GENESYS™

Genesys Rules System Deployment Guide

Genesys Rules System 8.1.4

1/2/2022

Table of Contents

Welcome

Genesys Rules System Deployment

Welcome to the Genesys Rules System 8.1.x deployment pages. This document describes how to install and configure Genesys Rules System.

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

Overview

Overview

Genesys Rules System Architecture

Genesys Composer

New Features by Release

Migration to 8.1.4

Preparing for Installation

Preparing for Installation

Configuring the Rules Repository Database

Installing Genesys Rules Engine

Deploying GRE in Genesys Administrator

Creating the GRE Application object in Configuration Manager

Installing the GRE Component

GRE Configuration Options

Installing Genesys Rules Authoring Tool

Deploying GRAT in Genesys Administrator

Installing the GRAT Component

GRAT Configuration Options

Values for DAP Connection Parameters

Creating Application Clusters &

Objects

Creating an Application Cluster in Configuration Manager

Creating an Application Cluster in Genesys Administrator

Creating the GRAT Application Object in Configuration Manager

Further Configuration

Deploying WAR Files

Configuring WebSphere 8.5

Logging Options

Installing Genesys Rules Development Tool

Installing Genesys Rules Development Tool

Post Installation

Testing the Installation

Installing GRS on Unix Platforms

High Availability Support

Troubleshooting

Configuration Considerations

Configuration Diagrams

GRS Configuration Options

Locating the GDRT Version Number

The log4j.properties File

Localization

Installing Language Packs

Uninstalling Language Packs

About Rules and Rule Templates

Rule Templates and Rules

Rule Life Cycle

Rule Templates

Examples of Rule Development

Rule Language Mapping

Rules and Rule Packages

About Business Structure

About Business Structure

Configuring the Business Structure

iCFD Business Structures

Role-Based Access Control

Role-Based Access Control

Role Permissions

User Logins

Business Hierarchy

Role Task Permissions

Template Script Objects

REST API

Rule Execution

Changes in 8.1.2

Error Handling

DROOLS 5 Keywords

DROOLS 5 Keywords

Overview

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

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 defined by a business analyst. These rules are evaluated in a Rules Engine based upon requests received from client applications. Some Genesys applications that can use the Rules Engine include:

- VXML applications executed by the Genesys Voice Platform

- SCXML applications executed by the Genesys Orchestration Server

- Genesys intelligent Workload Distribution (iWD) business processes executed by Genesys Interaction Server and Universal Routing Server.

The following figure illustrates the flow of a simple rule.
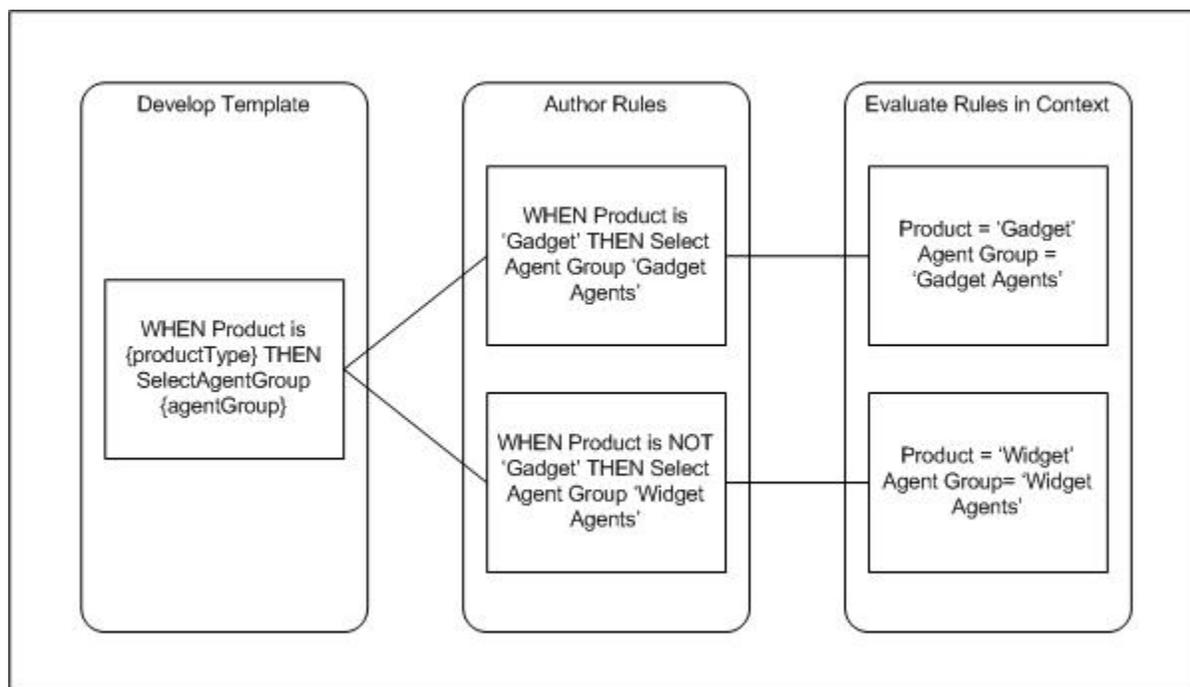
Simple Rule Flow

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.

> **Important**
> Support for hard-coded iCFD templates was removed in release 8.1.2.

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, Service Level Agreements, and other attributes of tasks.

Starting with release 8.1.0, 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 Genesys Web Engagement (GWE). 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.

- Which applications the rule package can be deployed to.

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 published to it, and users can select these user-defined template types or define new ones.

Genesys Rules System Architecture

The schematic below shows the main Genesys Rules System (GRS) building blocks, their interrelationships, and the external components that are involved.
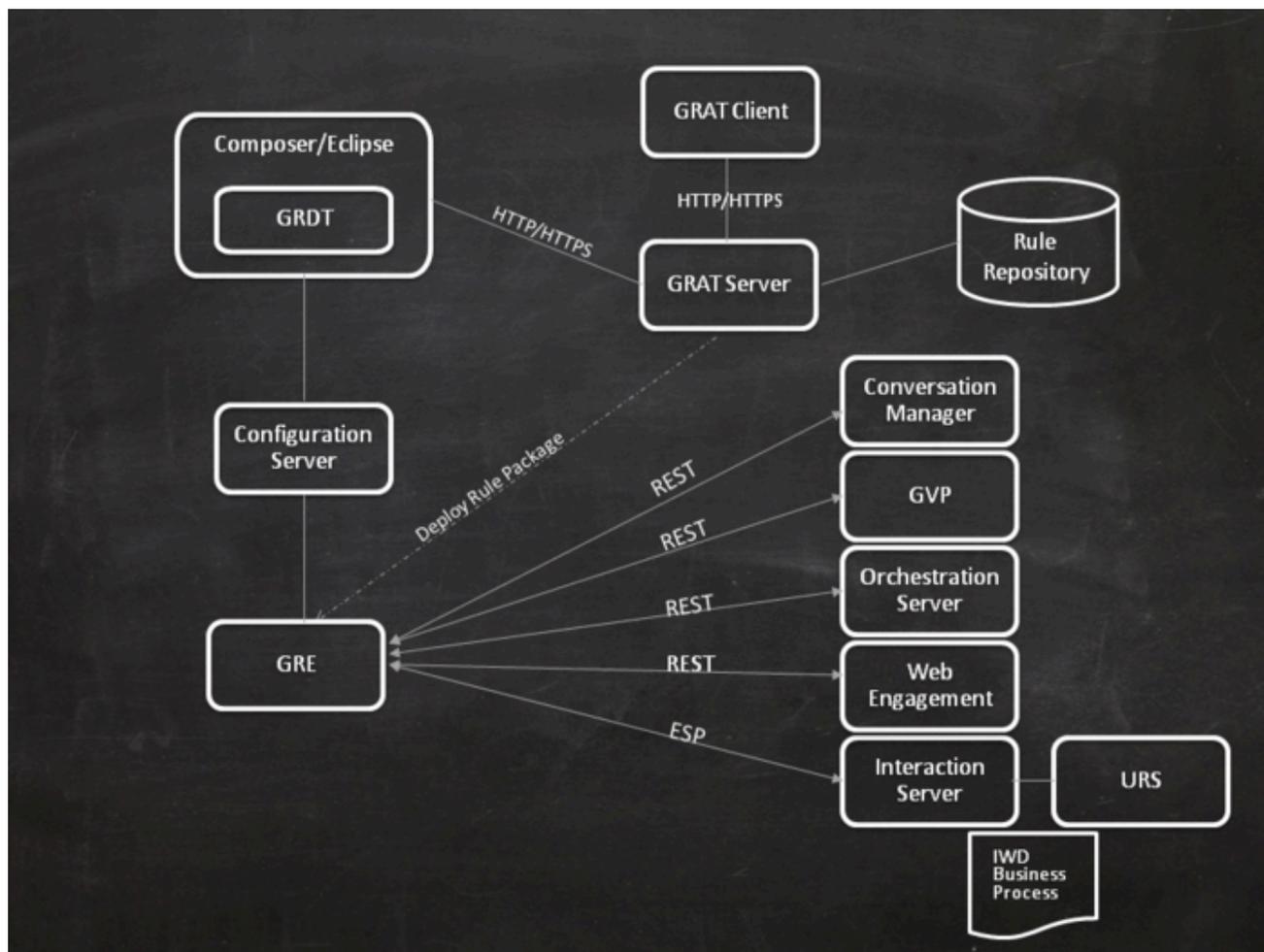
There are three main aspects of GRS:

- The Genesys Rules Development Tool (GRDT) 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) is a browser-based application 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) 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.

Genesys Composer's Business Rule Block

You can use Genesys Composer to create applications for rules evaluation requests. Composer 8.1.0 or higher is required, because that is the version in which the Business Rule block was introduced. Refer to the Business Rule Common Block help topic documentation for more information about how to use the Business Rule block.

New Features by Release

New in Release 8.1.4

- Configuration of the GRAT database connection is no longer part of the GRAT IP installation, but has moved to Genesys Configuration Server to be more consistent with other Genesys products. Configuration of the database connection credentials is now done via a Database Access Point (DAP). This change improves security and removes the need to re-install GRAT after a database password change. See Configuring the Rules Repository Database, Deploying GRAT in Genesys Administrator and Installing the GRAT Component.

- A new Check Permissions feature enabling users to view the role permissions granted to them by administrators for tenants and templates. See the GRAT online Help for details.

- Improvements to rule package deployment—In this release, the deployment of rule packages, which previously could take a long time and carried the risk of session timeouts, browser refreshes or other interruptions, has been made more responsive and quicker. Additionally, the process now recognizes any duplication in a deployment that is pending or scheduled (such as the same rule package, or the same snapshot, or to the same destination server/cluster). See the GRAT online Help for more details.

- Support for Google Chrome browser

Migration to 8.1.4

From 8.1.3

1. In 8.1.4, the repository database configuration has been moved to a Database Access Point (DAP). Create and configure a DAP and add it as a connection to the GRAT application object. You will no longer be prompted for database configuration information during the installation process.

   See the following procedures for details of how to create a DAP:

   - **Creating the Rules Repository Database for Configuration Server**
   - **Creating the Rules Repository Database for Genesys Administrator**

- Undeploy the 8.1.3 `.war` file from your application server.

  ### Important
  Do not just copy the 8.1.4 `.war` file over the 8.1.3 `.war` file in your application server directory. Genesys recommends undeploying the previous file first and leting the application server clean up its files, then deploying the new `.war` file.

- Deploy the 8.1.4 `.war` file to your application server.

  ### Important
  If you have a very large repository database, it may take several minutes the first time you deploy the 8.1.4 .war file, as GRAT must rebuild the index cache. Once this process is complete, the user will be able to log in to the system.

- Log into 8.1.4 Genesys Rules Authoring Server.

From 8.1.2 and 8.1.1

1. From the 8.1.1 or 8.1.2 Genesys Rules Authoring Server:

   a. Click on each tenant and export the templates associated with that tenant as an XML file.

   b. Click on each rule package that you wish to migrate and export as an XML file.

3. Create a new database for GRAT 8.1.4 (leaving the old one in place).

4. Install 8.1.4 Genesys Rules System.

5. Start 8.1.4 Genesys Rules Authoring Server. This creates the tables inside the new database.

6. Log into 8.1.4 Genesys Rules Authoring Server.

7. For each tenant, import the template XML file (from step 1a).

8. For each tenant, and under each solution, click on New Rule Package and import the corresponding rule package XML file (click the Auto-save option).

9. Redeploy each rule package to the corresponding 8.1.4 Genesys Rules Engine(s).

10. Optionally, from 8.1.4 Genesys Rules Deployment Tool, you may import the templates from the 8.1.4 Genesys Rules Authoring Server.

See the Genesys Rules Authoring Tool Help for explicit steps for exporting and importing templates and rules.

Important

Running an 8.1.4 Rules Authoring Server against an 8.1.2 repository can result in a corrupted repository that will no longer be useable by any version of the Rules Authoring Server.

Upgrading within the Same Release Family

Purpose

To apply correctly to GRAT and GRE a hot fix or maintenance release within the same release family (for example, from 8.1.300.00 to 8.1.301.07, or from 8.1.400.00 to 8.1.400.07).

Procedure

1. Install the new installation package. In 8.1.3 you will need to re-configure the database configuration information during installation. In 8.1.4, this is not necessary as it is retrieved from the DAP.

2. Undeploy the original `.war` file from your application server.

> ### Important
>
> Do not just copy the new `.war` file over the old `.war` file in your application server directory. Genesys recommends undeploying first and letting the application server clean up its files, then deploying the new `.war` file.

3. Deploy the new `.war` file to your application server.

> ### Important
>
> If you have a very large repository database, it may take several minutes the first time you deploy the new `.war` file, as GRAT must re-build the index cache. Once this process is complete the user will be able to log in to the system.

4. Log into the Genesys Rules Authoring server.

Installing Genesys Rules System (GRS)

Task Summary

The following table outlines the task flow for installation of GRS 8.1.4. The procedures in this table provide instructions about installing GRS components on Microsoft Windows. For information about how to install on UNIX-based operating systems, refer to Installing Genesys Rules System on UNIX Platforms.

Objective	Related Procedures and Actions
1. Prepare for installation and review prerequisites.	• Ensure that your environment meets the prerequisites that are outlined in Preparing for installation. • Ensure that the required CD is available.
2. Create the database for the Rules Repository.	• Configuring the Rules Repository database
3. Install the Genesys Rules Engine	• Genesys Administrator:Deploying the Genesys Rules Engine in Genesys Administrator • Configuration Manager:Creating the Genesys Rules Engine Application object in Configuration Manager • Installing the Genesys Rules Engine
4. Install the Genesys Rules Authoring Tool	• Genesys Administrator:Deploying the Genesys Rules Authoring Tool in Genesys Administrator • Genesys Administrator:Creating an Application Cluster in Genesys Administrator • Configuration Manager:Creating the Genesys Rules Authoring Tool Application objects in Configuration Manager • Configuration Manager: Creating an Application Cluster in Configuration Manager • Installing the Genesys Rules Authoring Tool

Objective	Related Procedures and Actions
5. Deploy the genesys-rules-authoring.war and genesys-rules-engine.war files to your application server.	Deploying the .war files
6. Install the Genesys Rules Development Tool	Installing the Genesys Rules Development Tool
7. Define your business structure	See About Business Structure.
8. Test the installation	Testing the Installation
9. Review the Troubleshooting section for configuration tips and considerations	See Troubleshooting.
10. Redeploy all standard rule packages that have been previously deployed only to pre-8.1.3 Genesys Rules Engines.	In release 8.1.3, the rules engine was updated from Drools 5.1 to 5.5. The rules engine (up to and including release 8.1.2) writes serialized objects to file. These serialized objects are no longer loadable due to the Drools upgrade. To avoid future upgrade issues, rules engines later than 8.1.3 will maintain the rules package in its DRL form.

