Genesys Customer Experience Insights Private Edition Guide

2/8/2026

Table of Contents

Overview

- About GCXI / RAA 6
- Architecture 8
- High availability and disaster recovery 13

Configure and deploy RAA

- Before you begin deploying RAA 17
- Configure RAA 23
- Provision RAA 32
- Deploy Reporting and Analytics Aggregates 33
- Upgrade, rollback, or uninstall RAA 39

Configure and deploy GCXI

- Before you begin deploying GCXI 41
- Configure GCXI 50
- Provision GCXI 54
- Deploy Genesys Customer Experience Insights 55
- Upgrade, roll back, or uninstall GCXI 63

Observability

- Observability in Genesys Customer Experience Insights 70
- GCXI metrics and alerts 76
- RAA metrics and alerts 81
- Logging 87

Contents

- 1 Overview
- 2 Configure and deploy RAA
- 3 Configure and deploy GCXI
- 4 Observability

Find links to all the topics in this guide.

Related documentation:

-
-
-

RSS:

- For private edition

Genesys Customer Experience Insights (GCXI) is a service available with the Genesys Multicloud CX private edition offering. Reporting and Analytics Aggregates (RAA) is a supporting service for GCXI; this manual describes both RAA and GCXI. GCXI can provide meaningful reports only if both Genesys Info Mart and RAA are deployed and available.

For information about GCXI reports and dashboards, see Historical Reporting with Genesys CX Insights. The documentation on this page describes Genesys CX Insights for *Genesys Multicloud CX and Genesys Multicloud CX private edition*.

> ### Important
> This page applies to **Genesys CX Insights Genesys Multicloud CX private edition** deployments only. In such deployments, always use software that you download from the Genesys JFrog repository, and refer to the Multicloud CX Release Notes. If you are deploying Genesys CX Insights in an **on-premises deployment**, always use software that you download from the Salesforce site, and refer to the on-premises GCXI Release Notes. For information about how to download the installation packages for on-premises deployments, talk to your Genesys representative, and follow the instructions in the Genesys CX Insights on-premises documentation.

Overview

Learn more about GCXI, its architecture, and how to support high availability and disaster recovery.

- About GCXI / RAA
- Architecture
- High availability and disaster recovery

Configure and deploy RAA

Find out how to configure and deploy RAA

- Before you begin deploying RAA
- Configure RAA
- Provision RAA
- Deploy Reporting and Analytics Aggregates
- Upgrade, rollback, or uninstall RAA

Configure and deploy GCXI

Find out how to configure and deploy GCXI

- Before you begin deploying GCXI
- Configure GCXI
- Provision GCXI
- Deploy Genesys Customer Experience Insights
- Upgrade, roll back, or uninstall GCXI

Observability

Learn how to monitor GCXI with metrics and logging.

- Observability in Genesys Customer Experience Insights
- GCXI metrics and alerts
- RAA metrics and alerts
- Logging

About GCXI / RAA

Contents

- 1 Supported Kubernetes platforms

Learn about Genesys Customer Experience Insights (GCXI) and how it works in Genesys Multicloud CX private edition.

Related documentation:

-
-
-
-

RSS:

- For private edition

GCXI provides reports and dashboards that summarize contact center activity. Reports display contact center activity using easy-to-read grids, while dashboards summarize a wider range of information using various visual devices draws. GCXI reports and dashboards enable business and contact center managers to make better business decisions for streamlining operations, reducing costs, and providing better services.

GCXI presents aggregated historical information from Genesys Info Mart or Intelligent Workload Distribution (IWD) Data Mart, and relies on Reporting and Analytics Aggregates (RAA) to present Genesys Info Mart data. RAA provides the mechanism for creating, maintaining, and populating a subset of tables and views in a Genesys Info Mart database that provide aggregated data of contact center operations for reporting and analytical purposes. This aggregation layer is a necessary and transparent component of GCXI. This document describes both RAA and GCXI.

GCXI is powered by MicroStrategy software and continuously aggregates data.

Supported Kubernetes platforms

Genesys Engagement Service (GES) is supported on the following Kubernetes platforms:

- Google Kubernetes Engine (GKE)
- OpenShift Container Platform (OpenShift)
- Azure Kubernetes Service (AKS)

See the Genesys Customer Experience Insights Release Notes and Reporting and Analytics Aggregates Release Notes for information about when support was introduced.

Architecture

Contents

- 1 Introduction
  - 1.1 RAA
  - 1.2 GCXI
- 2 Architecture diagram — Connections
- 3 Connections table

Learn about Genesys Customer Experience Insights architecture

Related documentation:

-
-
-
-

RSS:

- For private edition

Introduction

RAA

Genesys Customer Experience Insights (GCXI) relies on the Reporting and Analytics Aggregates (RAA) aggregation layer, which aggregates Genesys Info Mart data, and then stores the aggregated data in Genesys Info Mart. The RAA container contains only RAA and Netcat. Netcat is is used by RAA to expose Prometheus metrics, which optionally provide internal monitoring of RAA performance, containers, and so on. Prometheus metrics are for administrator and troubleshooting use only.

GCXI

GCXI includes the following containers:

- **gcxi** - The main GCXI container, which runs as a StatefulSet.
- **gcxi-control** - A supplementary container, which is used for GCXI first-install initialization and for cleanup.
- **alpine / ubi8-minimal** - An init container, used to set up log volume permissions. You can instead use any suitable container able to run simple bash commands.

