Genesys Callback Private Edition Guide

11/27/2021

Table of Contents

Contents

Find links to all the topics in this guide.

Related documentation:

-
-

Early Adopter Program

Genesys Multicloud CX private edition is being released to pre-approved customers as part of the Early Adopter Program. Please note that the documentation and the product are subject to change. For more details about the program, please contact your Genesys representative.

Genesys Engagement Service (GES) provides callback and click-to-call-in services, which are available with the Genesys Multicloud CX private edition offering.

Overview

Learn more about GES and Genesys Callback, its architecture, and how to support high availability and disaster recovery.

- About Genesys Engagement Service/Callback
- Architecture
- High availability and disaster recovery

Configure and deploy

Find out how to configure and deploy GES.

- Before you begin
- Configure Genesys Engagement Service
- Provision Genesys Engagement Service
- Deploy Genesys Engagement Service
- Upgrade, rollback, or uninstall Genesys Engagement Service

Operations

Learn how to monitor GES and the callback services with metrics and logging.

- Metrics and alerts
- Logging

About Genesys Engagement Service/Callback

Contents

Learn about Genesys Engagement Service/Callback and how it works in Genesys Multicloud CX private edition.

Related documentation:

-
-

Early Adopter Program

Genesys Multicloud CX private edition is being released to pre-approved customers as part of the Early Adopter Program. Please note that the documentation and the product are subject to change. For more details about the program, please contact your Genesys representative.

Welcome to the *Genesys Callback Private Edition Guide*. This document explains the provisioning, deployment, configuration, and start procedures for Genesys Callback. The microservice that provides the callback functionality is called Genesys Engagement Service. Because this guide covers the deployment of the service, Genesys Engagement Service and GES terminology is used in much of the descriptive text and in any sample commands.

Genesys offers classic callback services, allowing consumers to request a callback as soon as an agent with the correct skills is available or to schedule the callback for a specific day and time that is convenient. In addition, Genesys offers robust and feature-rich callback services so you can use Push Notifications and CAPTCHA widgets with your callback offering. You can monitor and manage your callback services in a UI, which includes a view of your callback queues. Key components of the consumer's app or web journey can be preserved for agent or reporting use.

On top of the traditional callback services and scenarios, Genesys also offers the Click-To-Call feature, which lets consumers call your contact center by simply tapping a button in your mobile app.

If you are new to GES/Callback and plan to use Callback on the hosted Genesys Multicloud CX platform, see Provisioning Callback in Designer.

After you have completed Callback provisioning and testing to ensure that calls are routed correctly, and Callback users have been assigned to the correct roles and access groups, you can begin to use the Callback UI. The **Callback** tab displays the list of callback records. Users with sufficient permissions use the **Callback** tab to manage the callback records, including creating, editing, or cancelling callbacks.

Callback Administrators and Developers have access to a **Developer** tab in the Callback UI. Use the **Developer** tab to manage callback activity and features at a more technical level. For example, you can check for errors in Callback API queries or validate API keys. To learn more about the tools on the **Developer** tab, see Troubleshooting and validating functionality.

For information about the Callback user interface (UI), see Genesys Callback Administrator's Guide.

For Release Notes, see Callback Release Notes.

Genesys Callback provides the following callback scenarios when deployed in a Kubernetes cluster:

- Immediate callback
- Scheduled callback
- User-originated Click-To-Call-In

Callback supports integration with Push Notifications and CAPTCHA widgets.

Genesys Callback is enabled by Genesys Engagement Services REST APIs. The Callback APIs are:

- Callbacks: Create, retrieve, cancel callbacks.
- Estimated Wait Time: Retrieve Estimated Wait Time.
- Availability: Retrieve time slots for a callback, matching Office Hours.
- Call In: Request the phone number to call in.
- Queue Status: Retrieve information about a queue's readiness to accept callbacks.
- Statistics: Provides a proxy to the GWS Statistics API. To use the Callback Statistics API, you must first register your GWS credentials in the Callback UI (**Developer** > **Credential Management** > **GWS Credentials** tab).

Callback reporting

You can generate both real-time and historical reports for callbacks. For real-time reporting, you require Genesys Pulse. For historical reporting, you require Genesys Customer Experience Insights (GCXI). For more information about callback reporting, see Callback reports.

Architecture

Learn about Genesys Callback architecture.

Related documentation:

-
-

Early Adopter Program

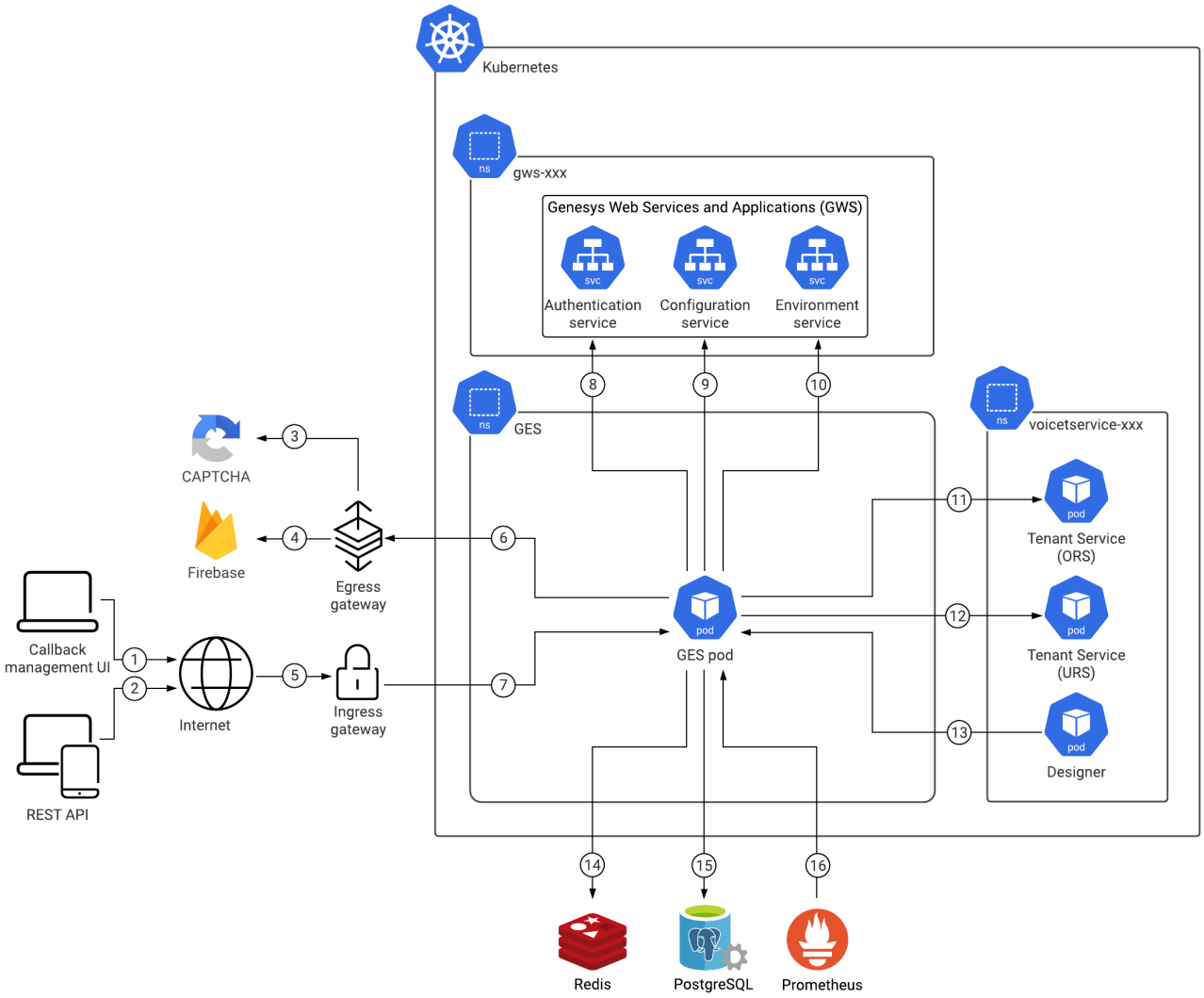
Genesys Multicloud CX private edition is being released to pre-approved customers as part of the Early Adopter Program. Please note that the documentation and the product are subject to change. For more details about the program, please contact your Genesys representative.

For information about the full Genesys Multicloud CX private edition architecture, see Architecture.

For architecture information about GES in relation to the Voice Microservices, including the Tenant Service, see the Voice Microservices Private Edition Guide and the Tenant Service Private Edition Guide.

