This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Genesys Rules System Deployment Guide

Genesys Rules System 8.5.3

12/29/2021

Table of Contents

Genesys Rules System Deployment

Welcome to the Genesys Rules System 8.5.x deployment pages. This document describes how to install and configure Genesys Rules System 8.5.3.

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

New Features by Release

Migration to 8.5.x

Upgrading within the Same Release

Preparing for Installation

Installing Genesys Rules System - Task Summary

Preparing for Installation

Configuring the Rules Repository Database for Configuration Manager

Configuring the Rules Repository for Genesys Administrator

Installing Genesys Rules Engine

Deploying GRE in Genesys Administrator

Creating the GRE Application object in Configuration Manager

Installing the GRE Component

Creating GRE Application Clusters

Importing the GRE Smart Cluster Template

Creating an Application Cluster in Configuration Manager

Creating an Application Cluster in Genesys Administrator

Installing Genesys Rules Authoring Tool

Deploying GRAT in Genesys Administrator

Installing the GRAT Component

Creating the GRAT Application Object in Configuration Manager

Creating GRAT Application Clusters

Creating the GRAT Application Cluster Object in GA

Installing GRAT (cont)

Deploying WAR Files

Performance Tuning with JVM

Configuring WebSphere 8.5

Installing Genesys Rules Development Tool

Installing Genesys Rules Development Tool

Post Installation

Testing the Installation

Installing GRS on Unix Platforms

High Availability Support

Configuring GRS for Secure Sockets Layer (SSL)

Troubleshooting

Configuration Considerations

Configuration Diagrams

Locating the GDRT Version Number

The log4j.properties File

GRE Server Status Check

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
Configuring a User

DROOLS 5 Keywords

DROOLS 5 Keywords

Overview

Genesys Rules System (GRS) 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.

For a comprehensive overview of GRS, including lifecycle and architecture information, please see the **Genesys Rules System Overview**.

New Features by Release

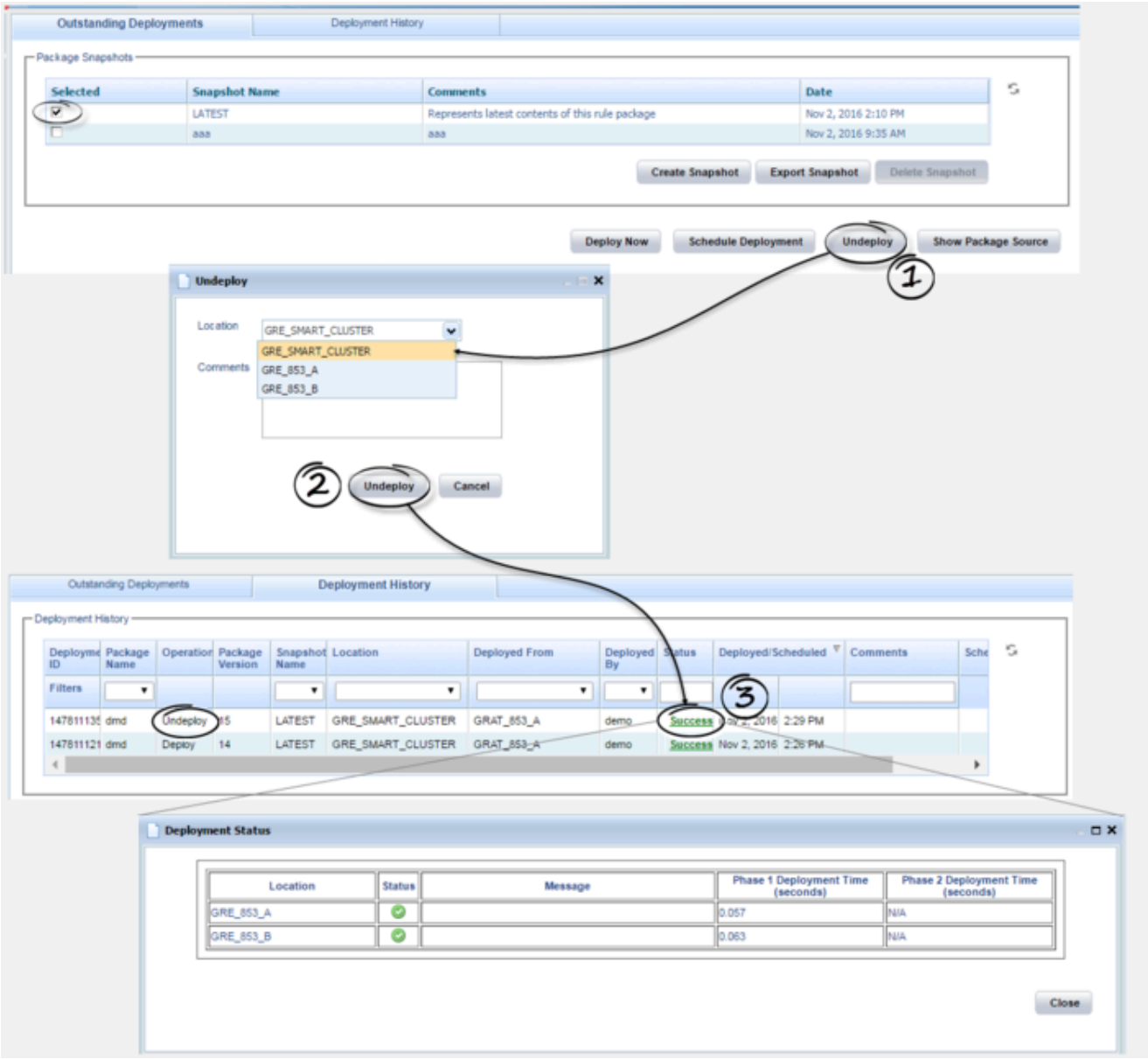
New in 8.5.303

Undeploy a Rules Package

You can now undeploy a specific rules package from a single GRE or cluster.

A new **Undeploy** button has been added to the Outstanding Deployments tab for users with the `PACKAGE_UNDEPL0Y` role privilege. Such users can now select a target GRE or cluster and undeploy a package from that location/cluster. Deployment history will display the undeployment.

Flow summary



Partial undeployment

Partial undeployment—the mirror of partial deployment—is also supported. A new configuration option—allow-partial-cluster-undeployment—which is a mirror to the existing allow-partial-cluster-deployment has been introduced.

If this option is set to `true`, the undeployment request is applied to "active" nodes and the rule package is undeployed. When the inactive GRE node(s) come back online or are re-connected and auto-synchronization is enabled, they will auto-synchronize and undeploy the rule packages automatically. If one or more (but not all) nodes are inactive, the status is set to `Partial`.

If this option is set to `false`, then GRAT undeploys the rule package only if all nodes of the cluster are active. If not, it is reported as a `Failed` status and the rule package is not undeployed from any node.

Changes to GRAT Help

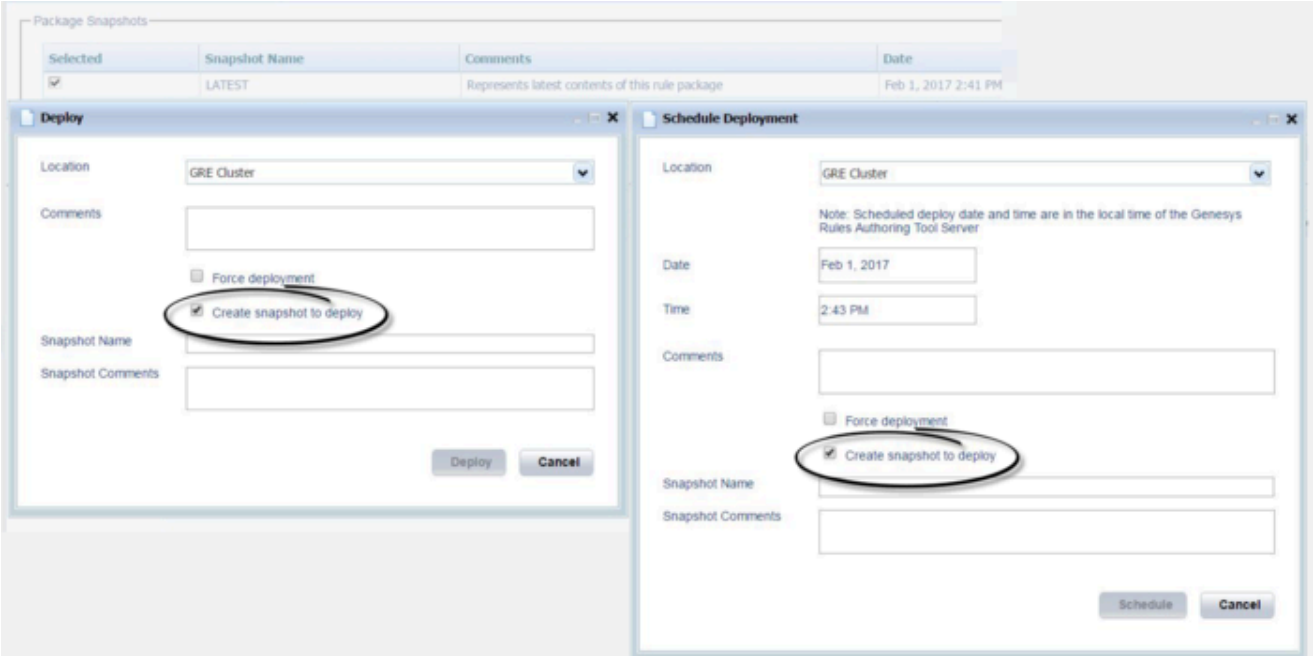
See Deploying a Rules Package for more details.

Limitation

> **Important**
>
> The Undeploy feature is not available for the Genesys Web Engagement rules engine or for rules packages that are based on the Complex Event Processing (CEP) template.

Automated snapshots on deployment

GRAT can now be configured to take snapshots automatically at deployment. In the **Deploy** and **Schedule Deployment** dialogs, a **Create snapshot to deploy** check box is now displayed for the LATEST version only of a rules package, for users with `DEPLOY` and `CREATE_SNAPSHOT` permissions.



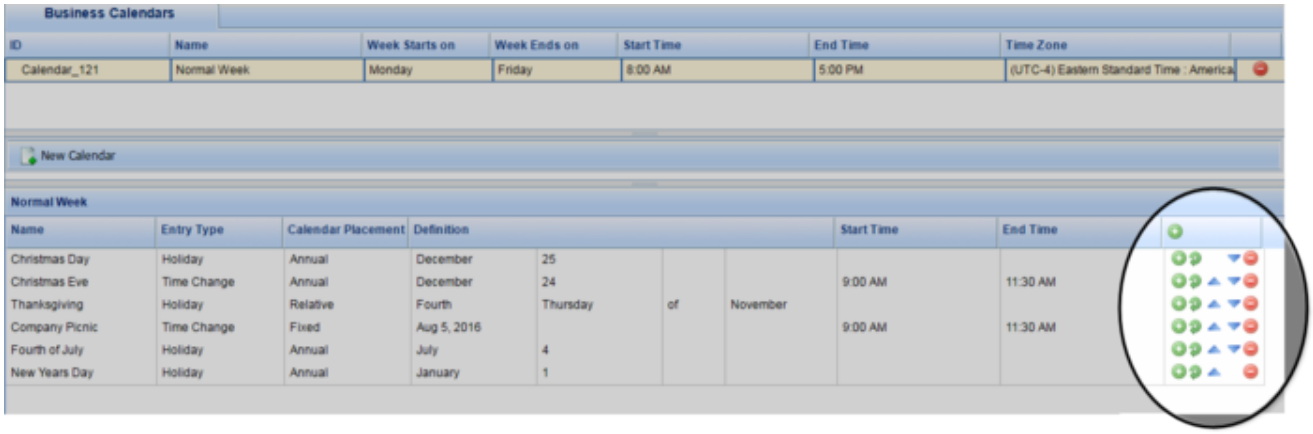
There are some changes to the operation of the configuration option force-snapshot-on-deployment as a result:

- If the value of this option is `false`, the **Create snapshot to deploy** checkbox is displayed unchecked and enabled and users with the above permissions can select it.

- If the value of this option is `true`, the **Create snapshot to deploy** checkbox is displayed checked and disabled and the snapshot is created and deployed automatically. In the case of scheduled deployments, the snapshot is created immediately and deployed at the scheduled time.

Business Calendar Rule Priority

Calendar exceptions can now be assigned a priority order to ensure that no clashes occur between exceptions. Now, for any exception that affects the same day as another exception, the topmost exception takes precedence. User can now also move calendar exceptions up or down in priority order, and copy and paste an exception. A new column in the exceptions panel has features for moving exceptions up and down, adding new ones and copying exiting ones.



GRAT Database Connection Monitor

GRAT now detects any loss of connectivity to the repository database and reports the issue via **status.jsp** for load balancers.

Previously, GRS only reported the `SYSTEM_STATUS_DB_NOT_CONNECTED` status when there a problem with the database connection was encountered during GRAT initialization, but during the rest of the lifetime of GRAT only reported `SYSTEM_STATUS_OK` even when the connection failed.

Now, GRAT continuously monitors the database connection status at a regular configurable interval. If the database connection fails, the monitor now sets the status to `SYSTEM_STATUS_DB_NOT_CONNECTED`. Later when the database connection is detected to be normal, the monitor reset the status to normal.

Additionally, if the GRAT repository could not be initialized at GRAT startup because of incorrect settings or because the repository was unavailable, the connection monitor now initializes the repository when the database connection is re-instated without requiring a restart of GRAT.

Two new configuration options control the monitor:

- enable-repository-connection-monitor
- repository-connection-monitor-interval

GRE Memory Monitor Enhancements

The Memory Monitor in GRE has been enhanced as follows:

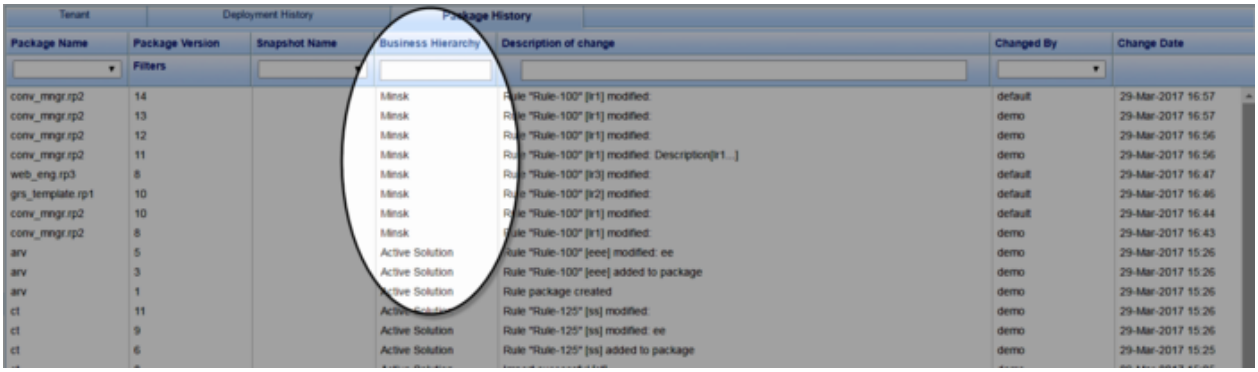
- The maximum value of the memory-monitor-threshold option has been changed from 80 to 99 and the minimum to 1.
- New option enable-memory-monitor-over-threshold-memory-cleanup (default=`true`) has been implemented. With value `true`, Java garbage collection is requested when memory goes above threshold.
- New option enable-memory-monitor-update-status (default=`true`). With value `false`, memory monitor is decoupled from the **status.jsp** function and the LCA status update function.

Package History Enhancements

The Package History tab now includes the following enhancements:

Business Hierarchy Column

A new **Business Hierarchy** column is introduced that can now calculate and identify where the affected rule package or rule package item sits in the business hierarchy. For example: **CM Samples > Sales > Closing**.

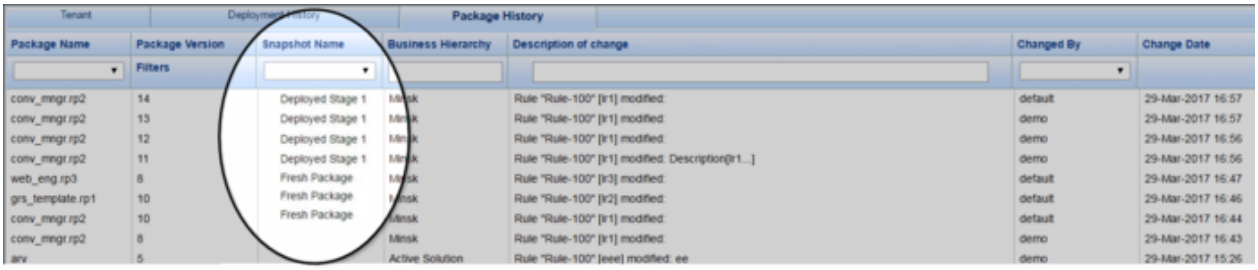


The new column shows values only for actions performed from this release onwards—older action (prior to release 8.5.303) do not appear.

> **Important**
>
> If a business structure node name change is required, this should be done only when there is no active user session (GUI or API) in GRAT or when GRAT is shut down.

Snaphot Name Column

The **Snapshot Name** column now shows the name of the snapshot in which a change was made.



Access to Business Structure History

Package history now shows only changes in the business structure nodes to which the user has access. Thi access eck can be overridden by a new role privilege—`PACKAGE_HISTORY_ADMIN_VIEW`. This allows you to view complete package history for a rule package without checking access to the business hierarchy subnodes used inside the rule package. Even with this role privilege enabled, the package history is only shown for packages that the user can view.

Change By Column Changes

The **Change By** column is visible only to users with relevant role privileges. This is controlled by new role privilege `PACKAGE_HISTORY_VIEW_CHANGED_BY`.

Rule Export Enhancements

When exporting Rule Templates as XML, users now can choose which rule packages to export. Previously, all rule packages from the repository were exported.

Test Scenario Enhancements

Tooltips in the Add Given and Add Expectation drop-down menus in Test Scenarios now display the **Fact** field description as defined in the template.

Linear Rule Segments

The maximum limit of 6 segments (text plus variables) on Conditions or Actions in linear rules has been increased to 9 in release 8.5.303. An error message is displayed if this limit is breached.

Platform Support

Tomcat 8 and 8.5 are supported in this release of GRS.

Configuration Options Reference Guide

Configuration options are documented from 8.5.303 onwards in the *Configuration Options Reference Guide*.

New in 8.5.302

Support for Role-Based Access Control at the Rules Package Level

> **Important**
> GRS requires Genesys Administrator 8.1.305.04 (minimum) for configuring package level permissions.

You can expand the graphics by clicking on them.

Background

Previously, GRAT used Configuration Server Roles to provide only global access control to all packages in a given node of the business hierarchy. The privileges, like `Modify Rule Package`, `Delete Rule Package`, `Modify Rule`, `Delete Rule` and so on, are granted to users via roles. With this approach, if a user is granted the `Modify Rule Package` (for example) privilege, then they can modify all the rule packages defined in a node of the GRAT business hierarchy.

Release 8.5.302 now provides package-level overrides to these global roles—role privileges can be restricted to specific rule packages by applying Role-Based Access Control at the rule package level. The new Rule Package Level Roles (roles created specifically for use with rule packages only) can be mapped to rule packages to override the global-level roles. These Rule Package Level Roles will have no effect if not mapped to a rule package.

New Role Permission—View Rule Package

View access for specific rule packages can now be controlled by using the new role permission **View Rule Package**. The new permission is applicable to only the rule package level.

Existing Role Permissions

All of the existing role permissions except **Create Rule Package** and template-related permissions are applicable at the rule package level too.

Example

In 8.5.302 you can now assign role permissions at both global/node level and at rule-package level to achieve the following outcome:

- Department A
  - Rule package 1
  - Rule package 2
  - Sales
    - Rule package 3
- Department B
  - Rule package 4

- User A—Can see Department A but not Department B
- User B—Can see Department B but not Department A
- User C—Can see rule package 1, but rule package 2 is hidden

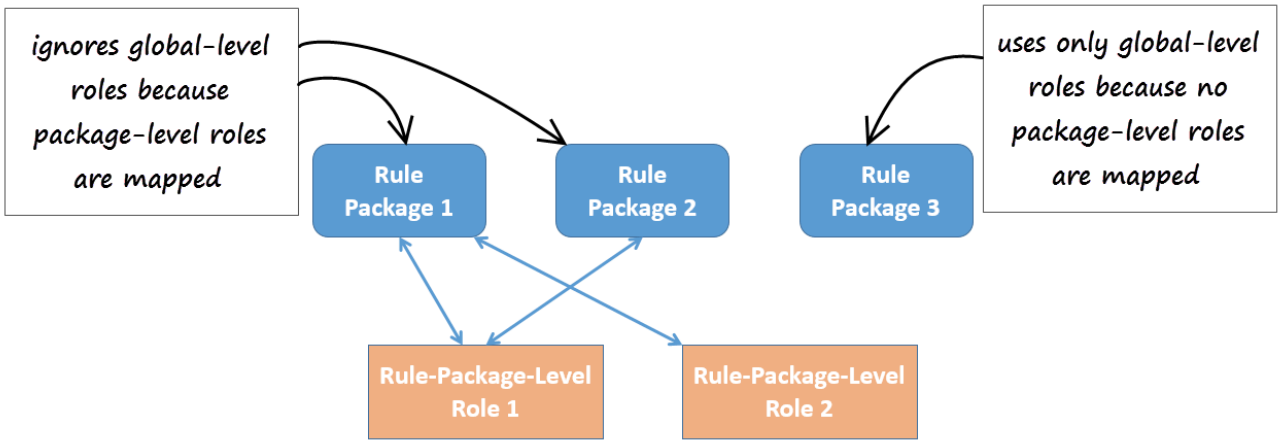
Location

To distinguish these new roles from global-level roles, they are placed in a new folder:

```
[Tenant] > Roles > GRS Rule Package Level Roles
```

Package-Level Overrides

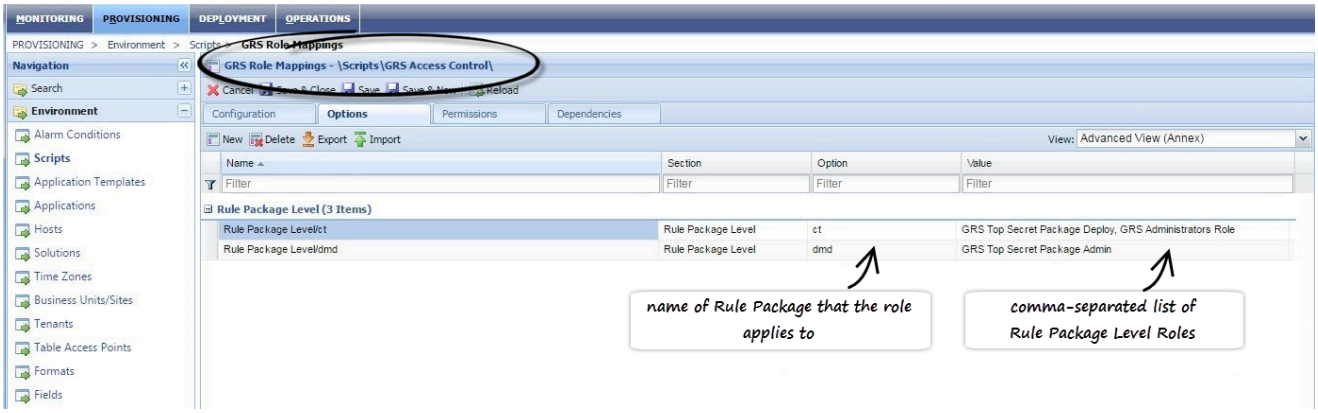
Where package-level roles are mapped to a rule package, they override global-level roles.