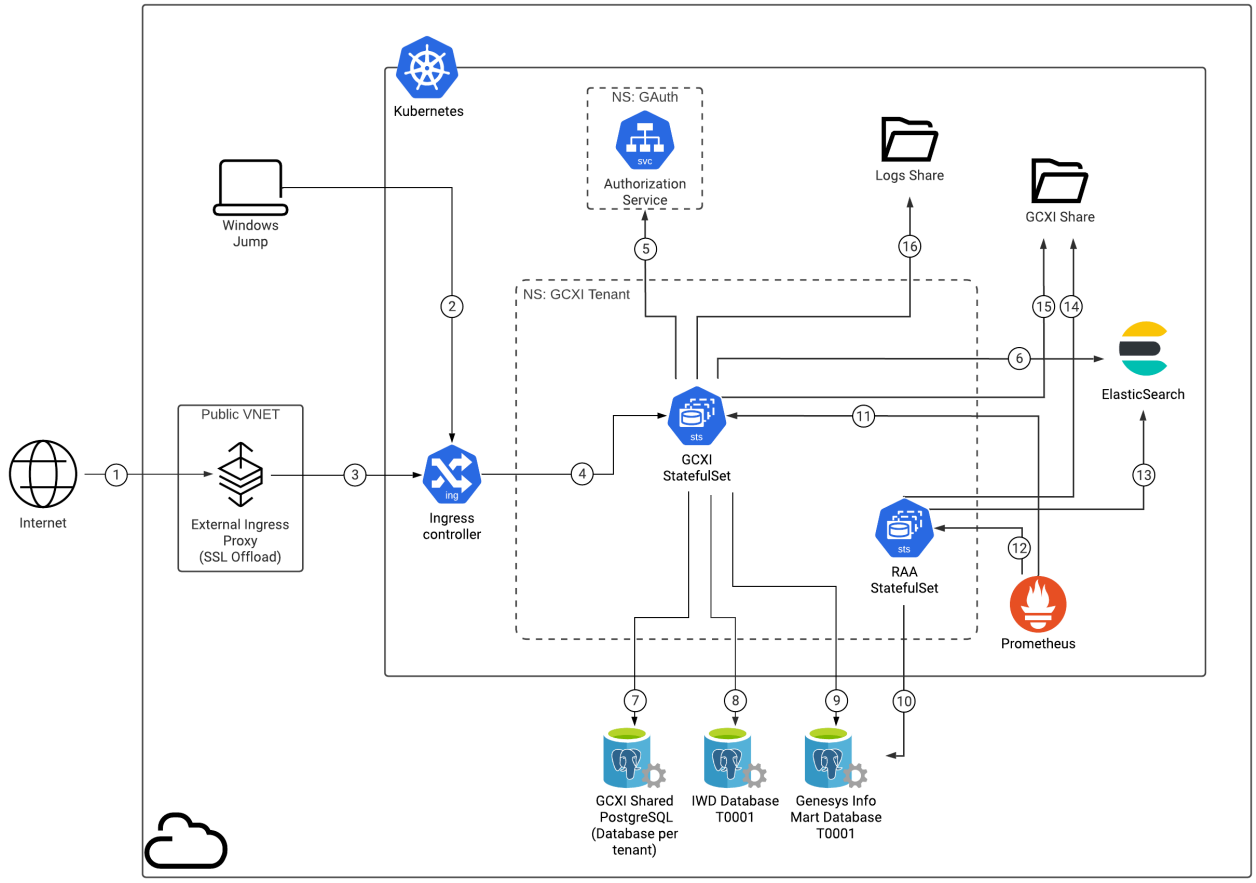
GCXI does not support Disaster Recovery, cross-regional deployment, or auto-scaling.

For information about the overall architecture of Genesys Multicloud CX private edition, see the high-level Architecture page.

See also High availability and disaster recovery for information about high availability/disaster recovery architecture.

Architecture diagram — Connections

The numbers on the connection lines refer to the connection numbers in the table that follows the diagram. The direction of the arrows indicates where the connection is initiated (the source) and where an initiated connection connects to (the destination), from the point of view of Genesys Customer Experience Insights as a service in the network.



Connections table

The connection numbers refer to the numbers on the connection lines in the diagram. The **Source**, **Destination**, and **Connection Classification** columns in the table relate to the direction of the arrows in the Connections diagram above: The source is where the connection is initiated, and the destination is where an initiated connection connects to, from the point of view of Genesys Customer Experience Insights as a service in the network. *Egress* means the Genesys Customer Experience Insights service is the source, and *Ingress* means the Genesys Customer Experience Insights service is the destination. *Intra-cluster* means the connection is between services in the cluster.

Connection	Source	Destination	Protocol	Port	Classification	Data that travels on this connection
1	Public Internet	Inbound Gateway	HTTPS	443	Ingress	Inbound web traffic.
2	Windows Jump	Ingress controller*	TCP	34952	Ingress	Connection from MicroStrategy Developer (an optional tool, installed on Windows). *This connection does not pass through the ingress controller in some configurations.
3	Ingress Proxy	Ingress controller	HTTPS	443	Intra-cluster	Inbound web traffic.
4	Ingress controller	GCXI StatefulSet	HTTP	8080	Intra-cluster	Ingress controller connects to GCXI StatefulSet.
5	GCXI StatefulSet	GAuth: Authorization Service	HTTP	80	Intra-cluster	GCXI queries the Genesys Authentication Service to validate user identity and obtain privilege information for the authenticated user.
6	GCXI StatefulSet	Logging destination, such as ElasticSearch	TCP	Logical connection only	Intra-cluster	Writes container logs.
7	GCXI StatefulSet	GCXI Shared PostgreSQL (Database per tenant)	TCP	5432	Egress	Exchanges internal information with the GCXI MicroStrategy meta database.

