties that you can specify for web service parameters.

Table 3: Web Service Parameter Properties

Property	Mandator/optional	Description
host	Mandatory	The host for the service.
base-path	Mandatory	The base path to access the service.
protocol	Optional	The default is `http`.
port	Optional	The default is `80`.
headers	Optional	Any custom HTTP headers that are needed for the service.
parameters	Optional	Any custom HTTP settings that are needed to tune the connection.

In general, the parameters values do not need to be set or changed.

Headers and parameters are lists in the following format:

```
key:value[,key:value]
```

Notes: You cannot specify headers or parameters that contain “, ” in the value.

If you are sending a message to the service, it is expected that `Content-Type` is specified in the header since it defines the overall message interaction with the server. An optional charset can be included. For example, `Content-Type:applicaton/json;charset=UTF-8`.

In the Genesys Rules Development Tool, you have to completely define the message to be sent and it must be constant. No variable substitution is done. The XPath Query is used to pull values out of the response from the server. The response must be in XML or JSON, otherwise this will not work. A valid XPath query for the response must be specified. This depends entirely on the service you interface with.

Note: The message is sent to the server only once per session. This is done both for performance reasons and because the values in the response are expected to be relatively constant.

In the Genesys Rules Development Tool, the path for the parameter is added to the `base_path` in the script.

For example:

If the Script contains:

```
host = api.wunderground.com
base_path = /auto/wui/geo/ForecastXML/
```

and the GRDT specifies:

```
query type = List
XPath Query = //high/fahrenheit/text()
HTTP Method = GET
path = index.xml?query=66062
message (not set)
```

then the message that is sent is:

```
GET /auto/wui/geo/ForecastXML/index.xml?query=66062 HTTP/1.1
```

This will return the week's highs in Fahrenheit:

```
81
77
81
81
83
85
```

Workforce Management Parameters

In Configuration Server, Workforce Management Scripts must have a section called `wfm`. Table 4 lists the properties that you can specify for Workforce Management parameters.

Table 4: Workforce Management Parameter Properties

Property	Mandator/optional	Description
wfmCfgServerApplName	Mandatory	Configuration Server application name for the WFM server.
wfmCfgServerUserName	Mandatory	Configuration Server user name.
wfmCfgServerPassword	Mandatory	Configuration Server password.
wfmServerUrl	Mandatory	URL of WFM Server.

When configuring a new parameter of type "Workforce Management" under the Genesys Rules Development Tool, simply name the parameter and choose the WFM profile (script object just created) from the drop-down list. When the author is using this parameter, the GRAT will fetch the current list of WFM Activities from the WFM Server and display them to the rule author.

Rule Language Mapping

When rule developers create the conditions or actions in a rule template, they enter the rule language mapping. In Genesys Rules System 8.1, the Drools Rule Language is used. This version of Genesys Rules System is based on the 5.1 version of Drools. For more information about the Drools Rule Language, see
`http://downloads.jboss.com/drools/docs/5.1.1.34858.FINAL/drools-expert/html/ch04.html`

Note: Because URLs change frequently, search the Drools web site for the *Drools Expert User Guide*, and then look at the table of contents of that guide for the information on the Drools Rule Language.

The rule language mapping is not visible to the business user when they are authoring rules in the Genesys Rules Authoring Tool. Instead, the rule authors will see the Language Expression that the rule template developer enters. The language expression is a plain-language description that uses terminology that is relevant to the business user, instead of low-level code. Rule language mapping is provided in the examples in the following section.

Language Expressions

When building a rule template in GRDT, the Language Expression cannot use the open or closed parenthesis character. For example, the expression:

```
More than {parCallLimit} calls within {parDayLimit} day(s)
```

will result in an error when you try to save the rule in GRAT. But if you want the business user to see a parenthesis in GRAT, you can use backslash characters in your Language Expression. For example:

```
More than {parCallLimit} calls within {parDayLimit} day\(s\).
```

Examples of Rule Development

This section provides some examples of what a rule developer might configure in the Rules Development Tool. More detailed information about how to configure rule templates is provided in the *Genesys Rules Development Tool Help*.

For specific information about how rule templates are configured to be used with the Genesys intelligent Workload Distribution (iWD) solution, refer to the section about iWD and the Genesys Rules System in the *iWD 8.1 Deployment Guide*.

Example 1: Condition and Action

If a customer's age is within a specific range, a specific Agent Group will be targeted. In this scenario, the Condition is whether the customer's age falls within the range. In the Genesys Rules Development Tool, the conditions would be configured as follows:

Name: `Age Range`

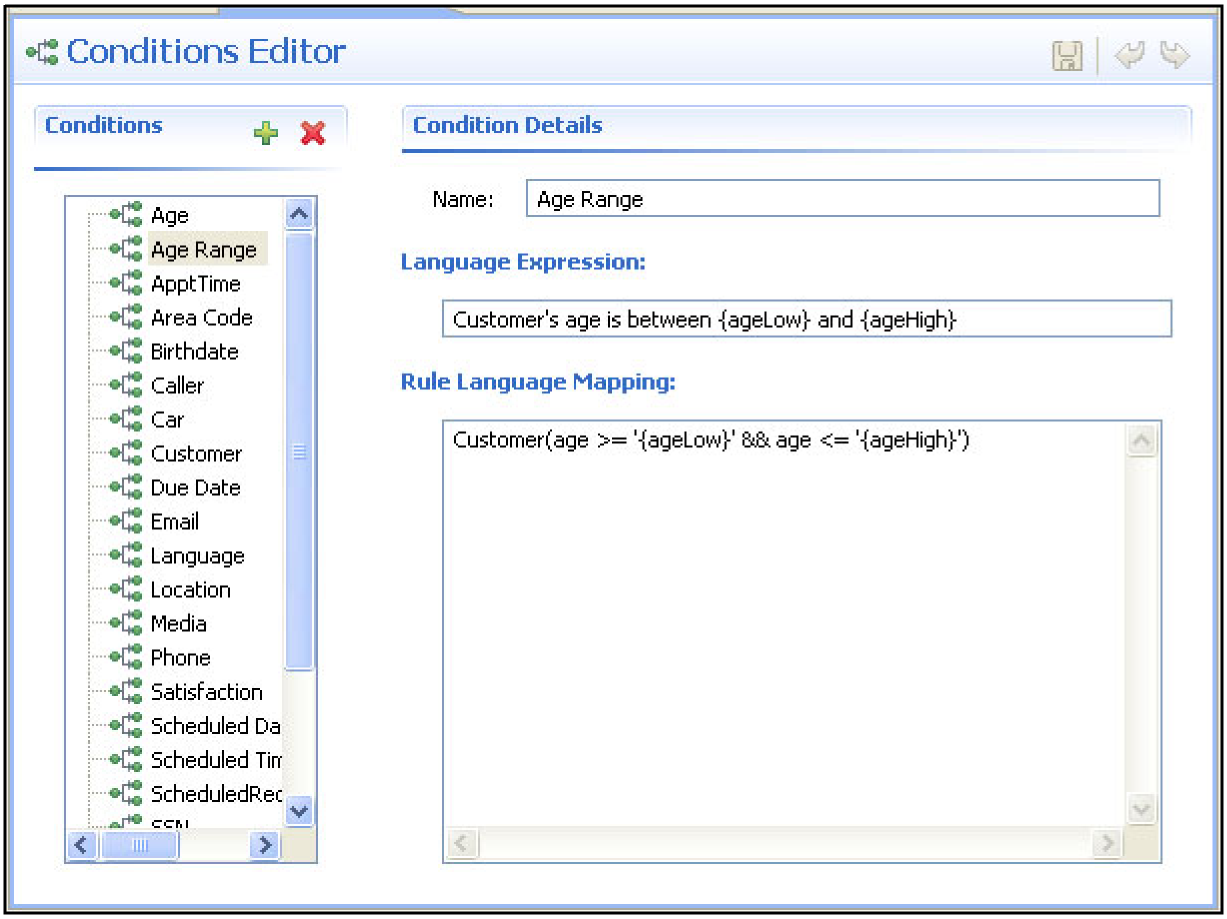
Language Expression: `Customer's age is between {ageLow} and {ageHigh}`

Rule Language Mapping:

`Customer(age >= '{ageLow}' && age <= '{ageHigh}')`

Warning! Do not use the word '`end`' in rule language expressions. Using '`end`' causes rule parsing errors.

Figure 8 shows how this condition would appear in the Genesys Rules Development Tool.



Name: Age Range

Language Expression:

Customer's age is between {ageLow} and {ageHigh}

Rule Language Mapping:

Customer(age >= '{ageLow}' && age <= '{ageHigh}')

Figure 8: Age Range Condition

In addition to testing that the Caller exists, the next condition also creates the `$Caller` variable which is used by actions to modify the `Caller` fact. The modified `Caller` will be returned in the results of the evaluation request.

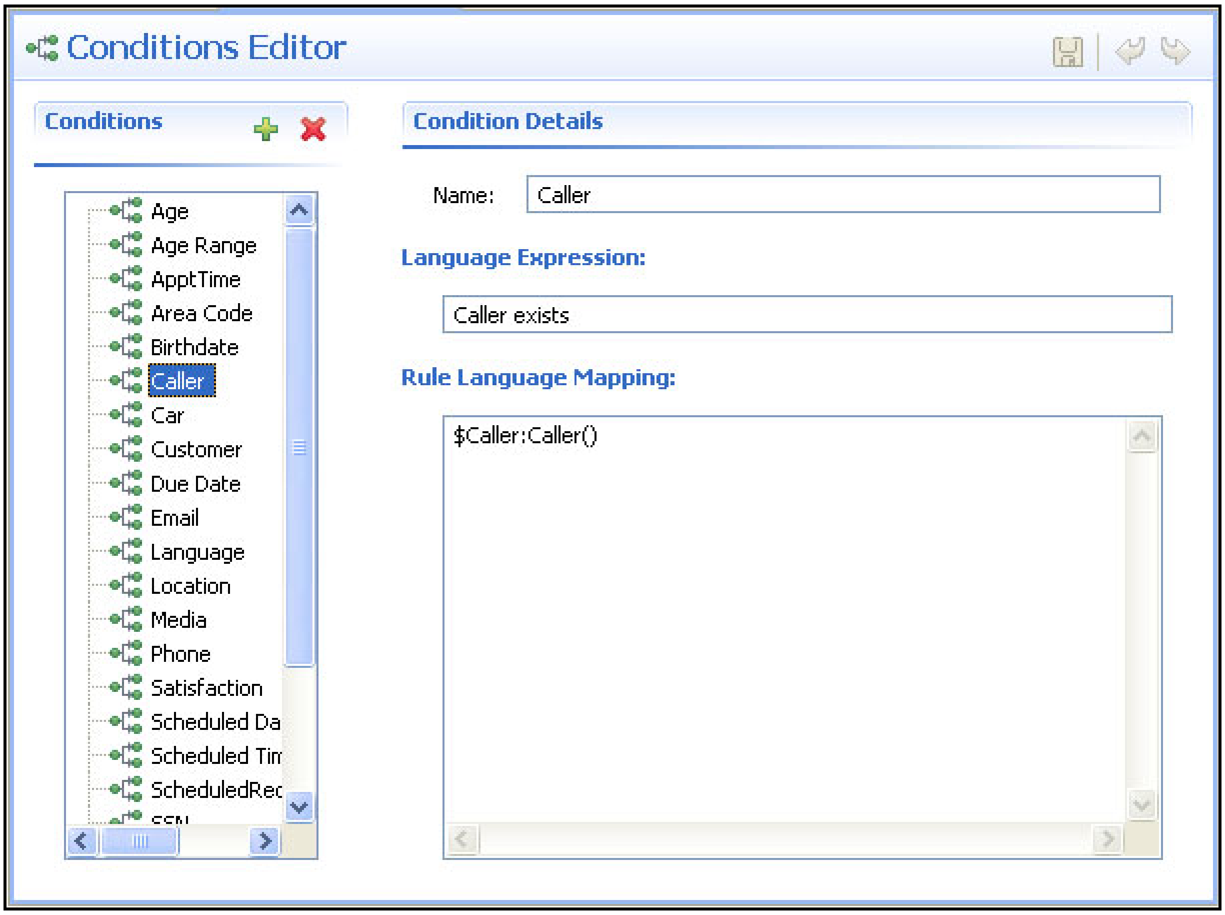
You cannot create a variable more than once within a rule, and you cannot use variables in actions if the variables have not been defined in the condition.

Name: `Caller`

Language Expression: `Caller exists`

Rule Language Mapping: `$Caller:Caller`

Figure 9 shows how this condition would appear in the Genesys Rules Development Tool.