The following diagram shows the Genesys Engagement Service architecture. There must be at least two GES nodes spread across availability zones, forming a single service for load balancing and high availability.

Configure Redis as primary-replica, non-clustered, single-shard distributed across availability zones. There is a primary node for read-write and one read-only replica, which is a standby in case the primary node experiences a failure.



The following table provides information about the objects and connections shown in the preceding architecture diagram.

Diagram reference number	Port number
1	443
2	443
3	443
4	443
5	443
6	443
7	3005

Diagram reference number	Port number
8	8095
9	8092
10	8091
11	9098
12	5580
13	3050
14	6379
15	5432
16	3050

High availability and disaster recovery

Find out how this service provides disaster recovery in the event the service goes down.

Related documentation:

-
-

Early Adopter Program

Genesys Multicloud CX private edition is being released to pre-approved customers as part of the Early Adopter Program. Please note that the documentation and the product are subject to change. For more details about the program, please contact your Genesys representative.

Service	High Availability	Disaster Recovery	Where can you host this service?
Genesys Engagement Service	N = N (N+1)	Not supported	Primary unit only

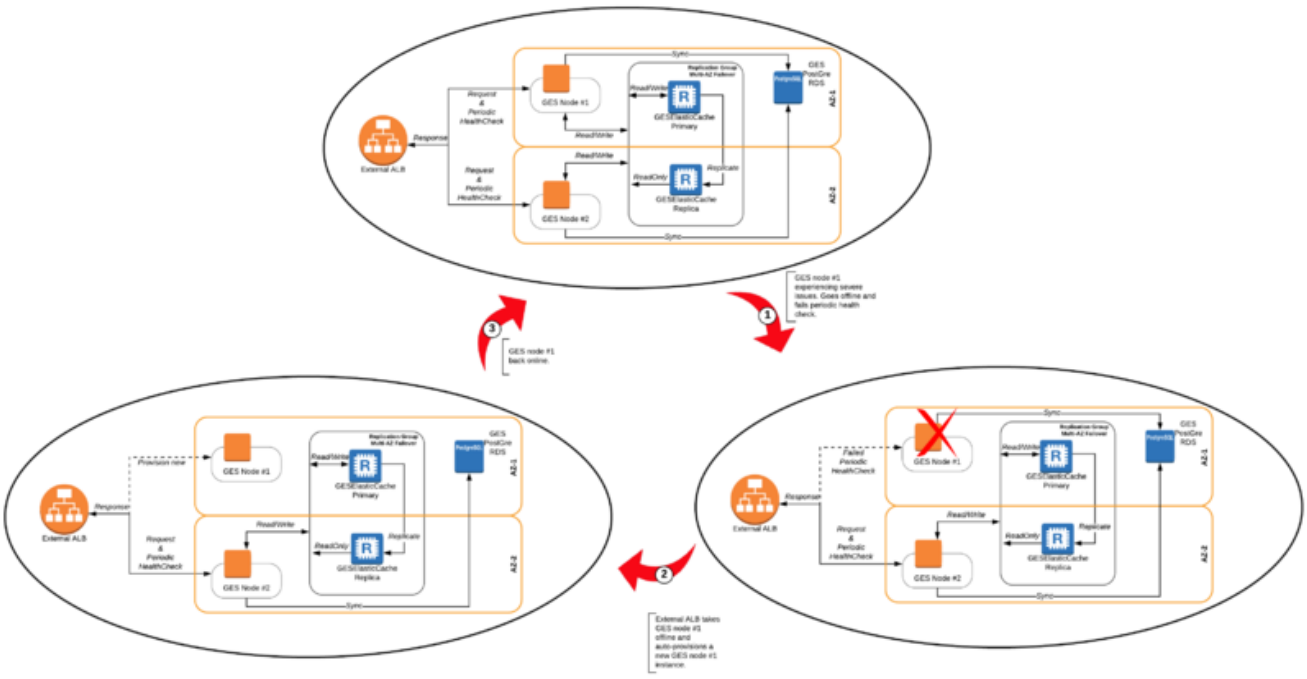
This information is under development: Flagged items aren't yet confirmed or have info coming soon; Checked items are valid.

See High Availability information for all services: High availability and disaster recovery

Genesys Engagement Service (GES) uses automated failure recovery mechanisms that allow GES to try to recover from critical failure scenarios such as a node failure or a partial or full Redis failure. The following sections illustrate how these failure recovery processes work.

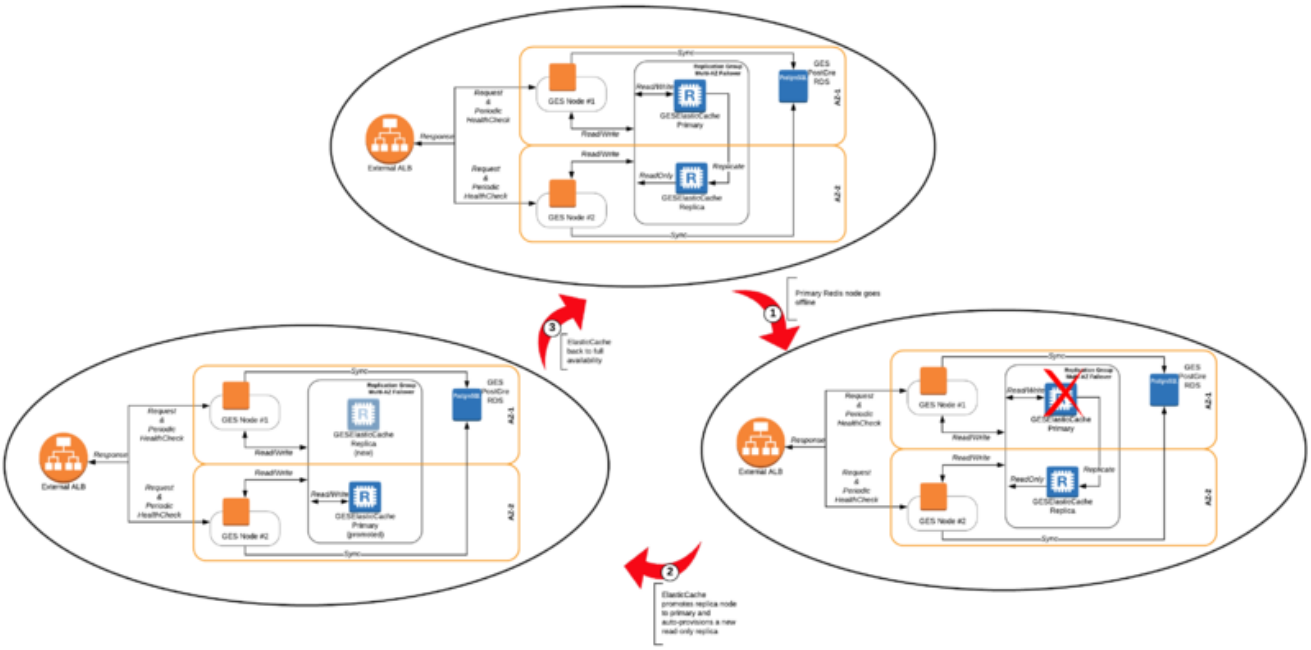
GES node failure

The following diagram depicts the automated recovery procedures when one of the GES nodes goes offline.



