Genesys Rules System Deployment Guide

High Availability Support

4/22/2025

High Availability Support

Contents

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

Single GRAT instance

GRAT is not considered a critical path application because it only handles the creation, editing and deployment of rules. Where only one GRAT instance is connected to a particular rules repository database at a time, if GRAT should fail, rule execution continues uninterrupted. Only rule editing becomes unavailable.

GRAT Status

GRAT has a `status.jsp` URL that can be used for a health check.

Status	Response Text/Meaning
HTTP 200	• `SYSTEM_STATUS_OK`—GRAT server is up and running
HTTP 503	• `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.

Configuring GRAT clusters

You can 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.

An n-node cluster configuration can be used to deliver High Availability and Load Balancing. For example, 2 or 3 GRATs can be configured in a cluster. It is not recommended to have more than 4 GRATs in a cluster, due to the network demands on the repository database. It is also not recommended to have GRATs which are geographically distant from the repository database server, due to the high network demands placed on the database. It is recommended to have all nodes of a GRAT cluster in the same region as the database server. Users can access the GRAT cluster from different regions via browser (the browser's GRAT bandwidth requirement is insignificant compared to the GRAT's databse bandwidth, so it is best to have users remote via browser and have GRAT close to the DB).

A load balancer can front-end the GRAT UI, and evenly distribute the load across the available healthy GRAT nodes in the cluster. 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.

The load balancer can poll each node's "health check" URL (**/genesys-rules-authoring/status.jsp**) to determine the health of that node in the cluster. In the event of a failure (or planned maintenance) of one of the GRAT nodes in the cluster, the load balancer will detect this (via the healthcheck URL) and no longer send traffic to the node that is down. Any users that were currently logged into that node would be reassigned to another node. They would be prompted to log in again to resume their work: however, any "unsaved" changes at the time of the node failure would be lost.

When a GRAT node is part of a cluster, you should in general set the value of its **clear-repository-cache** option to `false`.