Managing the Mapping of Roles

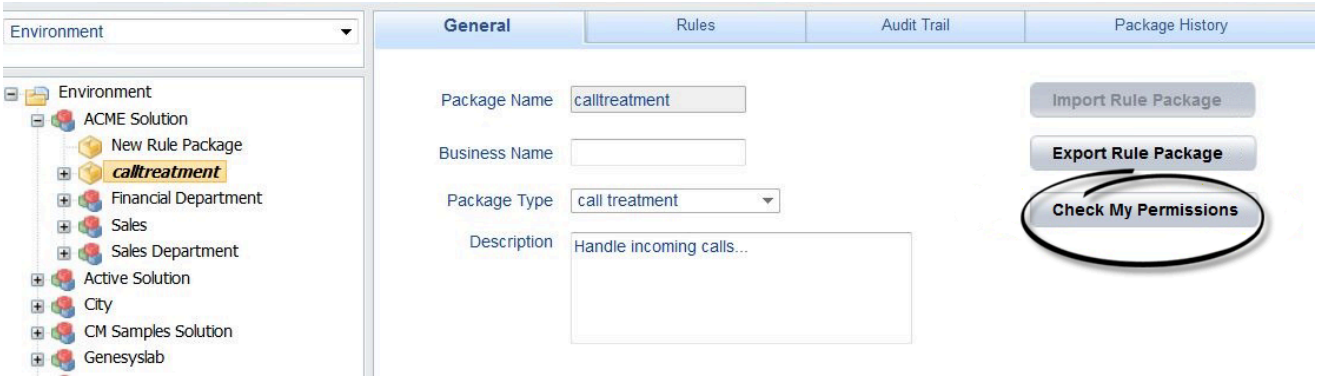
The mapping of the rule packages to Rule Package Level Roles is managed in Genesys Administrator or Genesys Administrator Extensions, in the options under section **Rule Package Level** of the **\Scripts\GRS Access Control\GRS Role Mappings** script. The example below is from Genesys Administrator.

> **Important**
>
> Because the delimiter in the list of roles is a comma, you can't use commas in the names of any role.



Viewing GRAT User Permissions

To enable GRAT users to view their current list of permissions, a **Check My Permissions** button is now also available at the rule-package level and shows the permissions at selected package level.

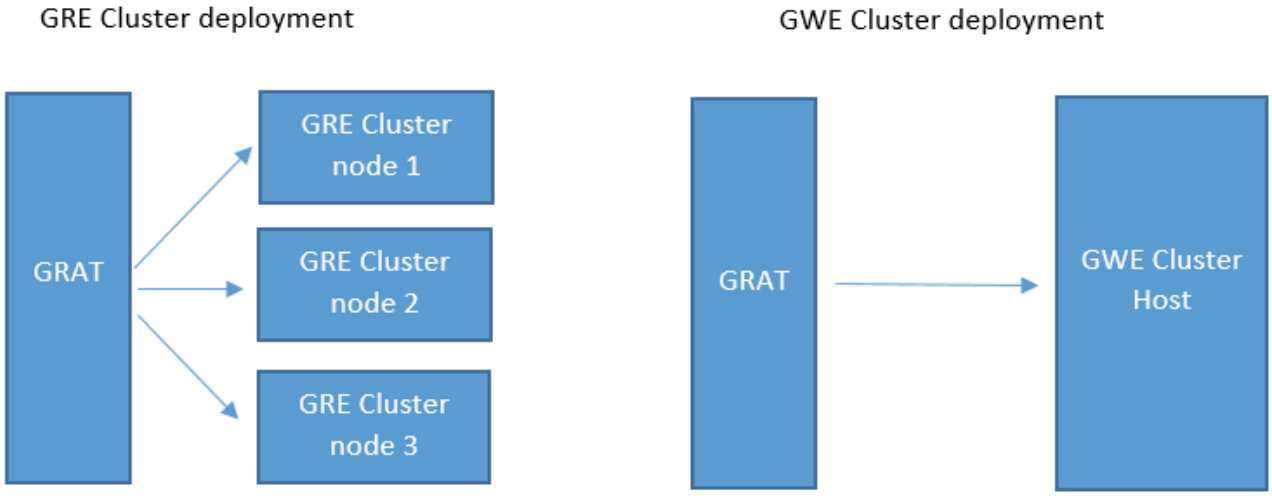


Support for Deployment to GWE Clusters

[+] DETAILS

In the case of GRE (Genesys Rules Engine) cluster deployment, GRAT gets the GRE nodes' information from the cluster and deploys the rule package individually to each of the cluster nodes.

But a GWE (Genesys Web Engagement) cluster is different—GWE cluster nodes are not available as connections. So the deployment must be targeted to the cluster host itself—that is, a host in the Configuration Server information of the GWE cluster.



The cluster host can be a GWE cluster node or a load balancer that randomly selects a GWE cluster node for deployment.

Streamlining of Removal of GRAT Nodes from Cluster

Removal of GRAT nodes from clusters has been streamlined.

[+] DETAILS

Removing a GRAT Node from the Cluster (pre-8.5.302)

1. Remove the GRAT node's Application ID from the list of connections in the cluster's Application object.
2. Manually remove the GRAT's entry in the `LOCAL_REVISIONS` table in the GRAT database (this is a single row, identified by the GRAT Application's DBID in the `JOURNAL_ID` column).

> **Warning**

> If this step is not performed, the clean-up thread (janitor) will not be effective.

Removing a GRAT Node from the Cluster (post-8.5.302)

In release 8.5.302, the manual cleanup of `LOCAL_REVISIONS` table (by the clean-up thread (janitor)) after node removal is not required. The cleaning task is now automated and controlled by the following new configuration options in the **settings** section of the GRAT cluster application:

- local-revisions-janitor-enabled
- local-revisions-janitor-sleep
- local-revisions-janitor-first-run-hour-of-day

Platform Support

- Support for Oracle 12c RAC implemented.
- Support for all variants of DB2 discontinued. GRAT will log a warning message if a DB2 variant is used as GRAT's database.

New in 8.5.301

Using GRAT Clusters for High Availability and Load Balancing

[+] DETAILS

Background

Before release 8.5.301, Genesys recommended maintaining a cold-standby backup GRAT server, connected to the same database repository as the primary, which could be initialized in the event of a primary GRAT server failure or disconnection.

A configuration option called `clear-repository-cache` could be set to force GRAT to clear and rebuild the local cache/search indices on startup. This allowed the backup server to synchronize with the repository, even if it had been shut down for months. Depending on the number of rule packages deployed, synchronization might be fast or slow.

What's New?

In release 8.5.301, you can now configure clusters of GRAT servers, which deliver much greater resilience and availability, enabling almost instant switchovers between GRAT nodes that are members of the cluster. All cluster members connect to the same database repository.

No single GRAT node is considered primary—they are all equal partners in the n-node cluster.

Each node maintains a journal of changes, which resides on a separate repository database table. Nodes can periodically poll the database to ensure that they mirror updates made on the other nodes. You can also add new nodes to the cluster and synchronize them with their peers.

- **New GRAT Cluster Application Template**

  The GRAT cluster configuration object must be based on a new application template—GRAT_Rules_Authoring_Tool_Application_Cluster_<*version*>.apd—t is delivered with this release. GRAT instances become members of the cluster when their Application object is added to the **Connections** tab in the cluster's Application object.

- **High Availability**

  An n-node cluster configuration can be used to deliver High Availability—for example, if a customer wants to use one GRAT server as the primary server and another as the secondary/backup warm-standby server, a load balancer could be configured to send all traffic to the primary server. The warm-standby instance periodically polls the journal table and keeps its cache/index files synchronized. In the event of a failure (or planned maintenance) of the primary GRAT server, the load balancer switches and sends traffic to the hot-standby secondary/backup GRAT server. The user would have to log in again, and any unsaved changes from their old session would be lost. However, they could resume in seconds instead of waiting for the cold-standby GRAT to become available.

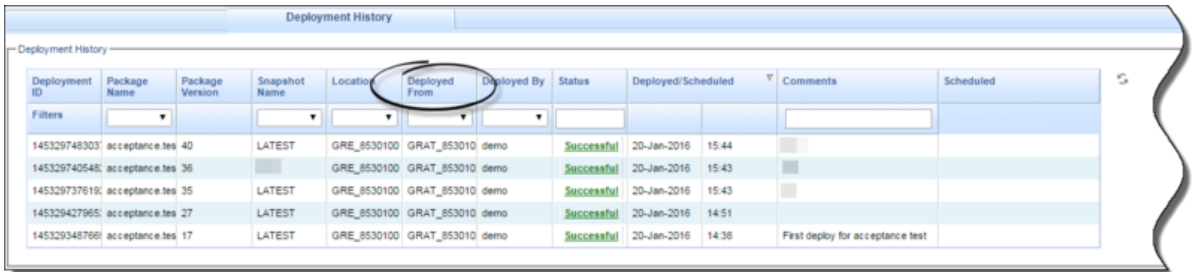
- **Load Balancing**

  A load balancer can evenly distribute the load across the available GRAT nodes in the cluster, or it can distribute the load based on other criteria, such as the geographic location of the browser. However, once a node has been selected, the load balancer must ensure that the same node is used for the duration of the session (session stickiness).

  The load balancer must provide session stickiness for the GRAT user interface requests by sending all the requests pertaining to a session to the same GRAT node that initiated the session after successful login. Similarly, in the case of the GRAT REST API, after successful login, the load balancer must send the subsequent requests to the same GRAT node that handled the login request. More information on the GRAT REST API Authentication mechanism is available **here.**

- **Scheduled Deployments of Rules Packages**

  Scheduled deployments of rules packages can now be viewed, edited and cancelled by GRAT users on GRAT instances that did not originally perform the scheduling—previously, only the original GRAT user/instance that performed the scheduling could do this. Any GRAT instance that modifies the scheduled deployment takes over the responsibility for handling that deployment. The deployment history, including a new **Deployed From** field that indicates which node last scheduled the deployment, is now available to all the nodes in the cluster (visible in the **Deployment History** tab).



> **Important**
>
> The new **Deployed From** field is also visible in standalone GRAT instances—it will display the ID of the standalone GRAT instance.

Deploying the GRAT Clustering Feature

1. Import the GRAT_Rules_Authoring_Tool_Application_Cluster_<*version*>.apd template into your Genesys configuration environment.
2. Create a GRAT cluster Application object based on the new template. Adjust the configuration options as needed.
3. For each GRAT node to be added to the cluster, add its Application ID as a connection in the cluster's Application object.
4. Add the DAP (Database Access Point) Application object to be used by the GRAT cluster as a connection in the cluster's Application object.

   > **Important**
   >
   > An existing standalone GRAT database can be used.

5. If you are re-using an existing standalone GRAT instance in the cluster, re-deploy the GRAT application

to ensure that its local cache is re-initialized.

> **Important**
>
> - Any change in the cluster configuration will only take effect upon re-start of the GRAT servers.
> - For high availability, GRAT instances must have a high-speed connection to the database. Slow connections may result in the type of issues commonly seen when the local repository cache is corrupted.

Removing a GRAT Node from the Cluster

1. Remove the GRAT node's Application ID from the list of connections in the cluster's Application object.
2. Manually remove the GRAT's entry in the `LOCAL_REVISIONS` table in the GRAT database (this is a single row, identified by the GRAT Application's DBID in the `JOURNAL_ID` column).

> **Warning**
>
> If this step is not performed, the clean-up thread (janitor) will not be effective.

New Configuration Options

Click here for details of the new configuration options that are set on the GRAT application cluster.

Additional Columns Support

- The previous limit of 30 columns in GRAT Decision Tables and Test Scenarios has been increased to 50 columns.

New in 8.5.3

Support for A/B/C Split Testing

[+] DETAILS

Overview

[+] DETAILS

What problem does A/B/C Split Testing solve?

If your organization has a complex set of rule packages and rules to define business decision-making, if you want to be able to compare alternative outcomes and scenarios, then rather than just changing a condition and hoping for the best you probably want to phase changes in slowly in order to measure the impact over time.

A/B/C Split Testing (hereafter, **Split Testing**) allows you to compare the business outcomes of alternative rule scenarios before rolling out significant changes to the way you make your business decisions. With Split Testing you can make, test and review changes incrementally to test their effects before committing to a particular change set.

For example, you might want to test several subtle changes in the way applications for credit cards are treated, in order to identify which has the best business outcome. You could split test for one income level against another, or one age range against another, or between different customer segments, or indeed any conditions in your rule package.

Flexible Approaches to Split Testing

Split Testing offers several approaches. For example, you can choose to:

- Unconditionally force either path A or B or C.
- Tie Split Testing to certain conditions (whether the customer is from a particular city, of a certain age, not a Gold customer, and so on).
- Weight the paths for A, B and C by percentage. This allows you to take a more statistical approach and use larger test data sets.
- Tie Split Testing to a business calendar (for example only do split testing on Fridays after 5 PM or Saturday morning 9 AM - 12 PM).

New Split Testing Template

A new template ships with GRS 8.5.3—GRSSplitTest_template.xml—and this provides some basic Facts (Conditions and Actions) for the Split Testing feature. To implement the feature, GRAT users must import this template (from the **Samples** folder) and attach it to all rules packages for which the Split Test functionality is required.

> **Important**
>
> The new template—`GRSSplitTest_template.xml`—is shipped with type `samples`. This

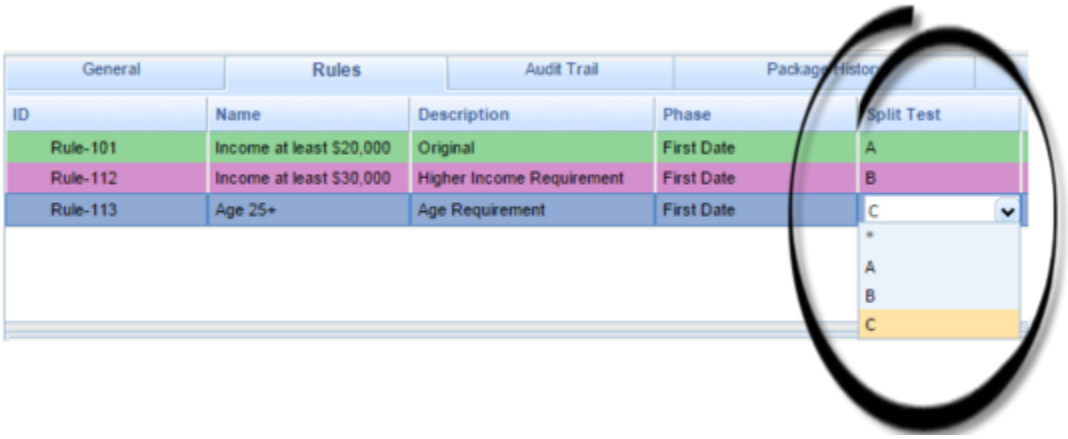
means it can only be added to rule packages of type `samples`, because GRAT only shows templates in the list with compatible types. To use this template with other rule package types, you can import the template from GRAT into GRDT, change the name (for example, `GRSSplitTestForMyType`) and the type (to match your rule package type) and publish to GRAT. Then you can use with another package type.

Changes in GRAT

[+] DETAILS

New Split Test Column in Rule Packages

A new Split Test column displays at the rule package level.

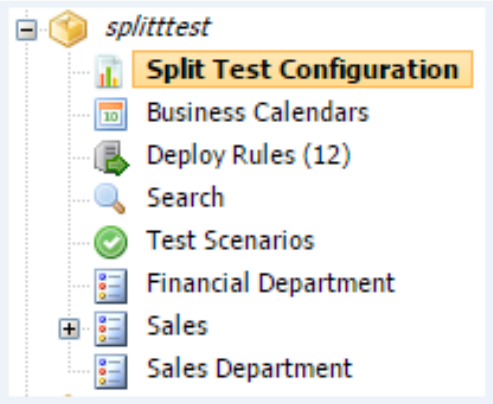


When the Split Test template has been imported into GRAT, this drop-down field will show the values imported from the template (A, B and C are the default values in the shipped template). If the template has not been imported, only the wildcard (*) symbol will be available.

When a path is selected in this column, the rows color-code for easier viewing. The wildcard symbol (*) means that the Split Test condition is ignored, so this rule will always fire.

New Split Test Configuration Node

A new node—Split Test Configuration—appears in the left navigation tree. You can use this node to create new (and display existing) Split Test rules that will apply at the rule package level. In rules created by clicking this new node, the **Split Test** column is hidden to avoid confusion.

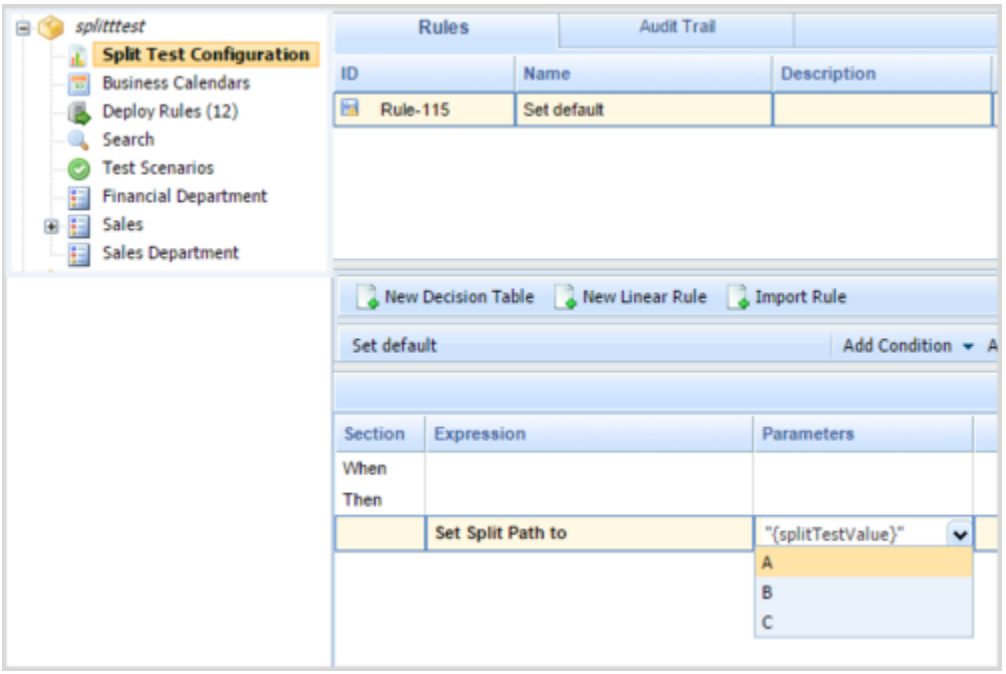


What Can I Do with Split Test Rules?

[+] Force the Split Test Path to A, B or C

Unconditionally force either path A or B or C

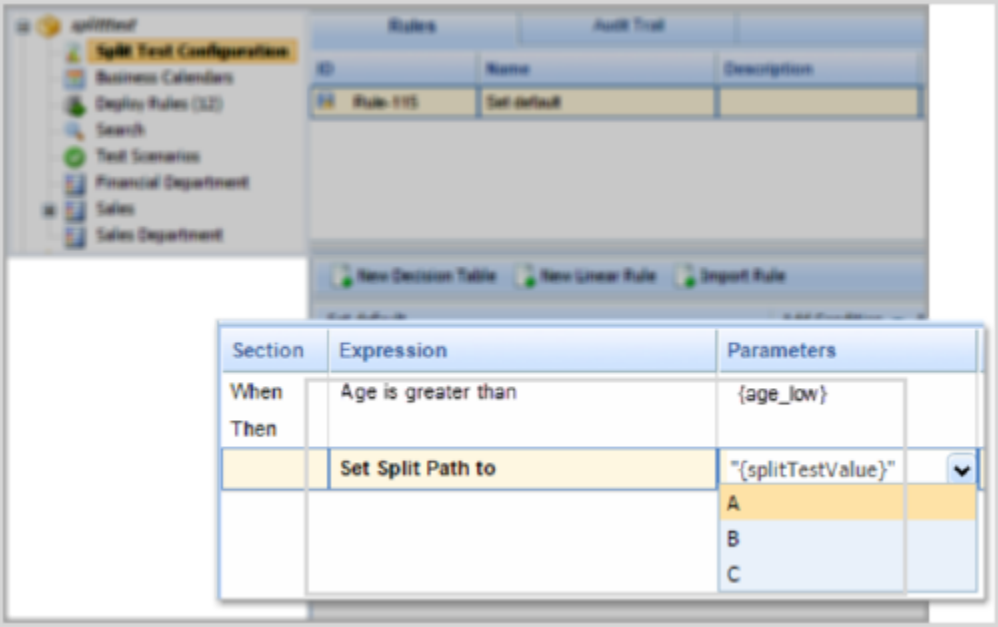
In rules created by clicking this new node, the `splitTestValue` can be selected as a Parameter for the rule. You can use this simple rule to force the Split Test path to any of the values in the template (the default shipped values in the template are `A`, `B` and `C`, but you can change these in the template if required).