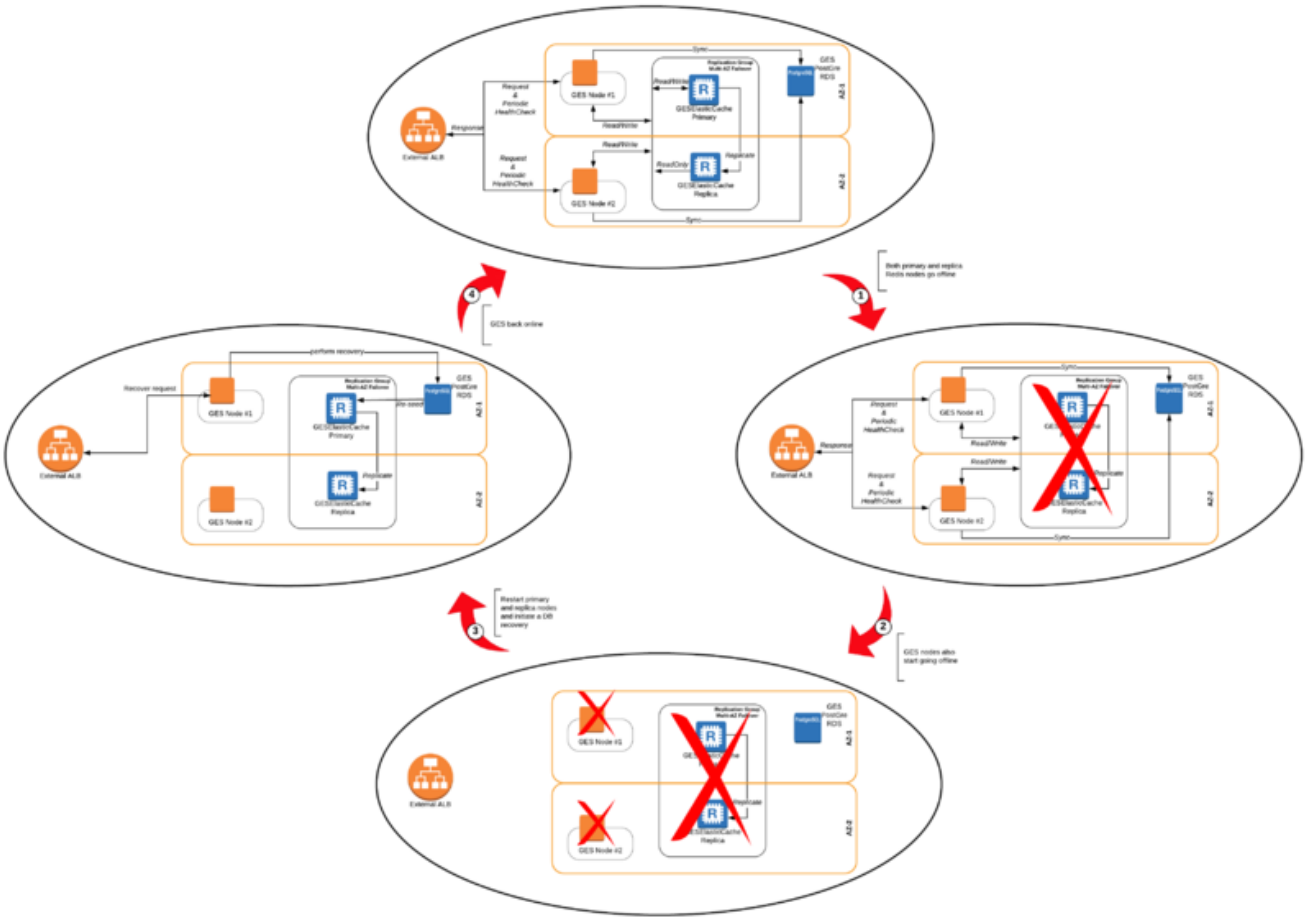
Partial Redis failure

The following diagram depicts the automated recovery procedures when the primary Redis node goes offline.



Full Redis failure

The following diagram depicts the automated recovery procedures when both the primary and replica Redis nodes go offline.



Before you begin

Contents

Find out what to do before deploying Genesys Callback.

Related documentation:

-
-

Early Adopter Program

Genesys Multicloud CX private edition is being released to pre-approved customers as part of the Early Adopter Program. Please note that the documentation and the product are subject to change. For more details about the program, please contact your Genesys representative.

Limitations and assumptions

Not applicable

Download the Helm charts

Genesys Engagement Service (GES) is the only service that runs in the GES Docker container. The Helm charts included with the GES release provision GES and any Kubernetes infrastructure necessary for GES to run, such as load balancing, autoscaling, ingress control, and monitoring integration.

For information about how to download the Helm charts, see Downloading your Genesys Multicloud CX containers.

See Helm charts and containers for Callback for the Helm chart version you must download for your release.

Third-party prerequisites

For information about setting up your Genesys Multicloud CX private edition platform, see Software requirements.

Third-party services

Name	Version	OpenShift	GKE	Purpose	Shared service?	Notes
Redis	6.x	Redis Enterprise Operator	Redis Helm chart	(Mandatory) Used for caching. Only distributions of Redis that support Redis cluster mode are supported.	No	GES requires a dedicated, non-clustered Redis instance.
PostgreSQL	11.x			(Mandatory) Relational database.	Optional	
Consul	1.9.5 - 1.9.x		Consul Helm chart	(Mandatory) Service discovery, service mesh, and key/value store.	Yes	

Storage requirements

The primary contributor to the size of a callback record is the amount of user data that is attached to a callback. Since this is an open-ended field, and the composition will differ from customer to customer, it is difficult to state the precise storage requirements of GES for a given deployment. To assist you, the following table lists the results of testing done in an internal Genesys development environment and shows the impact that user data has when it comes to the storage requirements for both Redis and Postgres.

Test	Redis size	Postgres size (MB)
10,000 Scheduled Callbacks with no user data	26.51 MB	41.1 MB
10,000 Scheduled Callbacks with 10 KB of user data	64.44 MB	252.91 MB
10,000 Scheduled Callbacks with 100k of user data	110.58 MB	595.79 MB

Note: *This is 100k of randomized string in a single field in the user data.*

Hardware requirements

Genesys strongly recommends the following hardware requirements to run GES with a single tenant. The requirements are based on running GES in a multi-tenanted environment and scaled down accordingly. Use these guidelines, coupled with the callback storage information listed above, to gauge the precise requirements needed to ensure that GES runs smoothly in your deployment.

GES

(Based on t3.medium)

- vCPUs: 1
- Memory: 2 GiB
- Network burst: 5 Gbps

Redis

(Based on cache.r5.large) Redis is essential to GES service availability. Deploy two of the Redis caches in a cluster; the second cache acts as a replica of the first. For more information, see Architecture.

GES requires a dedicated, non-clustered Redis instance. Callback data is stored in Redis memory.

- vCPUs: 1
- Memory: 8 GiB
- Network burst: 10 Gbps

PostgreSQL

(Based on db.t3.medium)

- vCPUs: 2
- Memory: 4 GiB
- Network burst: 5 Gbps
- Storage: 100 GiB

Sizing calculator

The information in this section is provided to help you determine what hardware you need to run GES and third-party components. The information and formulas are based on an analysis of database disk storage and Redis memory usage requirements for callback data. The numbers provided here include only storage and memory usage for callbacks. Additional storage and memory is required for configuration data and basic operations.

Requirements per callback

Each callback record (excluding user data) requires approximately 6.5 to 7.0 kB of database disk storage, plus additional disk storage for the user data. Each kB of user data consumes approximately 3.0 kB of disk storage.

Each callback record (excluding user data) requires approximately 4.5 to 5.5 kB of Redis memory, plus an additional 1.25 kB for each kB of user data.

Use the following formulas to estimate disk storage and Redis memory requirements:

- Estimate database disk storage requirements for callback data:
  number of callbacks per day> × (7 kB + (3 kB × kB of user data per callback>)) × 14 days

- Estimate Redis memory requirements for callback data:
  number of callbacks per day> × (5.5 kB + (1.25 kB × kB of user data per callback>)) × 14 days

For example, if a tenant has an average of 100,000 callbacks per day with 1kB user data in each callback:

- The database storage requirement is approximately 14 GB.
- The Redis memory requirement is approximately 9.5 GB.

NOTE: Each callback record is stored for 14 days. If you average about 10k scheduled callbacks every day, and the scheduled callbacks are all booked as far out as possible (that is, 14 days in the future), the number of callbacks to use in storage and memory calculations is 28 days × 10k callbacks per day = 280k callbacks.

Redis operations

The Redis operations primarily update the connectivity status to other services such as Tenant Service (specifically ORS and URS) and Genesys Web Services and Applications (GWS).

When GES is idle (zero callbacks in the past, no active callback sessions, no scheduled callbacks), GES generates about 50 Redis operations per second per GES node per tenant.

Each Immediate callback generates approximately 110 Redis operations from its creation to the end of the ORS session.

For Scheduled callbacks, assuming each callback generates 110 Redis operations when the ORS session is active (based on Immediate callback numbers), there is 1 additional Redis operation for each minute that a callback is scheduled.

For example, if a callback is scheduled for 1 hour from the time it was created, the number of Redis operations is approximately 60 + 110 = 170.

For a callback scheduled for 1 day from the time it was created, it generates approximately 60 × 24 + 110 = 1550 Redis operations, using the following formula for the number of Redis operations per callback:
number of callbacks> × (110 + number of minutes until scheduled time>)

Because the longevity of a callback ORS session depends on the estimated wait time (EWT), the total number of Redis operations performed by GES per minute varies, based on both the number of callbacks in the system and the EWT of the callbacks.