Name: Caller

Language Expression:

Caller exists

Rule Language Mapping:

```
$Caller:Caller()
```

Figure 9: Caller Condition

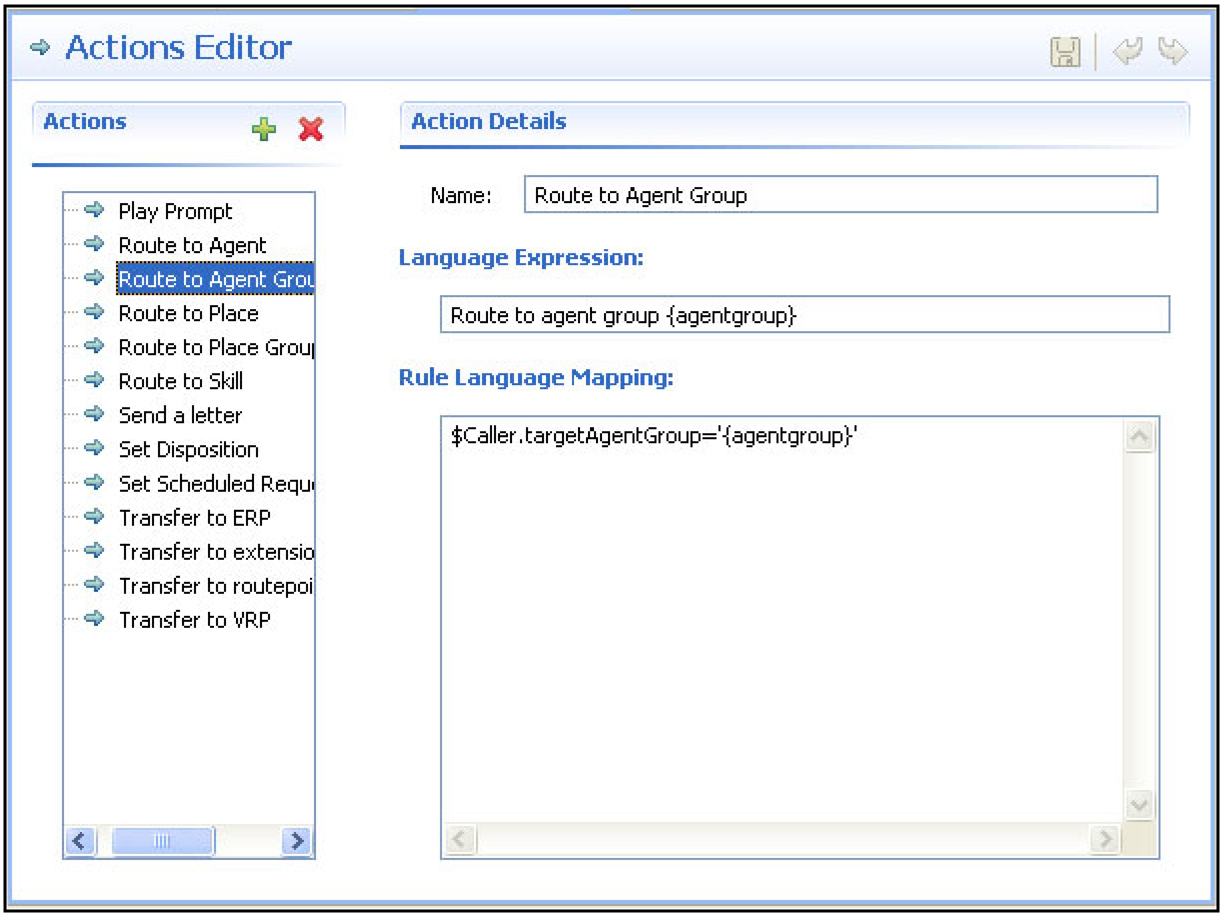
The action would be configured as follows:

Name: `Route to Agent Group`

Language Expression: `Route to agent group {agentGroup}`

Rule Language Mapping: `$Caller.targetAgentGroup='{agentgroup}'`

Figure 10 shows how this action would appear in the Genesys Rules Development Tool.



- Play Prompt
- Route to Agent
- Route to Agent Grou
- Route to Place
- Route to Place Grou
- Route to Skill
- Send a letter
- Set Disposition
- Set Scheduled Requ
- Transfer to ERP
- Transfer to extensio
- Transfer to routepoi
- Transfer to VRP

Name: Route to Agent Group

Language Expression:

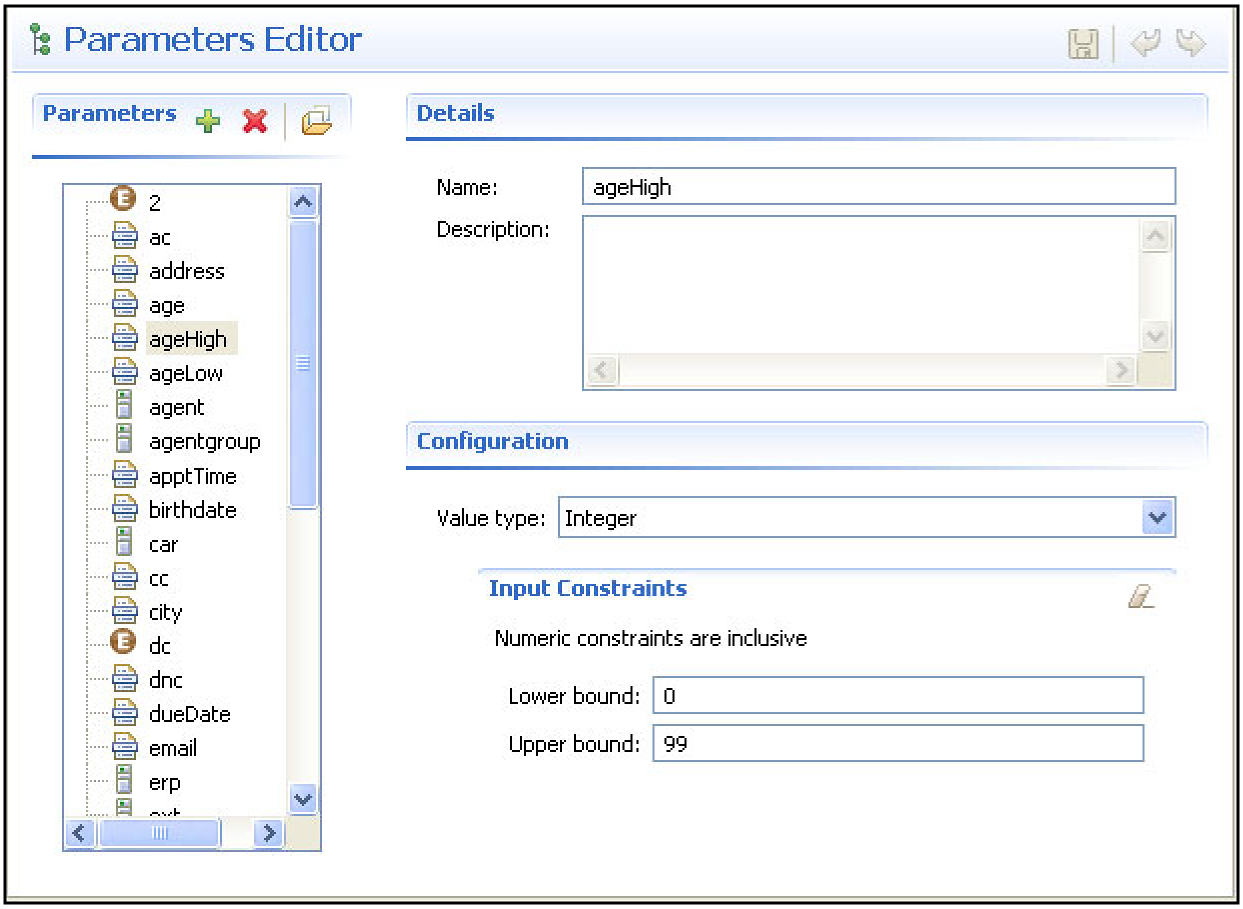
Route to agent group {agentgroup}

Rule Language Mapping:

```
$Caller.targetAgentGroup='{agentgroup}'
```

Figure 10: Target Agent Group Action

The condition in this example has two parameters: `{ageLow}` and `{ageHigh}`. The action has the `{agentGroup}` parameter. Parameters are also configured in the Genesys Rules Development Tool. Figure 11 shows a sample `{ageHigh}` parameter. Refer to the *Genesys Rules Development Tool Help* for more details about how to configure parameters.



Name: ageHigh

Description:

Value type: Integer

Numeric constraints are inclusive

Lower bound: 0

Upper bound: 99

Figure 11: {ageHigh} Parameter

The way the preceding example would work is as follows:

1. The rule developer creates a fact model (or the fact model could be included as part of a rule template that comes out of the box with a particular Genesys solution). The fact model describes the properties of the `Customer` fact and the `Caller` fact. In this case we can see that the `Customer` fact has a property called `age` (probably an integer) and the `Caller` fact has a property called `targetAgentGroup` (most likely a string).
2. The rule developer creates the `ageLow` and `ageHigh` parameters, which will become editable fields that the business user will fill in when they are authoring a business rule that uses this rule template (but see "Mapping Multiple Instances of a Rule Parameter to a Single Parameter Definition" on page 78 for post-8.1.2 releases). These parameters would be of type `Input Value` where the `Value Type` would likely be `integer`. The rule developer optionally can constrain the possible values that the business user will be able to enter by entering a `Lower Bound` and/or an `Upper Bound`. The rule developer also creates the `agentGroup` parameter, which will likely be a selectable list whereby the business user would be presented with a

drop-down list of values that are pulled from Genesys Configuration Server or from an external data source. The behavior of this parameter depends on the parameter type that is selected by the rule developer.

3. The rule developer creates a rule action and rule condition as previously described. The action and condition include rule language mappings that instruct the Rules Engine as to which facts to use or update based on information that is passed into the Rules Engine as part (of the rule evaluation request coming from a client application such as an SCXML application).
4. The rule developer publishes the rule template to the Rules Repository (but see “Creating Test Scenarios Before Deployment” on page 78 for post-8.1.2 releases).
5. The rules author uses this rule template to create one or more business rules that utilize the conditions and actions in the combinations that are required to describe the business logic that the rules author wants to enforce. In this case, the previously described conditions and action above likely would be used together in a single rule, but the conditions and action could also be combined with other available conditions and actions to create different business policies.
6. The rules author deploys the rule package to the Rules Engine (application server(but see “Deploying to Application Clusters” on page 78 for post 8.1.2-releases).
7. A client application such as a VXML or SCXML application invokes the Rules Engine and specifies the rule package to be evaluated. The request to the Rules Engine will include the input and output parameters for the fact model. In this example, it would have to include the `age` property of the `Customer` fact. This age might have been collected through GVP or extracted from a customer database prior the Rules Engine being called.
8. Based on the value of the `Customer.age` fact property that is passed into the Rules Engine as part of the rules evaluation request, the Rules Engine will evaluate a particular set of the rules that have been deployed. In this example, it will evaluate whether `Customer.age` falls between the lower and upper boundaries that the rules author specified in the rule.
9. If the rule evaluates as `true` by the Rules Engine, the `targetAgentGroup` property of the `Caller` fact will be updated with the name of the Agent Group that was selected by the business rules author when the rule was written.
10. The value of the `Caller.targetAgentGroup` property will be passed back to the client application for further processing. In this example, perhaps the value of `Caller.targetAgentGroup` will be mapped to a Composer application variable which will then be passed into the Target block to ask the Genesys Universal Routing Server to target that Agent Group.

Differences in Release 8.1.2

Mapping Multiple Instances of a Rule Parameter to a Single Parameter Definition

At the point of creating parameters, instead of create the `ageLow` and `ageHigh` parameters (as in pre-8.1.2 releases) the rule template developer can now create a single `{age}` parameter and use the underscore notation shown in the example below to create indices of it for scenarios in which multiple instances of parameter with the same type (`age`) are required (most commonly used with ranges). For example:

```
{age_1}, {age_2}....{age_n}
```

These will become editable fields in the same way as in "Example 1: Condition and Action" on page 73. This feature is most typically used for defining ranges more efficiently.

Fact/Condition

In release 8.1.2, Facts can be referenced in conditions and actions by prefixing the fact name by a $ sign. For example, the fact `Caller` can be referenced by the name `$Caller`. GRS will implicitly generate a condition that associates the variable `$Caller` to the fact Caller (that is, `$Caller:Caller()`).