Preparing for Installation

The topics in this section enable you to prepare for installing the GRS software distribution artifacts.

- **Summary of Installation Steps**
- **Creating the Rules Repository Database with Configuration Manager**
- **Creating the Rules Repository Database with Genesys Administrator**

Configuring the Rules Repository Database for Configuration Manager

This procedure creates and configures in Configuration Manager the database that will be used as the Rules Repository.

Most database distributions include the JDBC connector that is needed; if this is not the case, you must download it from the vendor's site. Genesys does not provide the JDBC connector. Genesys Rules System 8.1 can use either Java 6 or 7.

Once the configuration below is complete, the same database configuration will be used whenever GRAT is installed or updated. There will be no need to specify it again. Also, if any of the database information changes (for example, DB Server location, DB name, DB user or DB password), users can simply update the DAP and restart GRAT.

Prerequisites

Either Oracle or Microsoft SQL or DB2.

Procedure

1. Create a new database using the normal DBMS procedures for the type you are working with. This is the database that will be populated by GRAT.

2. In Configuration Manager, right-click the `Environment` > `Applications` folder and select `New` > `Application`. This opens the `Browse` dialog box that lists the available application templates.

3. In the `Browse` dialog box, select the DAP template file, and click `OK`. This opens the `Properties` dialog box for the new DAP `Application` object.

4. On the `General` tab:

a. Enter a name for the DAP. A DAP can have the same name as the database itself. However, it is recommended that you make their names unique if you are using multiple access points for the same database.

b. Do not enter anything in the DB Server field.

c. Select the JDBC Connection check box. This will disable the DBMS Name field on the DB Info tab.

d. Ensure that the State Enabled check box is checked.

5. On the DB Info tab:
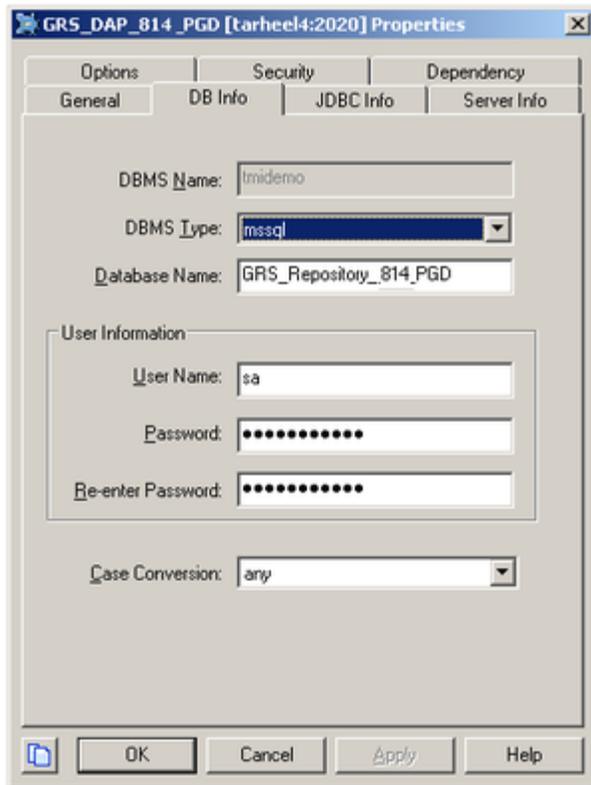
a. Enter the DBMS type, database name, username, and password.

b. Set Case Conversion to any, and leave the DBMS Name field clear.

6. On the JDBC Info tab, set the following values:

    a. Role field—Main

    b. Debug field—Unknown

    c. Query Timeout field—0 (zero)

4. On the Server Info tab, enter the host name and port number.
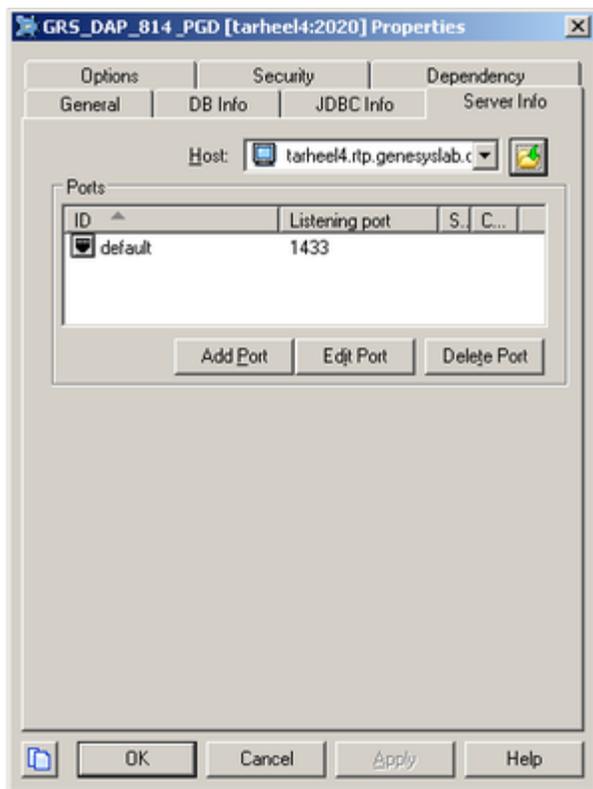
5. Add this newly created DAP to the Connections tab of the GRAT Application object.

Configuring the Rules Repository Database using Genesys Administrator

This procedure creates and configures the GRAT Rules Respository database using Genesys Administrator.

Most database distributions include the JDBC connector that is needed. If this is not the case, you must download it from the vendor's site. Genesys does not provide the JDBC connector. Genesys Rules System 8.1 can use either Java 6 or 7.

Once the configuration below is complete, the same database configuration will be used whenever GRAT is installed or updated. There will be no need to specify it again. Also, if any of the database information changes (for example, DB Server location, DB name, DB user or DB password), users can simply update the DAP and restart GRAT.

Prerequisites

RDBMS; Oracle or Microsoft SQL or DB2.

Procedure

1. Create a new database using the normal DBMS procedures for the type you are working with. This is the database that will be populated by GRAT. Create a DB user/password that will have full access to this new database.

2. In Genesys Administrator, navigate to the `Environment` > `Applications` > GRS folder.

3. Create a new application to be the new Database Access Point.
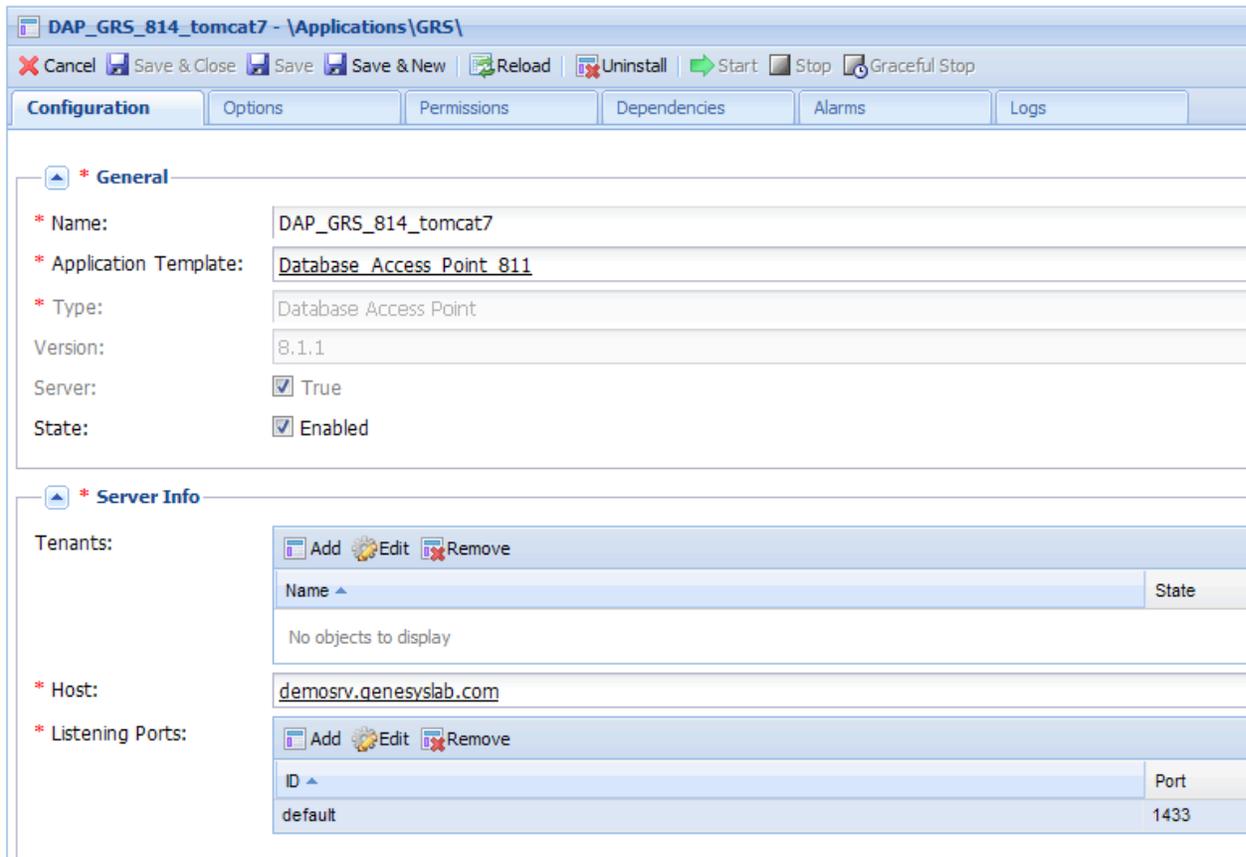
4. Select the Application Template type Database_Access_Point_811 (or later).

5. Ensure that the State: Enabled check box is checked.

6. In the Server Info panel, enter values for the Host and the Listening Ports of the DBMS server.

7. In the DB Info panel, enter JDBC as the Connection Type. This will disable the DBMS Name field.

8. Set the `Role` field to value `Main`.

9. Set the Debug field to value 0 (zero).

10. Set the `Query Timeout` field to value 0 (zero).

11. Select the value in the DBMS Type field (MSSQL, DB2 or Oracle).

12. Enter the name of the database created in Step 1. For Oracle, this is the "service name".

13. Enter the database username and password created in Step 1.

14. Ensure that the `Case Conversion` field has the value any.

15. Save your changes.

16. Add this newly created DAP as a Connection on the GRAT Application object. When GRAT initializes, it will use the information in this DAP to connect to the repository database.

Values for DAP Connection Parameters

During the installation of the GRAT, you will be prompted to enter various connection parameters for the database you are using as the Rules Repository (created in Creating the Rules Repository Database).

Default Values

The table below provides default values for the three supported database types (MSSQL, Oracle, and DB2). You should consult your database vendor's documentation for specific information. The last column in the table lists the JDBC drivers that you must copy to the `lib` directory of your application server.

Database Type	Example Connector Class	Default Database URL	JDBC Driver to be Copied
MSSQL	com.microsoft.sqlserver. jdbc.SQLServerDriver	jdbc:sqlserver://{host}:{port};databaseName={database_na...	sqljdbc4.jar
Oracle	oracle.jdbc.driver.OracleDriver	jdbc:oracle:thin:@//{host}:{port}/{SID}	ojdbc14.jar ojdbc14_g.jar ojdbc14dms_g.jar ojdbc6.jar ojdbc6_g.jar
DB2	com.ibm.db2.jcc.DB2Driver	jdbc:db2://{host}:{port}/{database_name}	db2jcc.jar db2jcc_license_cu.jar

Overriding Default Values on the Options Tab

More advanced users can use the DAPs `Options` tab to override the default values mentioned above; for example, if a database vendor makes changes to the JDBC driver class, or if additional options need to be specified on the DB URL.
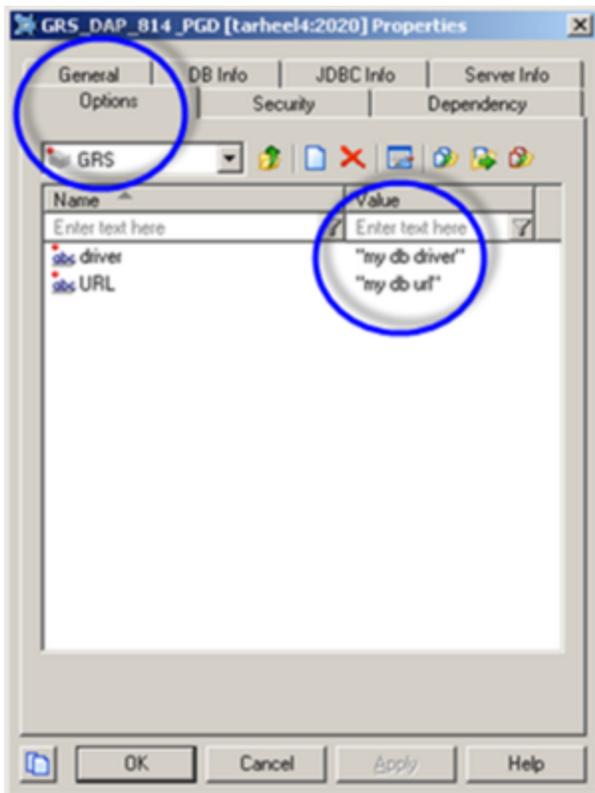
If the GRS section is present, the value of any options specified here overrides the defaults generated by GRAT.

Procedure

1. On the Options tab, create a section called GRS.

2. Use the URL field to override the URL value generated by GRAT.

3. Use the Driver field to override the default driver value generated by GRAT.

Installing GRE

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:

1. Create the application.
2. Run the installation package manually.

Deploying GRE in Genesys Administrator

Prerequisites

To install GRE on Configuration Servers 8.1.0 or later, Genesys Administrator 8.1.0 or later is required.

Procedure

1. Import the installation package into Genesys Administrator.

Import the installation package into Genesys Administrator

1. On the Deployment tab of Genesys Administrator, select Import.
2. Select Installation CD-ROM.
3. Click Next.
4. Browse to the MediaInfo.xml file on the CD or the CD image location on the network (the path must be in UNC format).
5. Click Next.
6. To import the installation package, select GRE for your operating system as well as the appropriate type in the list:
   - 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.
7. Select Next to start the import.
8. Click Finish when the import is complete.

2. Install the GRE IP.

Install the GRE IP

1. Select the `Deployment` tab in Genesys Administrator. The list of installation packages will now display GRE.

2. Right-click and select `Install Package` for the IP for your operating system and type.

3. 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

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

    h. Connection delay time in seconds

    i. Reconnect Attempts.

> ### Important
> 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.

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

11. 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.

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

3. Configure the Rules Engine application.

Configure the Rules Engine application

1. 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`.

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

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

- 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 GRE and GRAT. 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.

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

Settings in GRE

Description	Valid values	Default value	Takes effect
deployed-rules-directory (added to application template in 8.1.3)			
Specifies the directory in which to keep the working copy of deployed rule packages. When a		/GCTI/logs/GRS_Engine (8.1.3 onwards)	After restart

package is deployed, a copy of the deployed package is placed here. When the rules engine is restarted, all packages 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.