[+] Tie a Split Test Rule to a Condition

Tie Split Testing to a Condition

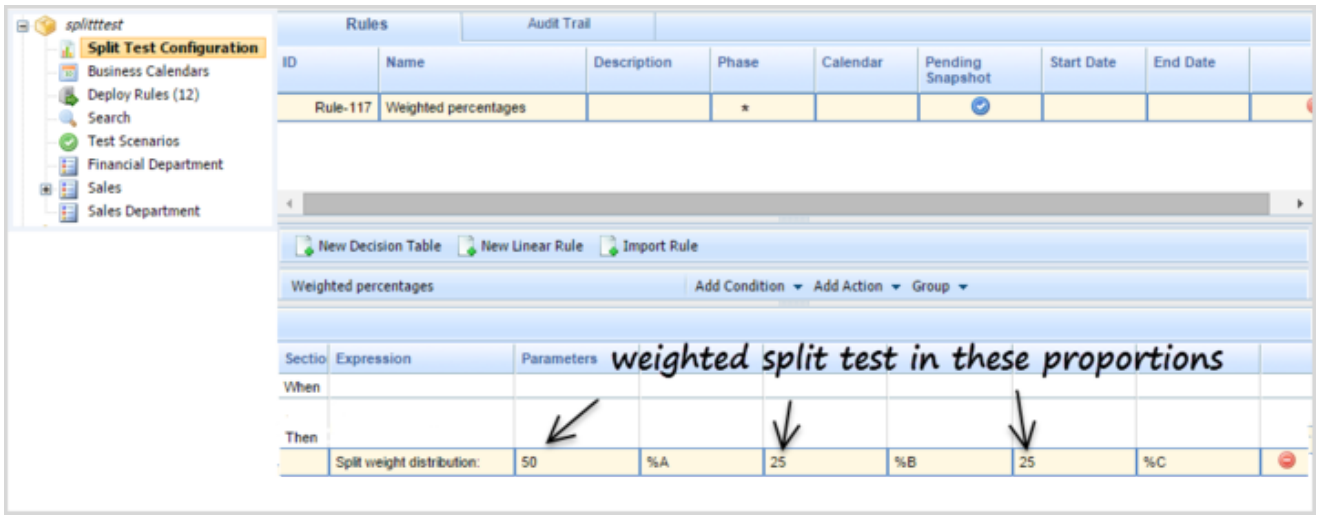
To add more flexibility and precision to your use of Split Test rules, you can add to your rule any other condition available from the template(s) attached to the rule package. The example below has a lower age limit:



[+] Set Up a Weighted Distribution

Weight the paths for A, B and C by percentage

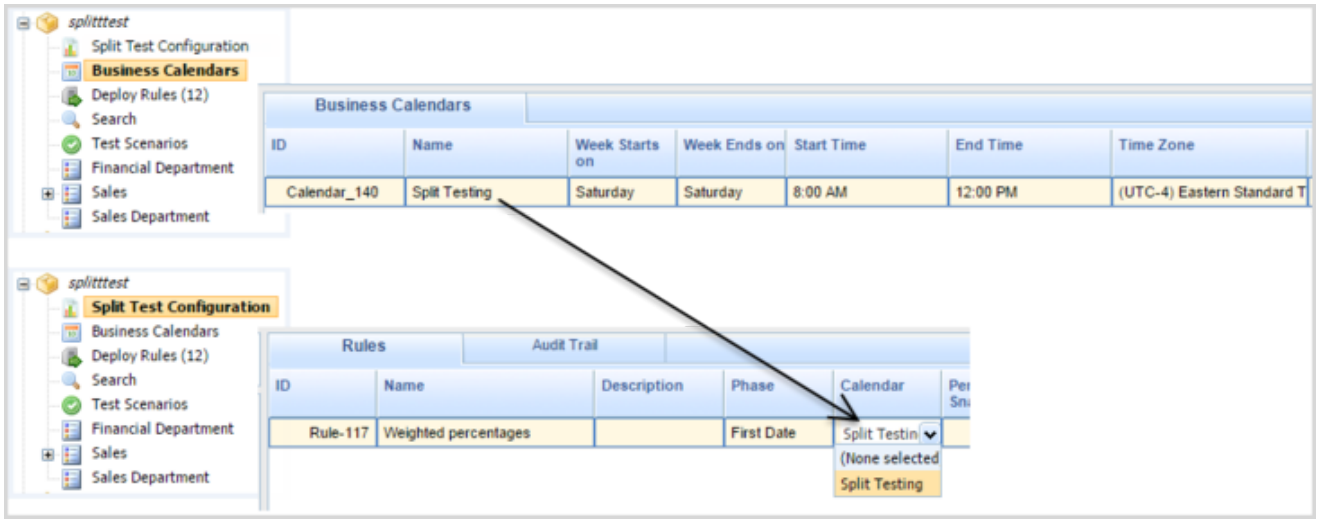
You could add a condition to perform Weighted Distribution Split Testing. In this example the Split Test rule distributes the incoming rule evaluation requests across the A, B and C paths in the percentage proportions 50/25/25. This should look something like this:



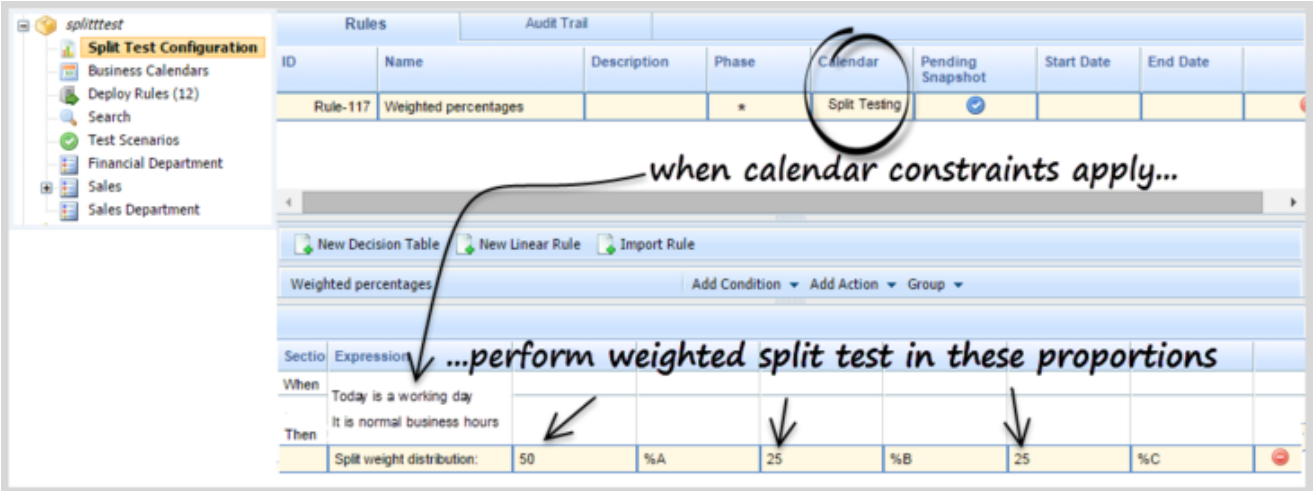
[+] Tie a Split Test Rule to a Business Calendar

Tie Split Testing to a Business Calendar

You can also tie Split Testing to a Business Calendar. For example, you might prefer to keep Split Testing outside normal business hours, or maybe run Split Testing in response to specific customer offers or other commercial events. To do this you can either use an existing Business Calendar or create a new one for your specific purpose. You then add this calendar to the rule to which you want to apply Split Testing.



So, when the parameters set for the Business Calendar apply, the weighted Split Testing begins and ends automatically.



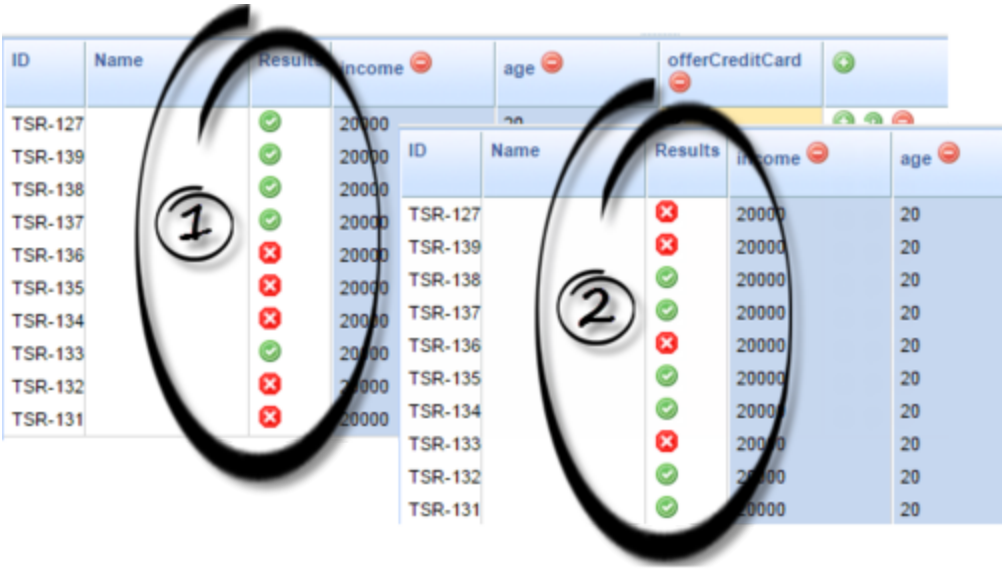
Using Test Scenarios to Check Your Split Test Logic

[+] Using Test Scenarios to Check Your Split Test Logic

GRAT's Test Scenario feature allows you to run pre-deployment checks on the logic of any rule package that you want to deploy. Because A/B/C Split Testing enables an additional level of logic to be built into a rule package, it's advisable to run more complex Split Test logic through a Test Scenario before deploying it to a production rules engine.

For example, if you wanted to check the Weighted Distribution example above before deploying the rule package to a production engine, you could create a test scenario to have 10 rows of identical data that only qualify when running path **A**. Each row in the test scenario simulates a separate rule evaluation and will therefore apply the Split Test Configuration rules first. If we see a green tick, then the original **A** path was taken. If we see a red cross, then either the **B** or **C** path was taken.

Because we configured the **A** path for 50% of the time, we should see about 50% green ticks every time we hit the **Run Test Scenario** button. Here are the outcomes of running the test scenario twice:



After testing that the A/B/C split logic is correct, you can deploy your rule package. At this point, the A/B/C split logic will run as you have it configured during normal rule evaluation. To adjust the A/B/C configuration (for example, make the **A** path 25% instead of 50% and the **B** path 75% instead of 50% and so on), you must make the changes in GRAT and then redeploy the rule package.

Allowing an Application to Override Split Test Configuration Rules

[+] Allowing an Application to Override Split Test Configuration Rules

In addition to setting the A/B/C path in the Split Test Configuration section, it is possible for the invoking application (GVP, IVR, ORS and so on) to set the value. This can either be used as a default, or can be used to override the logic in the Split Test Configuration.

If you want the invoking application to be able to override the normal logic in Split Test Configuration, then you can simply add the new `Split Test Path is not set` condition to your rule.

Section	Expression	Parameters
When		
	Split test path is not set	
Then		
	Set Split Path to	A

This new condition will check whether the value has already been passed in by the invoking application. If it has been passed in, then the Split Test logic will be bypassed. If not, then the normal Split Test rule logic will apply.

Changes to Platform Support

- Support for Java 8. In release 8.5.3 GRAT and GRE require Java 8.
- Support for JBOSS 7.x is discontinued.
- Support for RHEL 5 32-bit is discontinued.

New in 8.5.2

New Features in 8.5.2 (new document)

New in 8.5.1

New Features in 8.5.1 (new document)

New in 8.5.0

New Features in 8.5.0 (new document)

Migration

8.5.303

Migration is as for 8.5.3 from 8.5.2 subject to the following important points:

Important

- Business node access is required to view entries in package history generated after this release. The **Rule Package History - Admin View** privilege is required in order to override the new access check mechanism to view the previous package history.
- The **Changed By** column in Package History was previously accessible without a role privilege, but in release 8.5.303, users require a new role privilege—**Rule Package History - View Changed By**—to view this column.

Migration to 8.5.3 from 8.5.2

1. Undeploy the 8.5.2 `.war` file from your application server.

   ### Important

   Do not just copy the 8.5.3 `.war` file over the 8.5.2 `.war` file in your application server directory. Genesys recommends undeploying the previous file first and letting the application server clean up its files, then deploying the new `.war` file.

2. Upgrade your application server to use Java 8—see below.
3. Deploy the 8.5.3 `.war` file to your application server.
4. Set the values of any of the new configuration options.
5. Log into 8.5.3 Genesys Rules Authoring Server.

8.5.302.04 Requirements

8.5.302.04 requires Genesys Administrator 8.1.305.04 (minimum) to support GRAT's new RBAC features.

Java 8 Support

GRS 8.5.3 is now on the Java 8 platform. Java 8 gives improved performance and security over Java 7. GRS takes advantage of some of the new features in Java 8 to give you a more robust platform.

Since GRS 8.5.3 has been built on Java 8, it will no longer execute on the Java 7 platform. Therefore, your application servers will have to be reconfigured to use Java 8.

For Tomcat users

1. Download and install the **Java SE Runtime Environment 8** from Oracle.
2. Download and install the latest Tomcat 8 or 8.5 distribution (GRS does not currently support Tomcat 9).
3. During the installation, choose the Java 8 SDK location when prompted.
4. Set the appropriate memory settings. You can find recommendations for performance tuning **here**. You may need to adjust these depending upon your deployment.

> **Important**
>
> There is no longer a **`permgen`** setting for Java 8.

5. Deploy the 8.5.3 GRAT or GRE **`.war`** file.

For WebSphere Users

Currently, IBM only supports Java 8 on the WebSphere Liberty distribution. For configuration information for WebSphere Liberty click **here**.

A/B/C Split Testing Feature Template

A new template ships with GRS 8.5.3—**`GRSSplitTest_template.xml`**—and this provides some basic Facts (Conditions and Actions)for the Split Testing feature. To implement the feature, GRAT users must import this template (from the **Samples** folder) and attach it to all rules packages for which the Split Test functionality is required.

> **Important**
>
> The new template—**`GRSSplitTest_template.xml`**—is shipped with type **samples**. This means it can only be added to rule packages of type **samples**, because GRAT only shows templates in the list with compatible types. To use this template with other rule package types, you can import the template from GRAT into GRDT, change the name (for example, **`GRSSplitTestForMyType`**) and the type (to match your rule package type) and publish it to GRAT. Then you can use it with another package type.

Migration to 8.5.2 from 8.5.1

> **Warning**
> During migration, the format of DRL files in GRE's `deployed-rules-directory` will be changed from `.drl` to `.di`, so Genesys recommends backup of `.drl` files before migrating GRE to 8.5.2

1. Undeploy the 8.5.1 `.war` file from your application server.

   > **Important**
   > Do not just copy the 8.5.2 `.war` file over the 8.5.1 `.war` file in your application server directory. Genesys recommends undeploying the previous file first and letting the application server clean up its files, then deploying the new `.war` file.

2. Deploy the 8.5.2 `.war` file to your application server.
3. Set the values of any of the configuration options.
4. Log into 8.5.2 Genesys Rules Authoring Server.

Using the 8.5.2 Clustering Improvements

If you want to take advantage of the clustering improvements in release 8.5.2, you must import the "smart application" template, as follows;

From Genesys Administrator

1. Navigate to **Application Templates**.
2. Click **Upload Templates** (upper right corner).
3. Choose the `.apd` file `Genesys_Rules_Engine_Application_Cluster_852.apd`.
4. Click **Save** and **Close**.
5. Go to **Applications**.
6. Create a **New** application.
7. For the template, choose the application template that you just created in steps 1-4.
8. Fill in the mandatory fields, including the host (which is not used, but GA requires you to complete this field).
9. In the **Connections** section, add each GRE in the cluster.
10. On the **Options** tab, configure the three auto-synch options (see GRE Configuration options):
    - auto-synch-rules

- auto-synch-rules-interval
- auto-synch-rules-at-startup

See also **Cluster Improvements for Cloud**.

Migration to 8.5.1 from 8.5.001

1. Undeploy the 8.5.001 `.war` file from your application server.

> **Important**
> Do not just copy the 8.5.1 `.war` file over the 8.5.001 `.war` file in your application server directory. Genesys recommends undeploying the previous file first and letting the application server clean up its files, then deploying the new `.war` file.

2. Deploy the 8.5.1 `.war` file to your application server.
3. Set the values of any of the new configuration options.
4. Log into 8.5.1 Genesys Rules Authoring Server.

Migration to 8.5.001 from 8.5.0

1. Undeploy the 8.5.0 `.war` file from your application server.

> **Important**
> Do not just copy the 8.5.0001 `.war` file over the 8.5.0 `.war` file in your application server directory. Genesys recommends undeploying the previous file first and letting the application server clean up its files, then deploying the new `.war` file.

2. Deploy the 8.5.0001 `.war` file to your application server.
3. Set the values of any of the new configuration options.
4. Log into 8.5.0001 Genesys Rules Authoring Server.

Migration to 8.5.0/8.5.001

From 8.1.4

1. Undeploy the 8.1.4 `.war` file from your application server.

> **Important**
> Do not just copy the 8.5.0 `.war` file over the 8.1.4 `.war` file in your application server directory. Genesys recommends undeploying the previous file first and letting the application server clean up its files, then deploying the new `.war` file.

2. Deploy the 8.5.0 `.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 8.5.0 `.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.

3. Log into 8.5.0 Genesys Rules Authoring Server.

From 8.1.3

1. In 8.1.4, the repository database configuration was 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 using Configuration Server**
- **Creating the Rules Repository Database using Genesys Administrator**

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

> **Important**
> Do not just copy the 8.5.0 `.war` file over the 8.1.3 `.war` file in your application server directory. Genesys recommends undeploying the previous file first and letting the

application server clean up its files, then deploying the new `.war` file.

- Deploy the 8.5.0 `.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 8.5.0 `.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 8.5.0 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 8.5.0 (leaving the old one in place).
4. Install 8.5.0 Genesys Rules System.
5. Start 8.5.0 Genesys Rules Authoring Server. This creates the tables inside the new database.
6. Log into 8.5.0 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 8.5.0 Genesys Rules Engine(s).
10. Optionally, from the Genesys Rules Deployment Tool, you may import the templates from the 8.5.0 Genesys Rules Authoring Server.

See the GRAT online Help for explicit steps for **importing** and **exporting** templates, and **importing** and **exporting** rules packages.

> **Important**
> Running an 8.5.0 Rules Authoring Server against an 8.1.2 repository will result in a corrupted repository that will no longer be useable by any version of the Rules Authoring Server.

Repository Performance Enhancement for Oracle 11 users

For Oracle users, a performance enhancement is available that is enabled only when you create a new database repository when either initially installing, or migrating to, 8.5.x. For a new installation with a new Oracle database repository, no additional steps are required. If you are migrating to GRS release 8.5.x from an earlier release, to take advantage of the enhancement, do the following:

1. Click on each tenant and export the templates associated with that tenant as an XML file.
2. Click on each rule package that you wish to migrate and export as an XML file.
3. Create a new database for GRAT 8.5.x.
4. Start 8.5.x Genesys Rules Authoring Server.

With the enhancement, when GRAT initializes, it creates new database tables based on a new optimized schema, and database performance is improved.

> **Important**
>
> If you re-use your existing 8.1.3/8.1.4 repositories, GRAT does not re-create the tables with the new schema. In this scenario, there is no performance enhancement.

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 8.5.000.xx to 8.5.100.xx).

Procedure

1. Install the new installation package.
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. If installing GRAT, log into the Genesys Rules Authoring Server. If installing GRE, navigate to the home page to ensure it is operating successfully (for example; http://myserver:8080/genesys-rules-engine).

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**
- **Configure DAP Parameters/JDBC Drivers**

Installing Genesys Rules System Task Summary

The following table outlines the task flow for installation of GRS 8.5.x. 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 using Configuration Manager** • **Configuring the Rules Repository database using Genesys Administrator** • **Configure DAP Connection Parameters/ JDBC Drivers**

3. Install the Genesys Rules Engine	• Genesys Administrator:**Deploying the Genesys Rules Engine in Genesys Administrator**
	  • **Creating an Application Cluster in Genesys Administrator**
	• Configuration Manager:**Creating the Genesys Rules Engine Application object in Configuration Manager**
	  • **Creating an Application Cluster 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**

Objective	Related Procedures and Actions
	• **Creating a GRAT Application Cluster** • Configuration Manager: • **Creating the Genesys Rules Authoring Tool Application objects in Configuration Manager** • **Installing the Genesys Rules Authoring Tool**
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 8.1.2 (or earlier) Genesys Rules Engines. This step is not necessary for standard rule packages that have been deployed to 8.1.3 or later 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.

Configuring the Rules Repository Database using Configuration Manager

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

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.

> **Important**
> Genesys Rules System 8.5.3 can use only Java 8. Java 7 is no longer supported in release 8.5.3.

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

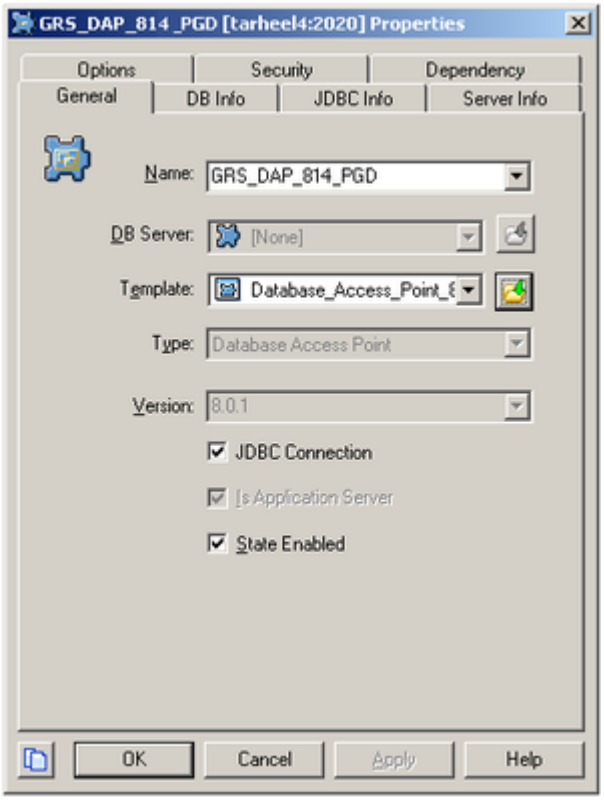
Either Oracle 11g or 12c or Microsoft SQL or DB2, or PostGRE SQL 9.x

> **Important**
> Support for Oracle 10g is discontinued in GRS 8.5.

Procedure

1. Create a new database using the normal DBMS procedures for the database type you are working with. There is no need to create tables within this database—it will be populated by GRAT automatically on startup.
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 0K. This opens the **Properties** dialog box for the new DAP `Application` object.