Connection	Source	Destination	Protocol	Port	Classification	Data that travels on this connection
8	GCXI StatefulSet	IWD Database	TCP	5432	Egress	Retrieves report data from Genesys intelligent Workload Distribution (iWD) database.
9	GCXI StatefulSet	Genesys Info Mart Database	TCP	5432	Egress	Retrieves report data from Genesys Info Mart database.
10	RAA StatefulSet	Genesys Info Mart Database	TCP	5432	Egress	Performs data processing in Genesys Info Mart database.
11	Prometheus	GCXI StatefulSet	HTTP	9101	Intra-cluster	Collects GCXI metrics for monitoring and alerting with Prometheus.
12	Prometheus	RAA StatefulSet	HTTP	9101	Intra-cluster	Collects RAA metrics for monitoring and alerting with Prometheus.
13	RAA StatefulSet	Logging destination, such as ElasticSearch	TCP	Logical connection only	Intra-cluster	Writes container logs.
14	RAA StatefulSet	GCXI Share	TCP	Logical connection only	Intra-cluster	Stores service files.
15	GCXI StatefulSet	GCXI Share	TCP	Logical connection only	Intra-cluster	Stores service files.
16	GCXI StatefulSet	Logs Share	TCP	Logical connection only	Intra-cluster	Writes log files to file share.

High availability and disaster recovery

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

Related documentation:

-
-
-
-

RSS:

- For private edition

Name	High Availability	Disaster Recovery	Where can you host this service?
Genesys CX Insights	N = 2 (active-active)	Not supported	Primary unit only
Reporting and Analytics Aggregates	N = 1 (singleton)	Limited active spare	Primary or secondary unit

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

By default, Genesys CX Insights (GCXI) provides High Availability support using StatefulSets with 2 x Pods in Active-Active mode, which form a MicroStrategy cluster. GCXI does not support autoscaling. GCXI does not support Disaster Recovery or any kind of cross-regional deployment. In most scenarios, GCXI is deployed in customer's primary region only. If GCXI is deployed in supplementary regions, each such deployment is completely independent from each other. Pods in different regions do not communicate with each other.

RAA is deployed in each region where Genesys Info Mart is deployed. Genesys Info Mart creates a complete production replica of the Genesys Info Mart database in place, and serves traffic during normal operations, but the data in the secondary region is used only in case of disaster. For more information about Genesys Info Mart, see High availability and disaster recovery.

Multi-region availability

Functionality	Pattern	Design	Dependencies	Limitation	Activate	Activate Time	Rollback	Rollback Time	Changes
GCXI	Pilot-Light	Deploy GCXI in the primary region, and replicate the database and File storage to the secondary region.	Ensure that the Genesys Info Mart database is available in the secondary region.	None	1. Tear down GCXI in the primary region. 2. Change secondary region replicated database and File storage to R/W, and replicate to the primary region (if available). 3. Deploy GCXI in the secondary region using the replicated database and File storage. 4. Notify tenant to use the secondary GCXI tile in Portal.	Under 15 mins	1. Tear down secondary region GCXI with a pipeline. 2. Deploy GCXI in the primary region. If the database and File storage is replicated to primary region, change to R/W and replicate to secondary region. If the database and File storage is not replicated to primary region, recreate them using backup	under 15 mins	• Change the GCXI pipeline to convert replicated database and File Storage to R/W, and deploy GCXI. • Ensure that the PostgreSQL database is replicated across regions. • Change the File storage to GZRS Azure files to replicate the data: Reports - email or link. Cube - in memory - refreshes

Functionality	Pattern	Design	Dependencies	Limitation	Activate	Activate Time	Rollback	Rollback Time	Changes
							data from the secondary region. 3. Update the URL for GCXI.		all links periodically. • Add secondary GCXI tile.
RAA	Limited active spare	RAA stores its data in Genesys Info Mart database. As is the case with Genesys Info Mart, RAA DR replica is in place and serves traffic during normal operations, but the data is used only in case of disaster.	You must ensure that the GIM database available in the secondary region, and that the GCXI shared part (infrastructure) is deployed in the secondary region.	None	During initial tenant deployment, ensure that RAA is deployed in the primary or DR region.	Not applicable.	Not applicable.	Not applicable.	Not applicable.

Primary Region failure

Functionality	Operational Impact	Representative Experience	Customer Experience
GCXI	without GCXI	No Impact	No Impact

	functionality 15 mins		

Secondary Region Failure

Functionality	Operational Impact	Representative Experience	Customer Experience
GCXI	No Impact	No Impact	No Impact

Data Sovereignty and Regulations

Functionality	Design	Changes
GCXI	There is potential sensitive data in reports.	

Before you begin deploying RAA

Contents

- 1 Limitations and assumptions
- 2 Download the Helm charts
- 3 Third-party prerequisites
- 4 Storage requirements
  - 4.1 GIM secret volume
  - 4.2 Config volume
  - 4.3 Health volume
- 5 Network requirements
- 6 Secrets
- 7 Genesys dependencies
- 8 GDPR support

Find out what to do before deploying Reporting and Analytics Aggregates (RAA).

Related documentation:

-
-
-
-

RSS:

- For private edition

Limitations and assumptions

The RAA container works with the Genesys Info Mart database; deploy RAA only after you have deployed Genesys Info Mart.

The Genesys Info Mart database schema must correspond to a compatible Genesys Info Mart version. Execute the following command to discover the required Genesys Info Mart release:

```
docker run -it --entrypoint /bin/java gcxi/raa: -jar GIMAgg.jar -version
```

RAA container runs RAA on Java 11, and is supplied with the following of JDBC drivers:

- MSSQL 9.2.1 JDBC Driver
- Postgres 42.2.11 JDBC Driver
- Oracle Database 21c (21.1) JDBC Driver

Genesys recommends that you verify whether the provided driver is compatible with your database, and if it is not, you can override the JDBC driver by copying an updated driver file to the folder **lib\jdbc_driver_** within the mounted config volume, or by creating a co-named link within the folder **lib\jdbc_driver_**, which points to a driver file stored on another volume (where is the RDBMS used in your environment). This is possible because RAA is launched in a config folder, which is mounted in a container.

Download the Helm charts

To learn what Helm chart version you must download for your release, see Helm charts and containers for Genesys Customer Experience Insights.

You can download the gcxi helm charts from the following repository:

```
https://pureengage.jfrog.io/ui/packages/helm:%2F%2Fgcxi-raa
```

For more information about downloading containers, see: Downloading your Genesys Multicloud CX containers.