If you choose to change the default value, ensure that the path exists and that the

application server can write to the specified directory.			
max-number-rule-executions			
The maximum number of rules to be executed during a request. This is used to detect unwanted recursion when `sequential-mode` is false. If this maximum is reached an error is reported. May be set to `-1` to denote no maximum.	Any positive integer or -1	10,000	Next rules execution
sequential-mode			
Indicates whether to run the rules engine in sequential mode. In sequential mode, after the initial data set, no more data can be inserted or modified. This allows for the rules engine to operate in a simplified way.	true/false	false	On rules deployment
verify-deployer-address			

Indicates whether to verify the TCP address of the application deploying rules to be that of an associated Genesys Rules Authoring Tool.	true/false	true	Immediately
esp-worker-threads (new in 8.1.2)			
Specifies the maximum number of worker threads available when using the ESP interface to execute rules.	Any positive integer	5	Immediately
load-packages-on-start (new in 8.1.4)			
Indicates whether to load deployed rule packages at application start up. If packages are not loaded at startup (value=`false`), then a package is loaded on its first execution request.	true/false	true	Immediately
json-hierarchical-driver (new in 8.1.4)			
With value `true`, the `JsonHierarchicalStreamDriver`	true/false	false	Immediately

class is used to serialize JSON responses. With value `false`, the `JettisonMappedXmlDriver` class is used. The Jettison driver is unaware of the original data type and will try to detect numerical values and omit the quotes, whereas the `JsonHierarchicalStreamDriver` will maintain the data type.			

- Save your changes.

Next Steps

- Deploy the `genesys-rules-engine.war` file to your application server. See Deploying the .WAR files.

Creating the GRE Application Object in Configuration Manager

Procedure

To create the application object for GRE in Configuration Manager, do the following:

1. Import the GRE application template into Configuration Manager.

Import the GRE Application Template into Configuration Manager

1. In Configuration Manager, navigate to the `Application Templates` folder.
2. Right-click the `Application Templates` folder, and select `Import Application Template`.
3. 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_Engine.apd.`.
   - For Management Framework 8.1 and earlier, select `Genesys_Rules_Engine_Generic_Server.apd.`.
- Click OK to save the template.

2. Configure the Rules Engine application.

Configure the GRE Application in Configuration Manager

1. Right-click the `Applications` folder and select `New` > `Application`.
2. Select the template that you imported in the previous procedure.
3. On the `General` tab, enter a name for the application, such as `Rules_Engine`.
4. On the `Tenants` tab, add the Tenants that will be available to the Rules Engine.

5. On the `Server Info` tab, select the Host on which the application will be installed.

6. Add a default listening port.

7. 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`.

8. 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.

9. On the `Options` tab, configure options. Logging options are as follows:

log

Description	Valid values	Default value	Takes effect
all			
Specifies the outputs to which an application sends all log events. The log output types must be separated by a comma when more than one output is configured. For example: all = stdout, logfile	• `stdout`—Log events are sent to the Standard output (`stdout`). • `stderr`—Log events are sent to the Standard error output (`stderr`). • `network`—Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database. Setting the `all` log level option to the network output enables an application to send log events of the Standard, Interaction, and Trace levels to Message Server. Debug-level log	stdout	After restart

Description	Valid values	Default value	Takes effect
	events are neither sent to Message Server nor stored in the Log Database. • `memory`—Log events are sent to the memory output on the local disk. This is the safest output in terms of the application performance. • `[filename]`—Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.		
expire			
Determines how many log files will be kept on disk. If set, expire specifies the maximum number of log files kept on disk.	Any number	(blank)	After restart
segment			
Determines whether a log output written to file is split in multiple segments. If it is, segment	Any number that represents the log size in megabyte	(blank)	After restart

Description	Valid values	Default value	Takes effect
specifies the maximum size of each segment file.			
standard			
Specifies the outputs to which an application sends the log events of the Standard level. The log output types must be separated by a comma when more than one output is configured. For example: `standard = stderr, network`	• `stdout`—Log events are sent to the Standard output (`stdout`). • `stderr`—Log events are sent to the Standard error output (`stderr`). • `network`— Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database. • `memory`—Log events are sent to the memory output on the local disk. This is the safest output in terms of the application performance. • `[filename]`—Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.	stdout	After restart
trace (not in application template by default)			

Description	Valid values	Default value	Takes effect
Specifies the outputs to which an application sends the log events of the Trace level and higher (that is, log events of the Standard, Interaction, and Trace levels). The log outputs must be separated by a comma when more than one output is configured. For example: `trace = stderr, network`	• `stdout`—Log events are sent to the Standard output (`stdout`). • `stderr`—Log events are sent to the Standard error output (`stderr`). • `network`—Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database. • `memory`—Log events are sent to the memory output on the local disk. This is the safest output in terms of the application performance. • `[filename]`—Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.	stdout	After restart
verbose			
Determines whether a log output is created. If it is, specifies the minimum level of log events generated. The log events levels, starting with the	• `all`—All log events (that is, log events of the Standard, Trace, Interaction, and Debug levels) are generated. • `debug`—The same as all. • `trace`—Log events of the Trace level and higher (that is, log events of the	standard	After restart

Description	Valid values	Default value	Takes effect
highest priority level, are Standard, Interaction, Trace, and Debug.	Standard, Interaction, and Trace levels) are generated, but log events of the Debug level are not generated. • `interaction`—Log events of the Interaction level and higher (that is, log events of the Standard and Interaction levels) are generated, but log events of the Trace and Debug levels are not generated. • `standard` Log events of the Standard level are generated, but log events of the Interaction, Trace, and Debug levels are not generated. • none—No output is produced.		

10. Configure the options on the Settings tab as follows:

Settings in GRE

Description	Valid values	Default value	Takes effect
deployed-rules-directory (added to application template in 8.1.3)			
Specifies the directory in which to keep the working copy of deployed rule packages. When a package is		/GCTI/logs/GRS_Engine (8.1.3 onwards)	After restart

deployed, a copy of the deployed package is placed here. When the rules engine is restarted, all packages 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.

If you choose to change the default value, ensure that the

path exists and that the application server can write to the specified directory.			
max-number-rule-executions			
The maximum number of rules to be executed during a request. This is used to detect unwanted recursion when `sequential-mode` is false. If this maximum is reached an error is reported. May be set to `-1` to denote no maximum.	Any positive integer or -1	10,000	Next rules execution
sequential-mode			
Indicates whether to run the rules engine in sequential mode. In sequential mode, after the initial data set, no more data can be inserted or modified. This allows for the rules engine to operate in a	true/false	false	On rules deployment

simplified way.			
verify-deployer-address			
Indicates whether to verify the TCP address of the application deploying rules to be that of an associated Genesys Rules Authoring Tool.	true/false	true	Immediately
esp-worker-threads (new in 8.1.2)			
Specifies the maximum number of worker threads available when using the ESP interface to execute rules.	Any positive integer	5	Immediately
load-packages-on-start (new in 8.1.4)			
Indicates whether to load deployed rule packages at application start up. If packages are not loaded at startup (value= `false`), then a package is loaded on its first execution request.	true/false	true	Immediately
json-hierarchical-driver (new in 8.1.4)			

With value `true`, the `JsonHierarchicalStreamDriver` class is used to serialize JSON responses. With value `false`, the `JettisonMappedXmlDriver` class is used. The Jettison driver is unaware of the original data type and will try to detect numerical values and omit the quotes, whereas the `JsonHierarchicalStreamDriver` will maintain the data type.	true/false	false	Immediately

11. Save your changes.

Installing the GRE Component

Purpose

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

Prerequisites

- Creating the GRE Application Object in Configuration Manager

Start

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 Creating the GRE Application Object in Configuration Manager. 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.

> **Important**
>
> 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

> **Important**
> In release 8.1.1 only, GRE may fail with `ClassNotFoundException` errors during rule execution. This problem does not occur in release 8.1.2 or 8.1.3. To prevent it, set the Java runtime option `-Dmvel2.disable.jit` option to `true`. Then restart the application server.

Next Steps

- Deploy the `genesys-rules-engine.war` file to your application server. See Deploying the .WAR files.

Installing GRAT

Genesys Administrator

Genesys recommends that you configure the GRAT by using Genesys Administrator. If you use Genesys Administrator, you can deploy the installation package from within Genesys Administrator.

Configuration Manager

You can configure the GRAT by using Configuration Manager if you are using an older version of Configuration Server, prior to 8.0.2, where Roles are not supported. If you use Configuration Manager, you will have to:

1. Create the applications.
2. Run the setup program manually.

Non-English Environments

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"/>
```

Deploying GRAT in Genesys Administrator

Purpose

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

Prerequisites

To install GRAT on Configuration Servers 8.1.1 or later, Genesys Administrator 8.1.1 or later is required.

Procedure

1. Import the GRAT IP into Genesys Administrator.

Import the GRAT IP into Genesys Administrator

Start

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`.

- Select Next to start the import.

- Click Finish when the import is complete.

2. Install the GRAT IP.

Install the GRAT IP

1. Select the Deployment tab in Genesys Administrator. The list of installation packages will now show the Genesys Rules Authoring Tool.

2. Right-click and select Install Package for the IP for your operating system and type.

3. 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.

> ### Important
> After the specified number of attempts to connect to the primary Configuration Server all fail, 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.

> **Important**
>
> 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.

11. 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.

3. Configure the GRAT application.

Configure the GRAT Application

To configure the GRAT server application:

1. 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 `Connections` tab, add a connection to the Database Access Point.

    d. On the `Options` tab, configure log options.

log

Description	Valid values	Default value	Takes effect
all			
Specifies the outputs to which an application sends all log events. The log output types must be	• `stdout`—Log events are sent to the Standard output (`stdout`). • `stderr`—Log events are sent to the Standard error output (`stderr`). • `network`—Log events are sent to	stdout	After restart

Description	Valid values	Default value	Takes effect
separated by a comma when more than one output is configured. For example: all = stdout, logfile	Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database. Setting the all log level option to the network output enables an application to send log events of the Standard, Interaction, and Trace levels to Message Server. Debug-level log events are neither sent to Message Server nor stored in the Log Database. • memory—Log events are sent to the memory output on the local disk. This is the safest output in terms of the application performance. • [filename]—Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.		
expire			
Determines how many log files will be kept on disk. If set, expire specifies	Any number	(blank)	After restart

Description	Valid values	Default value	Takes effect
the maximum number of log files kept on disk.			
segment			
Determines whether a log output written to file is split in multiple segments. If it is, segment specifies the maximum size of each segment file.	Any number that represents the log size in megabyte	(blank)	After restart
standard			
Specifies the outputs to which an application sends the log events of the Standard level. The log output types must be separated by a comma when more than one output is configured. For	• stdout—Log events are sent to the Standard output (stdout). • stderr—Log events are sent to the Standard error output (stderr). • network— Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database. • memory—Log	stdout	After restart

Description	Valid values	Default value	Takes effect
example: `standard = stderr, network`	events are sent to the memory output on the local disk. This is the safest output in terms of the application performance. • `[filename]`—Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.		
trace (not in application template by default)			
Specifies the outputs to which an application sends the log events of the Trace level and higher (that is, log events of the Standard, Interaction, and Trace levels). The log outputs must be separated by a comma when more than one output is configured. For example: `trace = stderr, network`	• `stdout`—Log events are sent to the Standard output (`stdout`). • `stderr`—Log events are sent to the Standard error output (`stderr`). • `network`—Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database. • `memory`—Log events are sent to the memory output on the local disk. This is the safest output in terms of the application performance. • `[filename]`—Log events are stored in a file with the specified name. If	stdout	After restart

Description	Valid values	Default value	Takes effect
	a path is not specified, the file is created in the application's working directory.		
verbose			
Determines whether a log output is created. If it is, specifies the minimum level of log events generated. The log events levels, starting with the highest priority level, are Standard, Interaction, Trace, and Debug.	• `all`—All log events (that is, log events of the Standard, Trace, Interaction, and Debug levels) are generated. • `debug`—The same as all. • `trace`—Log events of the Trace level and higher (that is, log events of the Standard, Interaction, and Trace levels) are generated, but log events of the Debug level are not generated. • `interaction`—Log events of the Interaction level and higher (that is, log events of the Standard and Interaction levels) are generated, but log events of the Trace and Debug levels are not generated. • `standard` Log events of the Standard level are generated, but log events of the Interaction, Trace, and Debug levels are not generated. • `none`—No output is produced.	standard	After restart

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.

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

Settings in GRAT

Description	Valid values	Default value	Takes effect
group-by-level (group rules by business level)			
There are three levels of rules: global, department, and process. With value `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,	true/false	true	Immediately

`level0` rules are executed first. Updates from this first pass then influence the department (`level1`) rules which are executed in the second pass. Updates from this second pass then influence any process rules (`level2`), which are executed in a third pass.

Note: The GRE option `sequential-mode` must be false when `group-by-level` is set to `true`.

When `group-by-level` is set to `false`, all rules are executed in a single pass. Changes made by a rule do not influence which other rules are executed (unless a Drools "update" or "insert" command is used).

CEP functionality • Genesys Web Engagement's CEP functionality strips out the rule attribute that indicates which level a rule is associated with. So, the setting of the `group-by-level` has no influence on rule execution.			
max-connections			
Specifies the maximum number of different users that may be connected to the server. Multiple connections from the same user ID are only counted once.	Any positive integer	99	After GRAT restart
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)	Any positive integer	30	Immediately

defined by the application server applies. If the value is less than or equal to 0, the session will not time out.			
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	Any positive integer	1	Immediately

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.			
strict-mode			
This option controls whether or not the rules authoring tool enables *strict* mode in the DROOLS rule compiler. Strict mode will cause the compiler to catch common mistakes when the rule author attempts to validate or save a rule.	true/false	true	Immediately
verify-deployer-address			
Indicates whether to verify the TCP address of the application deploying rules to be that of an associated Genesys Rules Engine.	true/false	true	Immediately
display-n-template-versions (new in 8.1.3)			
Specifies the maximum number of versions to	Minimum value 1	3	Immediately

display for any published template.			
deploy-response-timeout (new in 8.1.3 - not in application template by default)			
Specifies the timeout (in seconds) applied to the deployment of a rule package.	Any positive integer	300	Immediately
require-checkin-comment (new in 8.1.3)			
Specifies whether users must add a check-in comment when committing changes to rules. These comments show up when viewing package history. If the value is set to false (default), users can save changes to rules without specifying a comment.	true/false	false	Immediately
force-snapshot-on-deployment (new in 8.1.3)			
Specifies whether users can deploy only a package snapshot. If the value is true, users can only deploy a package snapshot. If false (default), users can deploy either the LATEST package or a	true/false	false	Immediately

snapshot.			
encoding (not in application template by default)			
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).			After GRAT restart
clear-repository-cache (new in 8.1.4)			
The GRAT server builds and maintains a cache of the rules repository database (for example, index files, and so on), and stores this on the file system under `WEB-INF/classes/`	true/false	false	After GRAT (re-)start

repository. The cache improves performance when accessing frequently used rules, calendars, and so on. However, this cache must stay synchronized with the rules repository database.

Normally, if GRAT is restarted, it re-uses the existing cache, which is synchronized with the rules repository database. In this case, the `clear-repository-option` should be set to `false` (default).

However, if you are configuring a second GRAT for warm standby (see High Availability Support), this option should be set to `true` for both the primary and the standby instances of GRAT. Since

either GRAT could be brought online in the event of a failure, this option forces GRAT always to rebuild the cache and re-synchronize it with the rules repository database. Setting this option to `true` can delay the startup of GRAT, since the cache must be rebuilt, but it ensures that it is properly synchronized with the rules repository database.			

6. 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. Make sure that the "System" access group has Read, Create, and Change permissions to the Scripts folder, and that you have applied these changes recursively.) 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.

7. Give the application Read permission for all roles, access groups and persons needed for GRAT.

8. 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.

Creating an Application Cluster in Configuration Manager

You can use a Configuration Server or Genesys Administrator application of type `Application Cluster` 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

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.

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.

Creating an Application Cluster in Genesys Administrator

Purpose

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

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`.

3. Go to `Provisioning > Environment > Applications`. 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 as a `Listening Port` 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.

Creating the GRAT Application Object 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.

Procedure

1. Import the GRAT application template for the server.

Import the GRAT application template for the server

To import the application template that is to be used for the server application:

1. In Configuration Manager, navigate to the `Application Templates` folder.
2. Right-click the `Application Templates` folder, and select `Import Application Template`.
3. 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`.
- Click OK to save the template.

2. Import the GRAT application template for the client.

Import the GRAT Application Template for the client

To import the template that is to be used for the client application:

1. Right-click the `Application Templates` folder.
2. Select `Import Application Template`.
3. Browse to the `templates` folder of the installation CD.
4. Select `Genesys_Rules_Authoring_Generic_Client_810.apd`.
5. Click OK to save the template.

3. Configure the server application.

Configure the GRAT Application

To configure the GRAT server application:

1. 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 `Connections` tab, add a connection to the Database Access Point.
    d. On the `Options` tab, configure log options.

log

Description	Valid values	Default value	Takes effect
all			
Specifies the outputs to which an application sends all log events. The log output types must be	• `stdout`—Log events are sent to the Standard output (`stdout`). • `stderr`—Log events are sent to the Standard error output (`stderr`). • `network`—Log events are sent to Message Server,	stdout	After restart

Description	Valid values	Default value	Takes effect
separated by a comma when more than one output is configured. For example: all = stdout, logfile	which can reside anywhere on the network. Message Server stores the log events in the Log Database. Setting the `all` log level option to the network output enables an application to send log events of the Standard, Interaction, and Trace levels to Message Server. Debug-level log events are neither sent to Message Server nor stored in the Log Database. • `memory`—Log events are sent to the memory output on the local disk. This is the safest output in terms of the application performance. • `[filename]`—Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.		
expire			
Determines how many log files will be kept on disk. If set, expire specifies the maximum	Any number	(blank)	After restart

Description	Valid values	Default value	Takes effect
number of log files kept on disk.			
segment			
Determines whether a log output written to file is split in multiple segments. If it is, segment specifies the maximum size of each segment file.	Any number that represents the log size in megabyte	(blank)	After restart
standard			
Specifies the outputs to which an application sends the log events of the Standard level. The log output types must be separated by a comma when more than one output is configured. For example:	• `stdout`—Log events are sent to the Standard output (`stdout`). • `stderr`—Log events are sent to the Standard error output (`stderr`). • `network`— Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database. • `memory`—Log events are sent to	stdout	After restart

Description	Valid values	Default value	Takes effect
`standard = stderr, network`	the memory output on the local disk. This is the safest output in terms of the application performance. • `[filename]`—Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.		
trace (not in application template by default)			
Specifies the outputs to which an application sends the log events of the Trace level and higher (that is, log events of the Standard, Interaction, and Trace levels). The log outputs must be separated by a comma when more than one output is configured. For example: `trace = stderr, network`	• `stdout`—Log events are sent to the Standard output (`stdout`). • `stderr`—Log events are sent to the Standard error output (`stderr`). • `network`—Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database. • `memory`—Log events are sent to the memory output on the local disk. This is the safest output in terms of the application performance. • `[filename]`—Log events are stored in a file with the specified name. If a path is not	stdout	After restart

Description	Valid values	Default value	Takes effect
	specified, the file is created in the application's working directory.		
verbose			
Determines whether a log output is created. If it is, specifies the minimum level of log events generated. The log events levels, starting with the highest priority level, are Standard, Interaction, Trace, and Debug.	• `all`—All log events (that is, log events of the Standard, Trace, Interaction, and Debug levels) are generated. • `debug`—The same as all. • `trace`—Log events of the Trace level and higher (that is, log events of the Standard, Interaction, and Trace levels) are generated, but log events of the Debug level are not generated. • `interaction`—Log events of the Interaction level and higher (that is, log events of the Standard and Interaction levels) are generated, but log events of the Trace and Debug levels are not generated. • `standard` Log events of the Standard level are generated, but log events of the Interaction, Trace, and Debug levels are not generated. • `none`—No output is produced.	standard	After restart

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.

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

Settings in GRAT

Description	Valid values	Default value	Takes effect
group-by-level (group rules by business level)			
There are three levels of rules: global, department, and process. With value `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,	true/false	true	Immediately

level0 rules are executed first. Updates from this first pass then influence the department (level1) rules which are executed in the second pass. Updates from this second pass then influence any process rules (level2), which are executed in a third pass.

Note: The GRE option `sequential-mode` must be false when `group-by-level` is set to `true`.

When `group-by-level` is set to `false`, all rules are executed in a single pass. Changes made by a rule do not influence which other rules are executed (unless a Drools "update" or "insert" command is used).

CEP functionality • Genesys Web Engagement's CEP functionality strips out the rule attribute that indicates which level a rule is associated with. So, the setting of the `group-by-level` has no influence on rule execution.			
max-connections			
Specifies the maximum number of different users that may be connected to the server. Multiple connections from the same user ID are only counted once.	Any positive integer	99	After GRAT restart
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)	Any positive integer	30	Immediately

defined by the application server applies. If the value is less than or equal to 0, the session will not time out.			
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	Any positive integer	1	Immediately

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.			
strict-mode			
This option controls whether or not the rules authoring tool enables *strict* mode in the DROOLS rule compiler. Strict mode will cause the compiler to catch common mistakes when the rule author attempts to validate or save a rule.	true/false	true	Immediately
verify-deployer-address			
Indicates whether to verify the TCP address of the application deploying rules to be that of an associated Genesys Rules Engine.	true/false	true	Immediately
display-n-template-versions (new in 8.1.3)			
Specifies the maximum number of versions to	Minimum value 1	3	Immediately

display for any published template.			
deploy-response-timeout (new in 8.1.3 - not in application template by default)			
Specifies the timeout (in seconds) applied to the deployment of a rule package.	Any positive integer	300	Immediately
require-checkin-comment (new in 8.1.3)			
Specifies whether users must add a check-in comment when committing changes to rules. These comments show up when viewing package history. If the value is set to false (default), users can save changes to rules without specifying a comment.	true/false	false	Immediately
force-snapshot-on-deployment (new in 8.1.3)			
Specifies whether users can deploy only a package snapshot. If the value is true, users can only deploy a package snapshot. If false (default), users can deploy either the LATEST package or a	true/false	false	Immediately

snapshot.			
encoding (not in application template by default)			
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).			After GRAT restart
clear-repository-cache (new in 8.1.4)			
The GRAT server builds and maintains a cache of the rules repository database (for example, index files, and so on), and stores this on the file system under `WEB-INF/ classes/`	true/false	false	After GRAT (re-)start

repository. The cache improves performance when accessing frequently used rules, calendars, and so on. However, this cache must stay synchronized with the rules repository database.

Normally, if GRAT is restarted, it re-uses the existing cache, which is synchronized with the rules repository database. In this case, the `clear-repository-` option should be set to `false` (default).

However, if you are configuring a second GRAT for warm standby (see High Availability Support), this option should be set to `true` for both the primary and the standby instances of GRAT. Since

| | either GRAT could be brought online in the event of a failure, this option forces GRAT always to rebuild the cache and re-synchronize it with the rules repository database. Setting this option to `true` can delay the startup of GRAT, since the cache must be rebuilt, but it ensures that it is properly synchronized with the rules repository database. | | |

6. 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. Make sure that the "System" access group has Read, Create, and Change permissions to the Scripts folder, and that you have applied these changes recursively.) 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.

7. Give the application Read permission for all roles, access groups and persons needed for GRAT.

8. 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.

4. Configure the client application.

Configure the Client Application

To configure the client application:

1. Right-click the `Applications` folder.
2. Select `New` > `Application`.
3. Select the `Genesys_Rules_Authoring_Generic_Client` template.
4. On the `General` tab, enter a name for the application, such as `Rules_Authoring_Client`.
5. Click Save.

Next Steps

- Installing the GRAT Component

Installing the GRAT Component

Purpose

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

Prerequisites

- Configuring the Rules Repository Database
- Creating the GRAT Application Objects in Configuration Manager

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 Creating the GRAT Application Objects in Configuration Manager. 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 Creating the GRAT Application Objects in Configuration Manager, specify the name of the rules authoring *client* application and click `Next`.
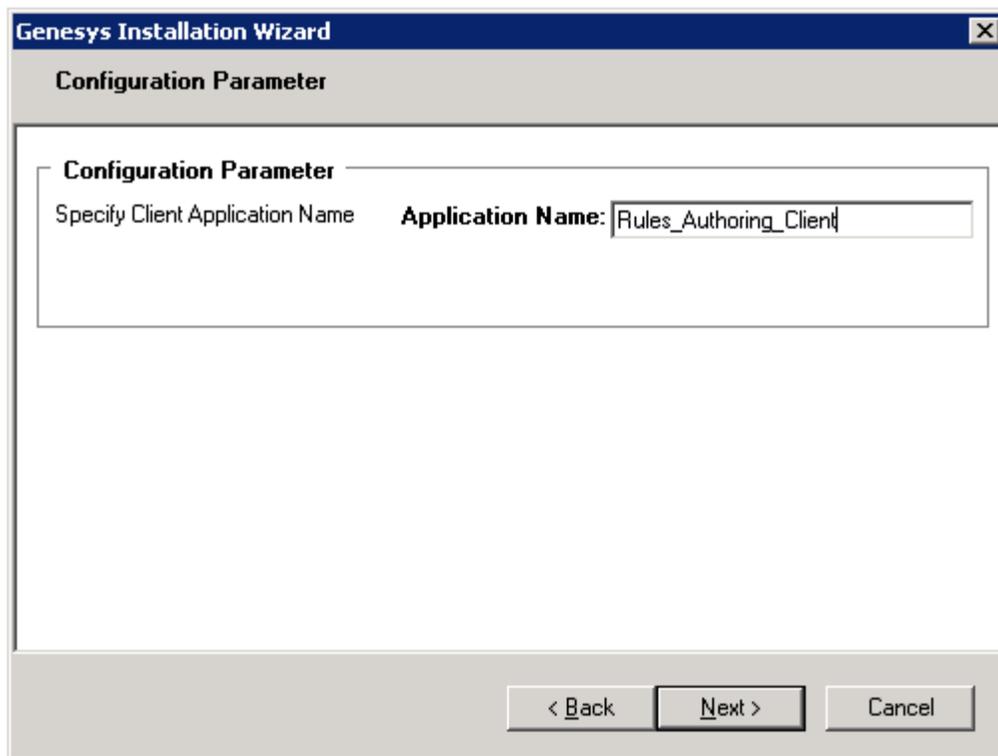
Specify the Rules Authoring Client Application Name

10. Select `Application Server Type`. Click `Next`.

11. Click `Install`.

12. Click `Finish`.

Next Steps

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

Deploying .WAR Files

The genesys-rules-authoring.war and genesys-rules-engine.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.

The .war files can be found in the destination folder that you specified when you installed the IPs.

- If you are using Tomcat, copy the files and paste them into the Tomcat webapps folder.

- If you are using WebSphere, deploy the .war files by using WebSphere Administrative Console.

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

Genesys recommends the following minimum JVM memory settings for your application server to ensure GRS components have enough memory for operation. This may need to be adjusted based on your configuration and depending upon any other applications deployed to your application server.

```
-XX:MaxPermSize=256m
-Xms256m
-Xmx1024m
```

Configuring WebSphere 8.5

Prerequisites

From release 8.1.3 of GRS, WebSphere 8.5 requires some once-only additional configuration to enable GRAT to deploy successfully to GRE.

Procedure

1. Extract `httpclient-4.1.1.jar` and `httpcore-4.1.jar` from the `WEB-INF/lib` directory of `genesys-rules-engine.war` and store them in:

   `${WAS_INSTALL_ROOT}\optionalLibraries`

2. Configure these two JAR files as Isolated Shared Libraries.

   a. From the WS Admin console select `Environment->SharedLibraries->New`

   b. Set the name to `sharedStuff`

   c. Set the classpath to:

      `${WAS_INSTALL_ROOT}/optionalLibraries/httpclient-4.1.1.jar`

      and

      `${WAS_INSTALL_ROOT}/optionalLibraries/httpcore-4.1.jar`

   d. Check the `Use an isolated class loader for this shared library` check box. Click Apply and Save.

3. Navigate to `Enterprise Applications->genesys-rules-engine->Shared library references` and add the `sharedStuff` shared library reference to the web module.

Installing the GRDT Component

Online Installation

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 version 3.5.0 or higher (but version 4 is not supported). 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.

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.

> **Important**
>
> 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 Role-Based Access Control for more information about roles.

> **Important**
>
> 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.

- 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 Installing the GRAT Component.

    - `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.

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

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

> **Important**
>
> If you are working with Genesys Technical Support, you will need to supply the exact version of the GRDT you are using. Refer to Locating the GRDT Version Number for information about how to find the version number.

Offline Installation

For environments where internet access is not available, copy the entire Composer directory to a 'sandbox' where internet is available, then install GRDT and the required dependencies. Once GRDT is working as expected copy the entire directory structure back to the production machine.

Next Steps

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

Testing the Installation

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

See Configuring a User for the GRAT to verify that the user has the correct permissions.

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). These are the same host and port that you entered in Installing the GRDT Component. The default name is `genesys-rules-authoring`, 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.

Installing GRS On Unix Platforms

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

Procedure

To install the GRE or the GRAT on UNIX systems:

1. Create the Application objects in Configuration Manager or Genesys Administrator. Please refer to the following topics:

   - Creating the Genesys Rules Engine Application Object in Configuration Manager

   - Deploying GRE in Genesys Administrator

2. Locate and run the `install.sh` scripts for each component (found in their respective directories on the CD).

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.
```