4. On the **General** tab:



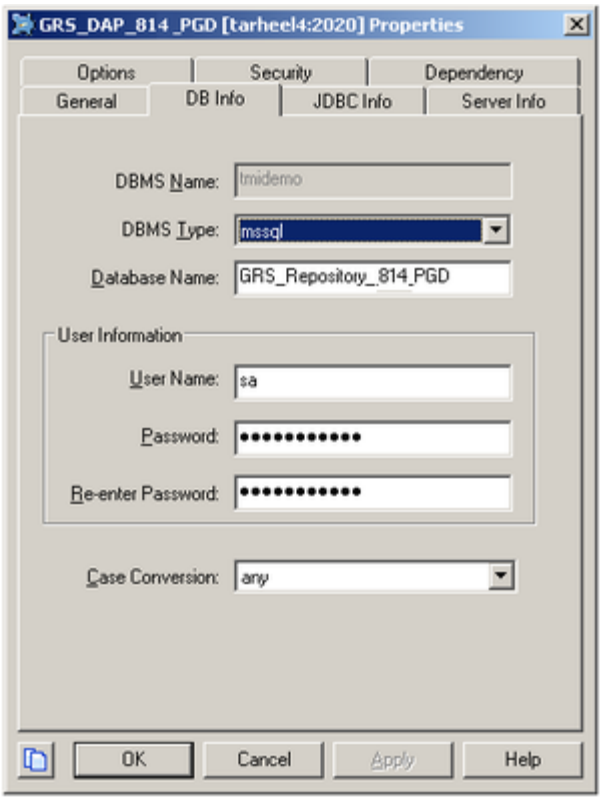
   a. Enter a name for the DAP. A DAP can have the same name as the database itself. However, it is recommended that you make their names unique if you are using multiple access points for the same database.

   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. The database username must have full permissions to this database in order to create the tables. GRAT uses these database credentials when it starts for the first time in order to create the necessary database schema.

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

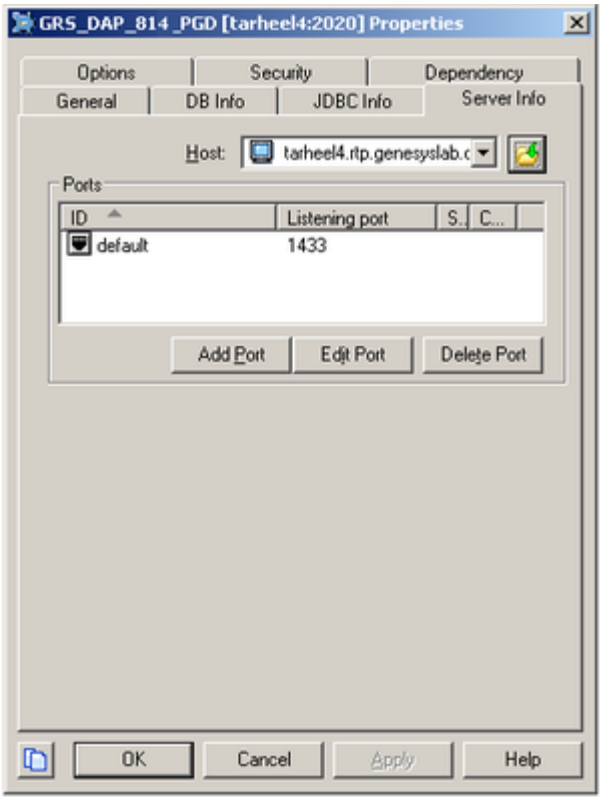
3. On the **JDBC Info** tab, set the following values:

   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 Repository 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 8.5.1 can use only Java 7. Java 6 is no longer supported in release 8.5.1.

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.

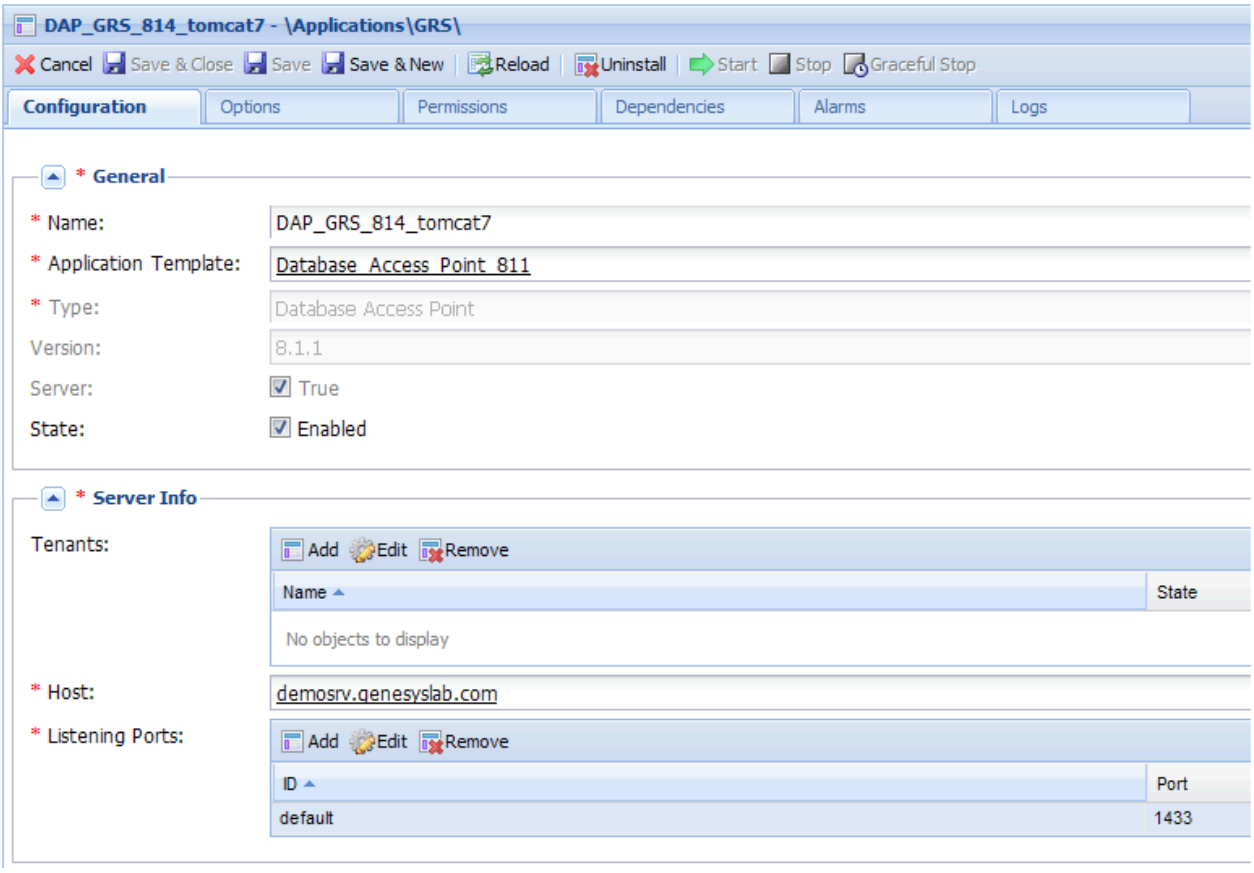
Prerequisites

Either Oracle 11g or 12c or Microsoft SQL or DB2 or PostGRE SQL 9.4

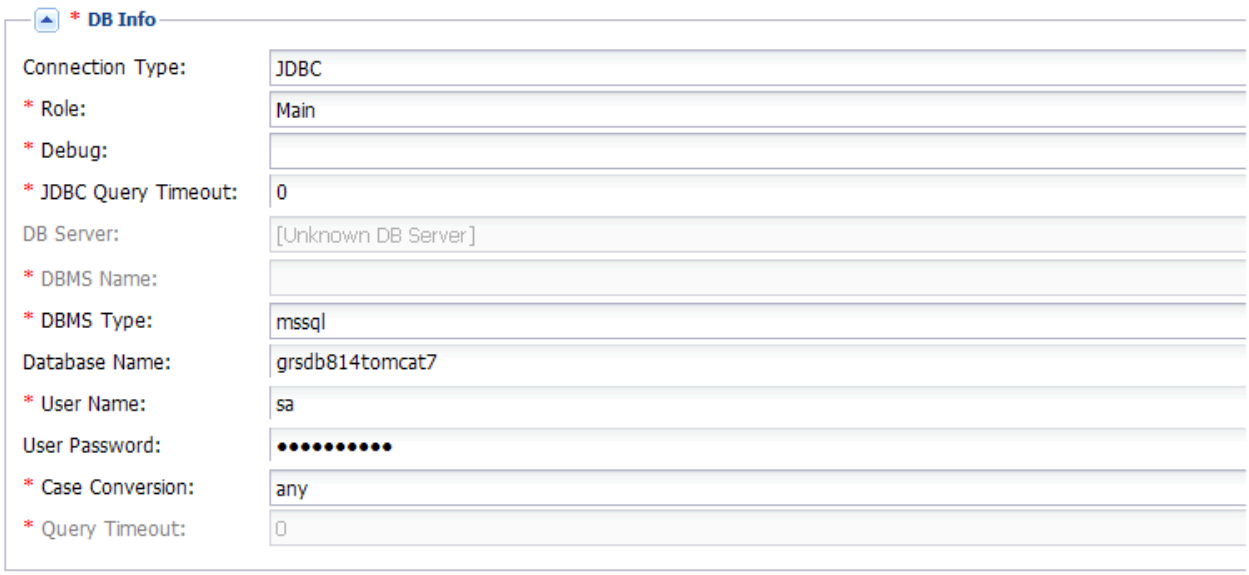
> **Important**
> Support for Oracle 10g is discontinued in GRS 8.5.

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, Oracle or PostGRE SQL).
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.

Configure DAP Connection Parameters/ JDBC Drivers

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**).

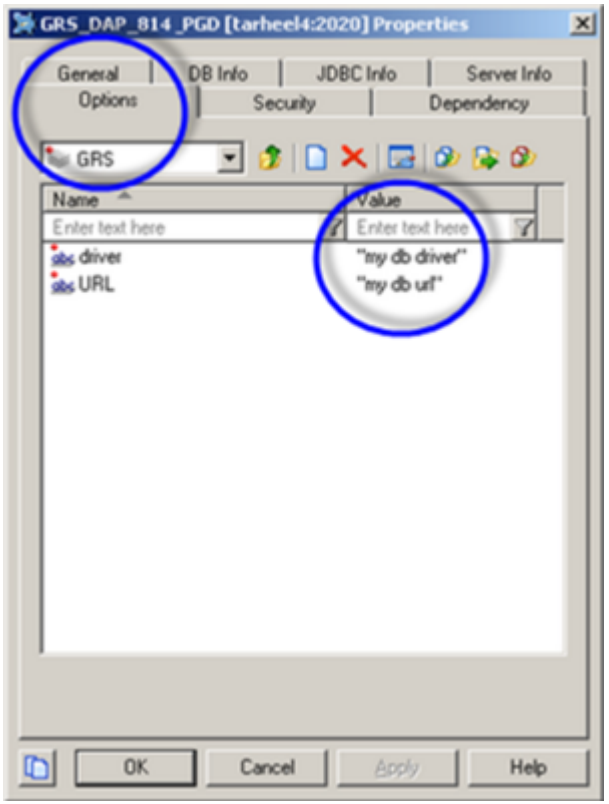
Default Values

> **Important**
>
> You should consult your database vendor's documentation for specific information. Choose the most current version of the JDBC driver that supports your operating system and Java version. The JDBC driver(s) must be copied to the **lib** directory of your application server. Typical examples are shown in the table.

Database Type	Connector Class	Default Database URL	Example JDBC Driver to be Copied (see note above)
MSSQL	com.microsoft.sqlserver.jdbc.SQLServerDriver	jdbc:jtds:sqlserver://{host}:{port};databaseName={database_name}	sqljdbc42.jar (from 8.5.3 onwards)
Oracle	oracle.jdbc.driver.OracleDriver	jdbc:oracle:thin:@//{host}:{port}/{SID}	ojdbc6.jar
DB2	com.ibm.db2.jcc.DB2Driver	jdbc:db2://{host}:{port}/{database_name}	db2jcc.jar db2jcc_license_cu.jar
PostGRE SQL	org.postgresql.Driver	"jdbc:postgresql://{host}:{port}/databaseName"	postgresql-9.3-1102.jdbc41.jar

Overriding Default Values on the Options Tab

More advanced users can use the DAP's `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**.

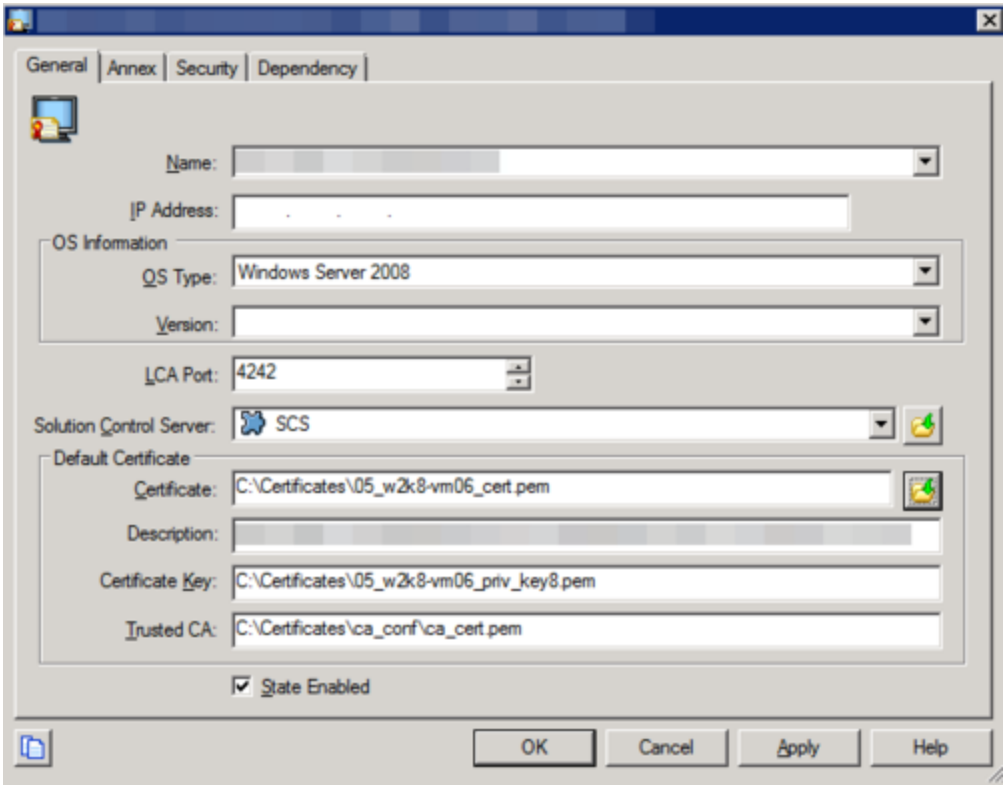
Security Certificates in Initial Release of 8.5.1

GRE and GRAT applications are unable to understand thumbprint certificates from the Windows Microsoft Certificate Store while connecting to Configuration Server or Message Server. Consequently they cannot establish a TLS connection to either Configuration Server or Message Server, and a security error is generated, such as this:

```
15:14:31.445 Alarm 21363 [ServerConnectionMonitor][Thread-2]: connect(): caught exception
while
opening connection to server '<GRAT/GREServerVersion>'. Nested exception: Could not configure
TLS.
```

Workaround

Create certificates in .PEM format and private-keys in PKCS#8 format—see the example below.



Paths to the physical certificates can be configured either on the Application or the Connection level, but not on the Host level (despite this being a general Genesys recommendation). For Host level certificates, if GRAT and GRE are located on the same host as Configuration Server or Message Server or other C++ applications, the secured connection will not be established because C++ based applications do not accept PKCS#8 format.

You can convert private-key from PEM to PKCS#8 format using the following OpenSSL command:

```
openssl pkcs8 -topk8 –nocrypt -in tradfile.pem -out p8file.pem
```

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 Summary

1. Import the installation package into Genesys Administrator.
2. Install the GRE IP.
3. Configure the Rules Engine application.

Procedure

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.

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**.

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`.
2. On the **Tenants** tab, add the Tenants that will be available to the Rules Engine.
3. On the **Connections** tab, add a connection to Message Server if you want to use network logging.
4. 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.
5. There are several optional configuration options to work with in the Configuration Options Reference Guide.
6. 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 Summary

1. Import the GRE application template into Configuration Manager.
2. Configure the Rules Engine application.

Procedure

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

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.`.
4. Click **OK** to save the template.

Configure the Rules Engine application

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 in the **[log]** section and the **[settings]** sections.
10. 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. Enter Application Name and click **Install**.
10. Click **Finish**.

End

Next Steps

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

Importing the GRE "Smart Cluster" Template

From Genesys Administrator

1. Navigate to **Application Templates**.
2. Click **Upload Templates** (upper right corner).
3. Choose the **Genesys_Rules_Engine_Application_Cluster_853.apd** file.
4. Click **Save** and **Close**.
5. Go to **Applications**.
6. Create a **New** application.
7. For the template, choose the application template that you just created in steps 1-4.
8. Fill in the mandatory fields, including the host (which is not used, but GA requires you to complete this field).
9. In the **Connections** section, add each GRE in the cluster.
10. On the **Options** tab, configure the three auto-synchronization options:
    - auto-synch-rules
    - auto-synch-rules-interval
    - auto-synch-rules-at-startup

Creating an Application Cluster in Configuration Manager

> **Important**
>
> If you do not require the auto-synchronization feature available in release 8.5.2, you can continue to use the cluster configuration for 8.5.1—click **here** for details. Only Genesys Rules Engine (GRE) supports the auto-synchronization feature.

You can use a Configuration Server application template of type `Genesys_Rules_Engine_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 a user deploys a package in GRAT, the deployment target list may contain cluster application names. Unlike release 8.5.1, partial deployments are possible when the allow-partial-cluster-deployment option in GRAT is set to value `true`. If this option is set to `false`, deployed packages are placed in service only after the deployment to all engines in the cluster is successful.

If deployment to any of the engines fails, details of the failure(s) are shown to the GRAT user and logged in the GRAT log. Partial deployments, where configured, are also displayed to the GRAT user and in the GRAT log.

Procedure

1. Import an application template of type `Genesys_Rules_Engine_Application_Cluster`, if one does not already exist in your environment.
2. Create a Configuration Server application of type `Genesys_Rules_Engine_Application_Cluster`.
3. Enter a host and other mandatory information.

   > **Important**
   >
   > Host information is required by the configuration user interface, but it is not used.

4. Add as connections to this cluster application the engine applications you wish to treat as a cluster. For each connection be sure to select a Port ID (it does not matter which port) for the Rules Engine Web Application (either GRE or Genesys Web Engagement).
5. Add the cluster application as a connection to the GRAT application.
6. 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

> **Important**
>
> If you do not require the auto-synchronization feature available in release 8.5.2, you can continue to use the cluster configuration for 8.5.1—click **here** for details. Only Genesys Rules Engine (GRE) supports the auto-synchronization feature.

Purpose

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

Procedure

1. Import an application template of type `Genesys_Rules_Engine_Application_Cluster`, if one does not already exist in your environment.
2. Create a Genesys Administrator application of type `Genesys_Rules_Engine_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. Make sure to select Genesys Rules Authoring Tool (GRAT) server as the Host in **Server Info** section.
6. Add as connections to this cluster application the engine applications you wish to treat as a cluster. For each connection be sure to select the default 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.