Note: The condition `$Caller:Caller()` requires a Caller object as input to rules execution for this condition to evaluate to true.

Creating Test Scenarios Before Deployment

In release 8.1.2, the rule author can create and execute test scenarios before deploying the rule. This would typically become Step 4 in "Example 1: Condition and Action" on page 73.

Deploying to Application Clusters

In release 8.1.2, rules can be deployed to application clusters defined in Genesys Administrator. Clusters may have multiple application of one type per cluster.

Example 2: Function

Functions are used for more complex elements and are written in Java. In this example, the function is used to compare dates. It would be configured as follows:

Name: `compareDates`

Description: `This function is required to compare dates.`

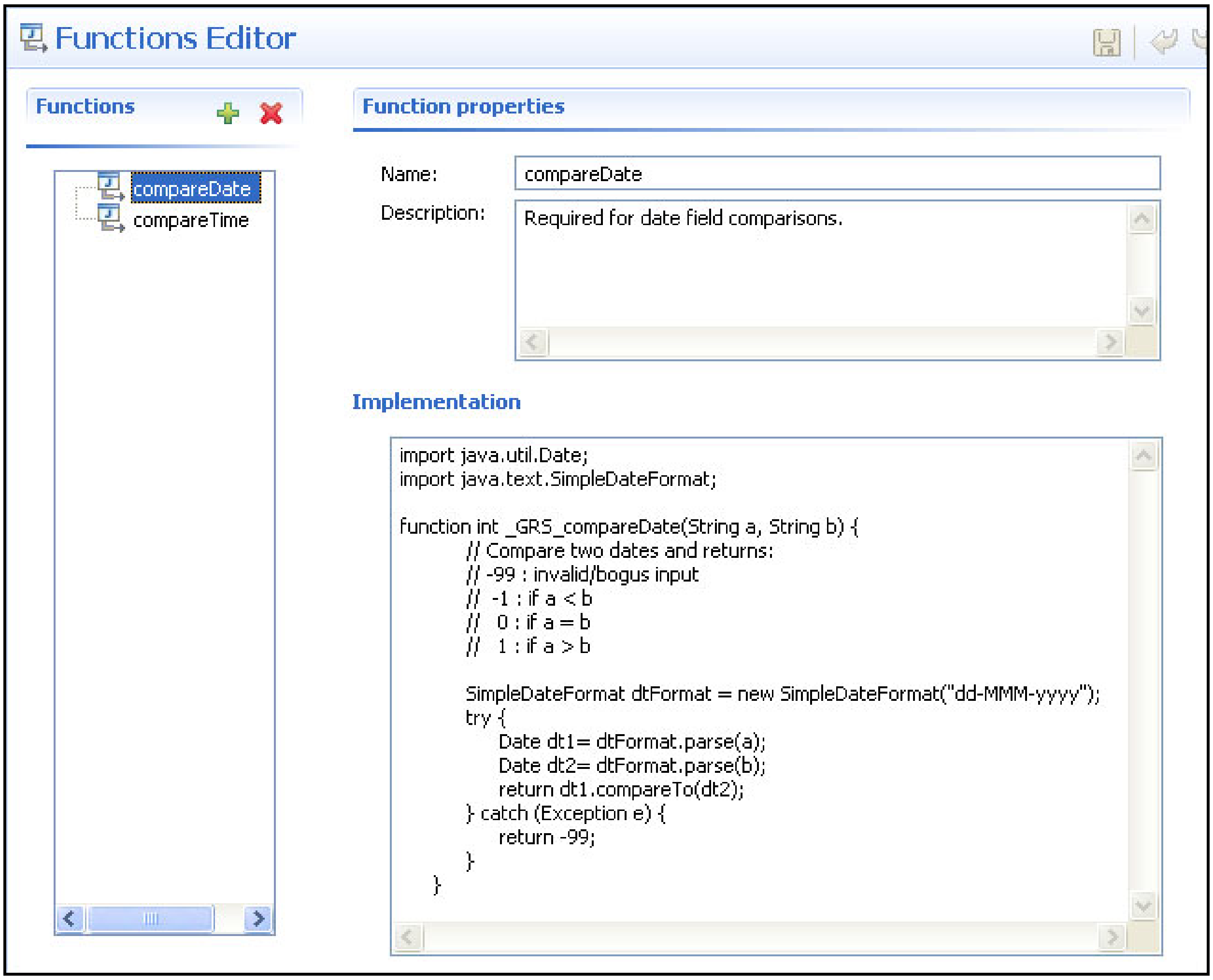
Implementation:

```
import java.util.Date;
import java.text.SimpleDateFormat;

function int _GRS_compareDate(String a, String b) {
            // Compare two dates and returns:
            // -99 : invalid/bogus input
            //  -1 : if a < b
            //   0 : if a = b
            //   1 : if a > b
            SimpleDateFormat dtFormat = new
SimpleDateFormat("dd-MMM-yyyy");
            try {
                  Date dt1= dtFormat.parse(a);
                  Date dt2= dtFormat.parse(b);
                  return dt1.compareTo(dt2);
            } catch (Exception e) {
                  return -99;
            }
      }
```

Note: For user-supplied classes, the .jar file must be in the CLASSPATH for both the GRAT and the GRE.

Figure 12 shows how this function would appear in the Genesys Rules Development Tool.



Name: compareDate

Description: Required for date field comparisons.

```
import java.util.Date;
import java.text.SimpleDateFormat;

function int _GRS_compareDate(String a, String b) {
        // Compare two dates and returns:
        // -99 : invalid/bogus input
        //  -1 : if a < b
        //   0 : if a = b
        //   1 : if a > b

        SimpleDateFormat dtFormat = new SimpleDateFormat("dd-MMM-yyyy");
        try {
            Date dt1= dtFormat.parse(a);
            Date dt2= dtFormat.parse(b);
            return dt1.compareTo(dt2);
        } catch (Exception e) {
            return -99;
        }
    }
```

Figure 12: compareDates Function

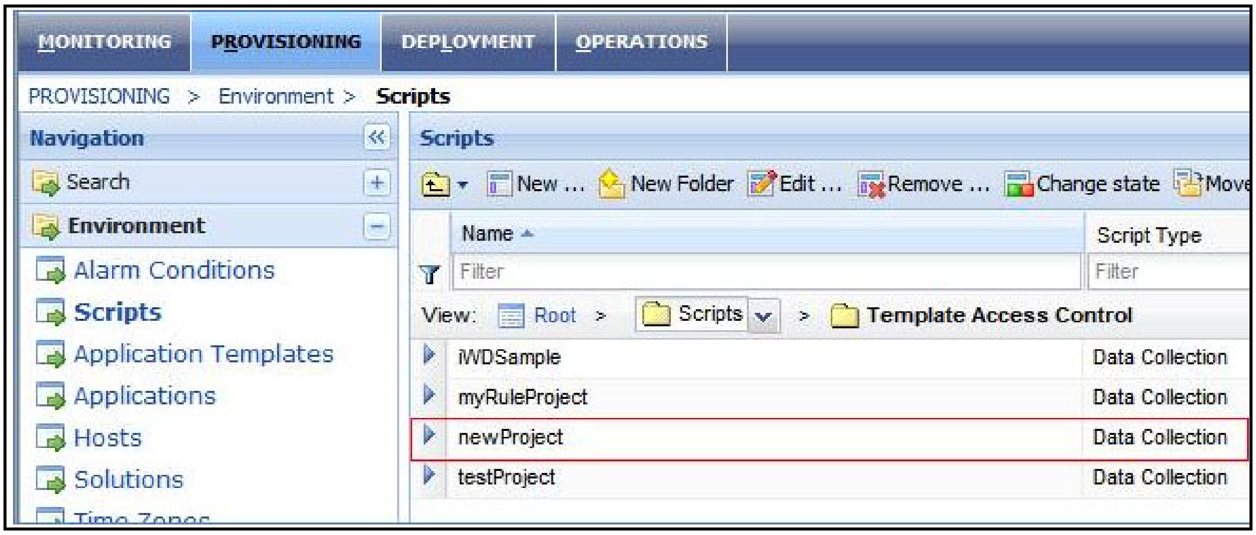
Deleting Rule Templates

Rule templates cannot be deleted through GRDT.

Deleting Templates—Releases Prior to 8.1.2

In releases prior to 8.1.2, to ensure that a template is no longer visible to rule authors when they create a new rule package, you must remove permissions on the `Script` object in Genesys Administrator or Configuration Manager. This way the rule template will not be visible to the rule author and cannot be used.

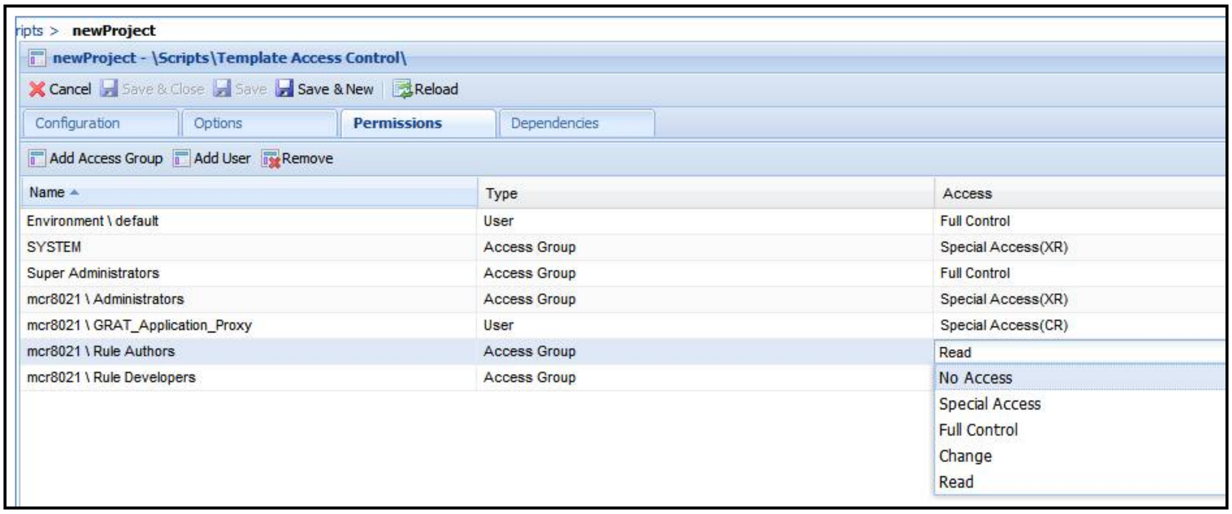
In Genesys Administrator or Configuration Manager, in the `Scripts` section, there is a folder called `Template Access Control`. It contains a `Script` object that corresponds to each rule template in the Rules Repository. (See Figure 13. The `Script` object `newProject` corresponds to a rule template of the same name). Use permissions to control which users and/or access groups should be able to use this template.



Name	Script Type
iWDSample	Data Collection
myRuleProject	Data Collection
newProject	Data Collection
testProject	Data Collection

Figure 13: Script Objects

Open the `Script` object and select the `Permissions` tab. You can select `No Access` (as shown in Figure 14) or, alternatively, select the Access Group or User from the list, and then click the `Remove` button.



Name	Type	Access
Environment \ default	User	Full Control
SYSTEM	Access Group	Special Access(XR)
Super Administrators	Access Group	Full Control
mcr8021 \ Administrators	Access Group	Special Access(XR)
mcr8021 \ GRAT_Application_Proxy	User	Special Access(CR)
mcr8021 \ Rule Authors	Access Group	Read
mcr8021 \ Rule Developers	Access Group	

Figure 14: Access Permissions for the Script Object

When the rules author logs into the Rules Authoring Tool, `newProject` will not be listed as an available rule template.

Deleting Templates—Release 8.1.2 and Higher

In release 8.1.2, rule templates can be deleted using the GRS Server Explorer in the GRDT, provided that:

- The user has rule template delete permissions, and;
- The rule is not used in any rule package.

Rules

Business rules are created in the GRAT by business analysts. Rules are created within a rule package. When a rule package is created, one or more rule templates can be selected for inclusion. The templates determine the conditions, actions, and so on that are available to use during creation of business rules.

For specific information about how business rules are used with the Genesys intelligent Workload Distribution (iWD) solution, refer to the section about iWD and the Genesys Rules System in the *iWD 8.1 Deployment Guide*.

Note: As well as creating a rule package, the GRAT enables you to import and export existing rule packages. This ability enables you, for example, to import a rules package from a test environment to a production environment, or to export a rules package for backup prior to upgrading.