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 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.
```

High Availability Support

GRE

The Genesys Rules Engine (GRE) can be set up in a cluster in order to provide a highly available configuration. GRE is considered a critical path application because the execution of rules depends upon at least one node in the system being available. Since GRE is stateless, each rule execution request can be dispatched to any node in the cluster, and should a node fail, another node could execute the request.

The load balancer can be set up to dispatch requests to each GRE node at random, or in a round-robin fashion. There is no need to configure "session stickiness" as there are no sessions to maintain between rule execution requests.

See the GRS High Availability video: .

GRAT

Unlike GRE, only one Genesys Rules Authoring Tool (GRAT) instance can be connected to a particular rules repository database at a time. GRAT is not considered a critical path application because it only handles the creation, editing and deployment of rules. If GRAT should fail, rule execution continues uninterrupted. Only rule editing becomes unavailable.

GRAT can be set up in a cold standby configuration. A standby GRAT can be installed as a mirror image on a separate machine and be configured to use the same configuration management application, same HTTP ports, and so on. Should the primary GRAT fail (hardware failure, network), the standby GRAT can be brought online quickly to restore service. Both the primary and standby GRATs can be connected to the same repository database; however, they should not be connected simultaneously. The rule author would have to log in again and resume their activity.

When configuring a standby GRAT, use option `clear-repository-cache=true` for both the primary

and backup GRAT instances. Setting this option to `true` can delay the startup of GRAT, since the cache must be rebuilt each time, but it ensures that it is properly synchronized with the rules repository database.

GRS Configuration Options

log

Description	Valid values	Default value	Takes effect
all			
Specifies the outputs to which an application sends all log events. The log output types must be separated by a comma when more than one output is configured. For example: all = stdout, logfile	• `stdout`—Log events are sent to the Standard output (`stdout`). • `stderr`—Log events are sent to the Standard error output (`stderr`). • `network`—Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database. Setting the `all` log level option to the network output enables an application to send log events of the Standard, Interaction, and Trace levels to Message Server. Debug-level log events are neither sent to Message Server nor stored in the Log Database. • `memory`—Log events are sent to the memory output on the local disk. This is the safest output in terms of the application performance.	stdout	After restart

Description	Valid values	Default value	Takes effect
	• `[filename]`—Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.		
expire			
Determines how many log files will be kept on disk. If set, expire specifies the maximum number of log files kept on disk.	Any number	(blank)	After restart
segment			
Determines whether a log output written to file is split in multiple segments. If it is, segment specifies the maximum size of each segment file.	Any number that represents the log size in megabyte	(blank)	After restart
standard			
Specifies the outputs to which an application sends the log events of the Standard level. The log output types must be separated by a comma when more than one output is configured. For example: `standard = stderr, network`	• `stdout`—Log events are sent to the Standard output (`stdout`). • `stderr`—Log events are sent to the Standard error output (`stderr`). • `network`— Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database. • `memory`—Log events are sent to the memory output on the local disk. This is	stdout	After restart

Description	Valid values	Default value	Takes effect
	the safest output in terms of the application performance. • `[filename]`—Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.		
trace (not in application template by default)			
Specifies the outputs to which an application sends the log events of the Trace level and higher (that is, log events of the Standard, Interaction, and Trace levels). The log outputs must be separated by a comma when more than one output is configured. For example: `trace = stderr, network`	• `stdout`—Log events are sent to the Standard output (`stdout`). • `stderr`—Log events are sent to the Standard error output (`stderr`). • `network`—Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database. • `memory`—Log events are sent to the memory output on the local disk. This is the safest output in terms of the application performance. • `[filename]`—Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.	stdout	After restart
verbose			

Description	Valid values	Default value	Takes effect
Determines whether a log output is created. If it is, specifies the minimum level of log events generated. The log events levels, starting with the highest priority level, are Standard, Interaction, Trace, and Debug.	• `all`—All log events (that is, log events of the Standard, Trace, Interaction, and Debug levels) are generated. • `debug`—The same as all. • `trace`—Log events of the Trace level and higher (that is, log events of the Standard, Interaction, and Trace levels) are generated, but log events of the Debug level are not generated. • `interaction`—Log events of the Interaction level and higher (that is, log events of the Standard and Interaction levels) are generated, but log events of the Trace and Debug levels are not generated. • `standard` Log events of the Standard level are generated, but log events of the Interaction, Trace, and Debug levels are not generated. • `none`—No output is produced.	standard	After restart

Settings in GRAT

Description	Valid values	Default value	Takes effect
group-by-level (group rules by business level)			

There are three levels of rules: global, department, and process. With value `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. Updates from this first pass then influence the department (`level1`) rules which are executed in the second pass. Updates from this second pass then influence any process rules (`level2`), which are executed in a third pass. **Note:** The GRE option `sequential-mode` must be false when `group-by-level` is set to `true`. When `group-by-level` is set to `false`, all rules are executed in a single pass. Changes made by a rule do not influence which other rules are executed (unless a Drools "update" or "insert" command is used). *CEP functionality* • Genesys Web Engagement's CEP functionality strips out the rule attribute that indicates which level a rule is associated with. So, the setting of the `group-by-level` has no influence on rule execution.	true/false	true	Immediately
max-connections			
Specifies the maximum number of different users that	Any positive integer	99	After GRAT restart

may be connected to the server. Multiple connections from the same user ID are only counted once.			
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 time out.	Any positive integer	30	Immediately
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.	Any positive integer	1	Immediately
strict-mode			
This option controls whether or not the rules authoring tool enables *strict* mode in the DROOLS rule compiler. Strict mode will cause the compiler to catch common mistakes when the rule author attempts to validate or save a rule.	true/false	true	Immediately
verify-deployer-address			
Indicates whether to verify the TCP address of the application deploying rules to be that of	true/false	true	Immediately

an associated Genesys Rules Engine.			
display-n-template-versions (new in 8.1.3)			
Specifies the maximum number of versions to display for any published template.	Minimum value 1	3	Immediately
deploy-response-timeout (new in 8.1.3 - not in application template by default)			
Specifies the timeout (in seconds) applied to the deployment of a rule package.	Any positive integer	300	Immediately
require-checkin-comment (new in 8.1.3)			
Specifies whether users must add a check-in comment when committing changes to rules. These comments show up when viewing package history. If the value is set to false (default), users can save changes to rules without specifying a comment.	true/false	false	Immediately
force-snapshot-on-deployment (new in 8.1.3)			
Specifies whether users can deploy only a package snapshot. If the value is true, users can only deploy a package snapshot. If false (default), users can deploy either the LATEST package or a snapshot.	true/false	false	Immediately
encoding (not in application template by default)			
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			After GRAT restart

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).			
clear-repository-cache (new in 8.1.4)			
The GRAT server builds and maintains a cache of the rules repository database (for example, index files, and so on), and stores this on the file system under `WEB-INF/classes/repository`. The cache improves performance when accessing frequently used rules, calendars, and so on. However, this cache must stay synchronized with the rules repository database. Normally, if GRAT is restarted, it re-uses the existing cache, which is synchronized with the rules repository database. In this case, the `clear-repository-option` should be set to `false` (default). However, if you are configuring a second GRAT for warm standby (see High Availability Support), this option should be set to `true` for both the primary and the standby instances of GRAT. Since either GRAT could be brought online in the event of a failure, this option forces GRAT always to rebuild the cache and re-synchronize it with the rules repository database. Setting this option to `true` can delay the startup of GRAT, since the cache must be rebuilt, but it ensures that it is properly synchronized with the rules repository database.	true/false	false	After GRAT (re-)start

Settings in GRE

Description	Valid values	Default value	Takes effect
deployed-rules-directory (added to application template in 8.1.3)			
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 rules engine is restarted, all packages 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. If you choose to change the default value, ensure that the path exists and that the application server can write to the specified directory.		/GCTI/logs/GRS_Engine (8.1.3 onwards)	After restart
max-number-rule-executions			
The maximum number of rules to be executed during a request. This is used to detect unwanted recursion when `sequential-mode` is false. If this maximum is reached an error is reported. May be set to -1 to denote no maximum.	Any positive integer or -1	10,000	Next rules execution
sequential-mode			
Indicates whether to run the rules engine in sequential mode. In sequential mode, after the initial data set, no more data can be inserted or modified. This allows for the rules engine to operate in a simplified way.	true/false	false	On rules deployment
verify-deployer-address			
Indicates whether to verify the TCP address of the application deploying rules to be that of an associated Genesys Rules	true/false	true	Immediately

Authoring Tool.			
esp-worker-threads (new in 8.1.2)			
Specifies the maximum number of worker threads available when using the ESP interface to execute rules.	Any positive integer	5	Immediately
load-packages-on-start (new in 8.1.4)			
Indicates whether to load deployed rule packages at application start up. If packages are not loaded at startup (value=false), then a package is loaded on its first execution request.	true/false	true	Immediately
json-hierarchical-driver (new in 8.1.4)			
With value `true`, the `JsonHierarchicalStreamDriver` class is used to serialize JSON responses. With value `false`, the `JettisonMappedXmlDriver` class is used. The Jettison driver is unaware of the original data type and will try to detect numerical values and omit the quotes, whereas the `JsonHierarchicalStreamDriver` will maintain the data type.	true/false	false	Immediately

Troubleshooting

This section contains the following topics:

- Configuration Considerations
- Configuration Diagrams
- Locating the GRDT Version Number
- The log4j.properties File
- GRE Server Status Check

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)

In a multi-tenant environment, the authorized tenant(s) must be added to the `Tenants` tab.

- 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.

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.

  - When deploying a rule package from the Rules Authoring Tool, if there are no "target" Rules Engines to select from, check that the correct tenants have been specified for both the Rule Authoring Tool and Rules Engines. Only those Rules Engines whose tenants match will be displayed.

- 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

- Secured: Optionally, select to activate deployment over a secured connection.

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. See Role-Based Access Control 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).

These first level nodes under Business Structure represent the Solution(s). Within each solution, 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 is 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. See About Business Structure 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
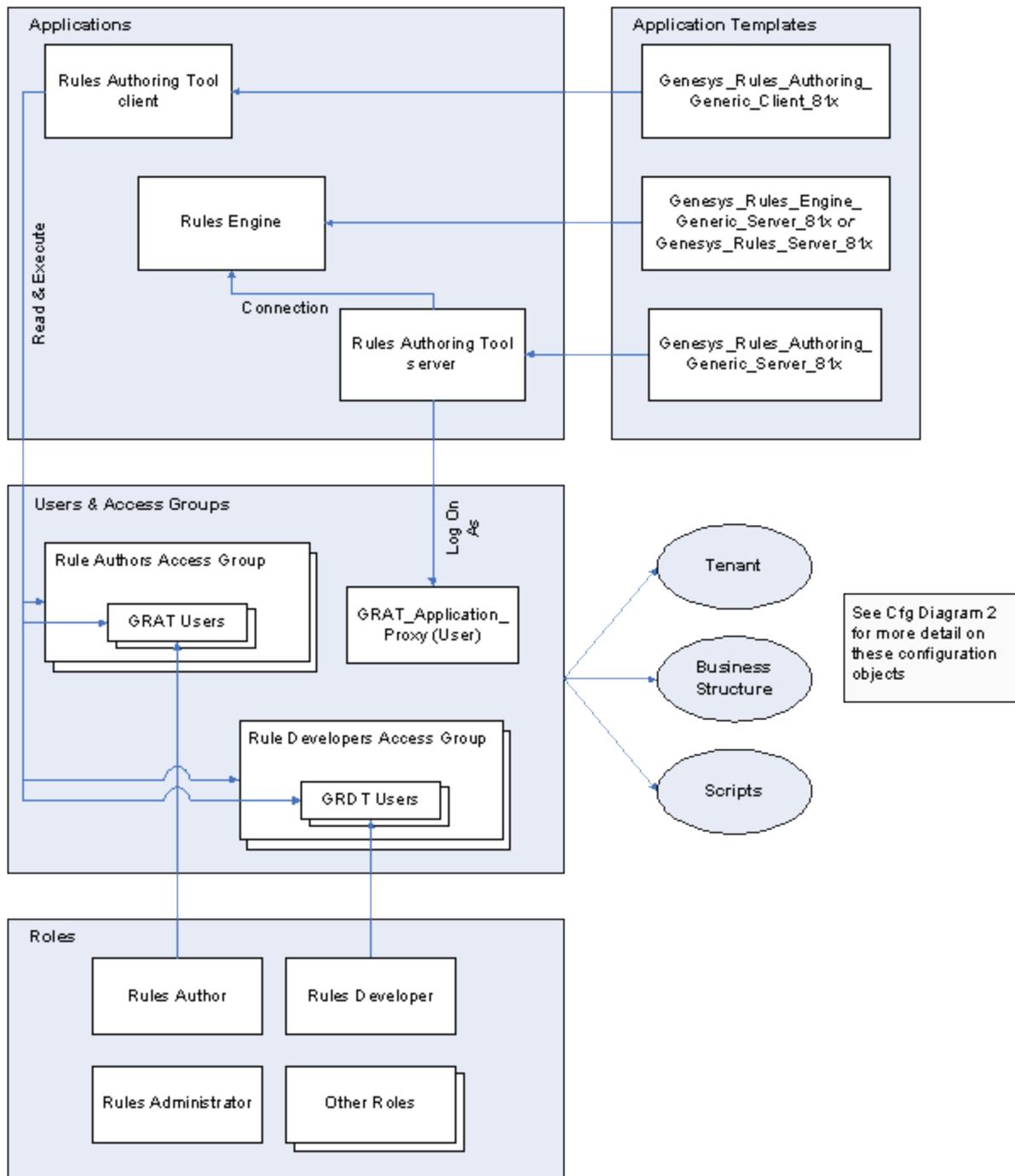
Configuration Diagram 1
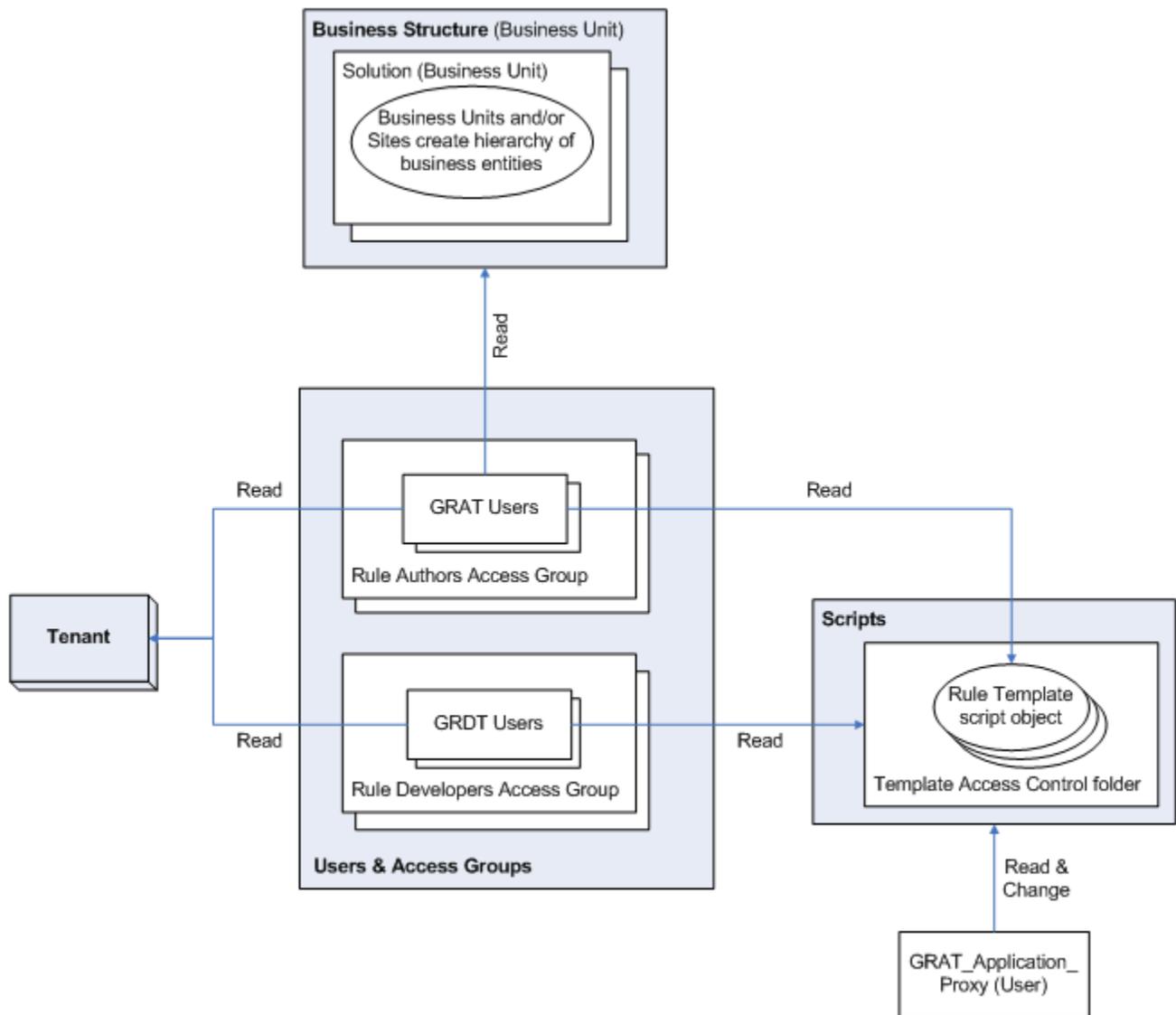
Configuration Diagram 2

Locating the GRDT Version Number

The GRDT is an Eclipse plug-in with a specific version number format that is not easily located. If you are working with Genesys Technical Support, you will need to supply the exact version of the GRDT you are using.

Procedure

To locate the version number:

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 column, you will see the version number (in the format `8.1.x.xxxxxxxxxxxx`, as shown in the diagram below).
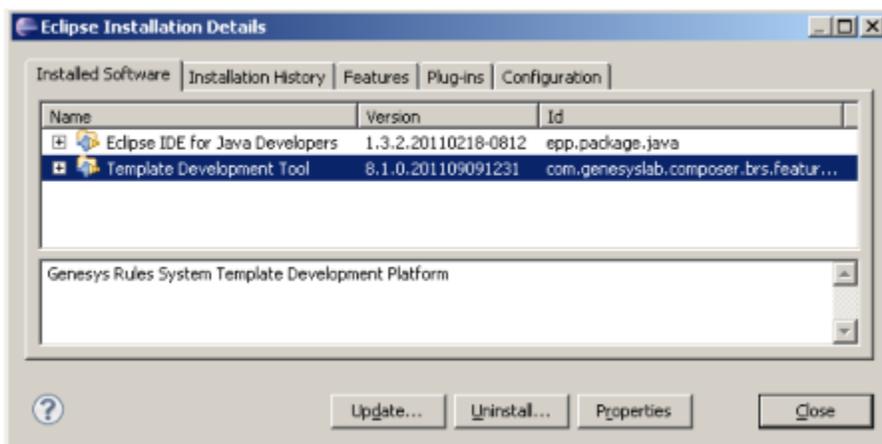
GRDT Version Number

> **Important**
> 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.

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

1. Locate the `log4j.properties` file. This file can be found in the `.war` file, which is located in the installation directory.

2. 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).

3. Open the file in a text editor, and update any logging parameters.

4. Save the file.

5. 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.

GRE Server Status Check

To test for a GRE server status "heartbeat," use the base GRE URL:

`http://<hostname>:<port>/genesys-rules-engine/`

This returns the following:

Localization

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 Installing Language Packs).

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.

Installing Language Packs

Installing a language pack on Windows

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.

> ### Important
> 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. 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.

Installing a language pack on UNIX

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.

> **Important**
>
> 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. 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.

Selecting a preferred language in Internet Explorer

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

1. Locate `Tools > Internet Options > Languages`.

2. Add the preferred language and move it to the top of the list.

3. Log out or refresh the browser.

Selecting a preferred language in Firefox

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

1. Locate `Tools > Options`.

2. Select the `Content` tab.

3. Add the preferred language and move it to the top of the list.

4. Log out or refresh the browser.

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 GRAT Component).

Rule Templates and Rules

Rule Templates

Releases up to and including 8.1.2

Rule templates are developed in the Genesys Rules Development Tool (GRDT). In releases up to and including 8.1.2, each time a rule template is published, a new version is created 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.

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.

Release 8.1.3

In release 8.1.3, multiple versions of templates can be created and stored for users to choose from in the Template Selection dialog. This dialog shows the last N versions of a template, where N is a value configured by using configuration option `display-n-template versions` in Genesys Administrator.

For example, if the configuration were set to show the last 3 versions of a template, the currently selected template is GRS Template version 2, and there are 5 versions in the repository, we would show GRS Template versions 5, 4 and 3, as well as GRS Template version 2. Users could choose between versions 3, 4, or 5.

Template Selection

Configuration Option

`display-n-template-versions`

Valid Values: Integer >=1

Default Value: 3

Description: Integer value specifying the maximum number of versions to display for any published template.

See also Deploying GRAT in Genesys Administrator for information about this configuration option.

Version Comment

In order to provide details about the differences between template versions, rules template developers in GRDT can now publish a version comment that describes specific changes made to individual template versions. This version comment appears in GRAT in the Template Selection table, and can be edited by the rule author in GRAT.

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

## 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.

Rule Life Cycle - Schematics
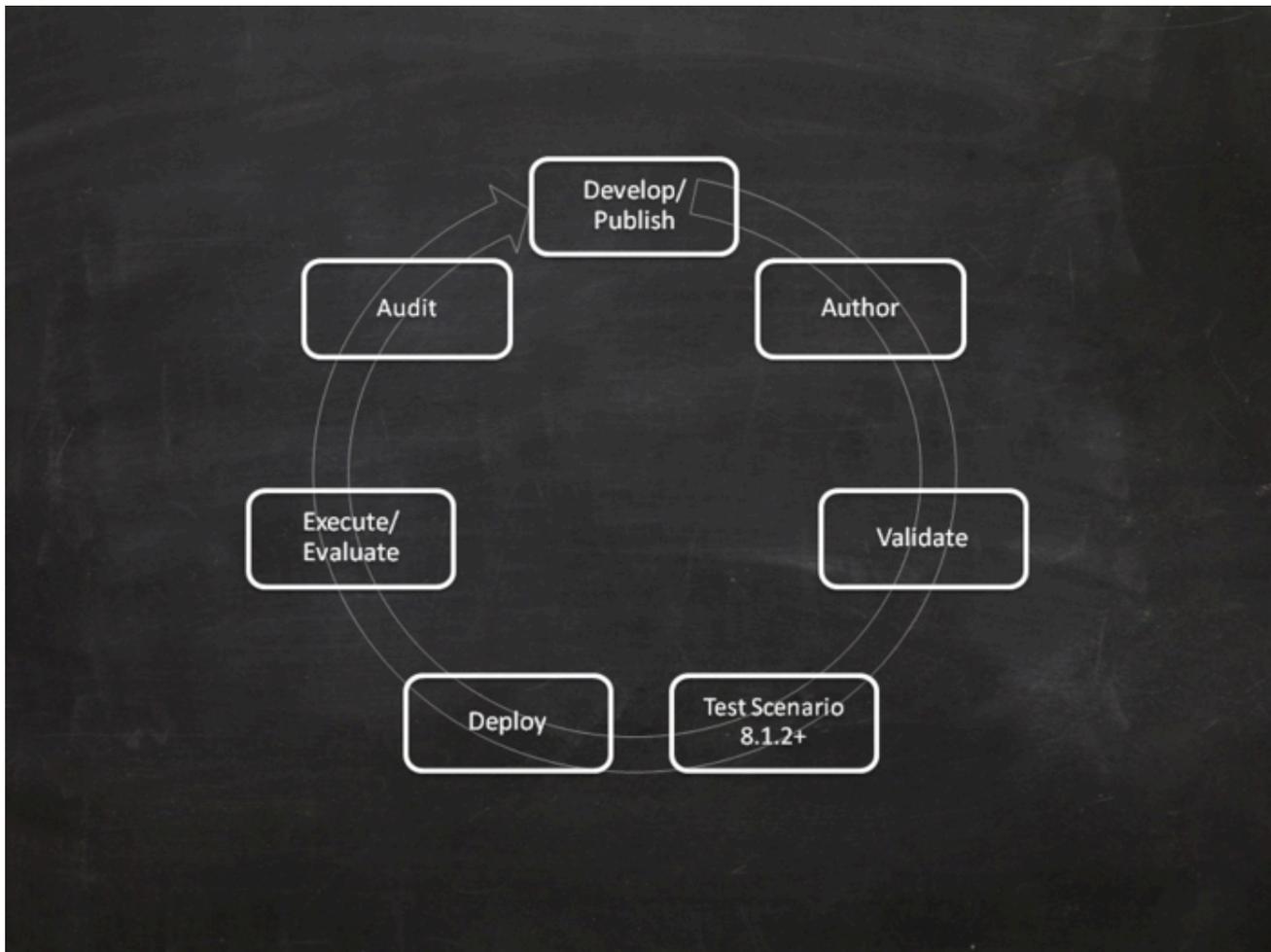
Rule Life Cycle by Component
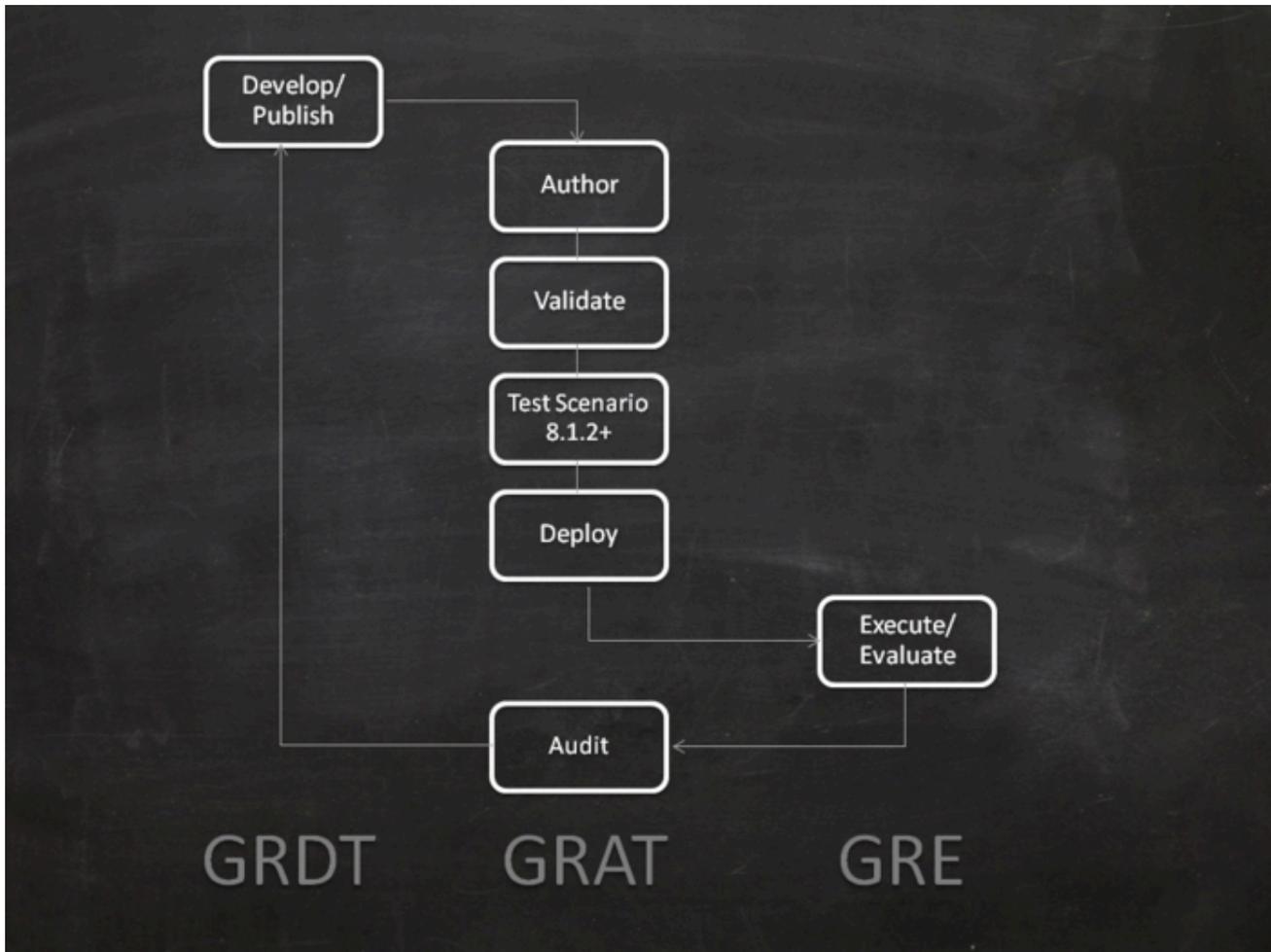
Rule Templates

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

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).

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. 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).

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

Database Parameter Properties

Property	Mandatory/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. The table below lists the properties that you can specify for web service parameters.

Web Service Parameter Properties

Property	Mandatory/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.

Property	Mandatory/optional	Description
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]
```