Use the following formula to estimate the number of Redis operations performed per minute:
Total number of Redis operations per minute = (50 base GES Redis operations per second × 60 seconds) + number of upcoming scheduled callbacks in the system> + ((total number of active callbacks> / EWT>) × 110)

Where:

- Total number of active callbacks = number of active immediate callbacks> + number of active scheduled callbacks>, and
- Number of active scheduled callbacks = (number of scheduled callbacks per time slot> / time slot duration>) × EWT>

For example, let's say we have the following scenario:

- Scheduled callbacks:
  - Time slot duration = 15 minutes

    - Maximum capacity per time slot = 100
    - Business hours = 24x7
    - Assume that all time slots are fully booked for the next 14 days
- Number of active immediate callbacks = 1,000
- Estimated wait time = 90 minutes

Using the preceding formulas, estimate the Redis operations per minute:

- Total number of scheduled callbacks = (100 × (60 / 15)) × 24 × 14 = 134,400
- Number of active scheduled callbacks = (100 / 15) × 90 = 600
- Number of upcoming scheduled callbacks = total number of scheduled callbacks> - number of active scheduled callbacks> = (134,400 - 600) = 133,800
- Total number of active callbacks = 1,000 + 600 = 1,600
- Total number of Redis operations per minute = (50 × 60) + 133,800 + ((1,600 / 90) × 110) = 138,756

Redis keys

Each callback creates three additional Redis keys. Given the preceding calculations for Redis memory requirements for each callback, the formula for the average key size is:
(5.5 kB + (1.25 kB × kB_of_user_data_per_callback>)) / 3

Network requirements

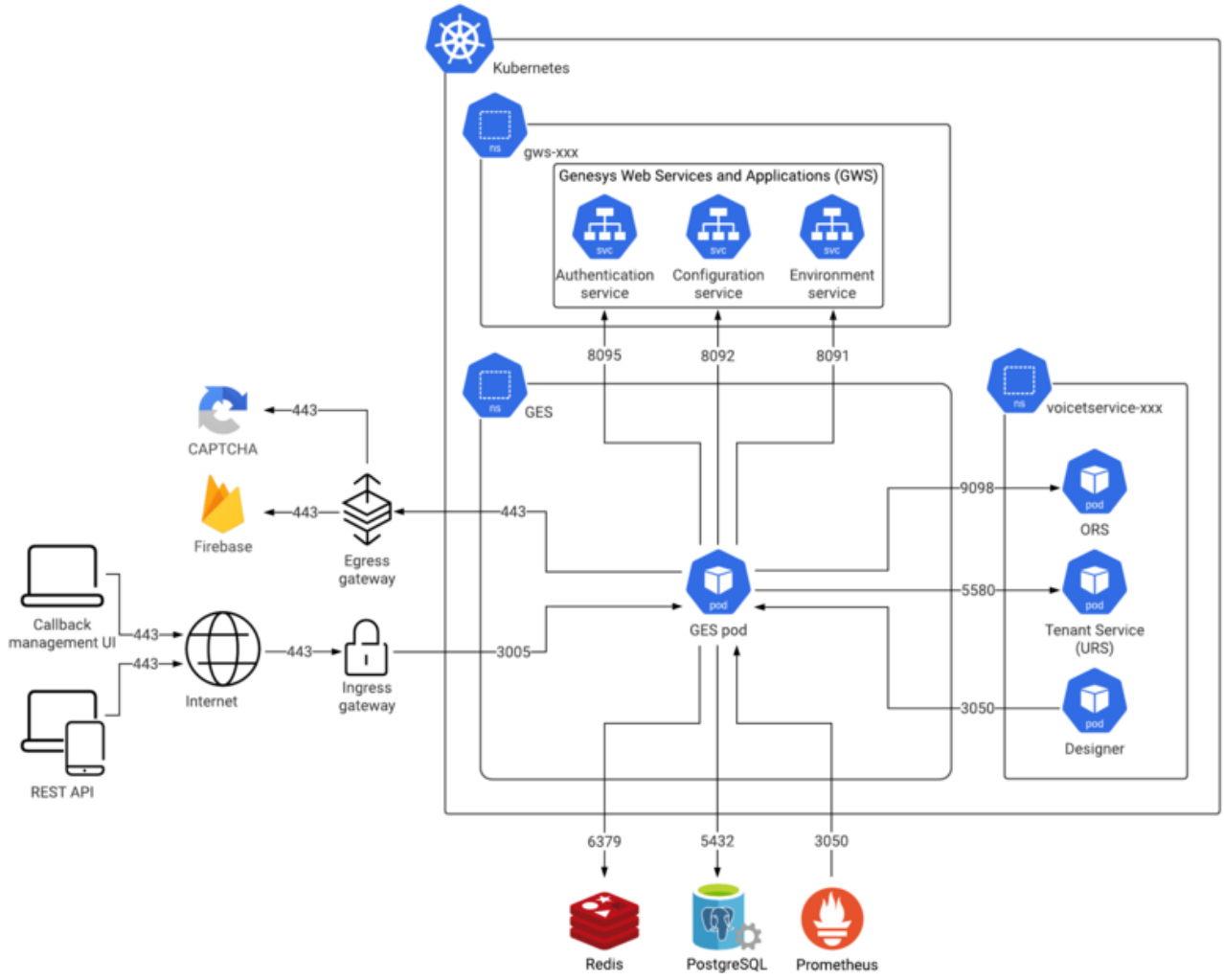
Incoming connections to the GES deployment are handled either through the UI or through the external API. For information about how to use the external API, see the Genesys Multicloud CX Developer Center.

Connection topology

The diagram below shows the incoming and outgoing connections amongst GES and other Genesys and third-party software such as Redis, PostgreSQL, and Prometheus. In the diagram, Prometheus is shown as being part of the broader Kubernetes deployment, although this is not a requirement. What's important is that Prometheus is able to reach the internal load balancer for GES.

The other important thing to note is that, depending on the use case, GES might communicate with Firebase and CAPTCHA over the open internet. This is not part of the default callback offering, but if you use Push Notifications with your callback service, then GES must be able to connect to Firebase over TLS. The use of Push Notifications or CAPTCHA is optional and not necessary for the basic callback scenarios.

GES requires a dedicated, non-clustered Redis instance.



Web application firewall rules

Information in the following sections is based on NGINX configuration used by GES in an Azure cloud environment.

Cookies and session requirements

When interacting with the UI, GES and GWS ensure that the user's browser has the appropriate session cookies. By default, UI sessions time out after 20 minutes of inactivity.

The external Engagement API does not require session management or the use of cookies, but it is important that the GES API key be provided in the request headers in the **X-API-Key** field.

For ingress to GES, allow requests to only the following paths to be forwarded to GES:

- /ges/
- /engagement/v3/callbacks/create

- /engagement/v3/callbacks/cancel
- /engagement/v3/callbacks/retrieve
- /engagement/v3/callbacks/availability/
- /engagement/v3/callbacks/queue-status/
- /engagement/v3/callbacks/open-for/
- /engagement/v3/estimated-wait-time
- /engagement/v3/call-in/requests/create
- /engagement/v3/statistics/operations/get-statistic-ex

In addition to allowing connections to only these paths, ensure that the `ccid` or `ContactCenterID` headers on any incoming requests are empty. This enhances security of the GES deployment; it prevents the use of external APIs by an actor who has only the CCID of the contact center.

TLS/SSL certificate configuration

There are no special TLS certificate requirements for the GES/Genesys Callback web-based UI.

Subnet requirements

There are no special requirements for sizing or creating an IP subnet for GES above and beyond the demands of the broader Kubernetes cluster.

Browser requirements

The Genesys Callback user interface is supported in the following browsers.

Browsers

Name	Version	Notes
Firefox	Current release or one version previous	Genesys also supports the current ESR release. Genesys supports the transitional ESR release only during the time period in which the new ESR release is tested and certified. For more information, see Firefox ESR release cycle. Firefox updates itself automatically. Versions of Firefox are only an issue if your IT department restricts automatic updates.
Chrome	Current release or one version previous	Chrome updates itself automatically. Versions of Chrome are only an issue if your IT department restricts automatic updates.
Microsoft Edge (Legacy)	Current release	
Microsoft Edge Chromium	Current release	

Genesys dependencies

GES has dependencies on several other Genesys services. You must deploy the services on which GES depends and verify that each is working as expected *before* you provision and configure GES. If you follow this advice, then – if any issues arise during the provisioning of GES – you can be reasonably assured that the fault lies in how GES is provisioned, rather than in a downstream program.

GES/Callback requires your environment to contain supported releases of the following Genesys services, which must be deployed before you deploy Callback:

- GWS
- Voice Microservices
- Designer

For detailed information about the correct order of services deployment, see Order of services deployment.

GDPR support

Callback records are stored for 14 days. The 14-day TTL setting starts at the Desired Callback Time. The `Callback TTL (seconds)` setting in the CALLBACK_SETTINGS data table has no effect on callback record storage duration; 14 days is a fixed value for all callback records.

