Genesys Callback Private Edition Guide

Observability in Genesys Callback

2/3/2026

Contents

Learn about the logs, metrics, and alerts you should monitor for Genesys Callback.

Related documentation:

-
-
-
-

RSS:

- For private edition

Monitoring

Private edition services expose metrics that can be scraped by Prometheus, to support monitoring operations and alerting.

- As described on Monitoring overview and approach, you can use a tool like Grafana to create dashboards that query the Prometheus metrics to visualize operational status.
- As described on Customizing Alertmanager configuration, you can configure Alertmanager to send notifications to notification providers such as PagerDuty, to notify you when an alert is triggered because a metric has exceeded a defined threshold.

The services expose a number of Genesys-defined and third-party metrics. The metrics that are defined in third-party software used by private edition services are available for you to use as long as the third-party provider still supports them. For descriptions of available Genesys Callback metrics, see:

-

See also System metrics.

Enable monitoring

Service	CRD or annotations?	Port	Endpoint/ Selector	Metrics update interval
	Supports both CRD (Service Monitor) and annotations	3050	/metrics	Real-time updates

Configure metrics

The metrics that are exposed by Genesys Engagement Service (GES) are available by default. No further configuration is required in order to define or expose these metrics. You cannot define your own custom metrics.

The Metrics page linked to above shows some of the metrics GES exposes. You can also query Prometheus directly or via a dashboard to see all the metrics available from GES.

What do Genesys Callback metrics monitor?

The GES/Callback metrics are divided into three categories:

- Metrics that have to do with the internal business logic of GES: This includes metrics that measure things like the number of callbacks booked, the number of click-to-call requests booked, callback monitor performance, and API usage.
- Metrics that measure the performance of GES: Disk, memory, and CPU usage, event loop lag, request handling times, and the health of connections to downstream services such as Redis, PostGres, GWS, ORS, and others.
- Alarms-type metrics, or alerts: These metrics are boolean flags that raise and lower whenever certain conditions are met.

Watching how the metrics change over time helps you understand the performance of a given GES deployment or pod. The sample Prometheus expressions show you how to use the basic metrics to gain valuable insights into your callback-related activity.

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 – 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 Alerting.

Alerting

Private edition services define a number of alerts based on Prometheus metrics thresholds.

Important
You can use general third-party functionality to create rules to trigger alerts based on metrics values you specify. Genesys does not provide support for custom alerts that you create in your environment.

For descriptions of available Genesys Callback alerts, see:

-

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).

One of the key things to understand about alerts in GES is that, while GES leverages Prometheus, the application manually triggers alerts when certain criteria are met. This internal alert is then turned into a counter that is incremented each time the conditions to trigger the alert are met. The counter is available on the `/metrics` endpoint. Prometheus rules capture the metric data and trigger the alert on Prometheus; also see Configure alerts. For more information about counters, see the Prometheus Metric types documentation.

Because alerts are implemented as counters in GES, you can leverage metrics to analyze how the counters increase over time for a given deployment or pod. For a list of helpful Prometheus expressions to use for this purpose, see Sample Prometheus expressions.

The following example shows an alert used in an Azure deployment; an increase in instances of the alert firing over a certain period of time triggers 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"
```

Configure alerts

Private edition services define a number of alerts by default (for Genesys Callback, see the pages linked to above). No further configuration is required.

The alerts are defined as **PrometheusRule** objects in a **prometheus-rule.yaml** file in the Helm charts. As described above, Genesys Callback does not support customizing the alerts or defining additional **PrometheusRule** objects to create alerts based on the service-provided metrics.

Logging

For solution-level documentation about logging, see Logging overview and approaches.

GES outputs logs to standard output (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 each 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.