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"/>
```

Security Certificates in Initial Release of 8.5.1

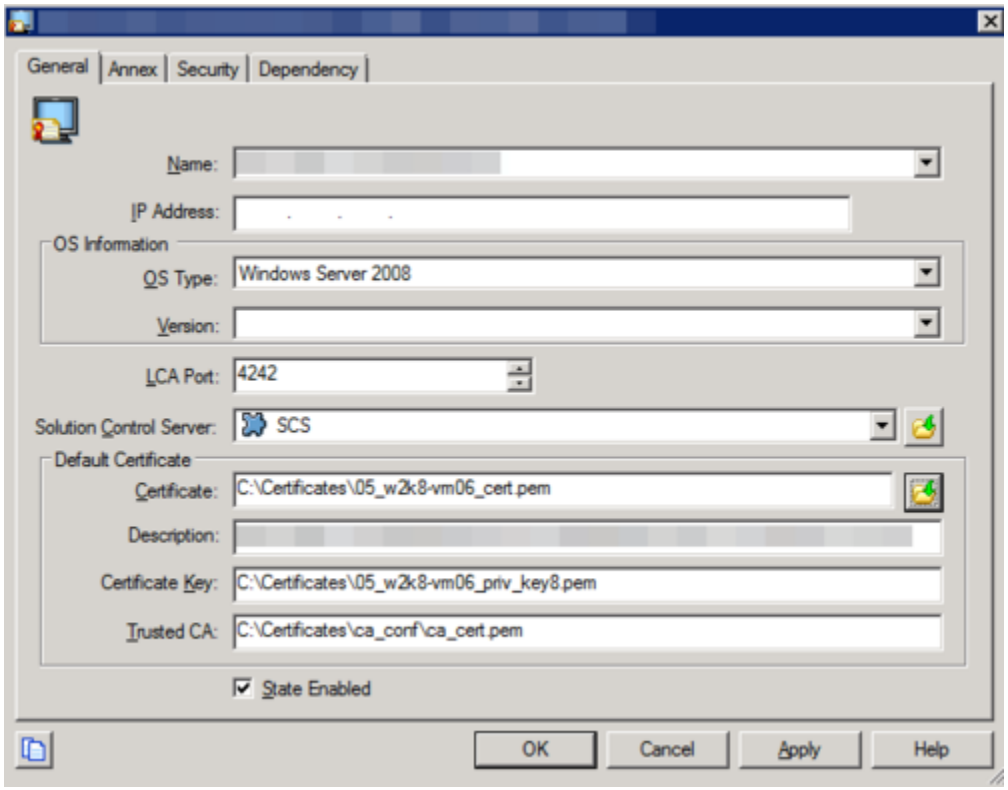
GRE and GRAT applications are unable to understand thumbprint certificates from the Windows Microsoft Certificate Store while connecting to Configuration Server or Message Server. Consequently they cannot establish a TLS connection to either Configuration Server or Message Server, and a security error is generated, such as this:

```
15:14:31.445 Alarm 21363 [ServerConnectionMonitor][Thread-2]: connect(): caught exception
while
opening connection to server '<GRAT/GREServerVersion>'. Nested exception: Could not configure
```

TLS.

Workaround

Create certificates in .PEM format and private-keys in PKCS#8 format—see the example below.



Paths to the physical certificates can be configured either on the Application or the Connection level, but not on the Host level (despite this being a general Genesys recommendation). For Host level certificates, if GRAT and GRE are located on the same host as Configuration Server or Message Server or other C++ applications, the secured connection will not be established because C++ based applications do not accept PKCS#8 format.

You can convert private-key from PEM to PKCS#8 format using the following OpenSSL command:

```
openssl pkcs8 -topk8 –nocrypt -in tradfile.pem -out p8file.pem
```

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 Summary

1. Import the GRAT IP into Genesys Administrator.
2. Install the GRAT IP.
3. Configure the GRAT application.

Import the GRAT IP into Genesys Administrator

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**.
5. 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`.
6. Select **Next** to start the import.
7. Click **Finish** when the import is complete.

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.

Configure the GRAT application

Configuration options are described here.

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 Summary

1. Import the GRAT application template for the server.
2. Import the GRAT application template for the client.
3. Configure the server application.
4. Configure the client application.

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_853.apd**.
    - For Management Framework 8.1 and earlier, select **Genesys_Rules_Authoring_Generic_Server_853.apd**.
4. Click **OK** to save the template.

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_853.apd**.
5. Click **OK** to save the template.

Configure the server application

Configuration options are described here.

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

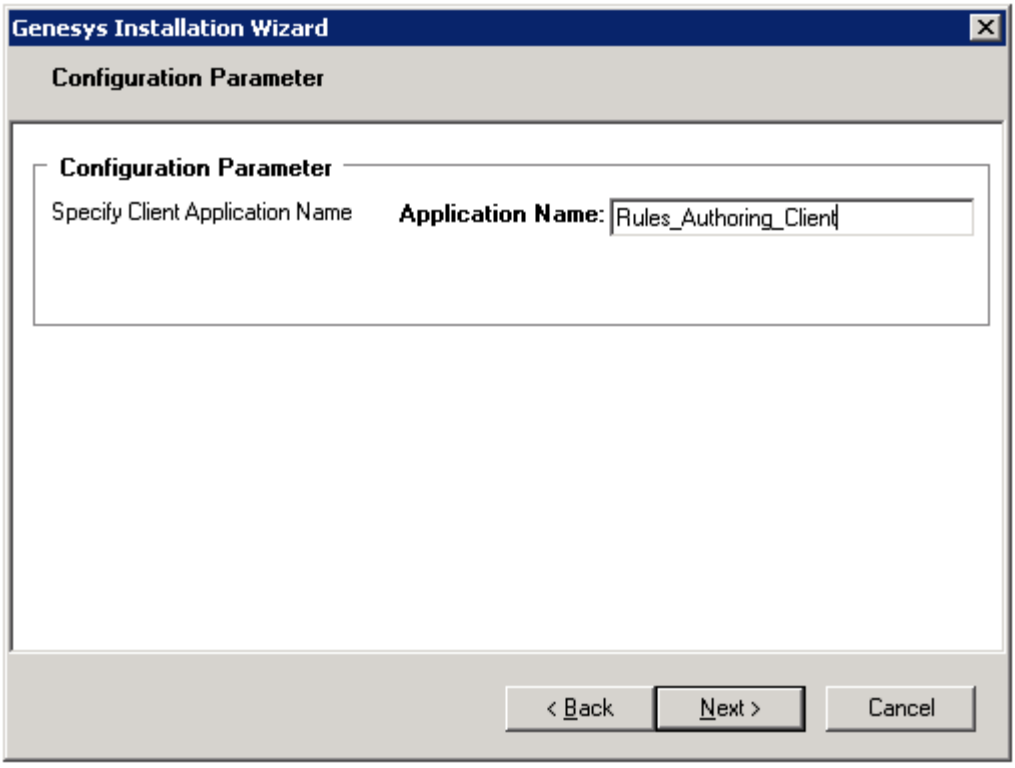
To run the installation package for the GRAT after the applications are configured in Configuration Manager.

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.

Creating the GRAT Cluster Application Object

Using Genesys Administrator

1. Navigate to **Application Templates**.
2. Click **Upload Templates** (upper right corner).
3. Choose the `.apd` file `GRAT_Rules_Authoring_Tool_Application_Cluster_853.apd`.
4. Click **Save** and **Close**.
5. Go to **Applications**.
6. Create a **New** GRAT cluster Application based on the new template. Adjust the **configuration options** as needed.
7. Fill in the mandatory fields, including the host (which is not used, but GA requires you to complete this field).
8. For each GRAT node to be added to the cluster, add its configuration Application as a connection in the cluster Application.
9. Add the DAP (Database Access Point) configuration Application to be used by the GRAT cluster as a connection in the cluster Application.

> ### Important
> An existing standalone GRAT database can be used.

10. If you are re-using an existing standalone GRAT instance in the cluster, re-deploy the GRAT application to ensure that its local cache is re-initialized.

> ### Important
> Any change in the cluster configuration will only take effect upon re-start of the GRAT servers.

> ### Important
> For high availability, GRAT instances must have a high speed connection to the

database. Slow connection may result in the type of issues commonly seen when the local repository cache is corrupted.

Deploying .WAR Files

> **Important**
>
> Make sure you have configured DAP connection parameters and JDBC drivers before 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 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.

> **Important**
>
> GRS 8.5.3 requires Java 8 and as a consequence of this requirement, you must deploy WebSphere Liberty—this is the only Websphere application server that supports Java 8.

- Refer to your Tomcat documentation for specific deployment instructions.
- WebSphere 8.5 configuration information can be found **here**.
- WebSphere Liberty configuration information can be found here.

Genesys recommends using **the information here** to tune performance for GRE and GRAT. This may need to be adjusted based on your configuration and depending upon any other applications deployed to your application server.

Configuring WebSphere 8.5

Procedure

1. Extract:
   - `httpclient-[VERSION].jar`
   - `httpcore-[VERSION].jar`
   - `jackson-core-asl-[VERSION].jar`
   - `jackson-mapper-asl-[VERSION].jar`

   from the **WEB-INF/lib** directory of `genesys-rules-engine.war` or `genesys-rules-engine-authoring.war` and store them in:

   ```
   ${WAS_INSTALL_ROOT}\optionalLibraries
   ```

   > **Important**
   > The `jackson-core-asl-[VERSION].jar` and `jackson-mapper-asl-[VERSION].jar` libraries are required for the 8.5.2 auto-synchronization feature.

2. Configure these four 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-[VERSION].jar
      ```

      and

      ```
      ${WAS_INSTALL_ROOT}/optionalLibraries/httpcore-[VERSION].jar
      ```

      and

      ```
      ${WAS_INSTALL_ROOT}/optionalLibraries/jackson-core-asl-[VERSION].jar
      ```

      and

      ```
      ${WAS_INSTALL_ROOT}/optionalLibraries/jackson-mapper-asl-[VERSION].jar
      ```

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

5. Navigate to **Enterprise Applications->genesys-rules-engine->Shared library references** and add the sharedStuff shared library reference to the Module.
6. Navigate to **Enterprise Applications->genesys-rules-authoring>Shared library references** and add the sharedStuff shared library reference to the Application and Module.

Configuring Clusters in Websphere in GRS 8.5.2

Background

Before release 8.5.2 of GRS, it was not possible to configure multiple cluster nodes running on the same machine and controlled by the same cluster manager because separate entries for the same host could not be created in **bootstrapconfig.xml** to represent different GRE nodes. The pre-8.5.2 format of the **bootstrapconfig.xml** allowed for a single node to be defined per host. The xml format was as follows:

```
<xs:complexType name="node">
      <xs:sequence>
       <xs:element name="cfgserver" type="cfgserver" minOccurs="1" maxOccurs="1"/>
       <xs:element name="lcaserver" type="lcaserver" minOccurs="0" maxOccurs="1"/>
       <xs:element name="application" type="application" minOccurs="1" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="host" type="xs:string"/>
      <xs:attribute name="ipaddress" type="xs:string"/>
</xs:complexType>
```

What's New?

In GRS 8.5.2, an additional attribute called **servername** has been added to the node definition. This makes it possible to define multiple nodes for the same host. The server name is defined via the WebSphere Application Server (WAS) Deployment Manager when the cluster node is created.

For example, you can replicate the “node” definition for each GRE that is running on the same host. Then, by adding **servername=**, you can make the entry unique. Each entry then points to the corresponding Configuration Server application for that GRE instance. In this way, a single bootstrapconfig.xml file can be used to define all nodes in the Websphere cluster, whether or not there are multiple GRE nodes defined on a given host.

To ensure backward compatibility, if no node is found within the **bootstrapconfig.xml** that matches both the hostname and **serverName** then the node that contains the **hostname** with no server name defined serves as the default.

Editing the bootstrapconfig.xml file

To edit this file, manually extract the `bootstrapconfig.xml` file from the `.war` file, edit and save the `bootstrapconfig.xml` file, then repackage the `bootstrapconfig.xml` file back into the `.war` file.

Sample bootstrapconfig.xml files

> **Important**
>
> Terminology—In the `bootstrapconfig.xml` files, the <node> element corresponds to an individual member of a WebSphere cluster.

For a cluster with one host and two server instances on that host

Below is a sample **`bootsrapconfig.xml`** definition for a GRE cluster running on one host, `GenSrv1000`, with server instances `server01` and `server02` on that host:


```
<bootstrap name="BRS_applications" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="bootstrapconfig.xsd">
  <node host="GenSrv1000" servername="server01"
   <cfgserver host="192.xxx.x.x" port="2020">
       <reconnectperiod></reconnectperiod>
       <reconnectattempts></reconnectattempts>
       <!-- <protocoltimeout>30</protocoltimeout> -->
       <clienttransport ipaddress="" port="" />
       <backup host="" port="" />
       <!-- <addp clienttimeout="30" servertimeout="30" /> -->
    </cfgserver>
    <application id="brs.rules.engine"> <cfgappname>GRE_85200-S01</cfgappname> </application>
 </node>
<node host="GenSrv1000" servername="server02"
   <cfgserver host="192.xxx.x.x port="2020">
       <reconnectperiod></reconnectperiod>
       <reconnectattempts></reconnectattempts>
       <!-- <protocoltimeout>30</protocoltimeout> -->
       <clienttransport ipaddress="" port="" />
       <backup host="" port="" />
       <!-- <addp clienttimeout="30" servertimeout="30" /> -->
    </cfgserver>
    <application id="brs.rules.engine"> <cfgappname>GRE_85200-S02</cfgappname> </application>
 </node>
</bootstrap>
```

For a cluster with two hosts and two server instances on each host

Below is a sample **`bootsrapconfig.xml`** definition for a GRE cluster running on two hosts, `GenSrv1000` and `GenSrv2000`, with server instances `server01` and `server02` on each host:


first host name and first server instance
first host name and second server instance
second host name and first server instance
second host name and second server instance
unique configuration application names

Configuring WebSphere Liberty in GRS 8.5.3

Support for Java 8 in GRS release 8.5.3 means that previous WebSphere version supported in Java 7 are not available. The only WebSphere available for Java 8 is WebSphere Liberty. Verification has been performed using version 8.5.5.7.

Configuring WebSphere Liberty to Run GRAT and GRE

1. Set the **JAVA_HOME** and **PATH** environment variables. These are used by the **./server** command.
2. Create the Liberty server by navigating to **[WAS Liberty Home]/bin** and executing the command **./server create grs**.
3. Navigate to **[WAS Liberty Home]/usr/servers/grs** and create a directory called **ExternalLibs**. Copy the JDT core .jar file to **ExternalLibs** from **[WAS Liberty Home]/lib**. The .jar file should be similar to **com.ibm.ws.org.eclipse.jdt.core.[version].jar**.
4. If GRAT is configured to use a database via a configuration Data Access Point, add the appropriate drive .jar in the **ExternalLibs** directory created above. Otherwise skip this step. For example, if using the PostgreSQL database, add a driver file like **postgresql-[version].jdbc4.jar** to the **ExternalLibs** directory.
5. Navigate to the **[WAS Liberty Home]/usr/servers/grs** directory and edit the **server.env** file to add **JAVA_HOME**. For example, if JAVA_HOME is `/usr/java8_64`, then add this line at the end of the file:

   ```
   JAVA_HOME=/usr/java8_64
   ```

6. Create file **jvm.options** in directory **[WAS Liberty Home]/usr/servers/grs** to set the JVM memory parameters—Genesys recommends using **the information here** to tune performance for GRE and GRAT.
7. Update the **server.xml** file (the server configuration file at **[WAS Liberty Home]/usr/servers/grs**) thus:
   a. Add **<webContainer deferServletLoad="false"/>** under the **<server>** element.
   b. Add **<applicationMonitor updateTrigger="disabled" />** under the **<server>** element.
   c. Add **host="*"** attribute to the **httpEndpoint** element if you want any hosts to be able to access this application.
   d. Add the following application elements under the root element (that is, the **<server>** element):

   ```
   <!-- GRAT Application: IMPORTANT! Remove classLoader element in below Application if
   not using an external database otherwise replace "[database driver file]" with the
   appropriate driver file name -->
   <application context-root="genesys-rules-authoring" type="war" id="genesys-rules-
   authoring"
          location="genesys-rules-authoring.war" name="genesys-rules-authoring">
          <classloader>
               <privateLibrary>
                         <file name="ExternalLibs/[database driver file].jar"
   ```

```
id="databasedriver"></file>
                </privateLibrary>
        </classloader>
</application>
<!-- GRE Application: IMPORTANT! Replace "[JDT core jar file name]" with the file
name -->
<application context-root="genesys-rules-engine" type="war" id="genesys-rules-engine"
       location="genesys-rules-engine.war" name="genesys-rules-engine">
       <classloader>
            <privateLibrary>
                        <file name="ExternalLibs/[JDT core jar file name].jar"
id="jdt"></file>
                </privateLibrary>
        </classloader>
</application>
```

> **Important**
>
> Make sure **<featureManager>** has only **<feature>jsp-2.2</feature>** and nothing else. Adding any other feature might interfere with the already existing libraries in the application.

5. For GRAT:
   a. Create directory **genesys-rules-authoring.war** under **[WAS Liberty Home]/usr/servers/grs/apps directory**.
   b. Extract the contents of the **genesys-rules-authoring.war** file into **genesys-rules-authoring.war** directory. Navigate to **genesys-rules-authoring.war** directory and execute command **jar -xvf [Path to GRAT .war]**.

3. For GRE:
   a. Create directory **genesys-rules-engine.war** under **[WAS Liberty Home]/usr/servers/grs/apps** directory.
   b. Extract the contents of the **genesys-rules-engine.war** file into **genesys-rules-engine.war** directory. Navigate to **genesys-rules-engine.war** directory and execute command **jar -xvf [Path to GRE .war]**.

3. Start the server by navigating to **[WAS Liberty Home]/bin** and executing EITHER command **./server run grs** OR command **./server start grs**.

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. 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**.
8. Click **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 GRDT 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 GRDT, 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** (or **http://<host>:<port>/genesys-rules-authoring/singlesignon.jsp** if this is configured) 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 8.5 Installation Script
-------------------------------------------------

Installing Genesys Rules Engine, version 8.5.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 : GRE85xxxxx_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/85xxxxx/linux

The target install directory /home/GRS/GRE/8.5.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/8.5.xxx.xx/linux
….

Installation of Genesys Rules Engine, version 8.5.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 8.5 Installation Script
------------------------------------------------

Installing Genesys Rules Authoring Tool, version 8.5.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 : GRAT8500010_rh5x64-vm1
2 : GRE8500004_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/8.5.xxx.xx/linux

The target install directory /home/GRS/GRAT/8.5.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/8.5.xxx.xx/linux
```

….

```
Installation of Genesys Rules Authoring Tool, version 8.5.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. The load balancer should only route rule evaluation requests to a node that returns an `HTTP 200/ SYSTEM_STATUS_OK`, as described in GRE Status below.

GRE Status

GRE has a `status.jsp` URL that can be used for a health check. The following statuses are available via `/genesys-rules-engine/status.jsp`.

Status	Response Text/Meaning
HTTP 503	• `SYSTEM_STATUS_CONFIG_SERVER_NOT_CONNECTED`—Configuration Server is not connected (same as pre-8.5.2 response) • `SYSTEM_STATUS_ENGINE_NOT_INITIALIZED`—Engine is not initialized • `SYSTEM_STATUS_CLUSTER_SYNCHING`—Engine synching with Cluster
HTTP 200	• `SYSTEM_STATUS_OK`—Ready to take rule execution requests (same as pre-8.5.2 response)

GRAT

Pre-8.5.301

Before release 8.5.301, 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 could 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.

GRAT Status

GRAT has a `status.jsp` URL that can be used for a health check. GRAT's `status.jsp` always returns `HTTP 200`. The response text must be queried to determine if the GRAT server is up or down.

Status	Response Text/Meaning
HTTP 200	• `SYSTEM_STATUS_OK`—GRAT server is up and running • `SYSTEM_STATUS_CONFIG_SERVER_NOT_CONNECTED`—GRAT server is not connected to Configuration Server • `SYSTEM_STATUS_DB_INITIALIZING`—GRAT server is currently initializing local cache from repository database. This can take several minutes for a large repository. • `SYSTEM_STATUS_DB_NOT_CONNECTED`—GRAT Server cannot connect to the repository database. Check the database status and/or check the database credentials that are specified in the DAP on the GRAT application object. • `SYSTEM_STATUS_UNKNOWN`—GRAT server is down. Check logs for more details.

8.5.301+

In release 8.5.301, you can now configure clusters of GRAT servers which deliver much greater resilience and availability, enabling instant switchovers between GRAT nodes that are members of the cluster. All cluster members connect to the same database repository. No single GRAT node is considered primary—they are all equal partners in the n-node cluster.

Each node maintains a journal of changes, which resides on a separate repository database table. Nodes can periodically poll the database to ensure that they mirror updates made on the other nodes. This design also allows new nodes to be added to the cluster and synchronized with their peers.

In addition, when a GRAT is part of a cluster, you should in general set the value of its `clear-repository-cache` option to `true`. For GRATs in a cluster, setting this value to `true` can help avoid the repository corruption errors that can occur if you forget to clear the cache when a node is re-added to the cluster or moved from a different cluster. However if there is a specific reason to avoid the possible slower startup of GRAT server that might ensue, then set the option to `false`.

If the GRAT startup delay is irrelevant or if GRAT startup is quick enough with the `clear-repository-cache` option set to `true`, then it should be set to `true`.

ADDITIONAL DETAILS.

Configuring GRS for Secure Sockets Layer (SSL)

Introduction

Purpose

This topic describes how set to up GRAT and/or GRE in SSL (Secure Sockets Layer) mode using the Java *Keytool* utility which should be available if Java is installed. In these steps, Keytool is used to generate the self-signed certificate and update Java's keystore to add the certificate. The browser is used to download the public certificate.

Supported Configurations

Supported configurations are:

- GRAT in SSL mode and GRE in non-SSL mode
- GRAT in non-SSL mode and GRE in SSL mode
- Both GRAT and GRE in SSL mode

Changes in GRE's Communication Port in Configuration

- Make sure the correct SSL communication port is provided. In Tomcat, by default, 8080 is a non-SSL port and 8443 is for SSL.
- The value for the Connection Protocol in the GRE Port must remain as `http` (**NOT** to be changed to `https`).
- The listening mode must be set to `Secured`.

Setting up GRAT in SSL Mode

> **Important**
>
> The steps here demonstrate how to create a self-signed certificate and use it for SSL. For security reasons, it is very important that you consult your organization's IT Security Administrator to check whether you must use a certificate signed by an external CA (Certificate Authority) like VeriSign or Thawte.

1. **[+] Create the Certificate if it is not already available.**

   On GRAT, use Keytool utility to create a self-signed certificate to be used for SSL.

   ```
   keytool -genkey -alias <alias name> -keyalg <security key algorithm> -validity 360
   ```

   > **Important**
   >
   > When prompted for input `What is your first and last name?`, enter the name of GRAT's Host object in Configuration Server. It must be either GRAT's hostname or the IP address. The value entered here is used in the commonName (CN) property of the certificate.

   For example:

   ```
   /usr/local/java/jdk1.7.0_79/jre/bin/keytool -genkey -alias linux-grat -keyalg RSA
   ```

   A self-signed certificate will be created by file name `.keystore` in the user's home directory. If the certificate must be signed by an external CA (Certificate Authority), a CSR needs to be created and submitted to the CA. You can use Keytool to create a CSR. Please see Java documentation for a complete list of Keytool options.

2. **[+] Enable SSL in the server configuration by using the Certificate and disable non-SSL mode.**

   For example, to enable SSL in the case of Tomcat, the SSL configuration in `.../[TOMCAT_HOME]/conf/server.xml` looks like this:

   ```
   <Connector port="8443" protocol="org.apache.coyote.http11.Http11Protocol"
   ```

```
maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS"
keystoreFile="/home/certificates/.keystore" keystorePass="changeit"/>
```

Where:

- `keystoreFile` is the path to the certificate file generated in step 1.
- `keystorePass` is the password created for the certificate in step 1.

3. **[+] On the GRE machine, get the public certificate of GRAT.**
   a. Open GRAT's link `https://[ GRAT IP address]:[SSL port number]/genesys-rules-authoring/index.jsp` in your browser.
   b. When it shows the warning about certificates, accept the certificate to be added in the browser's Trusted Certificates list.
   c. Once the certificate has been downloaded by the browser, export it using the browser's export certificate feature.

<toggledisplay linkstyle font-size:larger showtext="[+] BROWSER DETAILS" hidetext="[-] BROWSER DETAILS">

Browser	Procedure
IE 11	1. When the GRAT application is open in Internet Explorer using HTTPS, click the **Lock** icon in the address bar. (It is located beside the refresh icon on the right side in the address bar.) 2. Navigate to the **View Certificates link > Details** tab. 3. Click the **Copy to File...** button. 4. Select the format `DER encoded binary X.509 (.CER)` and click **Next**. 5. Enter the file name into which you want to save the certificate. Click **Next**, then **Finish**. OR 1. Navigate to **Internet Options > Content > Certificates**. 2. Locate the certificate. 3. Select the certificate and export it selecting `DER encoded binary X.509 (.CER)` format from the three format choices.

Browser	Procedure
Firefox 40.02	1. When the GRAT application is open in Firefox using HTTPS, click the **Lock** icon in the address bar. (It is located just before "https") 2. Click **More information...**. 3. Navigate to the **Security Tab > View Certificate button > Details Tab**. 4. Click the **Export...** button. OR 1. Navigate to **Options > Advanced > Certificates**. 2. Locate the certificate. 3. Select the certificate and export it.
Chrome 44.0	1. When the GRAT application is open in Chrome using HTTPS, click the **Lock** icon in the address bar. (It is located just before "https"). 2. In the popup that opens, navigate to **Connection tab > Certificate information link > Details** tab. 3. Click **Copy to File...**. 4. In the Certificate Export Wizard, enter the file name to which you want to save the certificate. Click **Next**, then **Finish**. OR 1. Navigate to **Customize > Settings >**. 2. Enter `certificate` in **Search Settings** and press the enter key. 3. Click **Manage Certificates...**. 4. Locate the certificate.

Browser	Procedure
	5. Select the certificate and export it.

4. **[+] On the GRE machine, add the public certificate to Java Keystore using the Java Keytool.**

   ```
   keytool -import -alias <alias> -keystore <cacerts_file> -trustcacerts -file <certificate_filename>
   ```

   > **Important**
   > This will prompt for the Java Keystore password. The default password for Java Keystore is changeit

   For example:

   ```
   /usr/local/java/jdk1.7.0_79/jre/bin/keytool -import -alias linux-grat -keystore /usr/local/java/jdk1.7.0_79/jre/lib/security/cacerts
    -trustcacerts -file /home/certificates/linux-grat
   ```

   Where:

   - `alias` is the alias to be used for this certificate.
   - `keystore` is the path to Java's Keystore in which we want to add the certificate. Make sure to update the Keystore of Java that is used by the Server.
   - `file` is the path to the certificate file (exported in step 3) that we can to add into Java Keystore.

5. **If you are using GRDT**, repeat steps 3 and 4 on the GRDT machine. Make sure to update the Host Configuration under **Preferences > Genesys Rules System > Repository Server** to use the `https` port and ensure that the **HTTPS** checkbox is selected.
6. **As for GRE and GRDT**, repeat step 3 and 4 for any other Java clients of GRAT which would need to use HTTPS to send requests to GRAT.

Setting up GRE in SSL mode

The procedure to set up GRE in SSL mode is similar to the procedure for GRAT. In step 3, use:

```
https://[ GRE IP address]:[SSL port number]/genesys-rules-engine/status.jsp
```

to get GRE's public certificate on the GRAT machine.

Similar to the steps above, where you added GRAT's public certificate to GRE's Java keystore, for GRE you need to add GRE's public certificate (exported from the browser) to GRAT's Java Keystore.

Performance Tuning with Java Virtual Machine

Use the following parameter settings in Java Virtual Machine (JVM) Garbage Collection (GC) to maximize the performance of GRE and GRAT.

> **Important**
> For performance reasons, GRAT and GRE should run under separate JVMs.

Required

```
-server -Xms2G -Xmx2G -XX:+UseG1GC -XX:MaxGCPauseMillis=200 -XX:ParallelGCThreads=5
-XX:ConcGCThreads=5 -XX:InitiatingHeapOccupancyPercent=10
```

Highly Recommended For GC log

These settings help in debugging JVM performance problems.

> **Important**
> Adjust the value for parameter **-Xlogg** parameter as needed—this is the log file path.