Third-party prerequisites

For information about setting up your Genesys Multicloud CX private edition platform, including Kubernetes, Helm, and other prerequisites, see Software requirements.

Storage requirements

This section describes the storage requirements for various volumes.

GIM secret volume

In scenarios where **raa.env.GCXI_GIM_DB__JSON** is not specified, RAA mounts this volume to provide GIM connections details.

1. Declare GIM database connection details as a Kubernetes secret in **gimsecret.yaml**:

```
apiVersion: v1
kind: Secret
metadata:
  namespace: gcxi
  name: gim-secret
type: kubernetes.io/service-account-token
data:
  json_credentials:
eyJqZGJjX3VybCI6ImpkYmM6cG9zdGdyZXNxbDovLzxob3N0Pjo1NDMyLzxnaW1fZGF0YWJhc2U+IiwgImRiX3VzZXJuYW1lIjoiPH
```

2. Reference **gimsecret.yaml** in **values.yaml**:

```
raa
  ...
  volumes:
    ...
    gimSecret:
      name: "gim-secret-volume"
      secretName: "gim-secret"
      jsonFile: "json_credentials"
   ...
```

Alternatively, you can mount the CSI secret using **secretProviderClass**, in **values.yaml**:

```
raa
  ...
  volumes:
    ...
    gimSecret:
      name: "gim-secret-volume"
```

```
      secretProviderClass: "gim-secret-class"
      jsonFile: "json_credentials"
   ...
```

Config volume

RAA mounts a config volume inside the container, as the folder **/genesys/raa_config**. The folder is treated as a work directory, RAA reads the following files from it during startup:

- **conf.xml**, which contains application-level config settings.
- custom ***.ss** files.
- JDBC driver, from the folder **lib/jdbc_driver_**.

RAA does not normally create any files in **/genesys/raa_config** at runtime, so the volume does not require a fast storage class. By default, the size limit is set to 50 MB. You can specify the storage class and size limit in **values.yaml**:

```
raa
  ...
  volumes:
    ...
    config:
      capacity: 50Mi
      storageClassName: ""

   ...
```

RAA helm chart creates a Persistent Volume Claim (PVC). You can define a Persistent Volume (PV) separately using the **gcxi-raa** chart, and bind such a volume to the PVC by specifying the volume name in the **raa.volumes.config.pvc.volumeName** value, in **values.yaml**:

```
raa
  ...
  volumes:
    ...
    config:
      pvc:
        volumeName: "my_raa_config_volume"
    ...
```

Health volume

RAA uses the Health volume to store:

- Health files.
- Prometheus file containing metrics for the most recent 2-3 scrape intervals.
- Results of the most recent **testRun init** container execution.

By default, the volume is limited to 50MB. RAA periodically interacts with the volume at runtime, so Genesys does not recommend a slow storage class for this

volume. You can specify the storage class and size limit in **values.yaml**:

```
raa
  ...
  volumes:
    ...
    health:
      capacity: 50Mi
      storageClassName: ""
   ...
```

RAA helm chart creates a PVC. You can define a PV separately using the **gcxi-raa** chart, and bind such a volume to the PVC by specifying the volume name in the **raa.volumes.health.pvc.volumeName** value, in **values.yaml**:

```
raa
  ...
  volumes:
    ...
    health:
      pvc:
        volumeName: "my_raa_helath_volume"
    ...
```

Network requirements

RAA interacts only with the Genesys Info Mart database.

RAA can expose Prometheus metrics by way of Netcat.

The aggregation pod has it's own IP address, and can run with one or two running containers. For Helm test, an additional IP address is required -- each test pod runs one container.

Genesys recommends that RAA be located in the same region as the Genesys Info Mart database.

Secrets

RAA secret information is defined in the values.yaml file (line 89).

For information about configuring arbitrary UID, see Configure security.

Genesys dependencies

RAA interacts with Genesys Info Mart database only.

GDPR support

Not applicable.

Configure RAA

Contents

- 1 Override Helm chart values
- 2 Container deployment
  - 2.1 Init containers
  - 2.2 Pod containers
  - 2.3 Test containers
- 3 Disk space requirements
  - 3.1 GIM secret volume
  - 3.2 Config volume
  - 3.3 Health volume
- 4 Configure security
  - 4.1 Arbitrary UIDs

Learn how to configure Reporting and Analytics Aggregates (RAA).

Related documentation:

-
-
-
-

RSS:

- For private edition

Override Helm chart values

Before you begin, download samples of the files used in this section: RAA configuration files. Within those files:

- Container environment variables are declared beginning at line 114 of the **Dockerfile**. See the comments accompanying each variable.
- Helm values are described in **values.yaml**. See the comments accompanying each Helm value.

You can override values in the Helm charts to configure Private Edition. For more information, see the suite-level documentation about how to override Helm chart values: Overriding Helm chart values.

1. Override Helm Chart Image repository values.

```
raa:
   ...
   image:
     registry: 'pureengage-docker-staging.jfrog.io'
     name: '{{- if $.Values.raa.image.registry -}}{{ $.Values.raa.image.registry
}}/{{- end -}}gcxi/raa:{{ include "raa.imageVersion" $ }}'
     pullSecrets:
        -
```

2. Override Helm Chart GIM DB values.
   The GIM database address and access parameters, base64 encoded, formatted as follows:

```
# echo '{"jdbc_url":"jdbc:postgresql://:5432/", "db_username":"", "db_password":""}' |
  base64

raa:
   ...
   env:
     GCXI_GIM_DB__JSON:
eyJkYl9ob3N0bmFtZSI6InBvc3RncmVzLXJ3LmluZnJhLnN2Yy5jbHVzdGVyLmxvY2FsIiwiZGJfbmFtZSI6ImV0bC1hcm8iLCJkYl9
ybmFtZSI6InBvc3RncmVzIn0K
```