You can configure rules for various business contexts (nodes that represent the various elements in your business structure hierarchy) or, for global rules, at the rule package level. In the Explorer Panel of the Rules Authoring Tool, each business context within the configured business structure is represented as a different node level. The order of execution of rules within a rule package depends on the node level; global rules are executed first, followed by rules at node level 1, and so on. Within a given node, you can modify the order of execution by using the up or down arrows on each rule. Rules will be executed from the top down. Refer to the *Genesys Rules Authoring Tool Help* for more information about how to configure rules and rule packages, and refer to Chapter 5 of this guide for information about how to configure your business structure.

Using the same example that was used in the rule language mapping section (see page 73), the following example shows how the action and condition might be used in a linear rule.

Example 1: Linear Rule

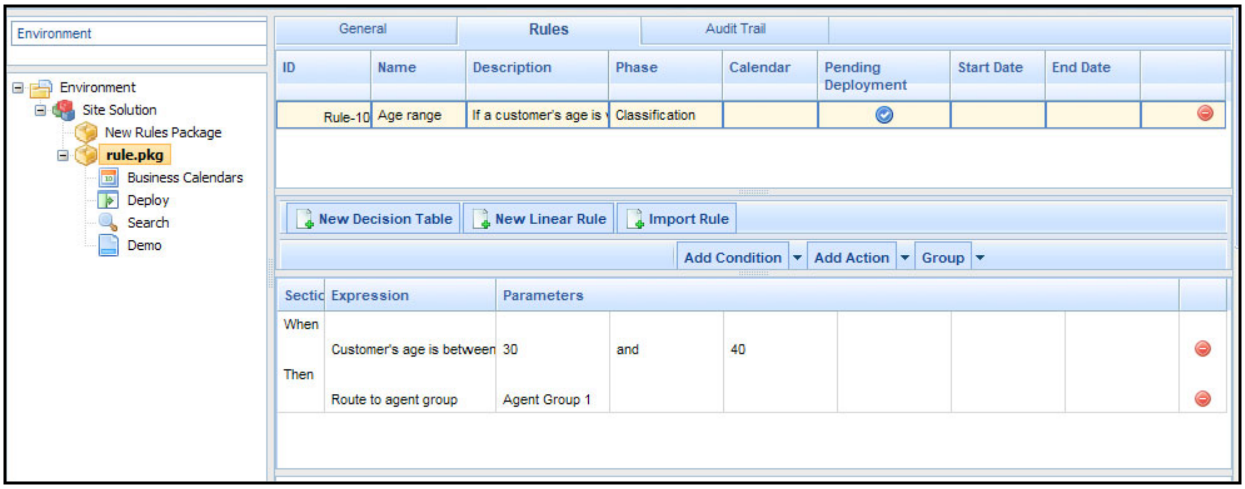
If a customer's age is within the range of 30-40 years, the customer's interaction will be routed to Agent Group 1.

In the Genesys Rules Authoring Tool, create a new linear rule. Enter the name, phase, and so on, as desired, and then add a condition and an action. The phases from which the rules author can select are dictated by the rule template that the rules author is using. There is an enumeration called Phases within the `_GRS_Environment` fact, that will be created whenever a new rules template project is created in the Genesys Rules Development Tool. If the Phases enumeration is not present, the rules author will simply see * in the `Phase` dropdown. In this case, Phase will not be considered when evaluating the rule package.

The `Add Condition` and `Add Action` drop-down lists are populated with all of the conditions and actions that were created in the rule templates that are included in the rule package. The drop-down lists contain the language expressions that the rule developers used during creation of the components, and not the rule language mapping. This makes it possible to create rules without knowing the rule language mapping or being familiar with Drools.

The parameters that are contained in each condition and action are represented by the names that are entered for them. The business rule author must replace this name either by entering a value (such as for an age range) or by selecting an option from the drop-down list (such as for an Agent Group).

So, to create this rule, the rules author would select `Age Range` as the condition and enter `30` as the `{ageLow}` parameter and `40` as the `{ageHigh}` parameter. The action would be `Target Agent Group`, and `Agent Group 1` would be selected from the `{agentGroup}` drop-down list. Figure 15 shows the linear rule in the Genesys Rules Authoring Tool.



ID	Name	Description	Phase	Calendar	Pending Deployment	Start Date	End Date	
Rule-10	Age range	If a customer's age is v	Classification		✓			●

Sectic	Expression	Parameters						
When								
	Customer's age is between	30	and	40				●
Then								
	Route to agent group	Agent Group 1						●

Figure 15: Sample Linear Rule

Example 2: Decision Table

Decision tables allow you to create a number of rules that have the same set of conditions (`WHEN`) and actions (`THEN`) that are to be used for a complex (structured) business case. Use decision tables to avoid dozens of linear rules that have an identical structure in the system.

Note: Choices in decision tables must be mutually exclusive to avoid ambiguity. This ensures that there is only one outcome per evaluation. If the choices are not mutually exclusive, multiple rows may be executed in no guaranteed order. The last row that is executed will determine the final result.

Figure 26 on page 110 of Appendix C shows a sample decision table.

Note: When you are editing rules, be careful not to clear your cookie data, as this might cause the rule to become stuck in a locked state until the session times out (the default is 30 minutes). Consult the documentation for the browser that you are using for more information about how to prevent a user from clearing cookie data.

Chapter

5 Business Structure

This chapter provides information about how to configure the business structure that is used by Genesys Rules System. It contains the following sections:

Defining the Business Structure

The business structure is a hierarchy of business units. No business structure is created out-of-box for Genesys Rules System; the business structure must be configured in Genesys Administrator or Configuration Manager. For customers who are using the Genesys Rules System with intelligent Workload Distribution, the business structure is created in iWD Manager and then synchronized with Configuration Server, after which it becomes available for use by the Genesys Rules System.

The business structure that you configure will be visible in the Genesys Rules Authoring Tool. Each rule package will display the business structure for the Tenant. Each Tenant can contain one more Solutions as the first level of the hierarchy, and rules can be defined at each level (node) of the business structure from Solutions down.

Note: Rules that are configured for the Solution, known as global rules, are executed first, followed by rules configured for the first node of the business structure, then rules configured for the second node, and so on. Global rules are only “global” within the defined rule package.

The business structure that you create can vary depending on a number of factors, including whether Genesys Rules System is to be used for iCFD.

Sample structures are provided in this chapter. The structure can be product- or business-specific.

Object permissions are used to determine which elements of a business structure are visible to various users. See Chapter 6, “Role-Based Access Control,” on page 89 for more information.

The following procedure explains how to configure a business structure.

Procedure: Configuring the business structure

Purpose: To configure your Tenant’s business structure. The business structure is created under Resources for single-tenant Configuration Server, or under a Tenant for a multi-tenant Configuration Server.

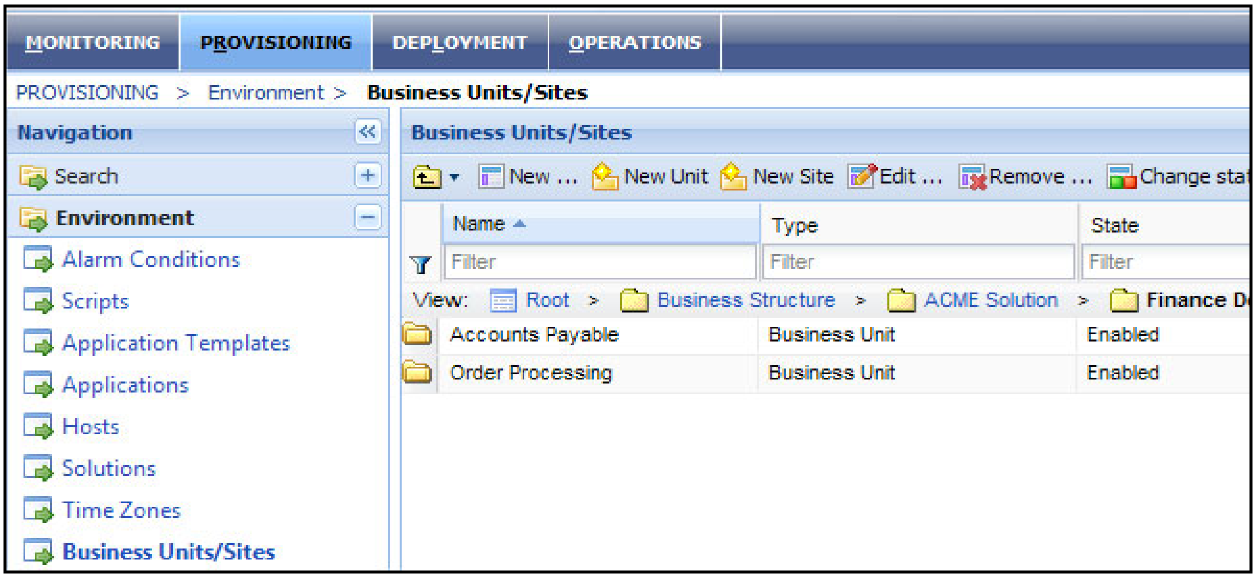
Start of procedure

1. Navigate to the Resources folder for a single-tenant Configuration Server, or to the specific Tenant for a multi-tenant Configuration Server, and open the Business Units/Sites (in Genesys Administrator) or Configuration Units (in Configuration Manager) folder. Create a new top-level folder named `Business Structure`.

Note: This folder *must* be named `Business Structure`.

2. Within the `Business Structure` folder, click either `New Unit` or `New Site` to create at least one more Business Unit or Site (it does not matter whether you create a site or a unit). This new site/unit will represent the Solution.Within the new folder (the Solution), additional levels of hierarchy can be created as needed, using either Business Units or Sites. The levels of hierarchy beneath the Solution level will represent the business context.

3. Multiple Solutions can be created by creating additional Business Units or Sites directly beneath the `Business Structure` folder. Figure 16 shows a sample business structure in Genesys Administrator, Figure 17 shows a sample business structure in Configuration Manager, and Figure 18 shows how the business structure will appear in the Genesys Rules Authoring Tool.



Name	Type	State
Filter	Filter	Filter
Accounts Payable	Business Unit	Enabled
Order Processing	Business Unit	Enabled