For more information, see **Link to come**.

Configure Genesys Engagement Service

Contents

Learn how to configure Genesys Callback.

Related documentation:

-
-

Early Adopter Program

Genesys Multicloud CX private edition is being released to pre-approved customers as part of the Early Adopter Program. Please note that the documentation and the product are subject to change. For more details about the program, please contact your Genesys representative.

Override Helm chart values

For additional information about overriding Helm chart values, see Overriding Helm Chart values in the *Genesys Multicloud CX Private Edition Guide*.

If you want to use arbitrary UIDs in your OpenShift deployment, you must override the **securityContext** settings in the **values.yaml** file, so that no user or group IDs are specified. For details, see Security context configuration, below.

Genesys Engagement Service (GES) has a Helm value layout inline with other Engage services, however, the use of environment variables as the primary way to configure GES means that setting up the helm values is something that one will need to pay particular attention to. Below is a "section by section" description of the different helm chart parameters, what they influence and anything else an operator might find important to know when provisioning GES within Private Edition.

All values to configure GES are within the `.ges` subsection of the `.Values.yaml` file. For example, to set affinity, you reference `.Values.ges.affinity`. For those values listed in a `misc` section, omit that from the path when referencing an environment variable.

labels

Parameter	Description	Default	Valid values
globalLabels	Labels that are going to be applied to all resources within the GES helm deployment. This is required.	{}	YAML object
podLabels	Labels that will be applied to all GES pods.	{}	YAML object

Parameter	Description	Default	Valid values
	This is required.		
serviceLabels	Labels that will be applied to just the GES service. This is required.	{}	YAML object

annotations

Parameter	Description	Default	Valid values
globalAnnotations	Annotations that are going to be applied to all resources within the GES helm deployment. This is required.	{}	YAML object
podAnnotations	Annotations that will be applied to all GES pods. This is required.	{}	YAML object

configMap

Set values in the config map. The values are divided into two sections:

- Environment variables that control the behavior of GES.
- Values supplied to GES that allow it to integrate with dependencies like Redis, Postgres, GWS, ORS, and so on.

env

log

Parameter	Description	Default	Valid values
level	Controls the level of logging output for GES.	DEBUG	(DEBUG, TRACE, ERROR, FATAL, INFO, WARN)
maxLen	The max length (in bytes) before GES log messages get auto-truncated.	2048	Number

server

Parameter	Description	Default	Valid values
internal_port	The port GES listens to for incoming requests from		Number

Parameter	Description	Default	Valid values
	within Engage. Typically 3050. This is required.		
external_port	The port GES listens to for incoming messages to the external APIs (typically from the outside world). Typically 2050. This is required.		Number
internal_url	A default URL that would allow for ORS to query this instance of GES. The default value assumes GES is within the GES namespace. This is required.	ges.ges	String

misc

Parameter	Description	Default	Valid values
regionAffinity	Determines the ORS/URS nodes that will be given priority to handle requests for a given GES nodes. If all servers within the priority region are down, a secondary region will be used. This is required.		String
devopsAccessGroups	The list of roles that are to be granted the DevOps permission in GES. This is required.	Platform Read Only,Platform Provisioning,Platform Internal Administrator	String

integrations

redis

Parameter	Description	Default	Valid values
host	The host of GES' redis instance. This can also be provisioned using secrets. This is required.		String

Parameter	Description	Default	Valid values
port	The port the Redis instance listens on. This can also be provisioned using secrets. This is required.		Number
secure	True if GES is to connect to Redis over TLS. This can also be provisioned using secrets. This is required.	ges.ges	Boolean

db

Parameter	Description	Default	Valid values
host	The host of GES' Postgres instance. This can also be provisioned using secrets. This is required.		String
name	The name of the DB provisioned for GES. This can also be provisioned using secrets. This is required.		string
secure	True if GES is to connect to Postgres over TLS. This can also be provisioned using secrets. This is required.		Boolean

gws

Parameter	Description	Default	Valid values
auth	The internal-facing hostname/port for GWS auth service. This is required.		String
env	The hostname/port for GWS env service. This is required.		String

Parameter	Description	Default	Valid values
conf	The hostname/port for GWS config service. This is required.		String
public_auth	The public-facing hostname/port for GWS auth service. This is required.		String

vmcs

Parameter	Description	Default	Valid values
redis_host	The hostname of the Voice Microservices Redis instance. This can also be supplied through the use of secrets, but if supplied through secrets, it will override the value supplied in the map. This is required.		String
redis_port	The port of the Voice Microservices Redis instance. This can also be supplied through the use of secrets, but if supplied through secrets, it will override the value supplied in the map. This is required.		String
redis_cluster_mode	True if Voice Microservices is running in cluster mode. Otherwise, False. This is required.		String
redis_stream_name	The Redis stream that GES should connect to to write information about new callbacks. This is required.	"NewCallbackStream"	String
ors_redis_location	A compound value made from defining the Redis host/port. Leave as the default value. This can also be supplied through the use of secrets, but if supplied through secrets, it will override the value supplied in the config map.	"{{ .Values.ges.configMap.integrations.vmcs.redis_host }}:{{ .Values.ges.configMap.integrations.vmcs.redis_port | default 10000 }}"	

Parameter	Description	Default	Valid values
	This is required.		

deployment

image

Parameter	Description	Default	Valid values
imagePullSecrets	A secret needed to authenticate with the docker registry. Might not be required; implementation is based on how you have set up your cloud. For example, in Genesys CX on Azure, the Pod ID is used to authenticate with the Azure Container Registry.	[ ]	List
imagePullPolicy	How often the pod will try to pull a fresh image on start up. This is required.	Always	String
registry	The docker registry that the GES image is to be pulled from. This is required.		String
image	The name of the docker image for GES. This is required.	genesys/ges	String

hpa

Parameter	Description	Default	Valid values
enabled	True when horizontal pod auto-scaling is enabled. Might not be required; implementation is based on how you have set up your cloud. For example, in Genesys CX on Azure, the Pod ID is used to authenticate with the Azure Container Registry.	True	Boolean
maximumReplicas	Maximum number of GES pods that will be running in a deployment.	5	String

Parameter	Description	Default	Valid values
minimumReplicas	Minimum number of pods that will run in a deployment.	1	String
targetMetrics	The metrics that the Kubernetes controller will consult when evaluating if additional GES pods are needed. For more information, see the Kubernetes documentation.		String
behavior	Defines the behavior when extra pods for GES are spun up and down, including how many pods should be added or removed at a time and how long the deployment is given to stabilize before spinning up or winding down extra pods. For more information, see the Kubernetes documentation.		

misc

Parameter	Description	Default	Valid values
env	A list of environment variables to be defined within the pod. These can be defined using simple key value pairs or using secrets as described in the Kubernetes documentation.	[ ]	List
envFrom	A list of config maps from which the GES pod will get the environment. Be sure to include all config maps and other entries from which you wish to pull environment variables. For more information, see the Kubernetes documentation.		List
min_ready_seconds	The amount of seconds that a pod must be in the ready state before the Kube controller begins routing traffic to it.	300	String
dnsConfig	Values related to the configuration of DNS resolution.		Object

Parameter	Description	Default	Valid values
	For more information, see the Kubernetes documentation.		
replicas	The number of replica GES deployments that are part of the cluster by default. This can be scaled up or down. For more information, see the Kubernetes documentation.	1	Number
nodeSelector	Contains any and all values that might aid the Kubernetes controller to determine what kind of cluster GES is to be scheduled on. For more information, see the Kubernetes documentation.		Object

resources

Parameter	Description	Default	Valid values
limits	Defines the limits of the resource requests that a GES pod can make from the cluster as well as the maximum amount to allot. For more information, see the Kubernetes documentation.	cpu: 1250m memory: 2048Mi	Object
requests	Defines the size of CPU/ RAM requests a GES pod can make from the cluster as well as the maximum amount to allot. For more information, see the Kubernetes documentation.	cpu: 75m memory: 512Mi	Object

ingress

ingressint

Parameter	Description	Default	Valid values
enabled	True when you want a shared app gateway to govern incoming connections to GES. This is required.	false	Boolean
annotations	Any annotations that are to be applied to the ingress object that governs how GES communicates with a shared App Gateway.		
hosts	The host name to reach GES from a shared app gateway.		
paths	The set of paths that the app gateway will forward to GES. Omitting these paths could cause the external APIs to break.	- /ges/ - /engagement/v3/callbacks/create - /engagement/v3/callbacks/cancel - /engagement/v3/callbacks/retrieve - /engagement/v3/callbacks/availability/ - /engagement/v3/callbacks/queue-status/ - /engagement/v3/callbacks/open-for/ - /engagement/v3/estimated-wait-time - /engagement/v3/call-in/requests/create - /engagement/v3/statistics/operations/get-statistic-ex	List
tls	Details of the TLS certificates for incoming connections to GES.		