3. Use one of the following methods to deliver configuration changes into the container:
   - Override Helm Chart config values using a TAR file:
     1. Prepare **confg.xml** and custom ***.ss** files.
     2. Compress the **confg.xml** and ***.ss** files into a tar(gzip) archive.
     3. Convert the archive to base64.
     4. Use the name of the compressed file in the override:

```
raa:
  ...
  deployment:
    configTar: H4sIADb/
0WAAA+2VXW+bMBSGua7U/+BxlVw4AcJA6pJJHmVZNSXN2uRiV8gJJrHKR2Qgavvr59KkbRKpII3QTTrPDcjHPq9f+xxYJHHQ
Q6j/CWNEH3PBkM8CmocZwvg5QtbrkC/
o09xiQA5xxP2BSpdLdTtSjN0N1DSf41WSC8zjjIkNDVW0Gag9LeKx2t2t7vKjPDhgNJPih/
kiKtNwGvJHhmXubWrsz4u8mcjZS9rdEhbTecgw3VAeFm/
b1OnxkqOdLILlG7eHm3G+D8lk4o3JyC1y3RCiore5UHFe3f0D63f3jvqjL35LnjKBfZpRHNF1J01PoVHS/
z3DNnb9b1i2JfvfsDQL+r8Jvuw3e3p+1pKFgF/
L4o49oNaKM0HFYvVwgX54V5dt1PJ5xOJU1vIFmt26N94lmRLvp/
tblzF2v5Zt9hzkIsj9jjO7nV6PnuKe3m7/
rYZRpmFU0iBDdzw9vZUymTrdeL9m7sxtyFOpWK3OHDKakKvh+F1z9XmrIGeUyFVzV8lXLZdWyVItN9ZQFTZUf4WMR741cEVV
TgAAAAAAAAAAAAAAAAAAAAAAAAD49/gDzO8AHwAoAAA=
```

     5. Override Helm Chart init container values.
        To enable the container, specify the name of the **configDelivery** container:

```
raa:
  ...
  statefulset:
    containers:
      configDelivery:
        name: '{{ $.Chart.Name }}-conf-delivery'
```

   - Override Helm Chart volume configuration values using mapping:
     1. The config and health volumes requires PVC, which are automatically created using the storage class name.

```
raa
  ...
  volumes:
    config:
      pv: {}
      storageClassName: azure-files
      pvc:
        name: "gcxi-raa-config-pvc"
        volumeName: ""
```

4. Override Helm Chart monitoring container values.
   This container exposes two ports for scraping of aggregation metrics and health metric by Prometheus or other monitoring tool. The monitor container is optional, and is disabled by default.

```
raa:
  ...
  statefulset:
  ...
    containers:
    ...
      monitor:
        name: "{{ $.Chart.Name }}-monitor"
```

```
    toolcmd:
      # interval of checking for a new file with command
      intervalSec: "20"

    metrics:
      portName: "metrics"
      containerPort: "9100"

    health:
      portName: "health"
      containerPort: "9101"
  ...
```

Container deployment

Deploy containers in the order they are described on this page.

Init containers

The RAA helm chart includes two explicit Init containers.

- **configDelivery init container**
  The optional configDelivery init container delivers required ***.xml** configuration and ***.ss** files to the RAA work dir. Those files must be contained in a GZIP archive encoded as base64, which is passed by one of two methods:
  - The option **--set-file raa.deployment.configTar=config.tar.gz.b64** of helm install/update.
  - The **configTar** value, specified in the YAML file.

  Default **conf.xml** and **user-data-map.ss** are supplied with the YAML chart. If these files are missing from the work directory, and are not provided by either of the methods described above, the Init container automatically copies the default files to the work directory.

  The configDelivery init container is disabled by default since you can manually create RAA configuration files in the mounted config volume. Enabling the configDelivery init container is useful in scenarios such as cloud deployments where access to the mounted work directory is restricted due to security policies, or due to use of ephemeral storage for config volume. Specify the container when you need to deploy a default configuration.

  To enable the container, specify the container name using the *configDelivery:* parameter in **values.yaml**:

  ```
  raa:
    ...
    statefulset:
    ...
      containers:
      ...
        configDelivery:
          name: "{{ $.Chart.Name }}-conf-delivery"
  ```

- **testRun init container**
  The optional testRun init container tests your configuration by running aggregation over an empty time range. SQL execution, even on empty data, checks the presence of the expected tables and

columns, which helps you to detect configuration and customization problems. The testRun init container is disabled by default, but Genesys recommends that you activate it to test your configurations.

To enable the container, specify the container name using the **testRun:** parameter in **values.yaml**:

```
raa:
  ...
  statefulset:
  ...
    containers:
    ...
      testRun:
        name: "{{ $.Chart.Name }}-test-run"
```

Logs and test results appear in the health volume **test** folder.

Pod containers

The RAA helm chart includes two explicit execution containers.

- **aggregation container**
  RAA requires the **aggregation** container to perform aggregation. The **aggregation** container is enabled by default.

  To enable the container, specify the container name using the **aggregation:** parameter in values.yaml:

  ```
  raa:
    ...
    statefulset:
    ...
      containers:
        aggregation:
          name: "{{ $.Chart.Name }}"
        ...
  ```

- **monitor container**
  The **monitor** container is an optional sidecar container that allows you to run the RAA tool command from a file, using parameters placed in a work directory. This functionality is useful in scenarios where you cannot use **kubectl**, for example due to security policies. The container also exposes two ports for scraping aggregation metrics and health metric by a monitoring tool, such as Prometheus. The **monitor** container is disabled by default.

  To enable the container, specify the container name using the **monitor:** parameter in **values.yaml**:

  ```
  raa:
    ...
    statefulset:
    ...
      containers:
      ...
        monitor:
          name: "{{ $.Chart.Name }}-monitor"

          toolcmd:
            # interval of checking for a new file with command
            intervalSec: "20"
  ```

```
    metrics:
      portName: "metrics"
      containerPort: "9100"

    health:
      portName: "health"
      containerPort: "9101"
  ...
```

Test containers

The RAA helm chart includes two containers for the helm test command. Run these containers in order (**testRunCheck** followed by **healthCheck**).