Warning:	You cannot specify headers or parameters that contain "," in the value. Warning: 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.

Workforce Management Parameter Properties

Property	Mandatory/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.

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. In 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 the Script Objects screenshot below. The Script object `newProject` corresponds to a rule template of the same name).
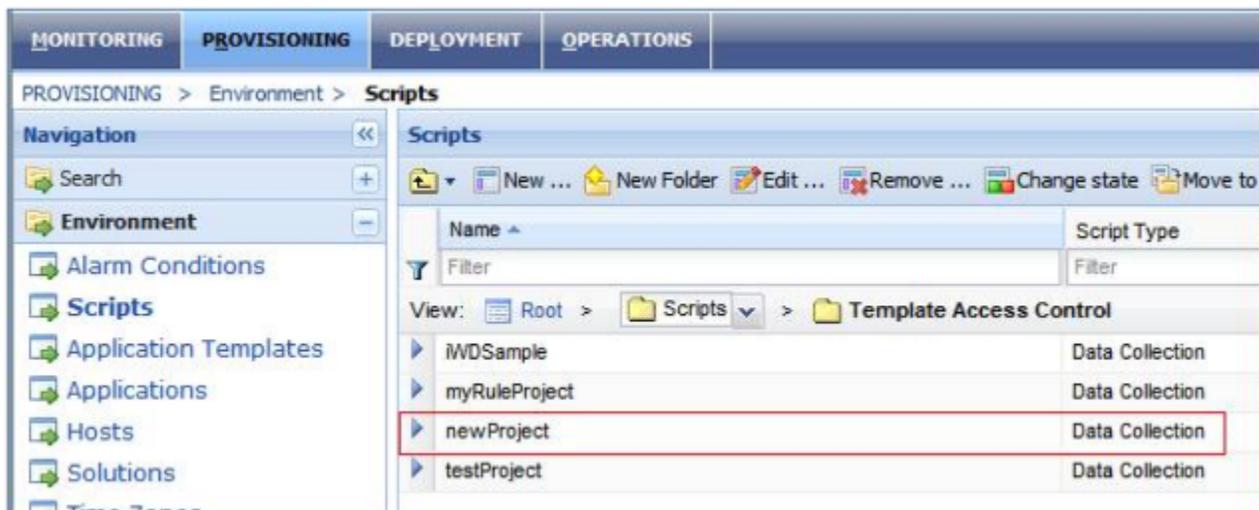
Script Objects

You can use permissions to control which users and/or access groups should be able to use this template.
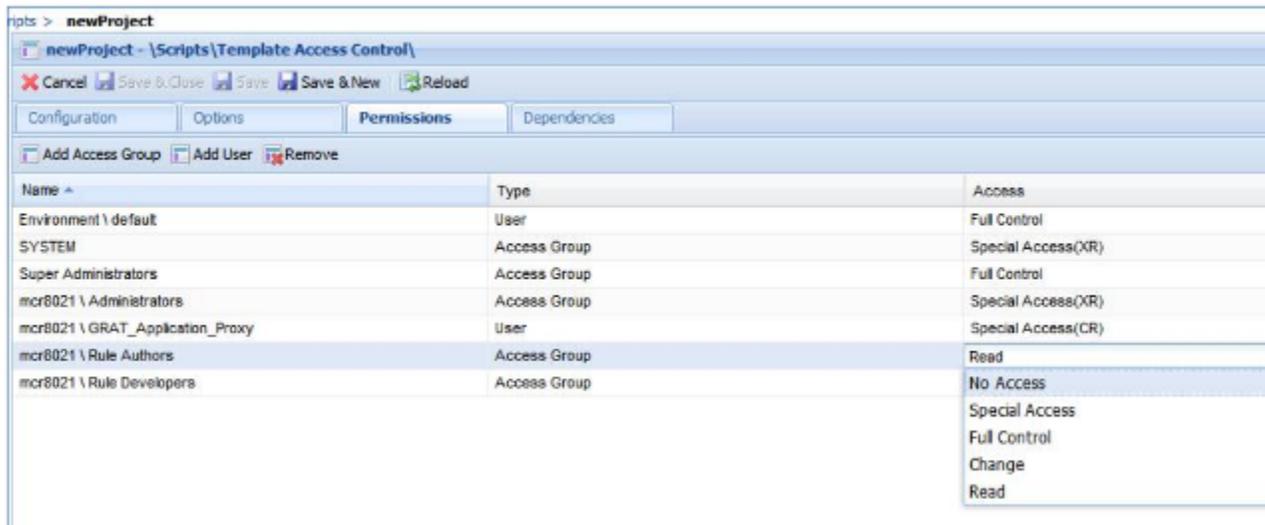
Script Objects

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

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.