monitoring

prometheus

Parameter	Description	Default	Valid values
use_service_monitor	True when you use a Service Monitor as the mechanism for service/pod discovery by the Prometheus framework.		boolean

Parameter	Description	Default	Valid values
	For more information, see Simple Management of Prometheus Monitoring Pipeline with the Prometheus Operator.		

grafana

Parameter	Description	Default	Valid values
enabled	True if Grafana dashboards will be deloyed from config maps defined in the Helm Chart.	False	Boolean

podDisruptionBudget

Parameter	Description	Default	Valid values
enabled	True when a PodDisruptionBudget is defined for GES.	True	Boolean
strategy	The strategy used to implement the pod disruption budget. For more information, see the Kubernetes documentation.		

secrets

The **secrets** section defines how different types of secrets are made available to the GES deployment through the use of volume mounts. Using this method is not strictly necessary, as it is possible to provide the same configuration information using Environment variables. At a high level, the kind of data that is made available through the use of secrets includes Redis/ORS Redis hosts and credentials, GWS client information as well as the DB credentials. For a more thorough explanation and examples of how GES can leverage the secrets capabilities, see Define secrets. For a detailed look at the capabilities of Kubernetes in general, see the Kubernetes documentation on secrets and the Kubernetes documentation on volumes and volume mounts, which GES uses to actually deliver the secrets.

volumeMounts

Parameter	Description	Default	Valid values	Notes
enabled	True when volume mounts are used for the deployment.	false	Boolean	Required

Parameter	Description	Default	Valid values	Notes
	This is required.			
list	A list of volume mounts to be provisioned for the GES deployment.			

volumes

Parameter	Description	Default	Valid values
enabled	True when volumes are used for the deployment. This is required.	false	Boolean
list	A list of volumes to be provided for the GES deployment.		

misc

Parameter	Description	Default	Valid values
config_folders	The list of folders that GES uses to read configuration information. These folders are the volumes mounted by the secrets infrastructure. For more information, see Configure security.	""	String

service

Parameter	Description	Default	Valid values
port_external	The port on which the service listens for external connections from outside the Genesys Multicloud CX solution. This is required. For more information, see the Kubernetes documentation.	80	Number
port_internal	The port on which the service listens for connections from inside the Genesys Multicloud CX solution. This is required.	8080	Number

Parameter	Description	Default	Valid values
	For more information, see the Kubernetes documentation.		
type	Defines whether or not (and how) a service is exposed to the outside world. This is required. For more information, see the Kubernetes documentation.	ClusterIP	See the Kubernetes documentation.

misc

Parameter	Description	Default	Valid values
serviceName	The service name you want to attach to a GES deployment. Only used in the Helm charts to help determine the name attached to the deployment.	ges	String
tolerations	Defines the tolerations to be applied to the GES pods. Pods with matching taints will be favored for scheduling. For more information, see the Kubernetes documentation.	{}	Object
affinity	Defines the pod affinity behavior used to determine which nodes the GES pods are scheduled on. For more information, see the Kubernetes documentation.	{}	

Configure Kubernetes

For information, see the following resources:

- the configMap section in the Helm values description
- the secrets section in the Helm values description
- Security
- Deploy the service

Configure security

In addition to supporting configuration through the supply of environment variables, GES supports configuration information being supplied through Kubernetes' secrets, which are subsequently mounted as volumes on the Kubernetes pod. This "secrets" mechanism is required when providing sensitive information to GES, such as usernames and passwords for connections to Redis, Postgres, GWS, and others. However, you can also use secrets to supply less-sensitive information to GES.

To successfully acquire the GES docker image, deploy a secret in the GES namespace that provides Kubernetes with necessary credentials to authenticate to the JFrog artifactory.

This section provides information about the security context settings and a general overview of how to enable secrets in the Helm charts through the use of volume mounts and includes a detailed look at how to integrate with other components using secrets.

Security context configuration

The security context settings define the privilege and access control settings for pods and containers. For more information, see the Kubernetes documentation.

By default, the user and group IDs are set in the **values.yaml** file as `500:500:500`, meaning the **genesys** user.

```
securityContext:
   runAsNonRoot: true
   runAsUser: 500
   runAsGroup: 500
   fsGroup: 500
```

Arbitrary UIDs in OpenShift

If you want to use arbitrary UIDs in your OpenShift deployment, you must override the **securityContext** settings in the **values.yaml** file, so that you do not define any specific IDs.

```
securityContext:
   runAsNonRoot: true
   runAsUser: null
   runAsGroup: 0
   fsGroup: null
```

Configuring a JFrog secret

For the GES pod to pull the docker image from the Genesys artifactory, your JFrog credentials must be available to the pod. This is done using a two-step process:

1. Create a secret in the GES namespace containing the username and password associated with your JFrog credentials.
2. Enter the name of the secret in the Values.yaml file. For information, see Updating the Values.yaml file.

JFrog secret layout

While instructions on how best to create the secret will vary between Kubernetes providers, what is consistent is that the secret containing the JFrog account details must be in a secret of type `kubernetes/dockerconfigjson`. You can find more information on this topic in the Kubernetes documentation. Regardless of platform, you can create the secret using the following `kubect` command:

```
kubectl create secret docker-registry  \
  --docker-username= \
  --docker-password= \
  --docker-email=
```

Updating the Values.yaml file

In your Values.yaml file, update the value of `.Values.ges.deployment.image.imagePullSecrets`. For example:

```
imagePullSecrets: - name:
```

Providing secrets to GES

At present, while it is possible to supply confidential values to GES through the facility in Kubernetes where secrets can be read into Pod environment variables, this is not always the case. Genesys strongly recommends that you make confidential information such as database usernames and passwords, GWS client information, and Redis keys available to GES through volume mounts. Volume mounts are described in this section, but for more information, see the Kubernetes documentation.

Secrets through volumes

GES incorporates secrets through the use of Volumes that are mounted to the GES Kubernetes pod. There are a number of ways in which you can define the secrets in the Kubernetes deployment, including manually defining secrets that are deployed along with the GES template, using tools such as Terraform, and so on. Instructions about how to leverage the different tools is outside the scope of this documentation. What's important is that you turn the secret into a volume.

To enable volumes, set the value `.Values.ges.secrets.volumes.enabled` to **true**, and then – for each secret that you want to make available – fill out a list entry. For example:

```
volumes:
  enabled: true
  list:
  - name:
    secret:
      secretName:  # This is the name of the secret within the Kubernetes Cluster
```

You can then mount the volume to GES. Set `.Values.ges.secrets.volumeMounts.enabled` to **true** and fill out a list entry. For example:

```
volumeMounts:
  enabled: true
  list:
    - mountPath:  # Best practise is just / but this can be different if requirements dictate
```

```
      name:
      readOnly: true
```

Finally, you must specify the folders that contain the configuration information for GES. To do this, enter the list of s in the form of a comma-separated string as the value for `.Values.ges.secrets.configFolders`.

While the examples here demonstrate how to provide secrets as typical Kubernetes secrets, GES also supports mounting secrets using the Container Storage Interface. Provided the CSI is configured correctly in the Kubernetes environment, you should be able to set this up with only some minor changes in the Helmvalues files.

Using secrets to integrate with other software

GES primarily uses secrets to create a secure mechanism through which it can obtain the credentials necessary to authenticate with services like Redis, GWS, the Postgres database, and the Voice Microservices Redis. This section provides information about how to accomplish this integration; this information is based on deployments on Azure.

REDIS

The REDIS-CACHEKEY must be supplied to GES through the use of an opaque secret. This is the password used to connect to the GES Redis instance. For this secret, the key is REDIS-CACHEKEY and the value is the password. You can provision this on its own or along with other secrets such as the database access information.

DB

The DB-USER and DB-PASSWORD must be supplied to GES through the use of an opaque secret. This is the username and password for GES to connect to the Postgres database. You can provision this on its own or along with other secrets such as the REDIS-CACHEKEY.

GWS

The AUTHENTICATION-CLIENT-ID and AUTHENTICATION-CLIENT-PASSWORD needed to authenticate with GWS must be supplied as an opaque secret. GES supports both encrypted and unencrypted client grants. However, if using an unencrypted grant, it is important that you encode the AUTHENTICATION-CLIENT-PASSWORD using the aes-128-ecb algorithm and an encryption key supplied by the password defined in the Helm value `.Values.ges.configMap.env.decrypt_password`.