Figure 16: Sample Business Structure in Genesys Administrator

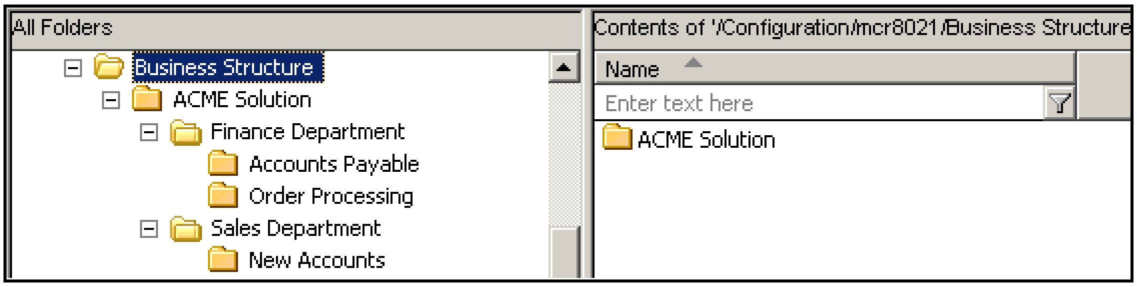


Figure 17: Sample Business Structure in Configuration Manager

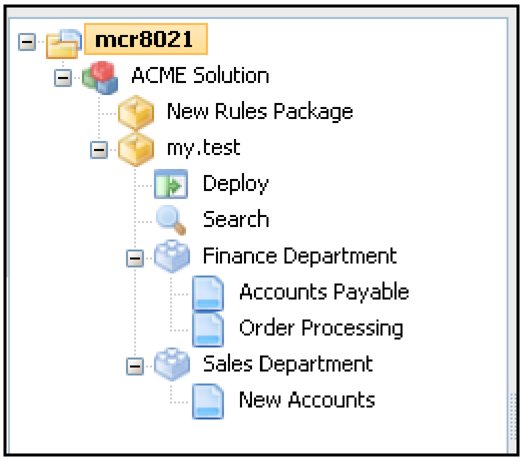
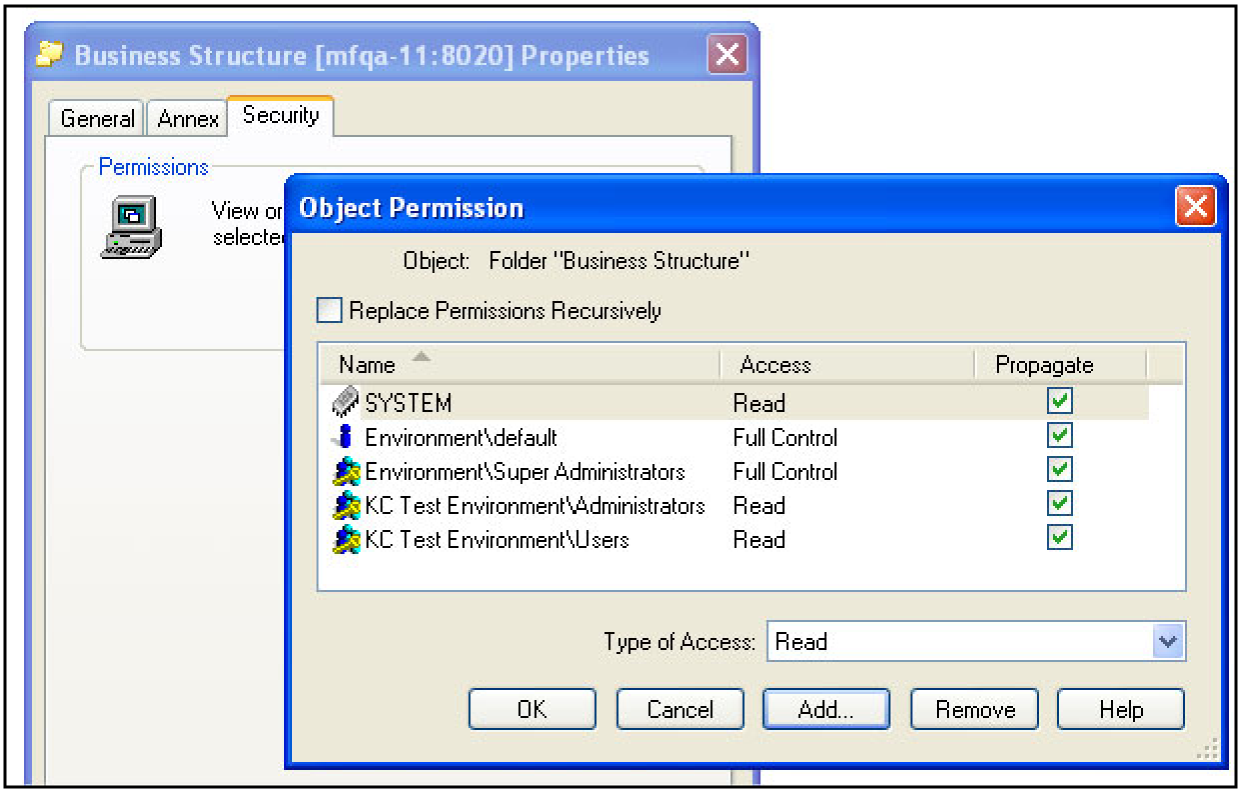


Figure 18: Sample Business Structure in the Genesys Rules Authoring Tool

4. Read permission to the `Business Structure` folder must be provided to the users and/or access groups that you want to use the Rules Authoring Tool. Normally this will be propagated automatically, if the user or access group

has permission to the Tenant object. If you do not want a user or access group to have permission to see all of the nodes of the business structure, you can control this by not giving that user or the access group(s) of that use read permission to those folders. Figure 19 shows that all members of the Users access group have Read permissions to the `Business Structure` folder.



Object: Folder "Business Structure"

☐ Replace Permissions Recursively

Name	Access	Propagate
SYSTEM	Read	✓
Environment\default	Full Control	✓
Environment\Super Administrators	Full Control	✓
KC Test Environment\Administrators	Read	✓
KC Test Environment\Users	Read	✓

Type of Access: Read

Figure 19: Permissions for the Business Structure folder

End of procedure

iCFD Business Structures

iCFD business structures can be configured in any way that best suits your business needs. For example, you could have separate Sites/Units for Product Types, Lines of Business, Departments, and so on. Genesys recommends that the business structure be no more than two or three levels deep, to help keep it manageable.

Chapter

6

Role-Based Access Control

This chapter provides information about how to configure role-based access control for Genesys Rules System8.1. It contains the following section:

Role-Based Access Control

Genesys Rules System role-based access control utilizes Configuration Server-defined access groups and roles to control visibility and access to rule packages, rule templates, rules, and business calendars. Because these objects are not stored in the Configuration Server database they will not have security permissions associated with them, as Configuration Server objects do. The GRAT server will utilize the access permissions for the container object, and the Genesys Rules System objects will inherit these access permissions.

Note: Role-based access control requires Configuration Server 8.0.2 or higher and Genesys Administrator 8.0.2 or higher.

Rule packages and business calendars inherit their access permissions from the `Tenant` object with which they are associated and the `Business Structure` folder access permissions. Business rules are associated with a specific node in the business structure. Their access permissions are inherited from the Configuration Server–defined node with which they are associated (the business structure nodes are created by using Configuration Manager or Genesys Administrator). Rule templates have `Script` objects created in Configuration Server that are used to hold the individual access permissions of the rule template. Additionally, rule templates inherit the access permissions from the business structure node with which they are associated.

Role Permissions

Genesys Rules System 8.1 defines a set of role permissions for governing the tasks that can be performed in the Genesys Rules Authoring Tool. The set of permissions is the following:

- Business Calendar - Create
- Business Calendar - Delete
- Business Calendar - Modify
- Business Calendar - View
- Business Rule - Create
- Business Rule - Delete
- Business Rule - Modify
- Business Rule - View
- Rule Template - Create
- Rule Template - Delete
- Rule Template - Modify
- Rule Package - Create
- Rule Package - Delete
- Rule Package - Modify
- Rule Package - Deploy

Additionally, in release 8.1.2, the following permissions are also defined:

- Test Scenario - Create
- Test Scenario - Modify
- Test Scenario - Delete
- Test Scenario - View
- Test Scenario - Execute

The combination of the *access* permissions and the *role* permissions will determine whether a task can be performed. For example, to view a rule a user must have Read permission for the node with which the rule is associated as well as the Business Rule - View role permission. To delete a rule, the user must have Read permissions for the node and the Business Rule - Delete role permission. In this example, Read access permission is also needed for the delete task, because the user will not have visibility to any object that is associated with the node without Read access permissions.

Role permissions for importing and exporting templates and rule packages must be set to the following values:

- To import a template, a user must have Create permission for the Rule Template.

- To export a template, a user must have read access to the Template Script Object representing the template. See “Template Script Objects” on page 92 for more information.
- To import a rule package, a user must have Create permission for both the Rule Package and Business Calendar.
- To export a rule package, a user must have View permission for the Business Calendar.

User Logins

The GRAT has multiple connections to Configuration Server. There is the server connection that is used by the Rules Authoring server to read application information and perform various server tasks, and each user that logs on to the GRAT has an individual client connection, which is limited based on the configuration of the user’s login.

Business Hierarchy

Each Tenant should contain a folder called `Business Structure` (for single-tenant Configuration Servers, the `Business Structure` folder must be created under `Resources`). Under that folder there can be multiple levels (nodes) of sites/business units that represent the business hierarchy for this Tenant. Each user login should be configured in Configuration Server with Read permissions for only the Tenants that will be visible to this user (if there is more than one Tenants) and Read permissions for only the nodes of the business hierarchy that this user can view. Users who have Rule View permissions can see all of the rules that are associated with a node that is visible to them. See Chapter 5, “Business Structure,” on page 85, for more information about business structures.

Role Task Permissions

When the Genesys Rule Authoring Tool has been deployed by using Genesys Administrator, role task permissions can be configured in Genesys Administrator. A new `Role` object can be created under `Provisioning > Accounts > Roles`. On the `Role Privileges` tab there will be a check box to add the privileges that are associated with the Genesys Rule Authoring Generic Server. Users can be granted a specific set of permissions by adding them as members of a role—either individually or as part of an access group. There are four groups of privileges:

- Rule Authoring—Create, Delete, Modify, and View
- Rule Packages—Create, Modify, Delete, and Deploy
- Rule Templates—Create, Modify, and Delete
- Test Scenarios—Create, Modify, View, Delete and Execute

- Business Calendars—Create, Delete, Modify, and View

Template Script Objects

`Script` objects are used to control visibility to templates. Whenever a template is created a `Script` object is created automatically in the `Template Access Control` folder under the `Scripts` folder to represent that template. A user must have read access to that `Script` object to be able to view that template. It is suggested that you give template developers View permissions to the `Template Access Control` folder and have that permission propagate to all sub-objects. This way template developers can immediately view any template that they may create. All other users will not be able to see the newly created templates until view permissions are explicitly granted for that template.

Configuring a User

The following procedure provides the basic steps for setting up users for the Rules Authoring Tool.

Procedure: Configuring a user for the Genesys Rules Authoring Tool

Purpose: To configure a user for the Genesys Rules Authoring Tool.

Start of procedure

1. Give the user Read access to all of the Tenants that they can access.
2. Add the user as a member of a role, with the desired permissions.
3. Give the user Read access to the `Business Structure` folder and all of the desired nodes for that user.
4. Give the user Read access to all of the desired templates through the `Script` objects.

End of procedure

Appendix

A REST API

This appendix describes the REST API that is supported by the GRE in 8.1. It contains the following sections:

Rule Execution

The Rules Engine accepts REST requests from clients through a configured port. Clients that want to execute a rule package will connect to this port and send a HTTP POST message to `http://{server-address:port}/{server-id}/knowledgebase/{packageName}`. This port is configured in the GRE application. See Procedure: Deploying the Genesys Rules Engine in Genesys Administrator, on page 26, for more information about how to configure this port.

The `server-id` is a configured value for the server and is not examined for the request. The `packageName` corresponds to the already deployed rule package that is to be evaluated.

The body of the HTTP request contains a `knowledgebase-request` in either XML or JSON format. If JSON is used, the Content-Type HTTP header must be set to `application/json`. A successful response will contain a `knowledgebase-response` message that contains the results of the evaluation.

The following schema defines the body of both the `knowledgebase-request` and `knowledgebase-response` message bodies.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">

  <xs:element name="knowledgebase-request">
    <xs:complexType>
      <xs:sequence>
```

```
        <xs:element name="globals" type="globals" minOccurs="0" maxOccurs="1"/>
        <xs:element name="inFacts" type="inFacts" minOccurs="0" maxOccurs="1"/>
        <xs:element name="inOutFacts" type="inOutFacts" minOccurs="0"
maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="knowledgebase-response">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="globals" type="globals" minOccurs="0" maxOccurs="1"/>
        <xs:element name="outFacts" type="outFacts" minOccurs="0"
maxOccurs="1"/>
        <xs:element name="inOutFacts" type="inOutFacts" minOccurs="0"
maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="globals">
    <xs:element name="named-fact" type="named-fact" maxOccurs="unbounded"/>
  </xs:complexType>
  <xs:complexType name="inFacts">
    <xs:group ref="factGroup" maxOccurs="unbounded"/>
  </xs:complexType>
  <xs:complexType name="inOutFacts">
    <xs:group ref="factGroup" maxOccurs="unbounded"/>
  </xs:complexType>
  <xs:complexType name="outFacts">
    <xs:group ref="factGroup" maxOccurs="unbounded"/>
  </xs:complexType>

  <xs:group name="factGroup">
    <xs:choice>
      <xs:element name="named-fact"/>
      <xs:element name="anon-fact"/>
    </xs:choice>
  </xs:group>

  <xs:complexType name="named-fact">
    <xs:sequence>
      <xs:element name="id" type="id"/>
      <xs:element name="fact" type="fact"/>
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="id">
    <xs:annotation>
      <xs:documentation>The identifier for a named fact</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>

  <xs:complexType name="anon-fact">
    <xs:sequence>
      <xs:element name="fact" type="fact"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="fact">
    <xs:annotation>
      <xs:documentation>Contained elements are named after the fields of the
class referred to by the class attribute. Element values are the values of the
fields</xs:documentation>
```

```
      </xs:annotation>
      <xs:attribute name="class" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:pattern value="\c+(\.\c+)*"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
    </xs:schema>
```

Changes in 8.1.2

In release 8.1.2, in addition to the action resulting from evaluation of the conditions in a rule that executes, the GRE returns to the application a list of the names of each rule executed, and in addition, for decision tables, the name of each row that was executed.

For the REST interface, the GRE returns an array of rule names within an `executionResult` block. For example:

```
<executionResult>
  <rulesApplied>
     <string>Rule-110 Dept ABC</string>
     <string>Rule-118 Process XYZ</string>
  </rulesApplied>
</executionResult>
```

For the ESP interface, the GRE returns the following two parameters on the ESP response:

- `NumberOfRulesApplied`
  A count of the number of rules or decision tables rows that evaluated to true.
- `RulesApplied`
  The names of the rules or decision tables rows that evaluated to true. The names are separated by semicolons.

Note: The rules are listed in the order of execution. If a rule is executed multiple times it will appear in the list multiple times.

Error Handling

If the GRE cannot process a received request, it issues an error response to the client. These errors are sent back to the client by using standard HTTP status codes. Table 5 describes these errors and the contents of the responses. The body of the error message will be in the same format as the request, as indicated by the Content-Type HTTP header. The text in the body of the error message will be replaced with appropriate information based on the request context.

Table 5: Error Codes and Responses

HTTP status code	Description	Cause	Body of the error message
400	Bad Request - URI not valid	The received URI does not match the GRE's REST specification.	`<error code=610>`URI needs to be of the form */knowledgebase/{packageName}*" `</error>`
404	Not Found - Package not found	The package for the received evaluation request was not found.	`<error code=620>`"Rule package *pkgName* was not found"`</error>`
406	Not Acceptable - Unable to convert message	The received evaluation request could not be converted to a valid `knowledgebase-request` message.	`<error code=602>`"Unable to convert. Error: *errorMsg*"`</error>`
406	Not Acceptable - Unable to process request	The received evaluation request could not be evaluated, because of an exception.	`<error code=602>`"Unable to process request. Error: *errorMsg*"`</error>`
500	Internal Server Error - Package deployment failed	The package could not be deployed, because of an internal error.	`<error code=621>`"Error allocating resources for rule package *packageName*"`</error>`
400	Bad Request - Package	The deployment of the package failed, because of compilation errors.	`<error code=622>`"Deployment of rule package *packageName* failed due to rule compile errors"`</error>`
400	Bad Request - Package	The deployment of the package failed, because of an exception.	`<error code=623>` "Error in deploying rule package *packageName*. *ErrorMsg.*" `</error>`

If the content type is `application/json`, the body of the error message is formatted as follows:

```
{
    error:{
        code:6xx,
        description:"error message"
    }
}
```

Appendix

B DROOLS 5 Keywords

This appendix lists hard and soft keywords introduced in DROOLS 5. It contains the following sections:

DROOLS 5 Keywords

Drools 5 introduces the concept of hard and soft keywords.

Hard Keywords

Hard keywords are reserved—you cannot use any hard keyword when naming domain objects, properties, methods, functions and other elements that are used in the rule text.

The following list of hard keywords must be avoided as identifiers when writing rules:

- `true`
- `false`
- `null`

Soft Keywords

Soft keywords are just recognized in their context, enabling you to use these words in any other place if you wish, although Genesys recommends avoiding them if possible to prevent confusion. The list of soft keywords is:

- `lock-on-active`
- `date-effective`
- `date-expires`
- `no-loop`
- `auto-focus`
- `activation-group`
- `agenda-group`
- `ruleflow-group`

- `entry-point`
- `duration`
- `package`
- `import`
- `dialect`
- `salience`
- `enabled`
- `attributes`
- `rule`
- `extend`
- `when`
- `then`
- `template`
- `query`
- `declare`
- `function`
- `global`
- `eval`
- `not`
- `in`
- `or`
- `and`
- `exists`
- `forall`
- `accumulate`
- `collect`
- `from`
- `action`
- `reverse`
- `result`
- `end`
- `over`
- `init`

You can use these (hard and soft) words as part of a method name in camel case, like `notSomething()` or `accumulateSomething()` without any issues.

Escaping Hard Keywords

Although the three hard keywords above are unlikely to be used in your existing domain models, if you absolutely need to use them as identifiers instead of keywords, the DRL language provides the ability to escape hard keywords on rule text. To escape a word, simply enclose it in grave accents, like this:

```
Holiday( `true` == "yes" ) //
```

Please note that Drools will resolve that reference to the method:

```
Holiday.isTrue()
```

Appendix

C Example

This appendix provides an almost complete end-to-end sample use case: rule template (including support for building test scenarios from release 8.1.2), rule package, deploy and execute. It contains the following sections:

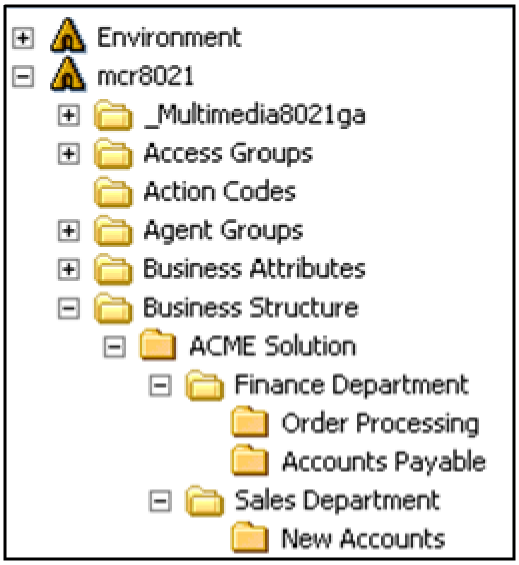
Use Case

We want to create a VXML self-service application for our company, ACME Corporation. Within that application, we will collect information from the customer that will allow us to determine the customer’s segment (that is, is the customer a Bronze, Silver, Gold, or Platinum customer), as well as the value of an order (in American dollars) that the customer has placed with our company. Based on the values for the customer segment and the order, we will use predefined business rules to determine whether to play a prompt to the customer that offers them a special promotion. In other words, the logic that will determine whether the special offer should be made to the customer will be defined within the business rules themselves, and not within the VXML application.

This example does not describe how the logic would be created in the VXML application to collect information from the customer, look up related information in a customer database (for example, to establish the value of the customer’s order), or play the prompt to the customer. It just demonstrates the use of business rules to supply the necessary information to the client application—in this case the VXML application—to allow the application to take the correct next step.

Business Structure

The business structure of our organization is defined under our tenant in Configuration Server. It consists of a single that is called "ACME Solution." Under this Solution there are two departments (Finance Department and Sales Department); under the Finance Department there are two processes (Order Processing and Accounts Payable; under the Sales Department there is a single process (New Accounts). Figure 20 shows the business structure as it appears in Configuration Manager. You can also manage the business structure in Genesys Administrator, although it does not appear as a hierarchical tree on the Administrator GUI.