```
-XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=5
-XX:GCLogFileSize=20M -Xloggc:[path to gc log file]
```

Highly Recommended for Out Of Memory Heap Dump

> **Important**
> Adjust the value for parameter **-XX:HeapDumpPath** parameter as needed.

```
-XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=[path to heap dump file]
```

Notes

- For most cases 2G for -Xms (heap space min) and -Xmx (heap space max), as in the above required settings, will suffice. In some marginal cases, you may need to increase or decrease the heap space.
- If GRE has a lot of rules deployed, then the Xms/Xmx values must be increased to leave enough heap space to execute the requests at the required rate.

Examples

- If rule packages alone consume about 3G of heap when loaded in memory, you should use approximately 6G as the value for Xms and Xmx.
- If there are only few rule packages (each with only few rules) then 1GB of heap space should be enough.

Important

GRE's Memory Monitor statistics log output can be used to determine the amount of heap space needed by GRE. Before enabling the Memory Monitor in production environment, please make sure to read about its purpose and adjust the settings according to the heap space.

Troubleshooting

This section contains the following topics:

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 the iWD GAX Plug-in 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).

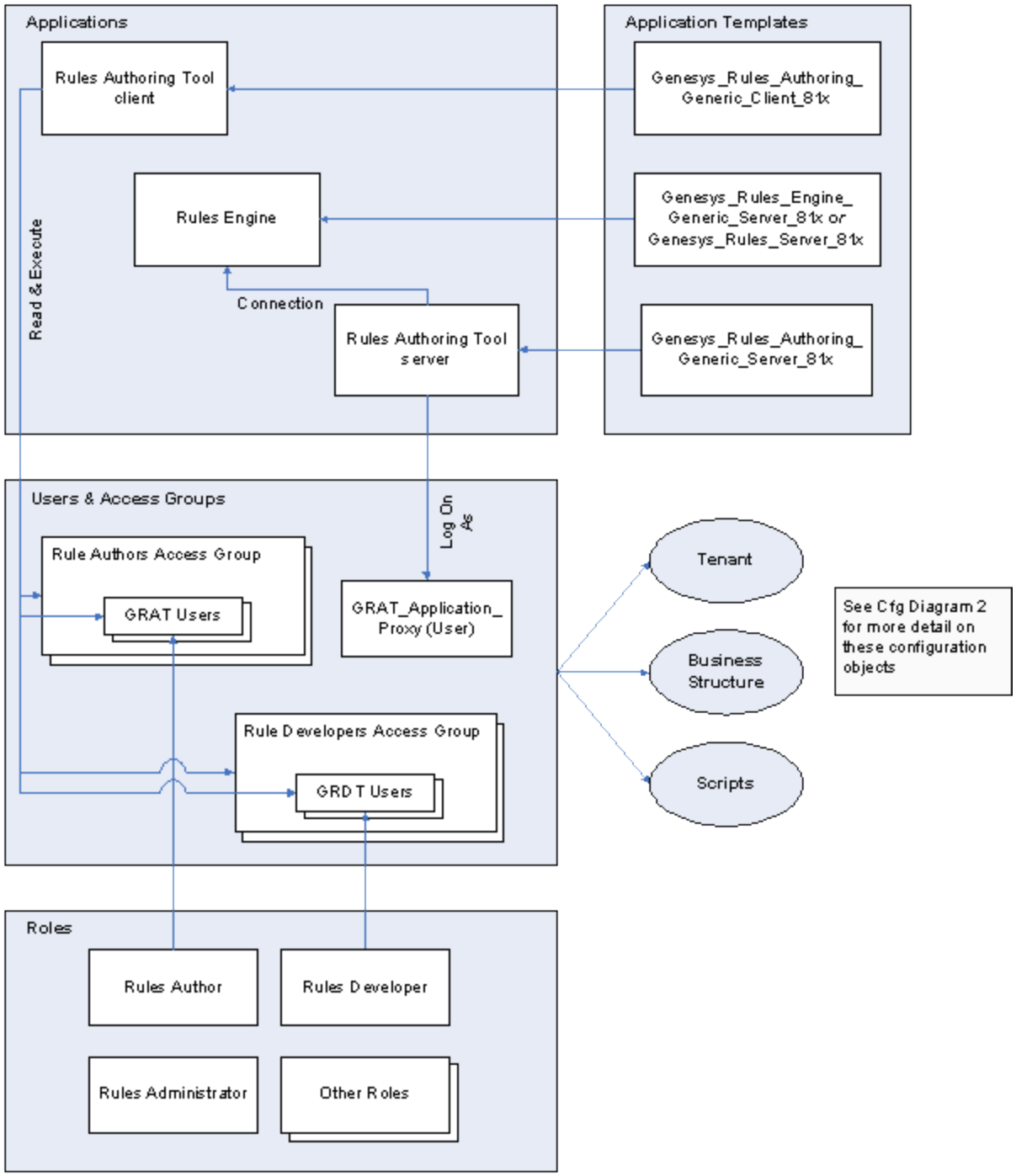
These first level nodes under Business Structure represent the Solution(s). Within each solution, additional levels of hierarchy may be created, as needed, using either Business Units or Sites. Those levels of hierarchy beneath the Solution level will represent the business context.

- 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.

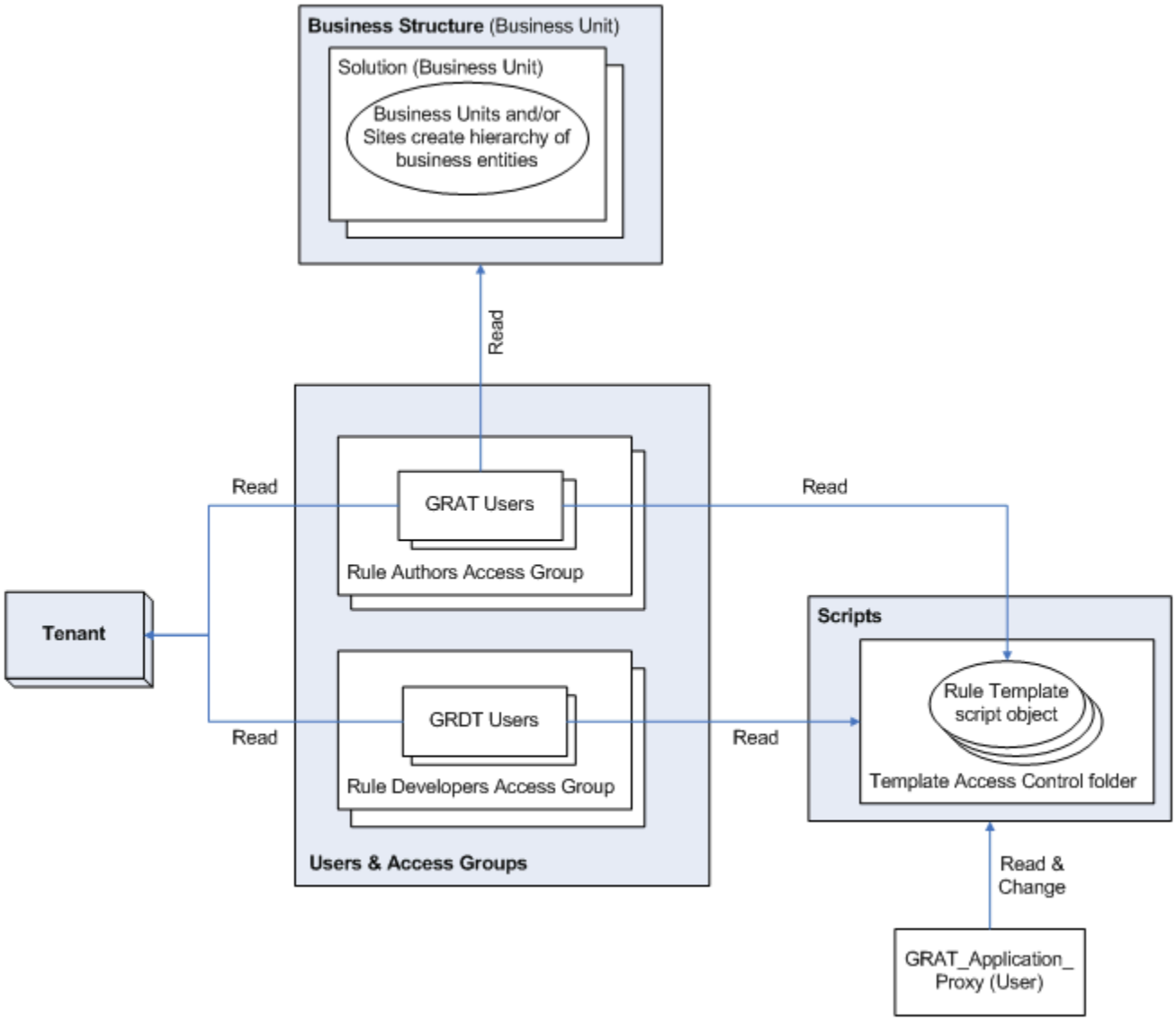
Scripts

- 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

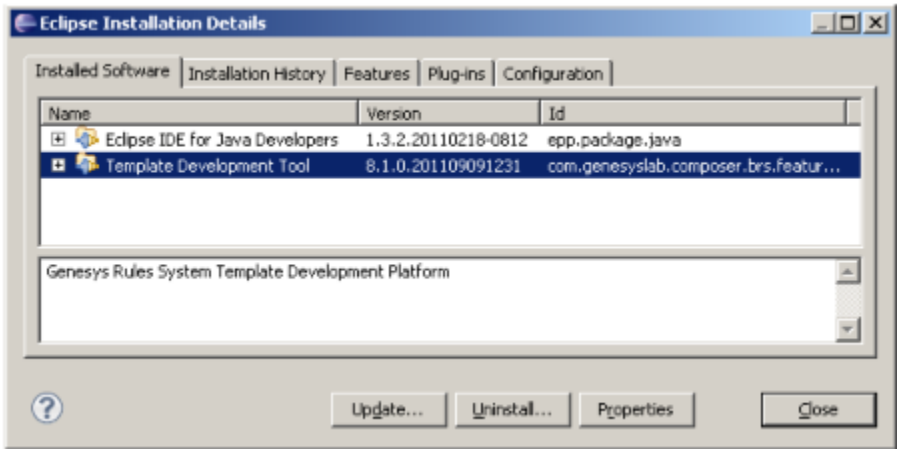
Locating the GRDT Version Number

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 `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.xxxxxxxxxxx`, 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.

Locating Log Files

Before connecting to Configuration Server, GRAT and GRE will log by default to the relative path:

- `"logs/GRATInit"`
- `"logs/GREInit"`

These logs are useful in debugging start-up issues; for example, if GRAT or GRE are unable to connect to Configuration Server.

- **In UNIX based systems,** these initial logs can be found under your application servers "logs" directory.
- **In Windows based servers,** these initial logs can be found in the `\Windows\SysWOW64\logs` directory.

You can change the initial location of these logs by editing the log4j.properties file as described in this section.

After GRAT or GRE connect to Config Server, the log location is determined by the application's configuration options (`log` section).

GRE Server Status Check

> **Important**
>
> Load balancers should not use this method, but rather check the GRE's status.jsp. See High Availability

.

To make a manual intervention for a GRE server status "heartbeat," use the base GRE URL:

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

This returns the following:

Genesys Rules Engine (Version 8.5.200.10) is running

This server allows you to execute rules/knowledge bases remotely using a RESTful interface.

Stateless services

URL:

http://{server address, port etc}/genesys-rules-engine/knowledgebase/{packageName}
a HTTP POST to this URL will perform a stateless execution of the knowledgebase/rules.
The {packageName} is the name of a rules package previously deployed to this server.

General instructions

A request may contain globals, inOutFacts, and inFacts. Globals and inOutFacts may be modified by the rules and returned in the response.

By default XML will be used. If you pass in a Content-Type header of application/json, then JSON will be used instead. JSON is both a more compact and more performant format. HTTP POST is used to access this service. Any libraries the rules use (such as fact pojos) will need to be in the WEB-INF/lib directory.

Sample XML request content:

```
<knowledgebase-request>
  <globals>
    <named-fact>
      <fact class="string">Some data</fact>
      <id>myGlobal</id>
    </named-fact>
  </globals>
  <inOutFacts class="named-fact-array">
    <named-fact>
      <fact class="com.genesyslab.brs.engine.Person">
        <name>Sam</name>
        <likes>Swiss</likes>
        <age>28</age>
      </fact>
      <id>customer</id>
    </named-fact>
  </inOutFacts>
  <inFacts>
    <anon-fact>
```

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.

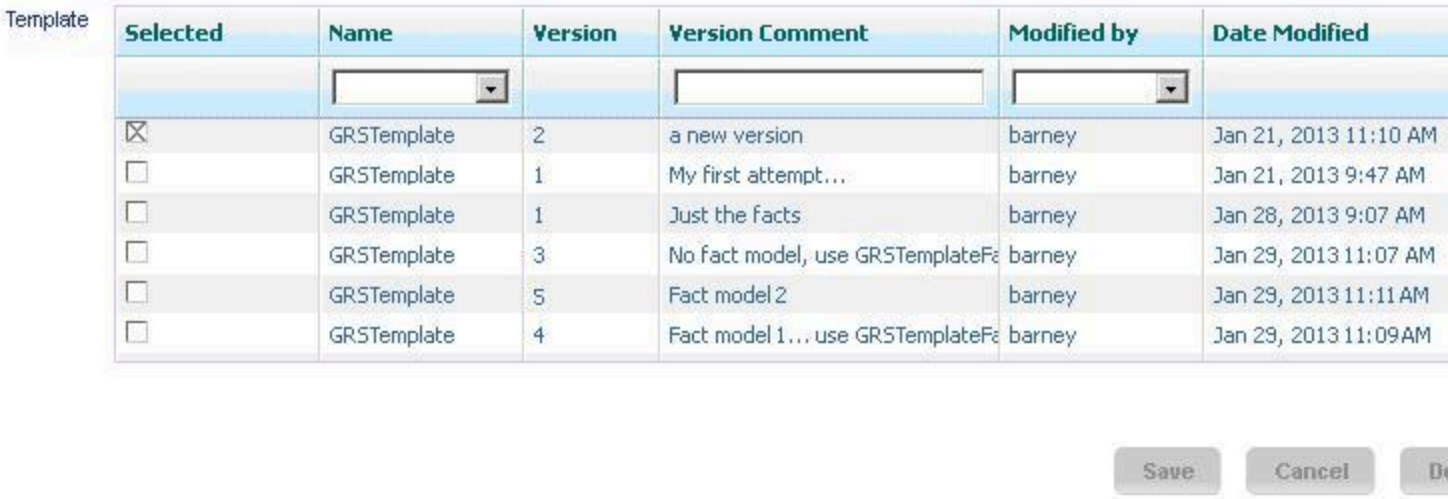
The rule developer should communicate to the rule author if a new version of the Rule Template is available and if they are advised to upgrade.

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

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.

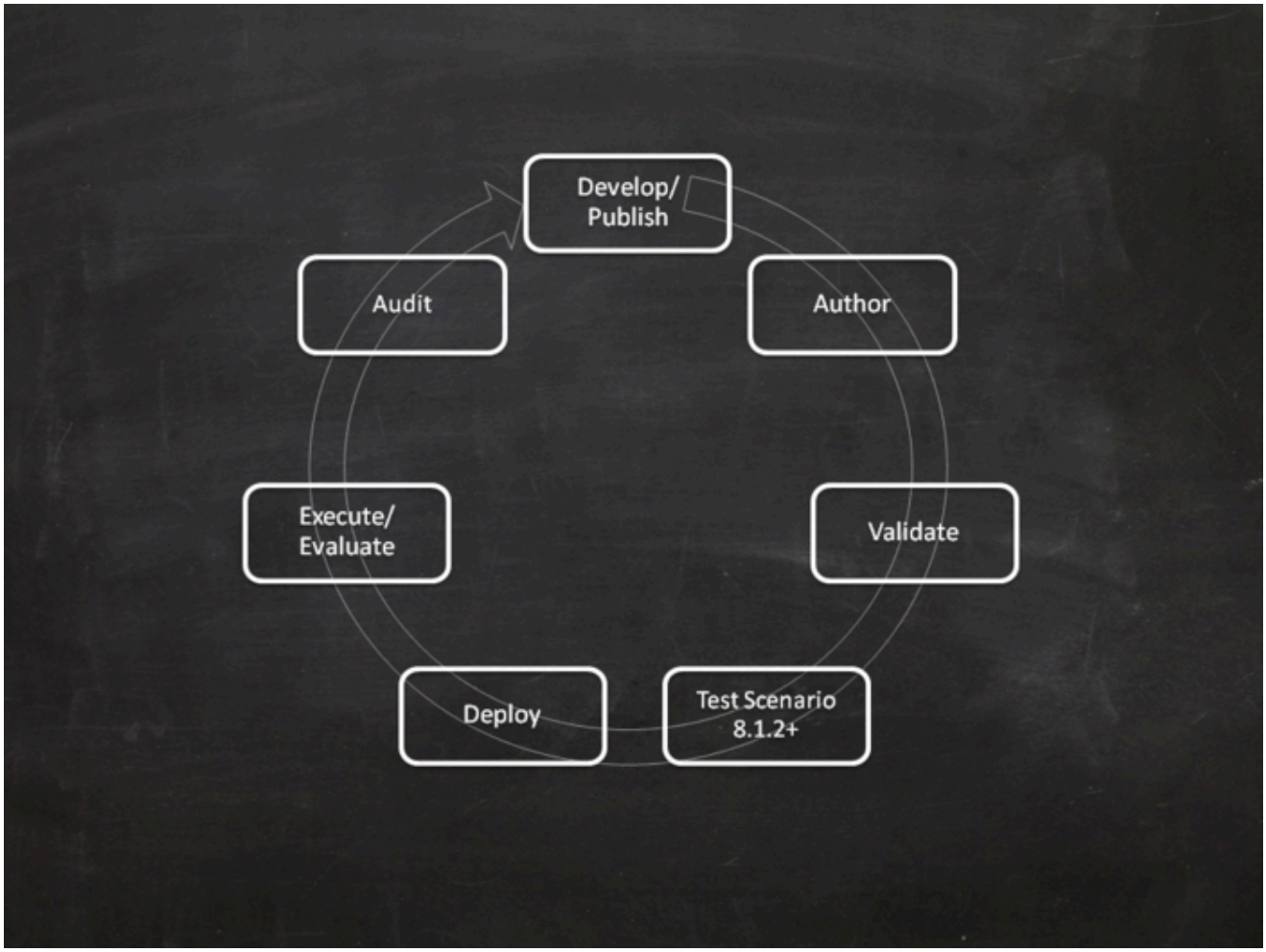
Refer to the Genesys Rules Development Tool Help for more information about rule templates and how to create them.

Rules

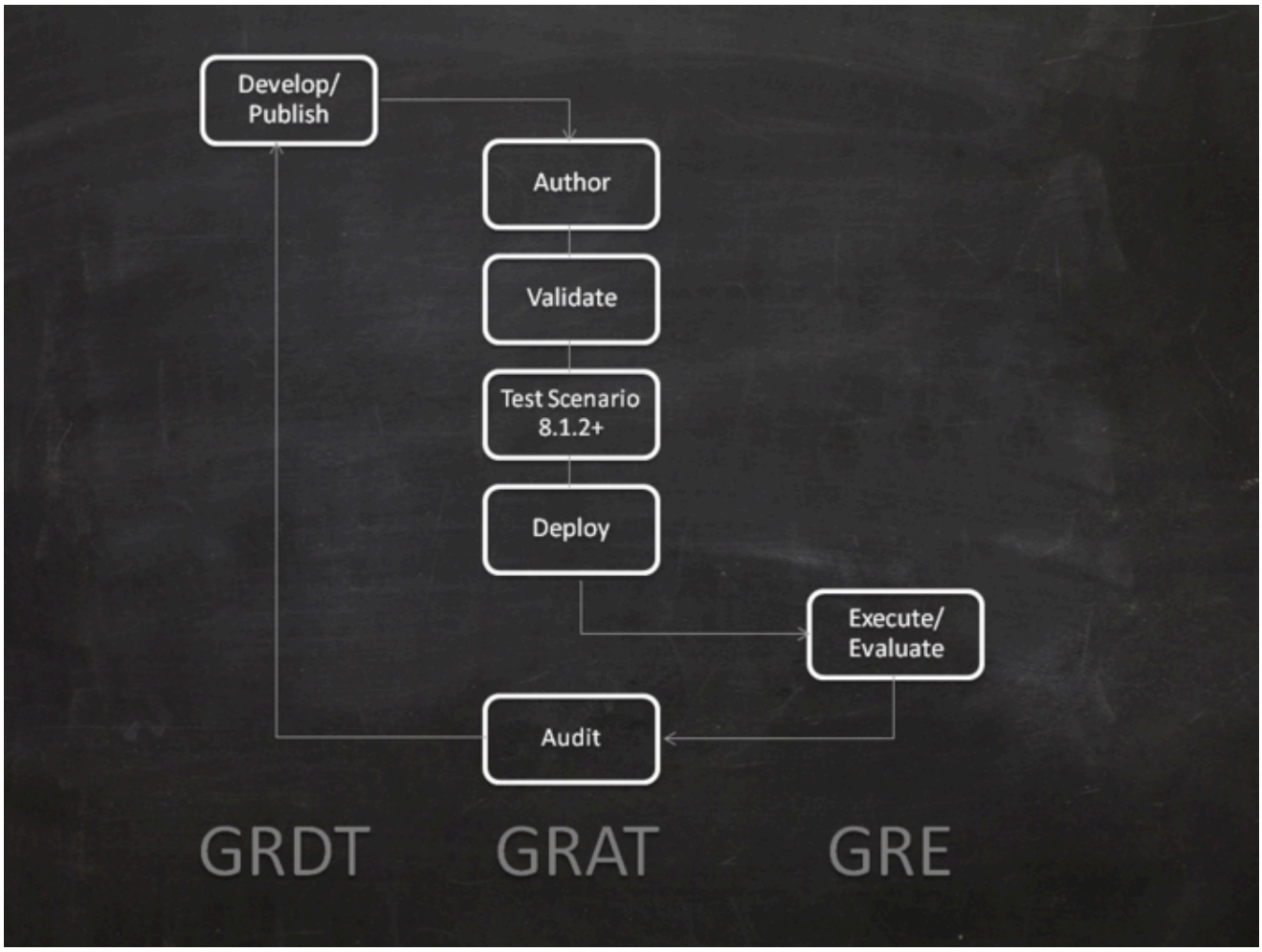
Business rules are created in the GRAT by business analysts. Rules are created within a rule package. When a rule package is created, one or more rule templates can be selected for inclusion. The templates determine the conditions, actions, and so on that are available to use during creation of business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

For specific information about how business rules are used with the Genesys intelligent Workload Distribution (iWD) solution, go **here**.

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.

|Actions Editor

|Conditions Editor

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.

|Enumerations Editor

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.

|Fact Model Editor

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 was enhanced 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.

|Fact Model Editor

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.

|Functions Editor

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.

|Parameters Editor

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`.
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
```