ORS

The password to the ORS Redis instance must be supplied to GES through the use of an opaque secret. Unlike other secrets, which can be set up as simple key-value pairs, ORS REDIS information must be supplied to GES in a key-value pair where the key is `voice-redis-ors-stream` and the value is a JSON string with the following configuration:

```
{
    "password":"" # Optional,
    "rejectUnauthorized":"true" # Required,
    "servername":" # Optional
}
```

While this is the required method to supply the ORS REDIS password, other configuration information can be supplied as well. If configuration information is present, it will override any configuration supplied through the Config Maps.

Provision Genesys Engagement Service

Contents

- Administrator

Learn how to provision Genesys Engagement Service.

Related documentation:

-
-

Early Adopter Program

Genesys Multicloud CX private edition is being released to pre-approved customers as part of the Early Adopter Program. Please note that the documentation and the product are subject to change. For more details about the program, please contact your Genesys representative.

Provision GES

For basic provisioning of GES to support the use of callback, see Provisioning Callback in Designer.

Guidance on how to configure Click-to-Call-In, CAPTCHA, Push Notifications, and other more advanced use cases can be found in the *Callback Administrator's Guide*.

You must assign users to specific Roles and Access Groups for access to the Callback UI. Genesys Callback uses Role-based-access-control (RBAC) to allow and restrict users' activities within the Callback UI. For more information about Callback-related Roles and Access Groups, see Controlling user access.

Deploy Genesys Engagement Service

Contents

Learn how to deploy Genesys Engagement Service.

Related documentation:

-
-

Early Adopter Program

Genesys Multicloud CX private edition is being released to pre-approved customers as part of the Early Adopter Program. Please note that the documentation and the product are subject to change. For more details about the program, please contact your Genesys representative.

Important

Make sure to review Before you begin for the full list of prerequisites required to deploy Genesys Engagement Service.

For solution-level deployment information, see [[PrivateEdition/Current/PEGuide/GetStarted|]].

Deploy the service

1. Because Genesys Engagement Service (GES) relies on other services, you must ensure that the following have been deployed and configured successfully before you deploy GES:
   - Voice Microservices
   - Designer
   - GWS
   - Agent Setup (you require Agent Setup to create user logins for GES and to assign the necessary privileges)
2. Create dedicated Redis and Postgres instances for GES. Note the hostname and the credentials to authenticate and then provision Kubernetes secrets to store the following:
   - REDIS-CACHEKEY
   - DB-USER
   - DB-PASSWORD
3. Create the GES project/namespace:

This will differ between environments. Be sure to check the instructions specific to the environment in Deploy in OpenShift.

4. Provision a set of GWS client credentials for GES. You can do this with the following `curl` command.

```
curl -X "POST" "http://<GWS-INTERNAL-URL>/auth/v3/ops/clients" \
     -H 'Content-Type: application/json; charset=utf-8' \
     -u 'opsAdmin:opsPass' \
     -d $'{
  "data": {
    "authorities": [
      "ROLE_INTERNAL_CLIENT"
    ],
    "contactCenterIds": [
      "<YOUR CCID>"
    ],
    "redirectURIs": [
      "https://ges-external-ges.apps.<DOMAIN_NAME>/ges/ui/login.html",
      "http://ges-internal-ges.apps.<DOMAIN_NAME>/ges/ui/login.html"
    ],
    "scope": [
      "*"
    ],
    "authorizedGrantTypes": [
      "client_credentials",
      "authorization_code",
      "refresh_token",
      "implicit",
      "password"
    ],
    "refreshTokenExpirationTimeout": 43200,
    "accessTokenExpirationTimeout": 43200,
    "clientType": "CONFIDENTIAL",
    "client_secret": "<YOUR CLIENT SECRET>",  # NOTE: This is your unencrypted GWS client secret.Encrypted it will be your AUTHENTICATION-CLIENT-SECRET
    "name": "ges_client",                     # It is possible to change this name
    "client_id": "ges_client"                 # It is possible to change this name. This is what is stored in AUTHENTICATION-CLIENT-ID
  }
}'
```

5. Provision secrets for GES:
   For more information about how GES leverages secrets and how they can be provisioned, see the secrets section.

6. Create the Helm values file:
   Using information in the Override Helm chart values section, create a Helm values file that describes the details of your GES deployment, secrets, and environment.

   A base Helm values file is included with the GES Helm charts that you receive from Genesys. The GES base file uses the filename, `ges-base-values.yaml`.

7. Install the GES Helm charts.

8. Validate the GES helm charts using:
   ```
   > helm upgrade --install --dry-run -n -f
   ```
   Review the output to ensure that the generated Helm manifest matches your expectations.

9. Install the GES Helm charts:
   ```
   > helm upgrade --install -n -f
   ```

Deploy in OpenShift

1. Create the GES Namespace/Project:

```
oc new-project ges
```

2. Switch to the GES project:

```
oc project ges
```

3. Create a secret to pull content from the JFrog Artifactory:

```
oc create secret docker-registry jfrog-stage-credentials \
--docker-server=<JFROG>-ARTIFACTORY-URL> \
--docker-username=<JFROG-USERNAME> \
--docker-password=<JFROG-PASSWORD> \
--docker-email=<emailid>

oc secrets link default jfrog-stage-credentials --for=pull
```

You can use a process similar to this one to provision the other secrets necessary for a GES deployment.

Validate the deployment

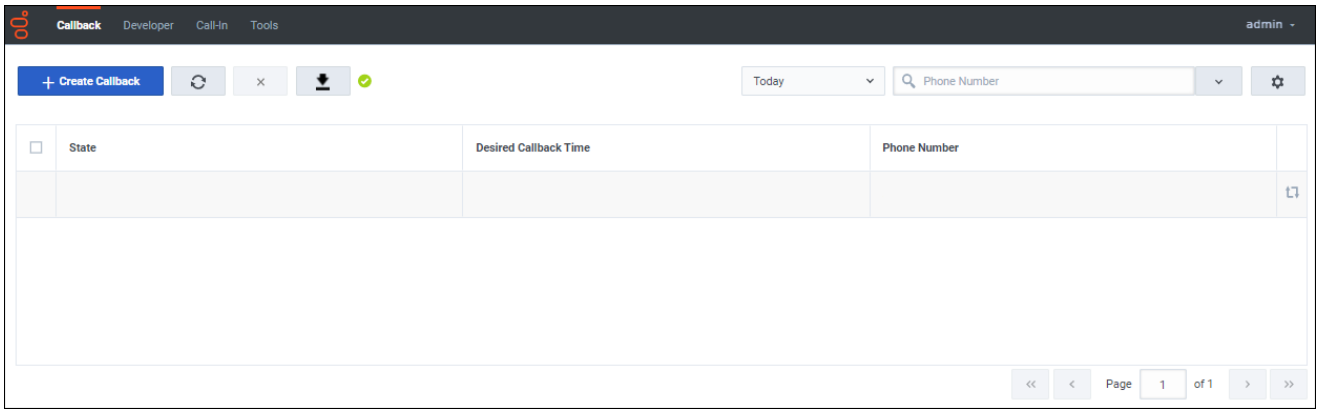
Once GES has been installed and provisioned following the processes outlined above the best way to do that is to check that Callback and another provisioned services work as intended. In addition, logging in to the UI is a good way to check that privileges are working as well as monitor that callbacks, call ins and other requests have been made successfully.

Validate the UI

Login to the UI.

With a user that has been assigned the **Callback Administrator** role, attempt to log into the Callback UI. You can do this at `/ges`.

If you successfully log in, then you see an empty **Callback** grid as the main feature of the screen:



Navigate to some of the other tabs and verify that no errors pop up in the lower corner of the screen as the system attempts to read and retrieve data.

If, at any point, data fails to load or you are locked out due to insufficient permissions, adjust the roles and privileges assigned to Callback UI users and try again.

Create a callback

There are two ways to create a callback request as an external user. You can create a callback request from the Callback UI or by using the /engagement API. For information about using the Callback UI to create a callback, see Managing callbacks. For information about using the APIs, see Genesys Multicloud CX REST APIs and tutorials for Callback.

Create Call-In request (optional)

The Click-to-Call-In scenario is optional functionality. If you plan to use the Click-to-Call-In feature, see the following pages for more information:

- Description of the Click-to-Call-In feature
- Provision the Click-to-Call-In scenario
- Configure Click-To-Call-In Groups
- View Click-To-Call-In records

Upgrade, rollback, or uninstall Genesys Engagement Service

Contents

Learn how to upgrade, rollback or uninstall Genesys Engagement Service.