Access Permission for Script Objects

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.

Examples of Rule Template 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: Condition and Action

Age Range Condition

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}')
```

Do not use the word 'end' in rule language expressions. This causes rule parsing errors.

The figure below shows how this condition would appear in the Genesys Rules Development Tool.
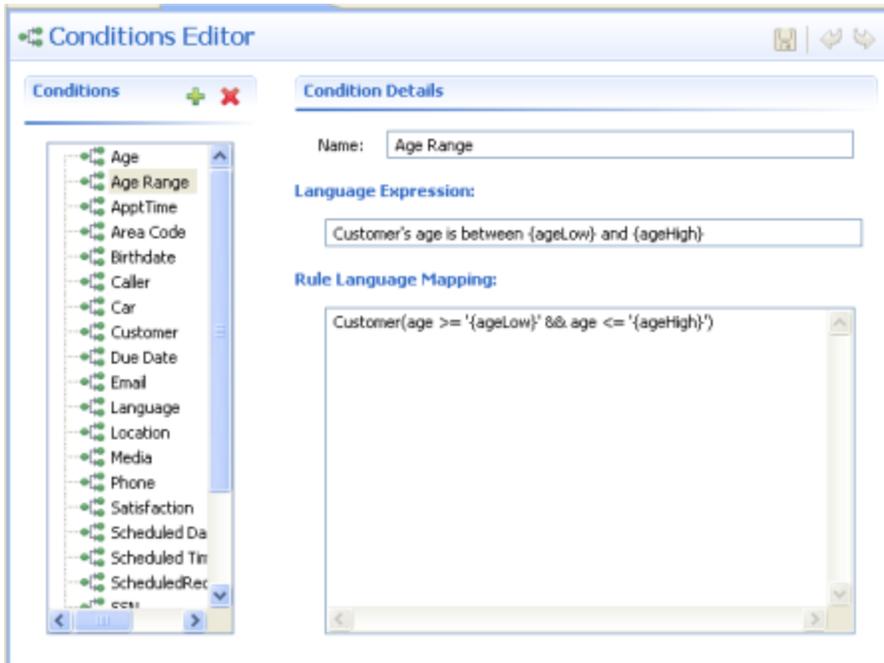
Age Range Condition

Caller 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
```

The figure below shows how this condition would appear in the Genesys Rules Development Tool.
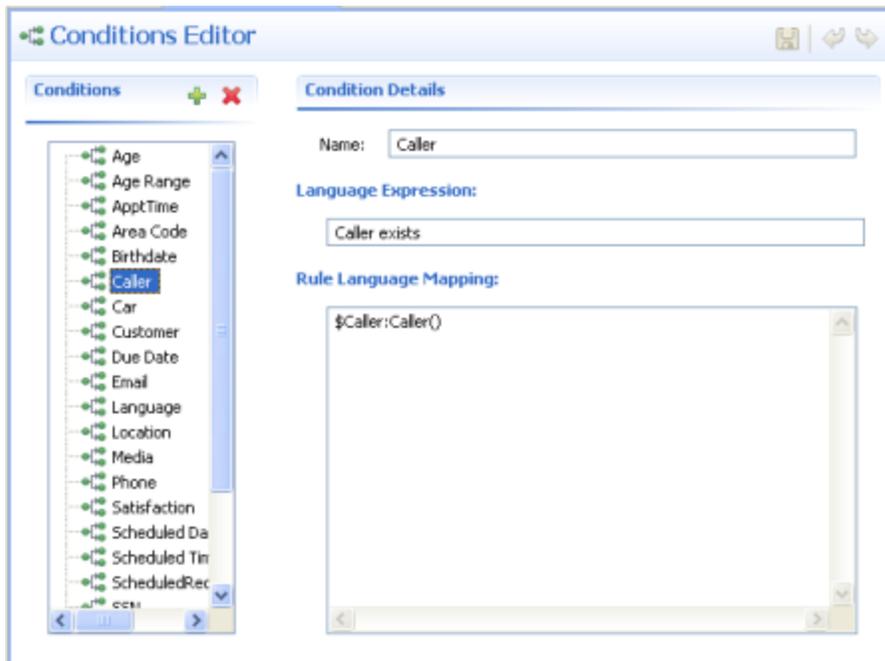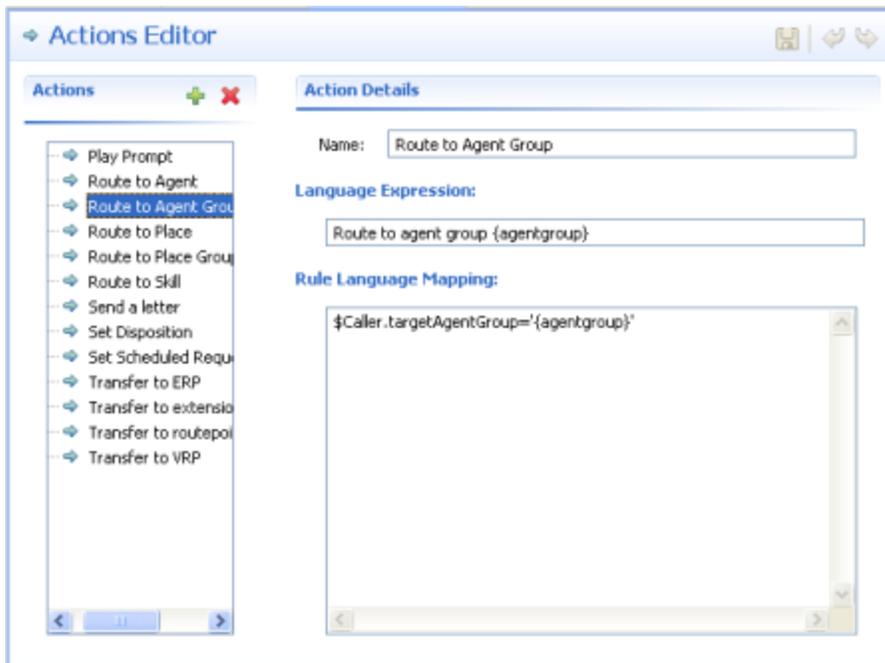
Caller Condition

Target Agent Group Action

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}'
```

The figure below shows how this action would appear in the Genesys Rules Development Tool.

Target Agent Group

The condition in this example has two parameters:

- {ageLow}
- {ageHigh}

The action has the {agentGroup} parameter. Parameters are also configured in the Genesys Rules Development Tool. The Parameters Editor screenshot shows a sample {ageHigh} parameter. Refer to the Genesys Rules Development Tool Help for more details about how to configure parameters.
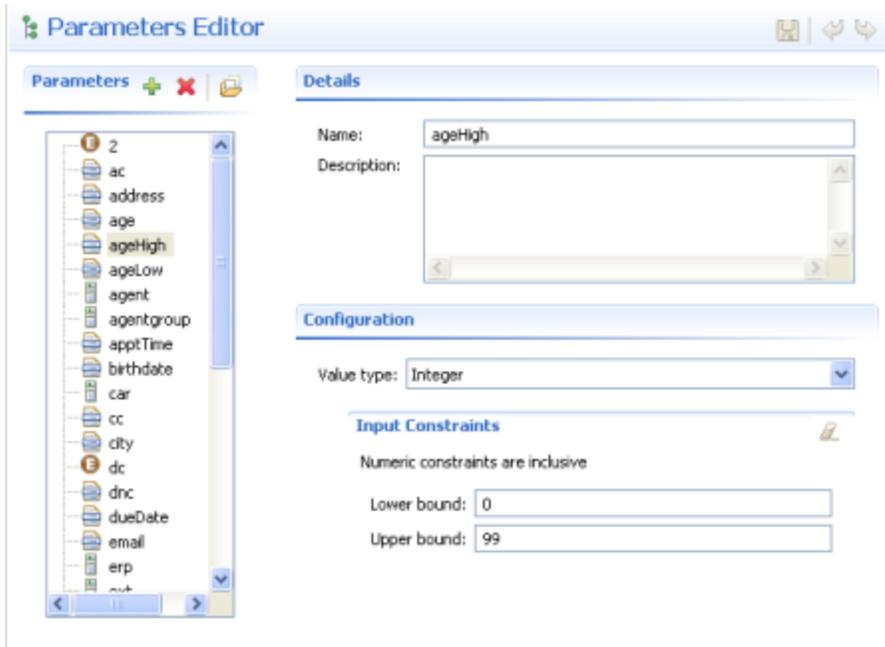
Parameters Editor Screen

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 Differences in Release 8.1.2). 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.

3. 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.

4. 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).

5. The rule developer publishes the rule template to the Rules Repository (but see Differences in Release 8.1.2 for post-8.1.2 releases).

6. 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.

7. The rules author deploys the rule package to the Rules Engine application server (but see Creating an Application Cluster in Configuration Manager for post 8.1.2-releases).

8. 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. 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. 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 Examples of Rule Development. 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()).

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.

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;
          }
     }
```
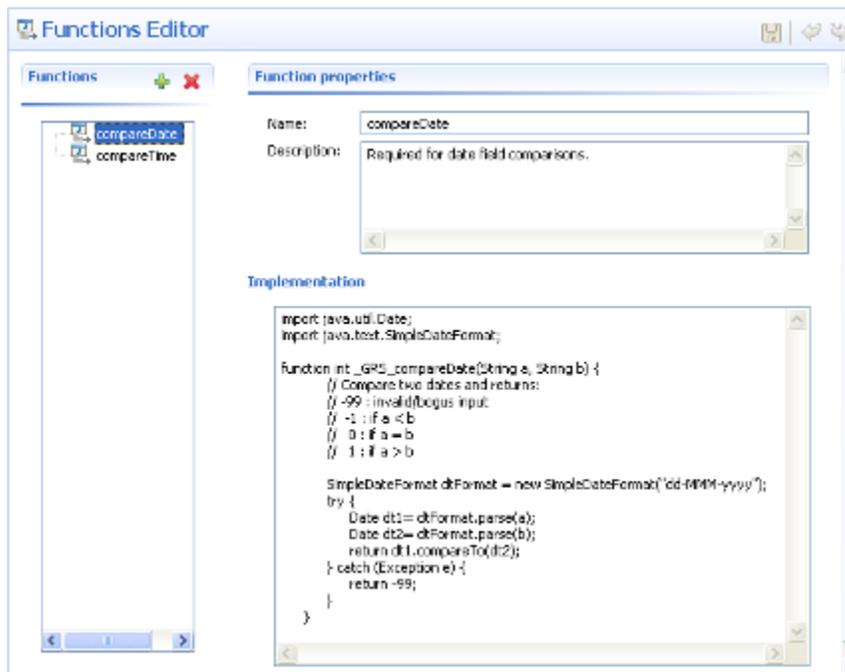
For user-supplied classes, the .jar file must be in the CLASSPATH for both the GRAT and the GRE.

The figure below shows how this function would appear in the Genesys Rules Development Tool.

compareDate Function

Example: Using a JSON Object

Release 8.1.3 allows template developers to create templates that enable client applications to pass Facts to GRE as JSON objects without having to map each field to the fact model explicitly.

> ### Important
>
> Rules based on templates that use this functionality do not support the creation of test scenarios at present.

This example shows how to create a template containing a class (called `MyJson`) for passing a JSON object.

Start

1. Create the following class and import it into a rule template:

   ```
   package simple;
    import org.json.JSONObject;
    import org.apache.log4j.Logger;

   public class MyJson {
           private static final Logger LOG = Logger.getLogger(MyJson.class);
           private JSONObject jsonObject = null;

           public String getString( String key) {
                   try {
                                   if ( jsonObject != null)
                                                   return jsonObject.getString(
   key);
                   } catch (Exception e) {
                   }
                   LOG.debug("Oops, jsonObect null ");
                   return null;
           }

           public void put( String key, String value) {
                   try {
                   if (jsonObject == null) {
                                   jsonObject = new JSONObject();
                   }
                   jsonObject.put( key, value);
                   } catch (Exception e) {
                   }
           }
   }
   ```

2. Create a dummy fact object with the same name (`MyJson`) in the template.

3. Add the `MyJson.class` to the class path of both GRAT and GRE.

4. Create the following condition and action:

```
Is JSON string "{key}" equal "{value}"
eval($MyJson.getString("{key}").equals("{value}"))
Set JSON string "{key}" to "{value}"           $MyJson.put("{key}", "{value}");
```

5. Use this condition and action in a rule within the `json.test` package. The following will be generated:

```
rule "Rule-100 Rule 1"
 salience 100000
   agenda-group "level0"
   dialect "mvel"
   when
        $MyJson:MyJson()
        and (
        eval($MyJson.getString("category").equals("test"))
        )
   then
        $MyJson.put("newKey", "newValue");
 end
```

6. Deploy the `json.test` package to GRE.

7. Run the following execution request from the RESTClient:

```
{"knowledgebase-request":{
 "inOutFacts":{"anon-fact":{"fact":{"@class":"simple.MyJson", "jsonObject":
{"map":{"entry":[{"string":["category","test"]},{"string":["anotherKey","anotherValue"]}]}}}}}}}
```

8. The following response is generated:

```
{"knowledgebase-response":{"inOutFacts":{"anon-
 fact":[{"fact":{"@class":"simple.MyJson","jsonObject":
{"map":{"entry":[{"string":["category","test"]},{"string":["newKey","newValue"]},
{"string":["anotherKey","anotherValue"]}]}}}}],
 "executionResult":{"rulesApplied":{"string":["Rule-100 Rule 1"]}}}}}
```

End

Rule Language Mapping

When rule developers create the conditions or actions in a rule template, they enter the rule language mapping. Up to and including Genesys Rules System 8.1.2, the 5.1 Drools Rule Language is used. Details of this can be found here:

```
http://downloads.jboss.com/drools/docs/5.1.1.34858.FINAL/drools-expert/html/ch04.html
```

However, for use in JBOSS environments, you should reference the 5.2 version here:

```
http://downloads.jboss.com/drools/docs/5.2.FINAL/drools-expert/html/ch05.html
```

For GRS 8.1.3, use the 5.5 versions, found here:

```
http://downloads.jboss.com/drools/docs/5.5.FINAL/drools-expert/html/ch04.html
```

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\(s\).
```

HTML Constructs

For security reasons, GRAT does not allow any HTML commands to be entered as parameters of a rule. For example, if a condition is:

```
Customer requests a callback on {day}
```

and {day} is defined as a string, we would not allow a rule author to enter the string:

```
Customer requests a callback on ‹b›Tuesday‹/b›.
```

All HTML constructs will be removed from the string. This applies to string parameters as well as dynamic list parameters such as business attributes, database or web service.

Rules and Rule Packages

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 About Business Structure for information about how to configure your business structure.

Using the same example that was used in the rule language mapping section (see Rule Language Mapping), the following example shows how the action and condition might be used in a linear rule.
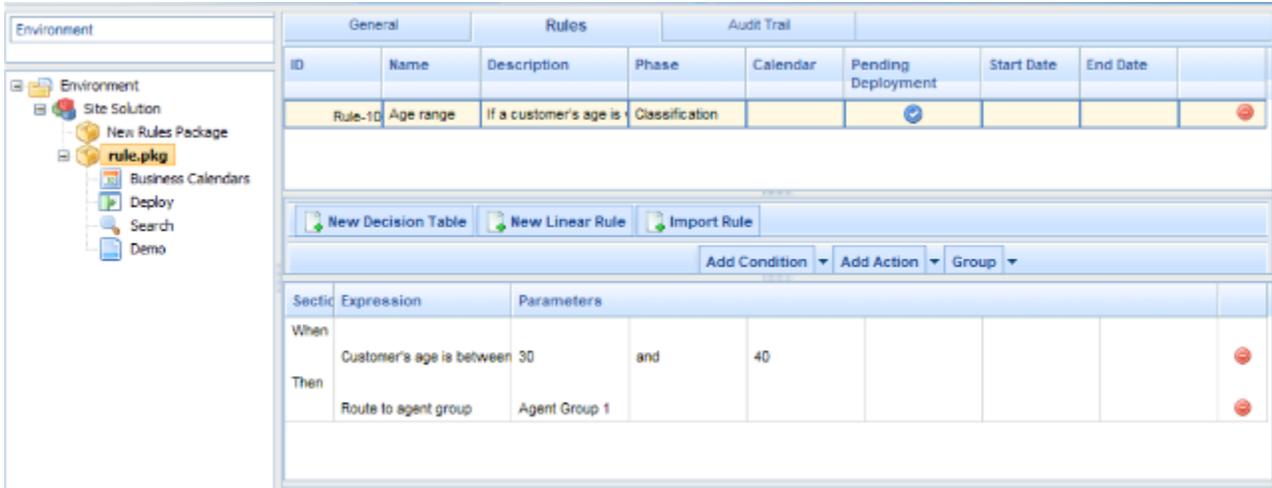
Example 1: Linear Rule

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. The figure below shows

the linear rule in the Genesys Rules Authoring Tool.

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.

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.
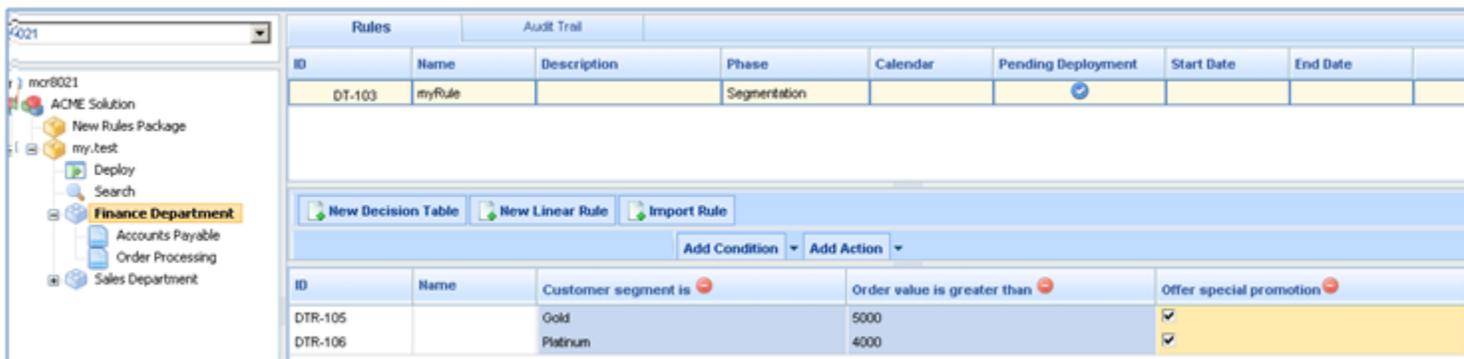
Sample Decision

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.

About 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.

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 Role-Based Access Control for more information.
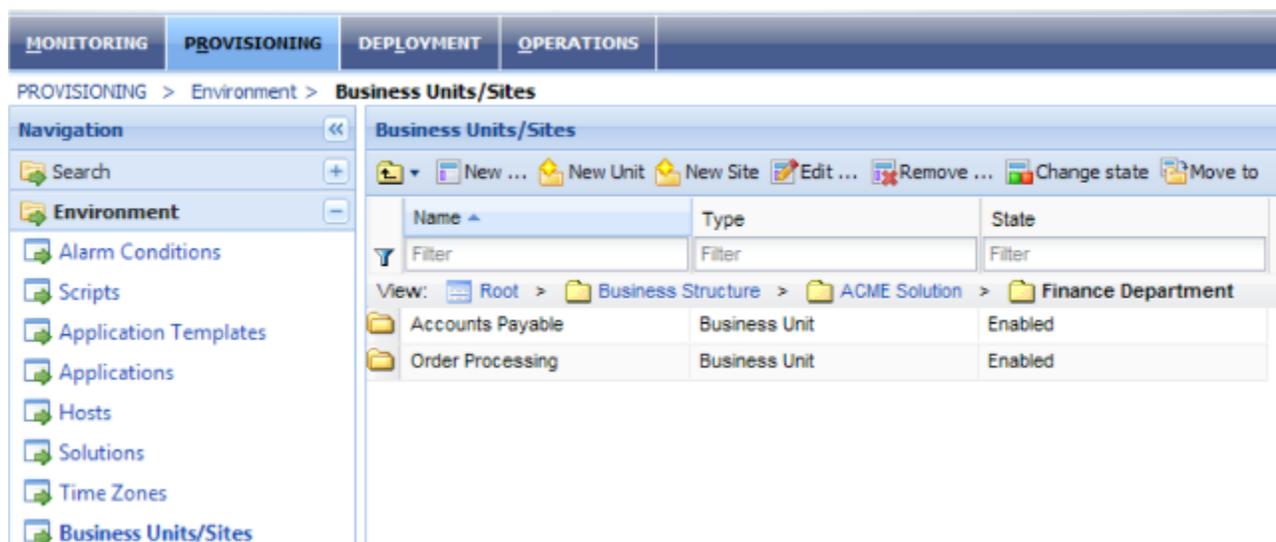
Configuring the Business Structure

Your Tenant's business structure is created under Resources for single-tenant Configuration Server, or under a Tenant for a multi-tenant Configuration Server.

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. This folder must be named Business Structure.

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.

Multiple Solutions can be created by creating additional Business Units or Sites directly beneath the Business Structure folder.
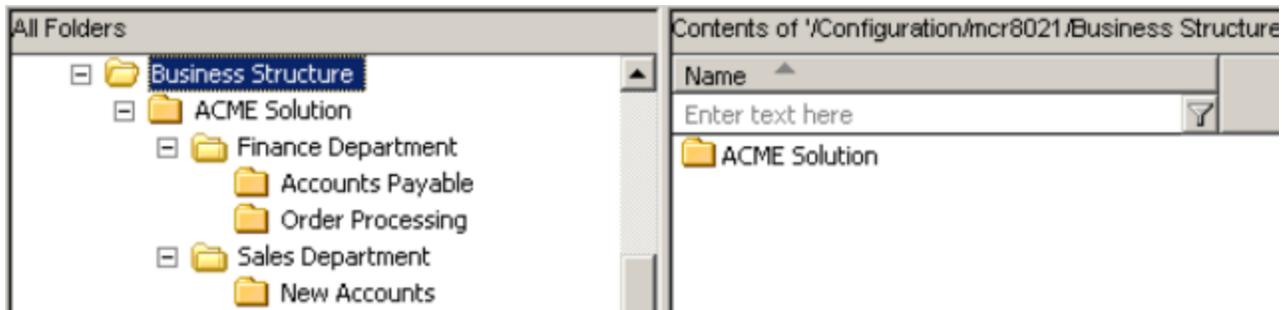
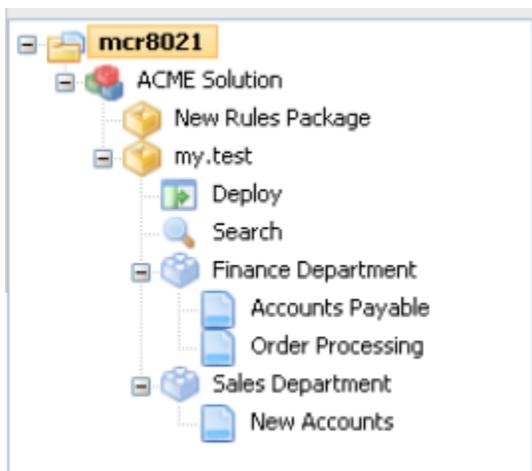## Sample Business Structure in Genesys Administrator

Sample Business Structure in Genesys Administrator

## Sample Business Structure in Configuration Manager

Sample Business Structure in Configuration Manager.

Sample Business Structure in the Genesys Rules Authoring Tool

Sample Business Structure in GRAT

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. The figure below shows that all members of the Users access group have Read permissions to the `Business Structure` folder.
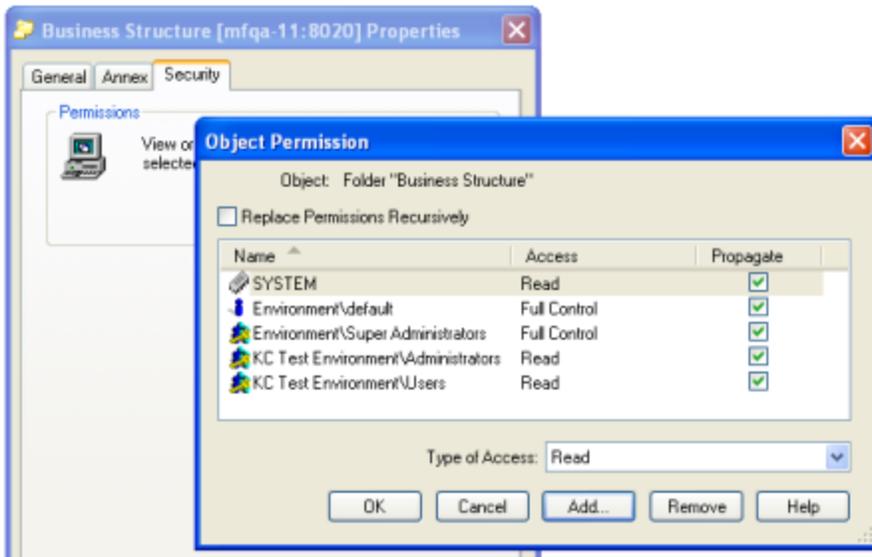
Business Structure Permissions.

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.

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.

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.

Permissions in Release 8.1

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

Additions in Release 8.1.2

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 for more information.

- To import or export rule packages, a user must have full permissions granted. For example, if a user does not have the ability to view business calendars or test scenarios, they won't be exported in the rule package XML. Conversely, if a user doesn't have permission to create calendars or test scenarios on import, they will not be able to create these resources from the imported rule package.

Additions in Release 8.1.3

In release 8.1.3, the following new permissions are defined:

- Snapshot - Create

- Snapshot - Delete

- Snapshot - View: User can view and export snapshots. If this is not enabled, users will only see LATEST in the list, which represents current 8.1.2 functionality where users can only deploy the latest version.

> ### Important
> Snapshot permissions are active on in the `Deployment` tab of GRAT, so all snapshot permissions also require `Rule Package - Deploy` permission.

User Logins

The GRAT has multiple connections to Configuration Server:

- The server connection that is used by the Rules Authoring server to read application information and perform various server tasks

- The individual client connection of each user who logs on to the GRAT. This 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 Tenant) 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 About Business Structures 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
- Snapshots—Create, Delete, 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.

Genesys recommends 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

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, or add the user as a member of an access group which can be part of a role.

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.

REST API

The following topics describe features of the REST API that are supported by Genesys Rules System:

- Rule Execution
- Changes in 8.1.2
- Error Handling

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 an HTTP POST message to:

```
http: //{server-address:port}/{server-id}/knowledgebase/{packageName}
```

This port is configured in the GRE application. See Installing the GRE Component 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. 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

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}"</er`
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>`
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>`

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

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

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	• activation-group	• package	• attributes	• template	• eval	• exists	• action	• init
• date-effective	• agenda-group	• import	• rule	• query	• not	• forall	• reverse	
• date-expires	• ruleflow-group	• dialect	• extend	• declare	• in	• accumulate	• result	
• no-loop	• entry-point	• salience	• when	• function	• or	• collect	• end	
• auto-focus	• duration	• enabled	• then	• global	• and	• from	• over	

You can use these (hard and soft) words as part of a method name in camel case, for example `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()
```

Working Example

This section provides an almost complete end-to-end sample use case:

- Rule template
- Test scenarios (from release 8.1.2)
- Rule package
- Deploy and execute

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 entity that is called "ACME Solution."

Under this Solution there are two departments:

- Finance Department
- Sales Department

Under the Finance Department there are two processes:

- Order Processing
- Accounts Payable

Under the Sales Department there is a single process:

- New Accounts

The figure below 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.
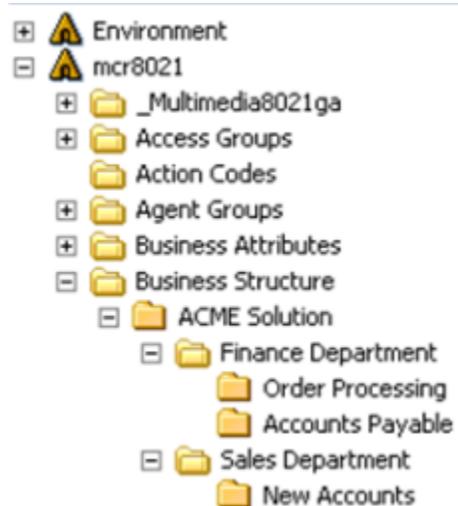
Business Structure

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 the two following tables for details of these parameters.

Rule Langauge Mapping

Rule Language Mapping Parameters

Name	Language Expression	Rule Language Mapping
Segment	Customer segment is {customerSegment}	`Customer(segment=='{customerSegment}')`
OrderValue	Order value is greater than {orderValue}	`Customer(order>{orderValue})`
SpecialOffer	Offer special promotion {specialOffer}	`$Customer.offer='{specialOffer}'`

Language Expression Details

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. 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.

The figure below shows how the enumeration is configured.

Enumeration Details

Note that for this template, because 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.

With the Drools language you cannot set the value of a Fact property by referring to the Fact's name. In the condition section you must first declare a variable and associate this variable with a Fact object. Once this association has been made within a condition then the variable can be used in actions (and other conditions) to reference fields contained within the fact. A period (".") is used to access the fields on a fact. Use a colon (":") 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 (`$Customer:Customer()` in the condition, `$Customer.offer` in the action).

Beginning with the 8.1.2 release, conditions are automatically added to declare variables which are referenced in actions. For a variable to be automatically declared, the variable name must be the name of the fact preceded by a '$' sign. So in this example, `$Customer` is referenced in an action so the condition `$Customer:Customer()` will automatically be added to the rule.

Prior to 8.1.2, variables must be declared by a condition within the template and added to the rule by the rule developer. With this example the following generic rule condition needs to be defined within the template and the rules developer would add this condition to any rule that referenced the `$Customer` variable. Note: A a variable cannot be declared twice.

```
Language Expression: Customer exists
Rule Language Mapping: $Customer:Customer()
```

Supporting Building Test Scenarios

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 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.
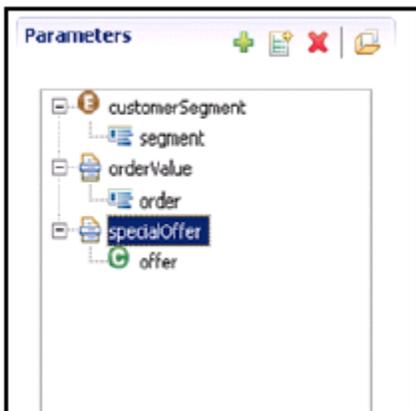
Mapping Parameters Popup Window

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.

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 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 Scenario Tab 2

We simply adjust the test scenario so that we now expect an offer for this customer by checking the special0ffer box. We now get a successful result when running the test.

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, which is created at the Finance Department node of the business structure.

Decision Table

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.

The third rule is defined as a linear rule and has been created at the Accounts Payable node of the business structure.

Linear Rule

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.

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

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 Rule Template.

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.