|DETAIL Parameters Editor

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. |Parameters Editor

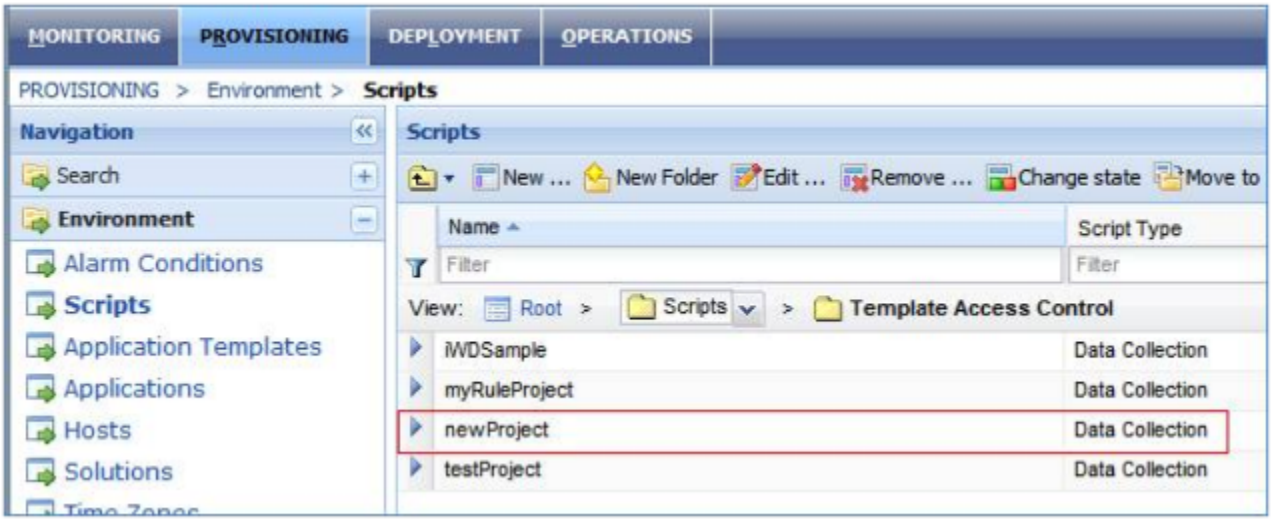
Deleting Rule Templates

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.

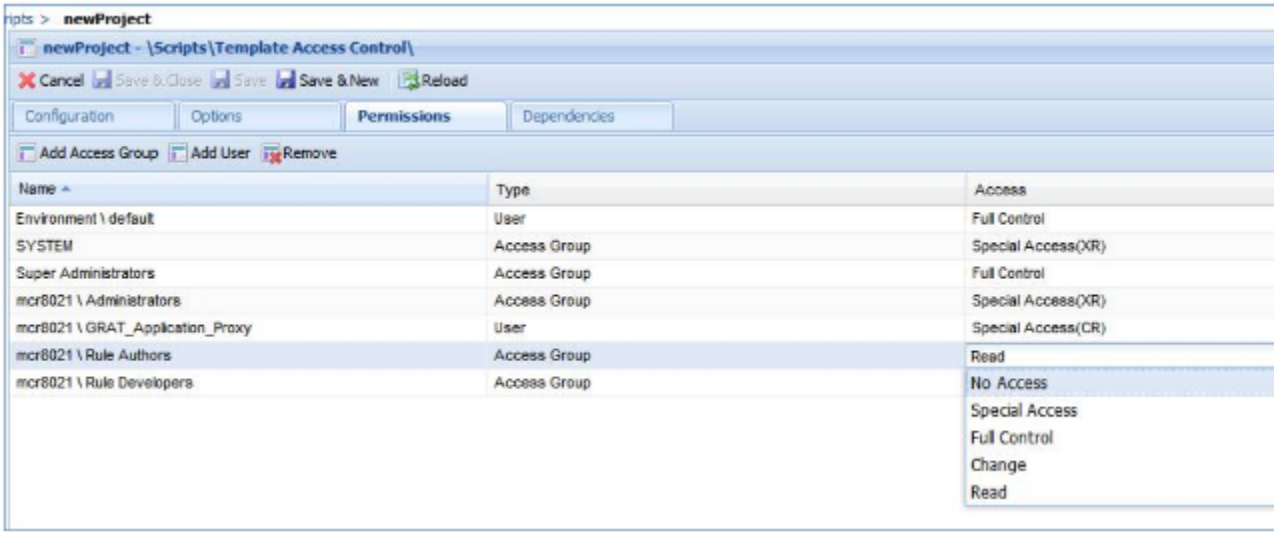
Script Objects

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 template 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 **iWD and Genesys Rules System**.

Example 1: Condition and Action

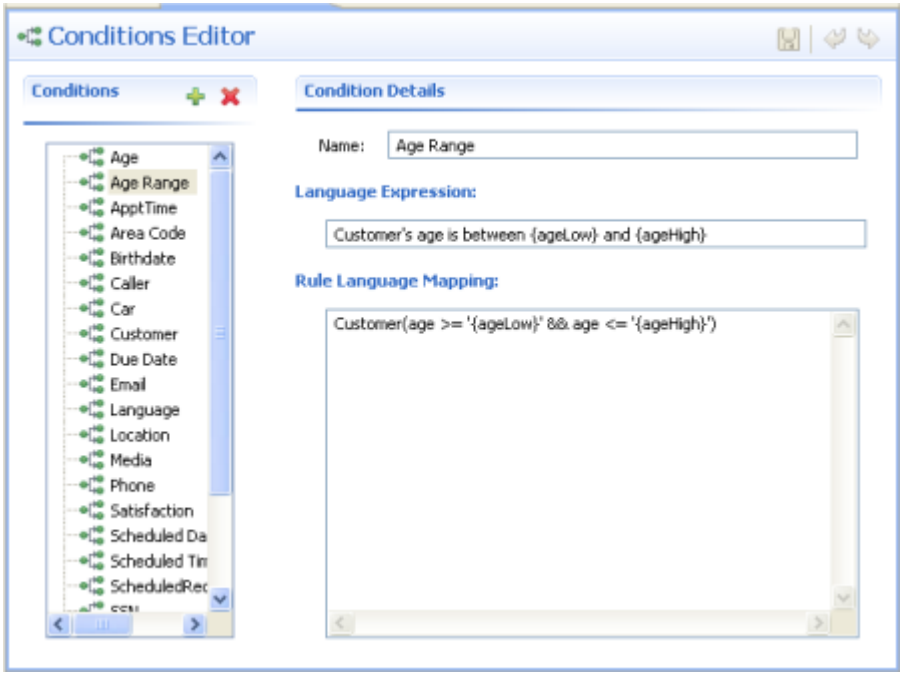
Age Range Condition

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

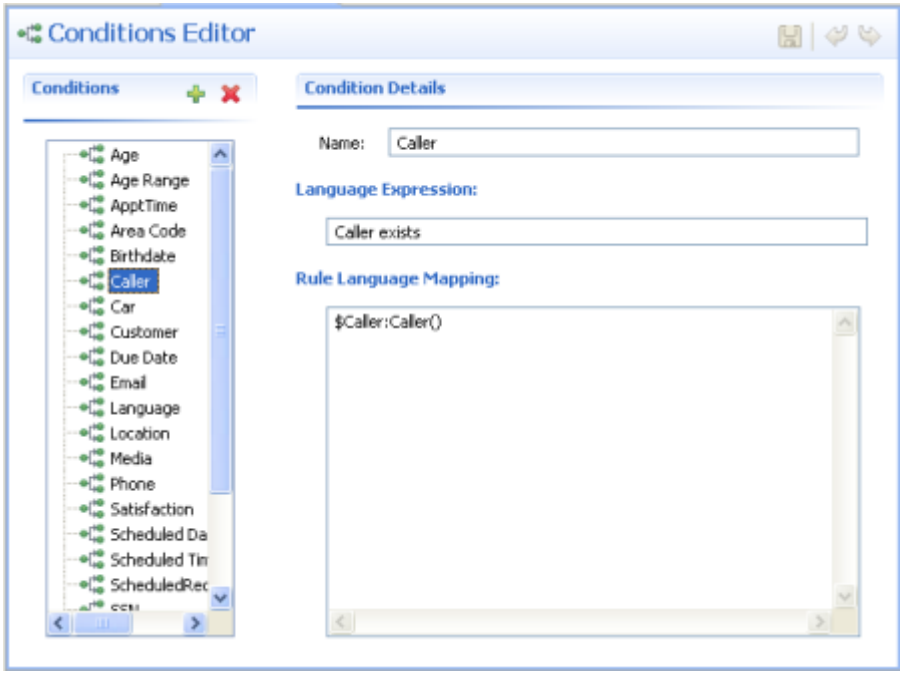
In addition to testing that the Caller exists, the next condition also creates the `$Caller` variable which is used by actions to modify the Caller fact. The modified Caller will be returned in the results of the evaluation request.

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

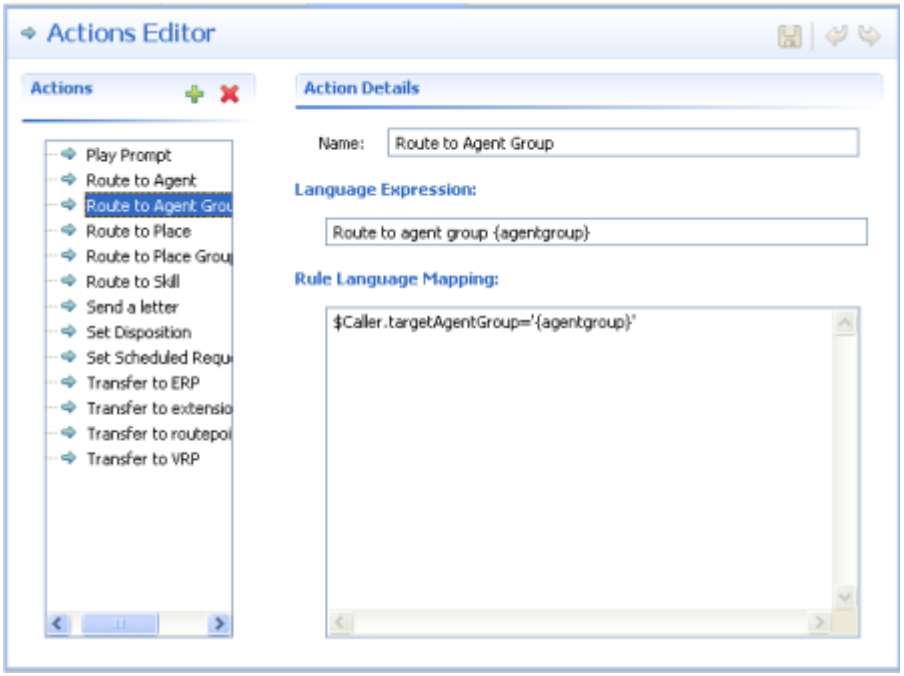
Target Agent Group Action

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.

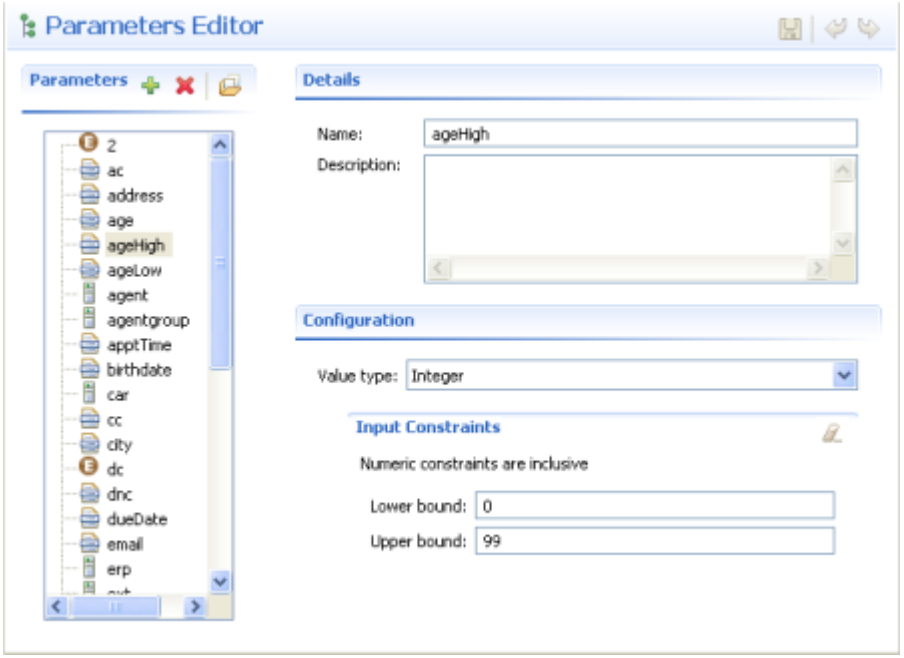


Target Agent Group

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 Since 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 Since 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.

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:
```

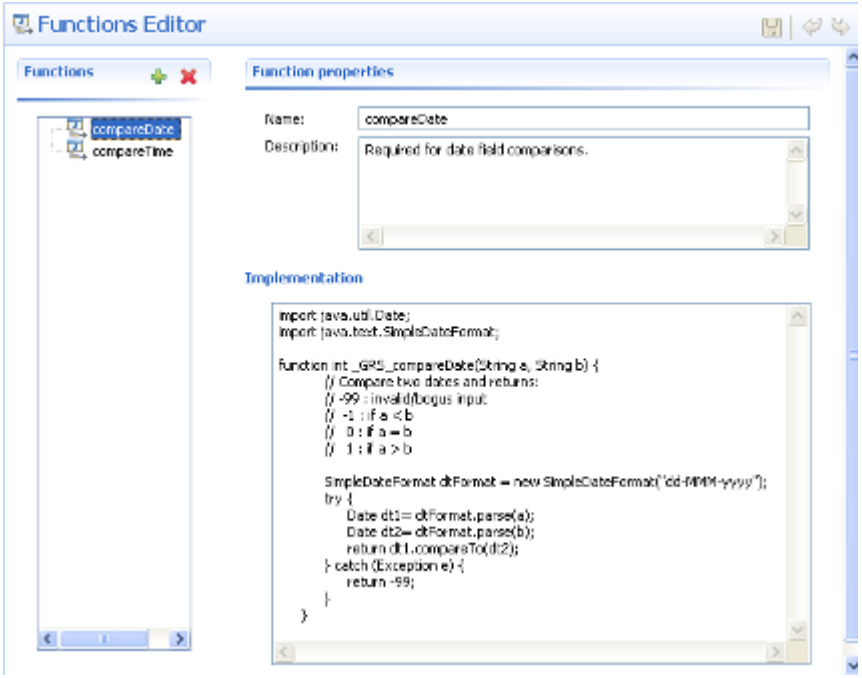
```
import java.util.Date;
import java.text.SimpleDateFormat;

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 3: Using a JSON Object

Since release 8.1.3, template developers can 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

Example 4: Developing Templates to Enable Test Scenarios

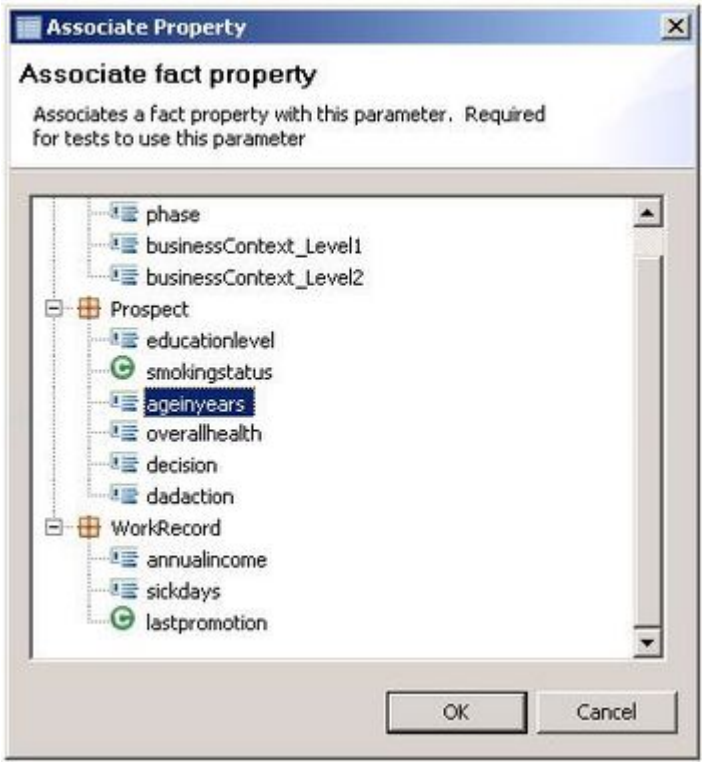
For Test Scenarios for Conversation Manager templates, see the **Best Practice/User Guide**.

Mapping Parameters to a Fact Model

Rules authors build rule-test scenarios using parameters, in the same way as they define rules. However, when the tests execute, GRAT maps the parameter values to the underlying fact model developed by the template developer, who understands the relationship between the parameters and the fact model. So, for rule testing to operate correctly, template developers must map parameters back to the underlying fact model.

For example, the {age} parameter may be related to the `ageinyears` field of the `Prospect` fact. So, if age is set to 25, then, when executing the test, GRAT needs to allocate a `Prospect` fact and set the `ageinyears` field to 25. The same is true for the expected results.

The **Associate Property** dialog enables this mapping.



In general, there should be a one-to-one mapping between a parameter and a fact. However, this may be too restrictive for all implementations. GRDT lets you map a single parameter to multiple fact values. For example, the {age} parameter could be defined once, but reused to represent both a customer's age and the age of an order.

So, {age} could map to both:

- `Customer.ageinyears`
- `Order.ageoforder`

Where this occurs, GRAT displays the parameter in the **Add Given** or **Add Expect** drop-down list in parentheses, so that the GRAT user can select the correct mapping

Example

In the following example, only {age} has this special designation because of the ambiguity in the definition.

`Add Given…`

- {age} (Customer.ageinyears)
- {age} (Order.ageoforder)
- {gender}
- {education}

To hide this ambiguity from the rule author, you should declare a different parameter for each usage: for example, `{customerAge}` and `{orderAge}`.

Using ESP-type Templates

There are some special considerations in developing templates for ESP-type templates, for products such as iWD.

With ESP templates, instead of building a set of facts and passing them to be executed, you create a `KeyValueCollection` (KVC) and populate it with key/value pairs of test data. In order to enable this mapping between a parameters and the correct key to use in the KVC collection, you need to create a dummy fact model in GRDT to represent the keys that will be inserted into the KVC.

For example, with iWD, this means modifying the iWD Standard Rules Table to insert a fact and a field for each key that is used in the template.

Fact Name

You will need to define a reserved fact name (for example, `_GRS_ESP_Fact`) that is processed differently. For this fact, the fields are mapped to a KVC instead of the traditional Fact model. The fact name must be used because there is no type associated with a fact. Types are only associated with individual fields.