Related documentation:

-
-

Early Adopter Program

Genesys Multicloud CX private edition is being released to pre-approved customers as part of the Early Adopter Program. Please note that the documentation and the product are subject to change. For more details about the program, please contact your Genesys representative.

Upgrade Genesys Engagement Service

GES uses a rolling upgrade approach.

During a prescribed maintenance window, upgrade the Helm deployment for GES by running the following command:

```
> helm upgrade --install -n -f
```

Versions must match	Ensure that the version of GES and the Helm chart match. While Helm charts might work with other versions of GES, this is not tested and Genesys cannot guarantee success.

Provided there are no issues or errors when running the Helm command, you can expect GES to work with any updates supplied in the latest version of the Helm charts. You can verify that GES has upgraded successfully by either looking at dashboards (if provisioned) or using the health check APIs for GES (through the internal port query `ges:3050/ges/v1/health/detail`) to make sure that all dependencies are working as provided. In some instances, there might be changes to the GES Helm manifest that, due to restrictions in Kubernetes, require the existing GES deployment to be uninstalled and then re-installed, rather than simply upgraded in place. Consult the Callback Release Notes to see if this will be necessary.

Rollback Genesys Engagement Service

To roll back to a previous iteration of GES, follow the same process you used to upgrade GES. Be sure to downgrade both the GES version and the Helm chart version that you're using. It might be necessary to use an older version of the Helm values file.

Uninstall Genesys Engagement Service

You can uninstall GES using the Helm `uninstall` command. Depending on how services like Postgres and Redis are provisioned for GES, you will have to decommission those services separately. Discussion of that process is outside the scope of this document.

Metrics and alerts

Contents

Learn which metrics you should monitor for and when to sound the alarm.

Related documentation:

-
-

Early Adopter Program

Genesys Multicloud CX private edition is being released to pre-approved customers as part of the Early Adopter Program. Please note that the documentation and the product are subject to change. For more details about the program, please contact your Genesys representative.

For information about configuring a monitoring tool, see **Link to come**.

Metrics and alerting

GES exposes default metrics about the state of the Node.js application; this includes CPU usage, memory usage, and the state of the Node.js runtime.

You'll find helpful metrics in the **GES Metrics** subsection, which includes some basic metrics such as REST API usage, the number of created callbacks, call-in requests, and so on. These basic metrics are created as counters, which means that the values will monotonically increase over time from the beginning of a GES pod's lifespan. For more information about counters, see Metric Types in the Prometheus documentation.

You can develop a solid understanding of the performance of a given GES deployment or pod by watching how these metrics change over time. The sample Prometheus expressions show you how to use the basic metrics to gain valuable insights into your callback-related activity.

For information about deploying dashboards and accessing sample implementations, see Grafana dashboards and Sample implementations.

Sample Prometheus expressions

For more information about querying in Prometheus, see Querying Prometheus.

Purpose	Prometheus snippet	Notes
Find the number of Callbacks Created within a given time range across all tenants.	sum(increase(ges_callbacks_created{tenant="$Tenant"}[$__range]))	The same type of expression can be used to track call-in requests, and other metrics.

Purpose	Prometheus snippet	Notes
Find the number of Callbacks Created per minute for a given tenant.	sum by (tenant) (rate(ges_callbacks_created{tenant=~"$Tenant"}[5m])) * 60	The same type of expression can be used to find callbacks, call-ins and other metrics.
Find the number of API failures per minute (across all tenants).	sum by (path, httpCode) (rate(ges_http_failed_requests_total{tenant=~"$Tenant"}[5m]) * 60)	
Find the API success rate over a selected time range.	1 - (increase(sum(ges_http_failed_requests_total{tenant=~"$Tenant"})[$__range]) / increase(sum(ges_http_requests_total{tenant=~"$Tenant"})[$__range]))	
Find the 15-minute rolling average response time by endpoint.	sum by (method, route, code)(increase(ges_http_request_duration_seconds_sum{pod=~"$Pod"}[15m])) / sum by (method, route, code)(increase(ges_http_request_duration_seconds_count{pod=~"$Pod"}[15m]))	
Find the 15-minute rolling average response time by pod.	sum by (pod)(increase(ges_http_request_duration_seconds_sum{pod=~"$Pod"}[15m])) / sum by (pod)(increase(ges_http_request_duration_seconds_count{pod=~"$Pod"}[15m]))	
Find the number of HTTP 401 errors per minute.	sum(rate(ges_http_failed_requests_total{httpCode="401", pod=~"$Pod"}[5m]) * 60)	Change the `httpCode` variable to query other response types.

Health metrics

Health metrics, that is, those metrics that report on the status of connections from GES to dependencies such as Tenant Service (ORS), GWS, Redis, and Postgres, do not work like the metrics described above. Instead, they are implemented as a gauge that toggles between "0" and "1". For information about gauges, see the Prometheus Metric types documentation. When the connection to a service is down, the metric is "1". When the service is up, the metric is "0". Also see How alerts work.

How alerts work

In a Kubernetes deployment, GES relies on Prometheus and Alertmanager to generate alerts. These alerts can then be fowarded to a service of your choice (for example, PagerDuty). For information about finding sample alerts, see Sample implementations.

While GES leverages Prometheus, GES also has internal functionality that manually triggers alerts when certain criteria are met. The internal alert is turned into a counter (see the Prometheus Metric types documentation) that is incremented each time the conditions to trigger the alert are met. The counter is made available on the `/metrics` endpoint. Use a Prometheus rule to capture the metric data and trigger the alert on Prometheus. The following example shows an alert used in an Azure deployment; note how the process watches the increase in instances of the alert being fired over time to trigger the Prometheus alert.

```
- alert: GES_RBAC_CREATE_VQ_PROXY_ERROR
annotations:
  summary: "There are issues managing VQ proxy objects on {{ $labels.pod }}"
labels:
  severity: info
```

```
  action: email
  service: GES
expr: increase(RBAC_CREATE_VQ_PROXY_ERROR[10m]) > 5
```

Health alerts in GES work a little differently. They are gauges, rather than counters. The gauge toggles between "0" and "1"; "1" indicates that the service is down and "0" indicates that the service is up. Because GES has an automatic health check that runs approximately every 15-20 seconds, the health alerts are generated when a connection has been in the DOWN state for a given period of time. The following example shows the ORS_REDIS_DOWN alert.

```
- alert: GES_ORS_REDIS_DOWN
        expr: ORS_REDIS_STATUS > 0
        for: 5m
        labels:
          severity: critical
          action: page
          service: GES
        annotations:
            summary: "ORS REDIS Connection down for {{ $labels.pod }}"
            dashboard: "See GES Performance > Health and Liveliness to track ORS Redis Health
over time"
```

Grafana dashboards

You can deploy the Grafana dashboards, included with the helm chart, when you deploy GES. Simply set the Helm value `.Values.ges.grafana.enabled` to **true**. This creates a config map to automatically deploy the dashboard.

In some cases, the dashboards might need adjustment to work appropriately with your Grafana version and overall Kubernetes setup. To make changes, unpack the helm chart .tar.gz file. Make the necessary upgrades to the `grafana/ges-dashboard-configmap.yaml` and `grafana/ges-performance-dashboard.yaml` files. Experienced users can make changes in the JSON files. Alternatively, you can use the web interface to set up the dashboard, export the JSON for the dashboard (following the Grafana dashboard export and import instructions), and then copy the JSON into the appropriate file. On a re-deploy of the Helm Charts, Grafana picks up the new dashboards.

Sample implementations

You can find sample implementations of alerts in the provided helm charts, in the `prometheus/alerts.yaml` file.

Sample dashboards, embedded in config maps, can be found in the `grafana/ges-dashboard.yaml` and `grafana/ges-performance-dashboard.yaml` files. These are for the business logic and performance dashboards respectively. You might need to make some adjustments to get the alerts and dashboards working; see Grafana dashboards.

Logging

Learn how to store logs for Genesys Callback.

Related documentation:

-
-

Early Adopter Program

Genesys Multicloud CX private edition is being released to pre-approved customers as part of the Early Adopter Program. Please note that the documentation and the product are subject to change. For more details about the program, please contact your Genesys representative.

For solution-level documentation about logging, see **Link to come**.

GES outputs logs to stdout.

GES log size is highly dependent on usage. A high-traffic enterprise that accepts many callbacks daily will have much larger log sizes than an enterprise with only a handful of callbacks a day. For your reference, a single Create Callback operation, invoked from the Callback UI, generates 16 log messages in 1 second at full trace level. Therefore, at a rate of 10 callbacks/second, there are 160 log messages per second.

You can make changes to logging settings using the Helm values file. For information about the `log` values included in the Helm charts, see the configMap section.