```
[+] Environment
[-] mcr8021
    [+] _Multimedia8021ga
    [+] Access Groups
        Action Codes
    [+] Agent Groups
    [+] Business Attributes
    [-] Business Structure
        [-] ACME Solution
            [-] Finance Department
                    Order Processing
                    Accounts Payable
            [-] Sales Department
                    New Accounts
```

Figure 20: Business Structure

Rule Template

The rule template that is created for this example consists of two facts:

- `_GRS_Environment`
- `Customer`

The `_GRS_Environment` fact is a mandatory fact for all Genesys Rules Systems rule templates. It is used to establish two important fact properties:

- `businessContext_Level*`—Used in the request to the Rules Engine, to determine the node(s) of the business structure at which to evaluate rules
- `phase`—Used within the request to the Rules Engine, to determine which rules to evaluate. Each rule that you create in Genesys Rules System must have a rule phase defined. The list of rule phases can be modified by changing the values of the enumeration that is called `Phases`, in the rule

template. In this example, the phase that is selected is called `segmentation`, so we can assume that the values for the `Phases` enumeration contains at least one value called `segmentation`, and possibly others.

The `Customer` fact contains three properties that we will use in our business rule:

- `segment`
- `order`
- `offer`

Our rule template contains two conditions and one action, as well as the necessary parameters that are used within these conditions and actions. See Table 6, Table 7, and Table 8. Figure 21 on page 107 shows how the enumeration is configured.

Table 6: Rule Conditions

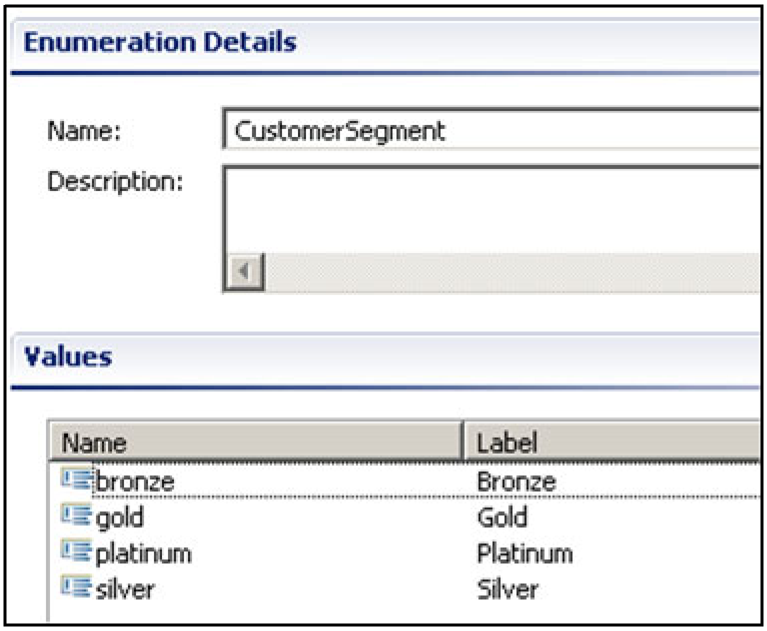
Name	Language Expression	Rule Language Mapping
Segment	Customer segment is {customerSegment}	`Customer(segment=='{customerSegment}')`
OrderValue	Order value is greater than {orderValue}	`$c:Customer(order>{orderValue})`

Table 7: Rule Actions

Name	Language Expression	Rule Language Mapping
SpecialOffer	Offer special promotion {specialOffer}	`$c.offer='{specialOffer}'`

Table 8: Rule Parameters

Name	Type	Comments
customerSegment	Enumeration	An enumeration must be created in the rule template that contains the values for Customer Segment from which the rules author will be able to select (for example, Bronze, Silver, and so on). Note that there are two properties that you must provide for each value of the enumeration: `Name` and `Label`. The `Label` is what will appear to the business rules author when the business rules author is using a rule condition or action that includes a parameter that references this enumeration. The `Name` is what is used in the request/response to/from the Rules Engine; therefore, case is important. For example, you may want to use uppercase for the labels of these enumeration values, and lowercase for the names. Figure 21 shows an example of how that might appear in the Genesys Rules Development Tool:
orderValue	Input Value (Numeric)	Optionally, you can supply upper and lower bounds for this parameter. If these are supplied in the template, the rules author will be constrained as to the values the rules author can provide in the rule condition that uses this parameter.
specialOffer	Input Value (Boolean)	Because the parameter type is `Boolean`, this will present a checkbox to the rules author when this parameter is used in the rule action.



Enumeration Details

Name: CustomerSegment

Description:

Values

Name	Label
bronze	Bronze
gold	Gold
platinum	Platinum
silver	Silver

Figure 21: Enumeration Details

Here are a few things to note about this rule template:

- Note that since the `orderValue` parameter is numeric, when it is used in a rule condition, there are no single quotation marks (‘’) surrounding it in the rule language mapping, whereas there are single quotation marks surrounding the `customerSegment` string parameter.
- A variable, `$c`, is declared as part of one of the rule conditions. The variable must be declared before it is used in another condition, or in an action. (Note that a variable may not be declared twice.) In this example, the variable is used in the rule action. An alternate way to declare the variable would be to have a generic rule condition such as the following:
  - Language Expression: `Customer exists`
  - Rule Language Mapping: `$c:Customer()`

  You would have to ensure that this rule condition is included in any rules that have other rule conditions (or a rule action) that use this variable.

  The reason that variables are used in this way is that you cannot set the value of a fact property directly, in the Drools language. Therefore, facts are updated through the use of variables.
- A “`.`”is used to access the fields on a fact. Use “`:`” when you want to create a variable in a condition. So, in the preceding example, the “`.`” is used in the rule language mapping for the action (`$c.offer` in the condition, `$c.offer` in the action).

Supporting Building Test Scenarios in Release 8.1.2

In order to support building test scenarios in release 8.1.2, the rule developer should provide a mapping between the parameters (which the rule author is familiar with) and the underlying fact model. In this way, when the rule author provides a sample value for say, “orderValue”, GRAT will know how to build the appropriate Fact object in order to run the test. In this case, it would create a Customer fact and set the `order` field to the specified value.

For this example, we will map the parameters to the fact model in the following way:

- `customerSegment -> Customer.segment`
- `orderValue -> Customer.order`
- `specialOffer -> Customer.offer`

In GRDT, right-click on each parameter and choose `Associate Property`. Then choose the appropriate Fact and field from the pop-up window (Figure 22).

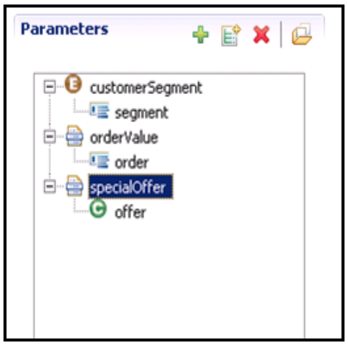


Figure 22: Mapping Parameters Window

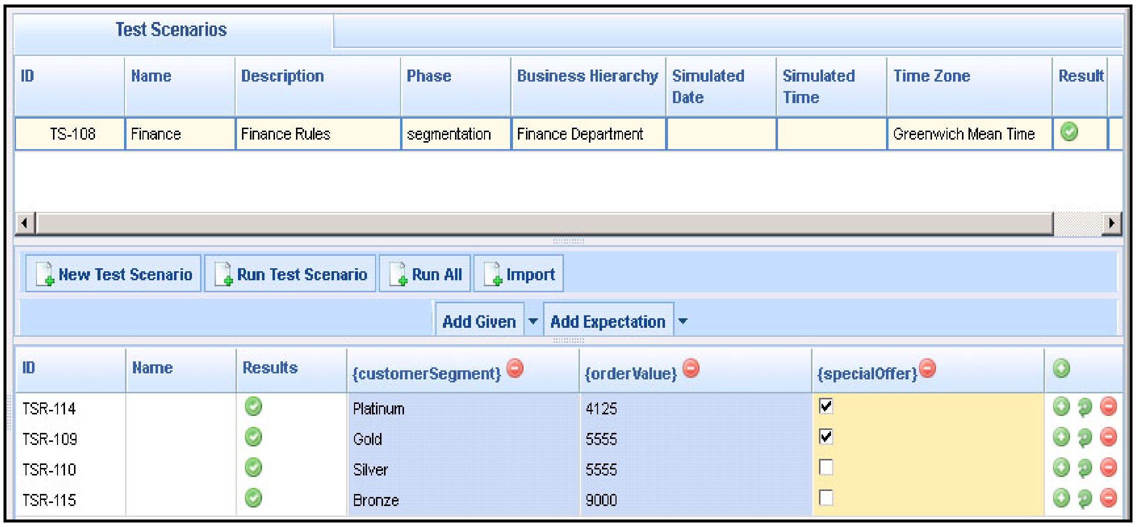
Navigate to the `Test Scenarios` tab and create a test scenario to test our decision table rule at the Finance Department node. Select test values `customerSegment` and `orderValue` from the `Add Given` drop-down. Then select `specialOffer` from the `Add Expectation` drop-down.

Now, insert rows of data. In these rows you can put some test values and also choose what your predicted or expected result should be.

When you click on the `Run Test Scenario` button, these test values will be passed into the rule package and the result will be compared to your expectations. If they match, you will see a green check mark in the `Results` column as shown in Figure 23 on page 109.

Note that we purposely passed in data that we predicted would return a positive result (for example, the customer gets the special offer) as well as a negative result (for example, the customer does not qualify).

These test scenarios are then saved and can be executed in the future when rules are added or modified.



Test Scenarios

ID	Name	Description	Phase	Business Hierarchy	Simulated Date	Simulated Time	Time Zone	Result
TS-108	Finance	Finance Rules	segmentation	Finance Department			Greenwich Mean Time	✓

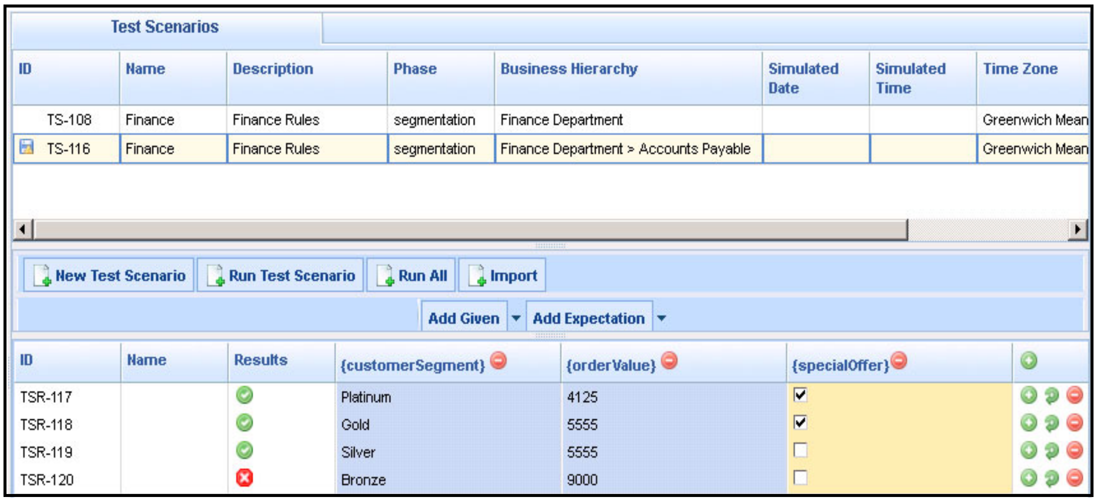
ID	Name	Results	{customerSegment}	{orderValue}	{specialOffer}	
TSR-114		✓	Platinum	4125	☑	
TSR-109		✓	Gold	5555	☑	
TSR-110		✓	Silver	5555	☐	
TSR-115		✓	Bronze	9000	☐	

Figure 23: Test Scenario Tab 1

Note, the 4th row of the table, shows our Bronze customer with an order value of 9000 NOT receiving a special offer. This is because the test was run against the `Finance Department` node of the hierarchy. In our example, we added a linear rule to the Accounts Payable department which addresses the Bronze customer.

We can now create a new test scenario which targets the Accounts Payable department and validates that, in this case, the Bronze customer gets an offer.

In our new test scenario (`TS-116`), we set the Business Hierarchy to the `Finance Department > Accounts Payable` department. We copy the same test data and when we run it, notice that the Bronze customer shows an unsuccessful result when our expectation is that they do NOT receive an offer (Figure 24).



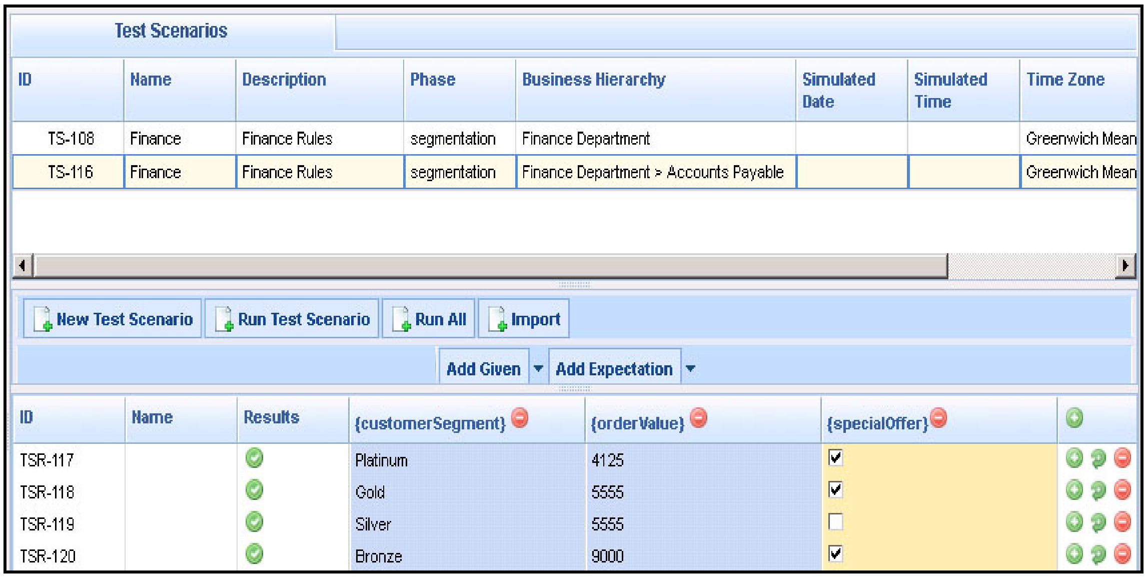
Test Scenarios

ID	Name	Description	Phase	Business Hierarchy	Simulated Date	Simulated Time	Time Zone
TS-108	Finance	Finance Rules	segmentation	Finance Department			Greenwich Mean
TS-116	Finance	Finance Rules	segmentation	Finance Department > Accounts Payable			Greenwich Mean

ID	Name	Results	{customerSegment}	{orderValue}	{specialOffer}	
TSR-117		✓	Platinum	4125	☑	
TSR-118		✓	Gold	5555	☑	
TSR-119		✓	Silver	5555	☐	
TSR-120		✗	Bronze	9000	☐	

Figure 24: Test Scenario Tab 2

We simply adjust the test scenario so that we now expect an offer for this customer by checking the `specialOffer` box. We now get a successful result when running the test.



ID	Name	Description	Phase	Business Hierarchy	Simulated Date	Simulated Time	Time Zone
TS-108	Finance	Finance Rules	segmentation	Finance Department			Greenwich Mean
TS-116	Finance	Finance Rules	segmentation	Finance Department > Accounts Payable			Greenwich Mean

ID	Name	Results	{customerSegment}	{orderValue}	{specialOffer}
TSR-117		✓	Platinum	4125	☑
TSR-118		✓	Gold	5555	☑
TSR-119		✓	Silver	5555	☐
TSR-120		✓	Bronze	9000	☑

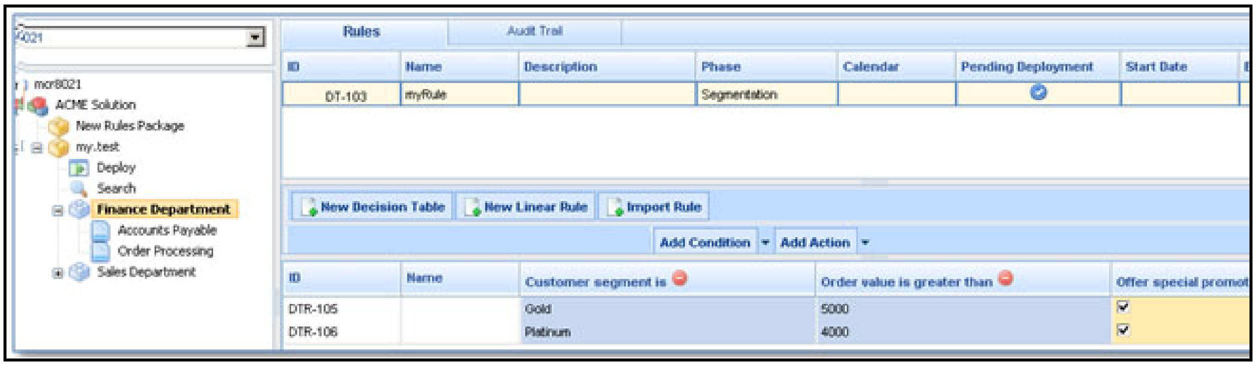
Figure 25: Test Scenario Tab 3

Rule Package

The rule package that is created for this example is called `my.test`. Three rules are defined within the package. Two of the rules are defined as two rows of a single Decision Table (see Figure 26), which is created at the `Finance Department` node of the business structure. The rule checks two conditions:

- The value of the `segment` property of the `Customer` fact
- The value of the `order` property of the `Customer` fact

If the conditions are all true, the rule will fire a single action, which is to set the value of the `offer` property of the `Customer` fact to `1`.

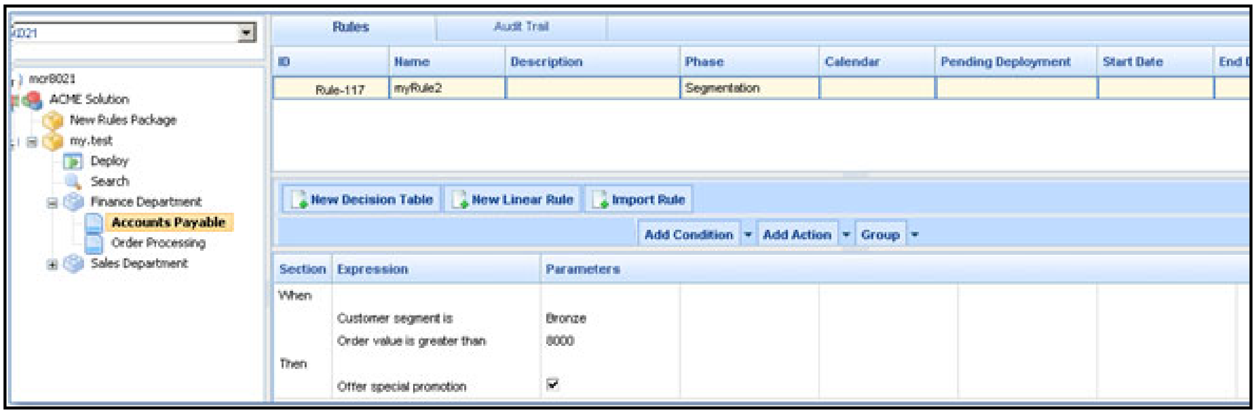


ID	Name	Description	Phase	Calendar	Pending Deployment	Start Date
DT-103	myRule		Segmentation		✓	

ID	Name	Customer segment is	Order value is greater than	Offer special promot
DTR-105		Gold	5000	☑
DTR-106		Platinum	4000	☑

Figure 26: Decision Table

The third rule is defined as a linear rule (see Figure 27) and has been created at the `Accounts Payable` node of the business structure. This rule checks the same conditions—and has the same action—as the rules that are defined at the `Finance Department` node. Normally, you might expect this rule to be a third row in the earlier decision table, at the `Finance Department` node. It is included here only to demonstrate how the rules at different nodes in the business structure can be evaluated.



ID	Name	Description	Phase	Calendar	Pending Deployment	Start Date	End D
Rule-117	myRule2		Segmentation				

Section	Expression	Parameters
When		
	Customer segment is	Bronze
	Order value is greater than	8000
Then		
	Offer special promotion	✓

Figure 27: Linear Rule

The my.test rule package is deployed to the Genesys Rules Engine or, in release 8.1.2, an application cluster.

Note: When two or more conditions are listed, there is an implied "*and*" between them. So, this rule is saying that "*when* the customer segment is bronze *and* the order value is greater than 8000, *then* offer special promotion".

The rule author can also choose other logical operators, such as *or*, *not*, *and not*, and so on.

Rule Evaluation

We want to call the GRE from our client (VXML) application. For the rule to be evaluated properly, we will have to populate the fact properties of the `_GRS_Environment` and `Customer` facts correctly.

To test this rule evaluation, you can use a REST client, such as the free Firefox REST Client add-on, or you can test the rule by using Composer's Business

Rule Block, which has a built in Test feature that provides sample values to the rule and evaluates the results.

Note: In most cases, you will use Genesys Composer to build applications that will invoke the GRE. However, to simplify rule testing, it might be more convenient to use a REST client in the manner that is described here.

The request to the Rules Engine will be a POST request. The URL we will use to make the POST request will be constructed as follows:

```
http://[server:port]/genesys-rules-engine/knowledgebase/[package]
```

where:

- `server` is the IP address or host name of the application server on which the rules engine is running
- `port` is the listening port of the application server. For example, 8080 is the default Tomcat port.
- `package` is the name of the rule package to evaluate. In this example it is `my.test`.

So, the URL might look like this:

```
http://myserver:8080/genesys-rules-engine/knowledgebase/my.test
```

We have to populate the request body with the request, in XML format. In the request body we specify the two fact classes, both of which are prefixed with the package name—for example, `my.test._GRS_Environment` and `my.test.Customer`—respectively.

For the `_GRS_Environment fact`, we have to provide values for the fact properties phase and `businessContext_Level*`. Note that your request can include multiple values for the `businessContext_Level*` fact property, depending on the node(s) of the business structure at which you want the Rules Engine to evaluate rules. In our case, let us assume that in this request, we want the Rules Engine to evaluate the rules at both the `Finance Department` level and the `Accounts Payable` level. In this case, in our request we will populate fact properties that specify both of these levels `(businessContext_Level1` and `businessContext_Level2`). Alternatively, if we omitted `businessContext_Level2` from the request, we could ask the Rules Engine to evaluate only the rules at the `Finance Department` level, which is `businessContext_Level1`.

Note also that if you had any rules configured at the “global” level (which are configured for the rule package itself by selecting the name of the package in the navigation tree, and then selecting the `Rules` tab), they will always be evaluated for every request, without having to specify anything explicitly in the `_GRS_Environment` fact property.

The other `_GRS_Environment` fact property that we must populate in the request is the `phase`. In our example, all rules were written for the `segmentation` phase.

For the `Customer` fact, we must provide values for the fact properties segment and order. We can provide whatever values we want, in order to test the results of the rule evaluation. Note that the value that you provide for the segment fact property is case-sensitive, as is the value for the phase fact property. See the description of the `customerSegment` enumeration in Table 7, “Rule Actions,” on page 105, for more details.

The following is an example of the request body:

```
<knowledgebase-request>
  <inOutFacts>

    <named-fact>
      <id>env</id>
      <fact class="my.test._GRS_Environment">
        <phase>segmentation</phase>
        <businessContext_Level1>Finance Department</businessContext_Level1>
        <businessContext_Level2>Accounts Payable</businessContext_Level2>
      </fact>
    </named-fact>

    <named-fact>
      <id>customer</id>
      <fact class="my.test.Customer">
        <segment>gold</segment>
        <order>6345.32</order>
      </fact>
    </named-fact>

  </inOutFacts>
</knowledgebase-request>
```

Based on our rule configuration, we would expect that the Rule Engine would return a value of `1` for the offer property of the `Caller` fact, indicating that under these conditions (customer is Gold and the customer’s order value is greater than $5,000.00), we want to offer them a special promotion. This is because the parameter (`specialOffer`) that is being used in the rule action is a `Boolean` type.

In this case, the response body will look like the following:

```
<knowledgebase-response>
  <inOutFacts>
    <named-fact>
      <id>env</id>
      <fact class="my.test._GRS_Environment">
        <businessContext__Level2>Accounts Payable</businessContext__Level2>
        <businessContext__Level1>Finance Department</businessContext__Level1>
        <phase>segmentation</phase>
      </fact>
    </named-fact>
    <named-fact>
      <id>customer</id>
      <fact class="my.test.Customer">
        <order>6345.32</order>
        <segment>gold</segment>
        <offer>1</offer>
      </fact>
    </named-fact>
  </inOutFacts>
  <executionResult>
    <rulesApplied>
      <string>Row 1 DT-103 myRule</string>
    </rulesApplied>
  </executionResult>
</knowledgebase-response>
```

If you pass in values in your request that the Rules Engine will not evaluate to `true`, based on all of the rules that you have deployed, no value for the `offer` fact property will be returned in the result. For example, if you set the value of `order` to `2345.32`, the response body will look like the following:

```
<knowledgebase-response>
  <inOutFacts>
    <named-fact>
      <id>env</id>
      <fact class="my.test._GRS_Environment">
        <businessContext__Level2>Accounts Payable</businessContext__Level2>
        <businessContext__Level1>Finance Department</businessContext__Level1>
        <phase>segmentation</phase>
      </fact>
    </named-fact>
    <named-fact>
      <id>customer</id>
      <fact class="my.test.Customer">
        <order>2345.32</order>
        <segment>gold</segment>
      </fact>
    </named-fact>
  </inOutFacts>
  <executionResult>
    <rulesApplied>
    </rulesApplied>
  </executionResult>
</knowledgebase-response>
```

Note that this is not the same as the value of `offer` being `0`. In this example, because all of the conditions in the rules were not met (evaluated as `true` by the Rules Engine), the action was not fired. Thus, `offer` has no value populated in the result. If you wanted the value of `offer` to be set to `0`, you would have to have a rule that included a rule action whereby the value of `offer` was unchecked (remember that it is a `Boolean` parameter so it is either checked or unchecked by the rules author). If all of the conditions of such a rule were evaluated as `true` by the Rules Engine, the result would set `offer` to `0`.

If you want the response to include the `offer` fact property, with no value, it must be included in the request (even if no value is provided). In this case the `my.test.Customer` fact class would look like the following in the request:

```
<named-fact>
  <id>customer</id>
  <fact class="my.test.Customer">
    <segment>gold</segment>
    <order>2345.32</order>
    <offer></offer>
  </fact>
</named-fact>
```

And the response body would include the following section:

```
<named-fact>
  <id>customer</id>
  <fact class="my.test.Customer">
    <order>2345.32</order>
    <segment>gold</segment>
    <offer></offer>
  </fact>
</named-fact>
```

You can also try populating the request with values that will be relevant to the rule at the `Accounts Payable` level of the business structure—for example, `segment = bronze` and `order = 9345.33`. In this case, you should also see the value of `order` set to `1` in the response body.

Supplements

Related Documentation Resources

The following resources provide additional information that is relevant to this software. Consult these additional resources as necessary.

Genesys Rules System

- *Genesys Rules Authoring Tool Help*, which is a guide to the GRAT user interface.
- *Genesys Rules Development Tool Help*, which is a guide to the GRDT user interface.

Genesys Composer

- *Composer Deployment Guide*, which describes how to install Composer and perform post-installation configuration.
- *Composer Help*, which provides integrated help information about using Composer to develop voice and routing applications.
- *Composer Routing Application User's Guide*, which provides details about creating routing applications.

Genesys

- *Genesys Technical Publications Glossary*, which ships on the Genesys Documentation Library DVD and which provides a comprehensive list of the Genesys and computer-telephony integration (CTI) terminology and acronyms used in this document.
- *Genesys Migration Guide*, which ships on the Genesys Documentation Library DVD, and which provides documented migration strategies for Genesys product releases. Contact Genesys Technical Support for more information.

- 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 Supported Operating Environment Reference Manual*
- *Genesys Supported Media Interfaces Reference Manual*

Consult these additional resources as necessary:

- *Genesys Hardware Sizing Guide*, which provides information about Genesys hardware sizing guidelines.
- *Genesys Interoperability Guide*, which provides information on the compatibility of Genesys products with various Configuration Layer Environments; Interoperability of Reporting Templates and Solutions; and Gplus Adapters Interoperability.
- *Genesys Licensing Guide*, which introduces you to the concepts, terminology, and procedures relevant to the Genesys licensing system.
- *Genesys Database Sizing Estimator 8.0 Worksheets*, which provides a range of expected database sizes for various Genesys products.

For additional system-wide planning tools and information, see the release-specific listings of System Level Documents on the Genesys Technical Support website. These documents are accessible from the `system level documents by release` tab in the Knowledge Base `Browse Documents` Section.

Genesys product documentation is available on the:

- Genesys Technical Support website at `http://genesyslab.com/support`.
- Genesys Documentation wiki at `http://docs.genesyslab.com/`.
- Genesys Documentation Library DVD and/or the Developer Documentation CD, which you can order by e-mail from Genesys Order Management at `orderman@genesyslab.com`.

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:

80fr_ref_06-2008_v8.0.001.00

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

Screen Captures Used in This Document

Screen captures from the product graphical user interface (GUI), as used in this document, may sometimes contain minor spelling, capitalization, or grammatical errors. 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.

Type Styles

Table 9 describes and illustrates the type conventions that are used in this document.

Table 9: Type Styles

Type Style	Used For	Examples
Italic	• Document titles • Emphasis • Definitions of (or first references to) unfamiliar terms • Mathematical variables Also used to indicate placeholder text within code samples or commands, in the special case where angle brackets are a required part of the syntax (see the note about angle brackets on page 120).	Please consult the *Genesys Migration Guide* for more information. Do *not* use this value for this option. A *customary and usual* practice is one that is widely accepted and used within a particular industry or profession. The formula, $x+1 = 7$ where x stands for . . .

Table 9: Type Styles (Continued)

Type Style	Used For	Examples
Monospace font (Looks like `teletype` or `typewriter text`)	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. • Code samples. Also used for any text that users must manually enter during a configuration or installation procedure, or on a command line.	Select the `Show variables on screen` check box. In the `Operand` text box, enter your formula. Click `OK` to exit the `Properties` dialog box. T-Server distributes the error messages 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. Enter `exit` on the command line.
Square brackets ([ ])	A particular parameter or value that is optional within a logical argument, a command, or some programming syntax. That is, the presence of the parameter or value is not required to resolve the argument, command, or block of code. The user decides whether to include this optional information.	`smcp_server -host [/flags]`
Angle brackets (< >)	A placeholder for a value that the user must specify. This might be a DN or a port number specific to your enterprise. **Note:** In some cases, angle brackets are required characters in code syntax (for example, in XML schemas). In these cases, italic text is used for placeholder values.	`smcp_server -host <confighost>`

Index