- **testRunCheck container**
  The optional **testRunCheck** container watches for the execution of the **testRun init** container, and reads the testRun results from the health volume. The testRunCheck container is enabled by default. If the **testRun init** container is not enabled, Genesys recommends that you disable **testRunCheck**.

  The disable this container, clear the value from the **testRunCheck** parameter in the **testPod** section of your **values.yaml** file:

  ```
  raa:
    ...
    testPods:

      testRunCheck:

        name: "{{ tpl .Values.raa.serviceName . }}-test-run-check"

        container:
          name: "{{ $.Chart.Name }}-test-run-check"

        labels: {}

        annotations:
          "helm.sh/hook-weight": "100"
          "helm.sh/hook": "test-success"
          "helm.sh/hook-delete-policy": "before-hook-creation"
        ...
  ```

- **healthCheck test container**
  The optional **healthCheck test** container runs **healthCheck**, displays healthCheck status information to the console, and prints the content of the current configuration files and health files to standard output. The container is enabled by default.

  The disable this container, clear the value from the **healthCheck** parameter, in the **testPod** section of your **values.yaml** file:

  ```
  raa:
    ...
    testPods:
      ...
      healthCheck:
        name: "{{ tpl .Values.raa.serviceName . }}-health-check"

        container:
          name: "{{ $.Chart.Name }}-health-check"
  ```

```
    labels: {}

    annotations:
      "helm.sh/hook-weight": "200"
      "helm.sh/hook": "test-success"
      "helm.sh/hook-delete-policy": "before-hook-creation"
    ...
```

Disk space requirements

This section describes the storage requirements for various volumes.

GIM secret volume

In scenarios where **raa.env.GCXI_GIM_DB__JSON** is not specified, RAA mounts this volume to provide GIM connections details.

1. Declare GIM database connection details as a Kubernetes secret in **gimsecret.yaml**:

```
apiVersion: v1
kind: Secret
metadata:
  namespace: gcxi
  name: gim-secret
type: kubernetes.io/service-account-token
data:
  json_credentials:
eyJqZGJjX3VybCI6ImpkYmM6cG9zdGdyZXNxbDovLzxob3N0Pjo1NDMyLzxnaW1fZGF0YWJhc2U+IiwgImRiX3VzZXJuYW1lIjoiPH
```

2. Reference **gimsecret.yaml** in **values.yaml**:

```
raa
   ...
   volumes:
     ...
     gimSecret:
       name: "gim-secret-volume"
       secretName: "gim-secret"
       jsonFile: "json_credentials"
    ...
```

Alternatively, you can mount the CSI secret using **secretProviderClass**, in **values.yaml**:

```
raa
  ...
  volumes:
    ...
    gimSecret:
      name: "gim-secret-volume"
      secretProviderClass: "gim-secret-class"
      jsonFile: "json_credentials"
   ...
```

Config volume

RAA mounts a config volume inside the container, as the folder **/genesys/raa_config**. The folder is treated as a work directory, RAA reads the following files from it during startup:

- **conf.xml**, which contains application-level config settings.
- custom ***.ss** files.
- JDBC driver, from the folder **lib/jdbc_driver_**.

RAA does not normally create any files in **/genesys/raa_config** at runtime, so the volume does not require a fast storage class. By default, the size limit is set to 50 MB. You can specify the storage class and size limit in **values.yaml**:

```
raa
  ...
  volumes:
    ...
    config:
      capacity: 50Mi
      storageClassName: ""

   ...
```

RAA helm chart creates a Persistent Volume Claim (PVC). You can define a Persistent Volume (PV) separately using the **gcxi-raa** chart, and bind such a volume to the PVC by specifying the volume name in the **raa.volumes.config.pvc.volumeName** value, in **values.yaml**:

```
raa
  ...
  volumes:
    ...
    config:
      pvc:
        volumeName: "my_raa_config_volume"
    ...
```

Health volume

RAA uses the Health volume to store:

- Health files.
- Prometheus file containing metrics for the most recent 2-3 scrape intervals.
- Results of the most recent **testRun init** container execution.

By default, the volume is limited to 50MB. RAA periodically interacts with the volume at runtime, so Genesys does not recommend a slow storage class for this volume. You can specify the storage class and size limit in **values.yaml**:

```
raa
  ...
  volumes:
    ...
    health:
      capacity: 50Mi
```

```
      storageClassName: ""
   ...
```

RAA helm chart creates a PVC. You can define a PV separately using the **gcxi-raa** chart, and bind such a volume to the PVC by specifying the volume name in the **raa.volumes.health.pvc.volumeName** value, in **values.yaml**:

```
raa
  ...
  volumes:
    ...
    health:
      pvc:
        volumeName: "my_raa_helath_volume"
    ...
```

Configure security

Arbitrary UIDs

If your OpenShift deployment uses arbitrary UIDs, you must override the securityContext settings in the (**gcxi-raa/**) **values.yaml** file (line 89), so that you do not define any specific IDs:

```
## optional
## a security context assigned to each container in pod
securityContext:
  runAsNonRoot: true
  runAsUser: null
  runAsGroup: null
  fsGroup: null
```

The default values (user ID = 500) are suitable for many other deployment scenarios:

```
## optional
## a security context assigned to each container in pod
securityContext:
  runAsNonRoot: true
  runAsUser: 500
  runAsGroup: 500
  fsGroup: 0
```

Provision RAA

- Administrator

Learn how to provision Reporting and Analytics Aggregates (RAA).

Related documentation:

-
-
-
-

RSS:

- For private edition

An installed Genesys Info Mart database is required for RAA.