Field Name

You must develop a convention (such as prefixing all field names with `grs_`) . This is because fact fields must start with a lower-case letter (a DROOLS restriction) and GRDT enforces this convention, but iWD key names all begin with IWD. Since the existing iWD key names are like "IWD_businessValue", you will need to adopt some naming convention. If the `grs_` prefix is present, GRAT will remove it and insert the remaining value into the KVC as the key (for example, `grs_IWD_channel` is inserted into KVC as `IWD_channel`)

The rule template developer must then map each parameter name (used in conditions/actions) to the appropriate field within `_GRS_ESP_Fact`.

The rule author can then use GRAT to create a rule and a test scenario for that rule.

Differences Since 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. This feature is most typically used for defining ranges more efficiently.

Fact/Condition

Since 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.

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 and higher, 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.

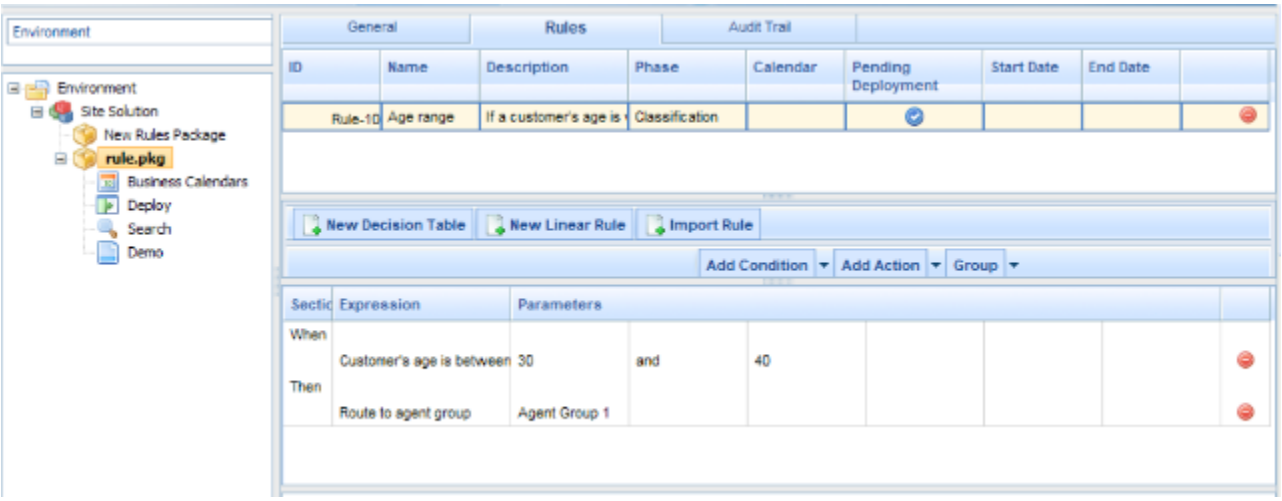
There is an enumeration called Phases within the `_GRS_Environment` fact, that will be created whenever a new rules template project is created in the Genesys Rules Development Tool. If the Phases enumeration is not present, the rules author will simply see * in the **Phase** dropdown. In this case, Phase will not be considered when evaluating the rule package.

> **Important**
>
> From GRS release 8.5.300.06 onwards, the `_GRS_Environment` Fact must be provided for all rule evaluations. In earlier releases, this Fact could be omitted for rules at the package level that did not use a Phase (that is, the Phase was defined to be *). From 8.5.300.06, an empty `_GRS_Environment` fact must be provided in this case.

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.

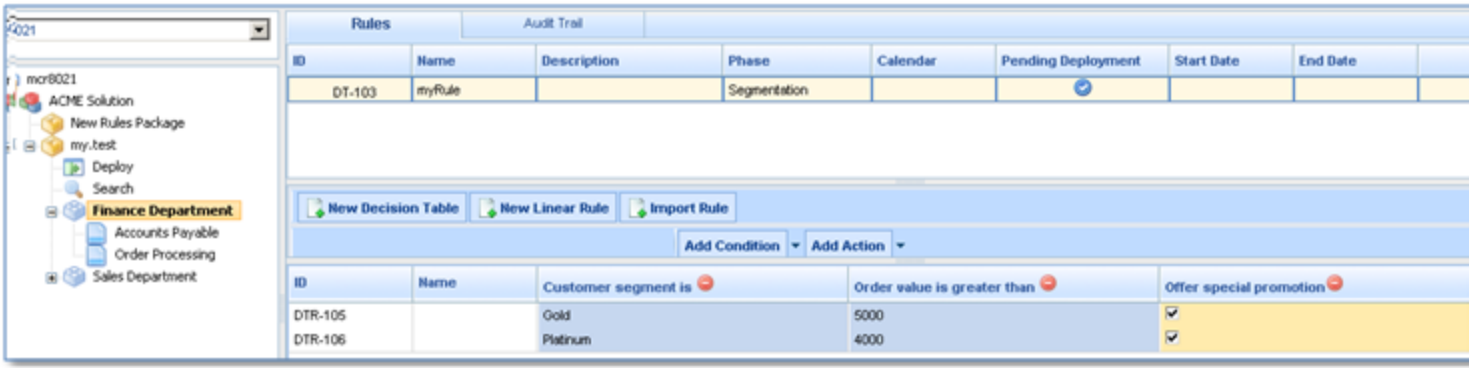


Sample Linear Rule

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 (iWD pre-8.5.0) or iWD GAX Plug-in (IWD post-8.5.0) 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.

Procedure

1. Navigate to the `Resources` folder for a single-tenant Configuration Server, or to the specific Tenant for a multi-tenant Configuration Server.
2. Open the `Business Units/Sites` folder (in Genesys Administrator) or `Configuration Units` (in Configuration Manager) folder.
3. Create a new top-level folder named `Business Structure`. This folder **must** be named `Business Structure`.
4. 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.
5. 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.

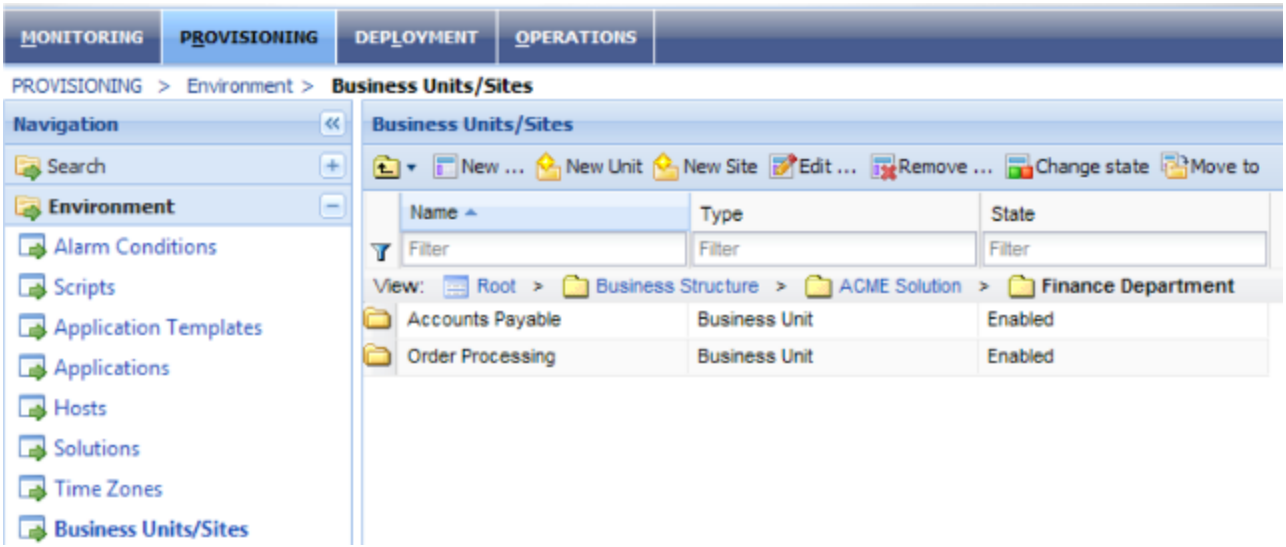
Warning

You cannot have the same node name across different departments in the hierarchy. So, you must either:

- Ensure that all node names within the business structure are unique, or;
- Add a condition to your template (for example, location) and have have it passed in as a new Fact field.

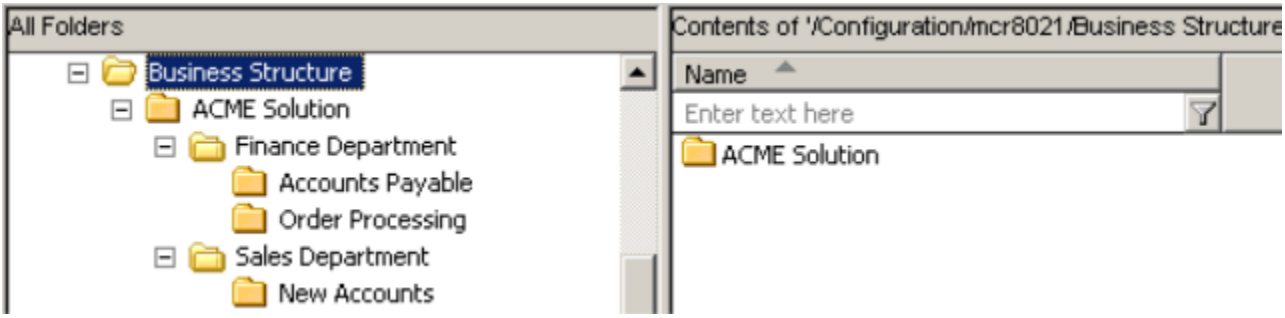
Multiple Solutions can be created by creating additional Business Units or Sites directly beneath the Business Structure folder.

Sample Business Structure in Genesys Administrator



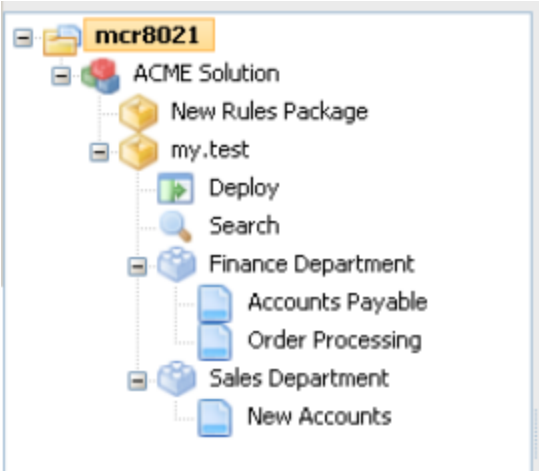
Sample Business Structure in Genesys Administrator

Sample Business Structure in Configuration Manager



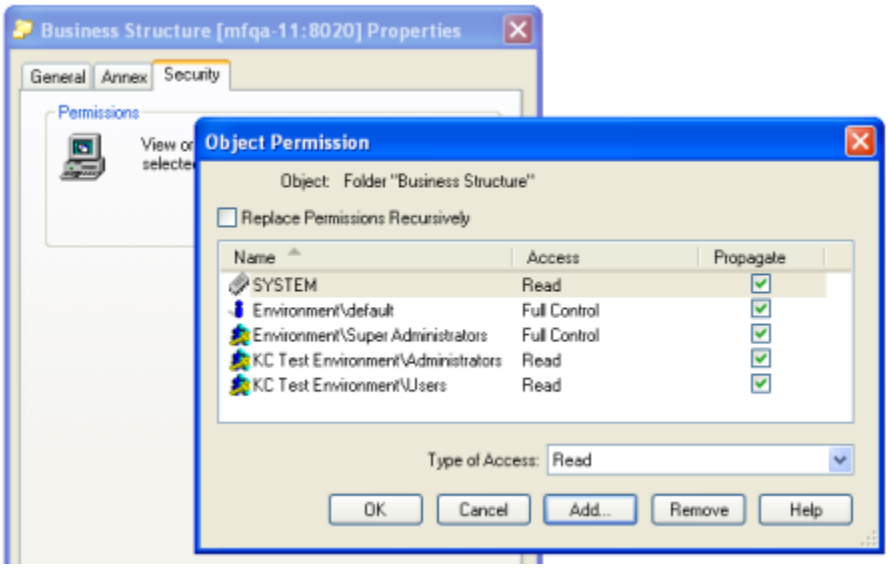
Sample Business Structure in Configuration Manager.

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.

For a full discussion of Role-Based Access Control, please refer to the **Genesys 8.1 Security Deployment Guide**.

Role Permissions for Rules and Rule Packages

Genesys Rules System 8.5 defines a set of role permissions for governing the tasks that can be performed in the Genesys Rules Authoring Tool.

Rules

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.

Rule Packages

- Release 8.5.302 provides package-level overrides to these global roles—a user's role privileges can be restricted to specific rule packages by applying Role-Based Access Control at the rule package level. See **Changes in 8.5.302**.

List of Permissions

- Business Calendar - Create
- Business Calendar - Delete
- Business Calendar - Modify
- Business Calendar - View
- Business Rule - Create
- Business Rule - Delete
- Business Rule - Modify
- Business Rule - View
- Business Rule - Edit Only - allows a user to edit and save only the parameter values of a rule. No other permissions are granted.

- Rule Template - Create
- Rule Template - Delete
- Rule Template - Modify
- Rule Package - Create
- Rule Package - Delete
- Rule Package - Modify
- Rule Package - Deploy
- From 8.5.303—Rule Package - Undeploy (also requires Rule Package - Deploy permission)
- From 8.5.303—Rule Package History - Admin View—Allows viewing of complete package history for a rule package without checking access to the business hierarchy subnodes used inside the rule package. Even with this role privilege enabled, the package history will only be shown for packages that the user can view.
- From 8.5.303—Rule Package History - View Changed By—Allows users to view Changed By information in Package History.
- Locks - Override
- Test Scenario - Create
- Test Scenario - Modify
- Test Scenario - Delete
- Test Scenario - View
- Test Scenario - Execute
- 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 of snapshots, which represents 8.1.2 functionality where users can only deploy the latest version.

Important

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

Changes in 8.5.302

Support for Role-Based Access Control at the Rules Package Level

> **Important**
> GRS requires Genesys Administrator 8.1.305.04 (minimum) for configuring package level permissions.

You can expand the graphics by clicking on them.

Background

Previously, GRAT used Configuration Server Roles to provide only global access control to all packages in a given node of the business hierarchy. The privileges, like `Modify Rule Package`, `Delete Rule Package`, `Modify Rule`, `Delete Rule` and so on, are granted to users via roles. With this approach, if a user is granted the `Modify Rule Package` (for example) privilege, then they can modify all the rule packages defined in a node of the GRAT business hierarchy.

Release 8.5.302 now provides package-level overrides to these global roles—role privileges can be restricted to specific rule packages by applying Role-Based Access Control at the rule package level. The new Rule Package Level Roles (roles created specifically for use with rule packages only) can be mapped to rule packages to override the global-level roles. These Rule Package Level Roles will have no effect if not mapped to a rule package.

New Role Permission—View Rule Package

View access for specific rule packages can now be controlled by using the new role permission **View Rule Package**. The new permission is applicable to only the rule package level.

Existing Role Permissions

All of the existing role permissions except **Create Rule Package** and template-related permissions are applicable at the rule package level too.

Example

In 8.5.302 you can now assign role permissions at both global/node level and at rule-package level to achieve the following outcome:

- Department A
  - Rule package 1
  - Rule package 2
  - Sales
    - Rule package 3

- Department B
  - Rule package 4

- User A—Can see Department A but not Department B
- User B—Can see Department B but not Department A
- User C—Can see rule package 1, but rule package 2 is hidden

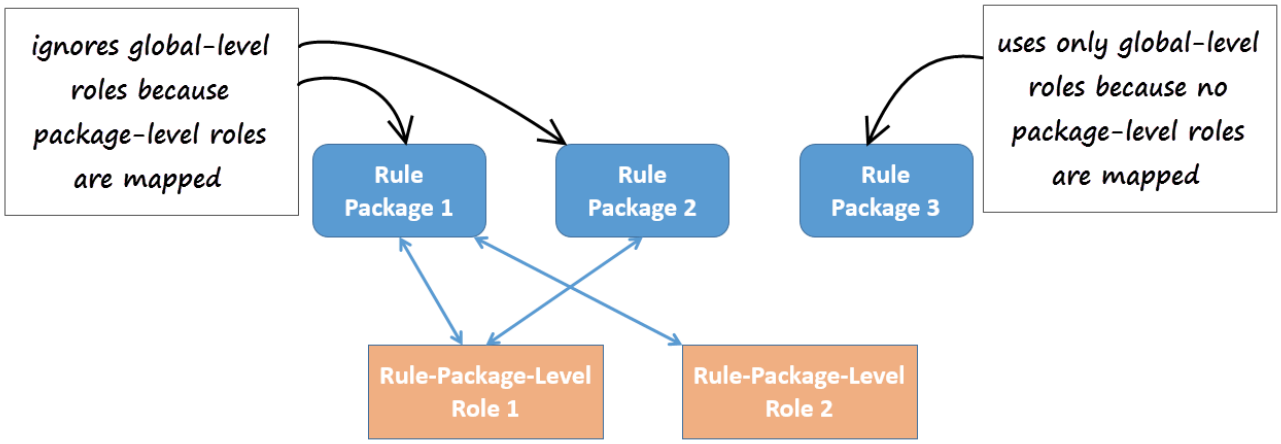
Location

To distinguish these new roles from global-level roles, they are placed in a new folder:

```
[Tenant] > Roles > GRS Rule Package Level Roles
```

Package-Level Overrides

Where package-level roles are mapped to a rule package, they override global-level roles.

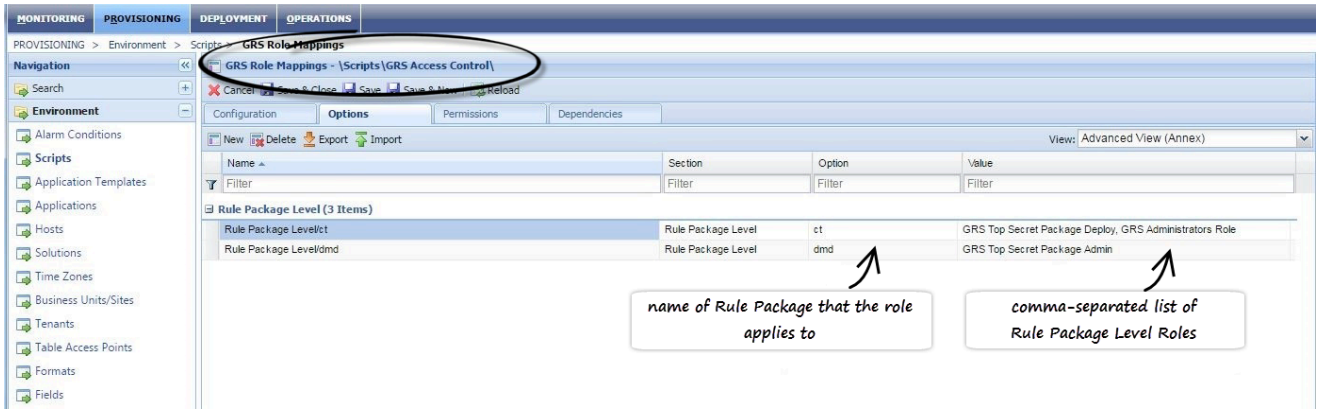


Managing the Mapping of Roles

The mapping of the rule packages to Rule Package Level Roles is managed in Genesys Administrator or Genesys Administrator Extensions, in the options under section **Rule Package Level** of the **\Scripts\GRS Access Control\GRS Role Mappings** script. The example below is from Genesys Administrator.

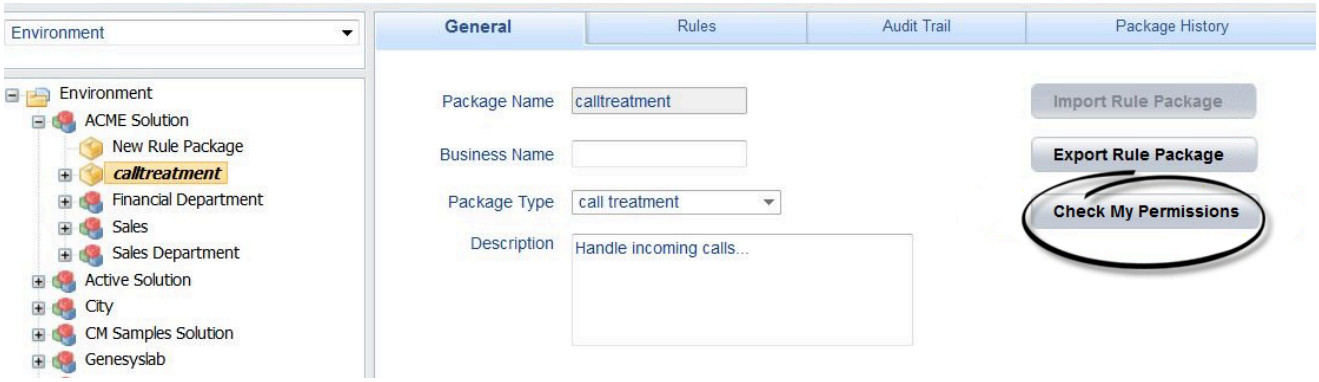
> **Important**
> Because the delimiter in the list of roles is a comma, you can't use commas in the

> names of any role.



Viewing GRAT User Permissions

To enable GRAT users to view their current list of permissions, a **Check My Permissions** button is now also available at the rule-package level and shows the permissions at selected package level.



User Logins

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 GRAT 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 is a check box to add the privileges that are associated with the Genesys Rule Authoring Generic Server.

You can grant users a specific set of permissions by adding them as members of a role—either individually or as part of an access group. There are six groups of privileges:

- Rule Authoring—Create, Delete, Modify, and View
- Rule Packages—Create, Modify, Delete, and Deploy (and View from release 8.5.302)
- 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

Templates and their Script Objects

Templates

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.
- 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.

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 GRAT.

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.

DROOLS 5 Keywords

Drools 5 introduces the concept of hard and soft keywords.

Hard Keywords

Hard keywords are reserved—you cannot use any hard keyword when naming domain objects, properties, methods, functions and other elements that are used in the rule text. The following list of hard keywords must be avoided as identifiers when writing rules:

- true
- false
- null

Soft Keywords

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

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

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

Please take a look at the GRS101 video series for a complete end-to-end tutorial on how to build a template from scratch, create rules and execute calls to the rules engine from Composer.