You configure RAA using **conf.xml** and **custom *.ss** files, which you can provide using any of the following methods:

- Place them on a mounted config volume.
- Using configDelivery container with configTar variable — used to supply custom configuration (set in configTar).
- Using configDelivery container without configTar variable — used to supply default configuration (included in the default RAA Helm chart).

Download examples of these files: RAA Provisioning

You can configure thresholds using Agent Setup.

Deploy Reporting and Analytics Aggregates

Contents

- 1 Assumptions
- 2 Prepare to deploy RAA
- 3 Install RAA
- 4 Validate the deployment

Learn how to deploy Reporting and Analytics Aggregates (RAA) into a private edition environment.

Related documentation:

-
-
-
-

RSS:

- For private edition

Assumptions

- The instructions on this page assume you are deploying the service in a service-specific namespace, named in accordance with the requirements on Creating namespaces. If you are using a single namespace for all private edition services, replace the namespace element in the commands on this page with the name of your single namespace or project.
- Similarly, the configuration and environment setup instructions assume you need to create namespace-specific (in other words, service-specific) secrets. If you are using a single namespace for all private edition services, you might not need to create separate secrets for each service, depending on your credentials management requirements. However, if you do create service-specific secrets in a single namespace, be sure to avoid naming conflicts.

Prepare to deploy RAA

Configure the following resources to support the RAA pod:

- Database connection details specified as `raa.env.GCXI_GIM_DB__JSON` value, or as a Kubernetes or vendor-specific secret. For more information, see Genesys Info MartGenesys info Mart secret volume.
- Config volume — Ensure that space is provisioned and available for RAA to mount the Config Volume to the POD containers (using specific helm values, or separately from the RAA helm chart, with the help of Kubernetes or vendor-specific tools).
- Health volume — Ensure that space is provisioned and available for RAA to mount the Health Volume mounting to the POD containers (using specific helm values, or separately from the RAA helm chart, with the help of Kubernetes or vendor-specific tools).

Configure at least the following values in **values-raa.yaml**:

```
raa:
  deployment:
    targetPlatform: "local"
```

```
  env:
    GCXI_GIM_DB__JSON:  |-

eyJqZGJjX3VybCI6ImpkYmM6cG9zdGdyZXNxbDovLzxob3N0Pjo1NDMyLzxnaW1fZGF0YWJhc2U+IiwgImRiX3VzZXJuYW1lIjoiPHVzZXI+Iiw
    XML_CONF: "default_conf.xml"
    STAT_SCRAPE_INTERVAL: 15
  image:
    registry: pureengage-docker-staging.jfrog.io
    pullSecrets:
      - pullsecret

  statefulset:
    containers:
      configDelivery:
        name: "{{ $.Chart.Name }}-conf-delivery"
      monitor:
        name: "{{$.Chart.Name }}-monitor"
        metrics:
          portName: "metrics"
          containerPort: "9100"
        health:
          portName: "health"
          containerPort: "9101"

  volumes:
    config:
      pv: {}
      storageClassName:
      pvc:
        name: "gcxi-raa-config-pvc"
        volumeName: ""
    health:
      pv: {}
      storageClassName:
      pvc:
        name: "gcxi-raa-health-pvc"
        volumeName: ""

  podMonitor:
    name : "{{ tpl $.Values.raa.serviceName $ }}-monitor"
    podMetricsEndpoints:
      - port: "{{ $.Values.raa.statefulset.containers.monitor.metrics.portName }}"
        path: "/{{ tpl $.Values.raa.statefulset.containers.monitor.metrics.portName $ }}"
        interval: "{{ $.Values.raa.env.STAT_SCRAPE_INTERVAL }}s"
      - port: "{{ tpl $.Values.raa.statefulset.containers.monitor.health.portName $
}}"
        path: "/{{ tpl $.Values.raa.statefulset.containers.monitor.health.portName $ }}"
        interval: 4m
        scrapeTimeout: 3m
    podTargetLabels:
      - service
      - servicename
```

Install RAA

Using the values you configured in **values-raa.yaml**, install RAA:

1. To deploy RAA, run the following bash script :

```
helm install -n gcxi gcxi-raa ./gcxi-raa-0.1.364.tgz -f values-raa.yaml
```

2. To check the installed Helm release, run the following bash script:

```
helm list -n gcxi
```

Release information appears, similar to the following:

```
NAME     NAMESPACE       REVISION        UPDATED
 STATUS          CHART           APP VERSION
 gcxi-raa      gcxi                                  1                       2021-06-22
 18:08:30.5041926 +0300 +0300                       deployed                gcxi-
 raa-0.1.364
```

3. To check the RAA project status, run the following bash script:

```
helm status gcxi-raa -n gcxi
```

Status information appears, similar to the following (with STATUS: deployed):

```
NAME: gcxi-raa
 LAST DEPLOYED: Tue Jun 22 18:08:30 2021
 NAMESPACE: gcxi
 STATUS: deployed
 REVISION: 1
```

Validate the deployment

The **testRun** init container activates validation testing. Helm test command runs all the test containers: one of the test containers tests execution results, while a second performs a healthCheck.

```
   containers:

      ## optional
      ## enables config delivery init container.
      ## container delivers required xml configuration and *.ss files to RAA work dir
      ## those files delivered from gzip archive encoded as base64 and specified via {{
.Values.raa.deplyment.configTar }} helm value
      ## default conf.xml and user-data-map.ss are supplied with chart
      ## they are copied to work dir when absent and helm value above is not specified
      configDelivery: {}

      #  ## container name template
      #  ## should be uncommented when configDelivery is enabled
      #  name: "{{ $.Chart.Name }}-conf-delivery"

      ## optional
      ## enables test run init container (GCXI-3463).
      ## it performs running of all the aggregates on empty data
      ## to test the correctness of configrugation and customization
      testRun:

        ## container name template
        name: "{{ $.Chart.Name }}-test-run"

      ## main aggregation container
      aggregation:

        ## container name template
```

```
      name: "{{ $.Chart.Name }}"

    ## optional
    ## enable side car container for monitoring and processing of:
    ## - new tool command requests from a file in work dir
    ## - new prometheus scrape requests
    monitor: {}

    #  ## container name template
    #  name: "{{ $.Chart.Name }}-monitor"

    #  ## optional
    #  ## enables command processing from file placed into work dir (GCXI-4052)
    #  ## it is helpful when kubect exec is forbidden by environment restrictions.
    #  toolcmd:

    #    ## interval of checking for a new file with command
    #    intervalSec: "20"

    #  ## optional
    #  ## enables raa metrics scraping in Prometheus format via http
    #  metrics:

    #    ## k8s port name
    #    portName: "metrics"

    #    ## port number
    #    containerPort: "9100"

    #  ## optional
    #  ## enables raa health metric scraping in Prometheus format via http
    #  health:

    #    ## k8s port name
    #    portName: "health"

    #    ## port number
    #    containerPort: "9101"


## optional
## defines if any test pod will be declared for helm release testing
## see links:
## - https://helm.sh/docs/topics/chart_tests/
## - https://helm.sh/docs/helm/helm_test/
testPods:

  ## optional
  ## checks the result of test run init container execution
  ## actual when {{ .Values.raa.statefulset.containers.testRun }} is specified.
  testRunCheck:

    ## pod name template
    name: "{{ tpl .Values.raa.serviceName . }}-test-run-check"

    container:
      ## container name template
      name: "{{ $.Chart.Name }}-test-run-check"

    ## optional
    ## test pod specific labels are adjusted to common labels
    labels: {}
```

```
  ## test pod specific annotations are adjusted to common annotations
  annotations:
    "helm.sh/hook-weight": "100"
    "helm.sh/hook": "test-success"
    "helm.sh/hook-delete-policy": "before-hook-creation"

## optional
## performs RAA health check
healthCheck:
  ## pod name template
  name: "{{ tpl .Values.raa.serviceName . }}-health-check"

  container:
    ## container name template
    name: "{{ $.Chart.Name }}-health-check"

  ## optional
  ## test pod specific labels are adjusted to common labels
  labels: {}

  ## test pod specific annotations are adjusted to common annotations
  annotations:
    "helm.sh/hook-weight": "200"
    "helm.sh/hook": "test-success"
    "helm.sh/hook-delete-policy": "before-hook-creation"
```

Upgrade, rollback, or uninstall RAA

Contents

- 1 Upgrade RAA
- 2 Rollback RAA
- 3 Uninstall RAA

Learn how to upgrade, rollback, or uninstall Reporting and Analytics Aggregates (RAA)

Related documentation:

-
-
-
-

RSS:

- For private edition

Upgrade RAA

Update existing helm chart:

```
helm upgrade gcxi-raa-lab helmrepo/gcxi-raa --values myvalues.yaml
```

Rollback RAA

Rollback helm chart:

```
helm rollback gcxi-raa-lab --recreate-pods
```

Uninstall RAA

Delete helm chart:

```
helm uninstal gcxi-raa-lab
```

Before you begin deploying GCXI

Contents

- 1 Limitations and assumptions
- 2 Download the Helm charts
- 3 Third-party prerequisites
- 4 Storage requirements
  - 4.1 Volumes Design
- 5 Preparing the environment
- 6 Network requirements
- 7 Ingress
- 8 WAF Rules
- 9 SMTP
- 10 TLS
- 11 Browser requirements
- 12 Genesys dependencies
- 13 GDPR support

Find out what to do before deploying Genesys Customer Experience Insights (GCXI).

Related documentation:

-
-
-
-

RSS:

- For private edition

Limitations and assumptions

GCXI can provide meaningful reports only if Genesys Info Mart and Reporting and Analytics Aggregates (RAA) are deployed and available. Deploy GCXI only after Genesys Info Mart and RAA.

Download the Helm charts

For more information about how to download the Helm charts in Jfrog Edge, see the suite-level documentation: Downloading your Genesys Multicloud CX containers

To learn what Helm chart version you must download for your release, see Helm charts and containers for Genesys Customer Experience Insights

GCXI Containers

- GCXI Helm chart uses the following containers.
  - **gcxi** - main GCXI container, runs as a StatefulSet. This container is roughly 12 GB; ensure that you have enough space to allocate it.
  - **gcxi-control** - supplementary container, used for initial installation of GCXI, and for clean-up.

GCXI Helm Chart Download the latest yaml files from the repository, or examine the attached files: Sample GCXI yaml files

Third-party prerequisites

For more information about setting up your Genesys Multicloud CX private edition platform, including Kubernetes, Helm, and other prerequisites, see Software requirements.

Third-party services

Name	Version	Purpose	Notes
A container image registry and Helm chart repository		Used for downloading Genesys containers and Helm charts into the customer's repository to support a CI/CD pipeline. You can use any Docker OCI compliant registry.	
PostgreSQL 12	12	Relational database (Genesys Customer Experience Insights meta requires this version)	Version 12 is required.
MicroStrategy		Genesys Customer Experience Insights provides this software but requires replacement of MicroStrategy 3rd party keys.	
Load balancer		VPC ingress. For NGINX Ingress Controller, a single regional Google external network LB with a static IP and wildcard DNS entry will pass HTTPS traffic to NGINX Ingress Controller which will terminate SSL traffic and will be setup as part of the platform setup.	
An SMTP relay		Facilitates email communications in an environment where GCXI reports or voicemails are sent as emails to contact center personnel. Genesys recommends PostFix, but you can use any SMTP relay that supports standard mail libraries.	

Storage requirements

GCXI installation requires a set of local Persistent Volumes (PVs). Kubernetes *local* volumes are directories on the host with specific properties: https://kubernetes.io/docs/concepts/storage/volumes/#local

Example usage: https://zhimin-wen.medium.com/local-volume-provision-242affd5efe2

Kubernetes provides a powerful volume plugin system, which enables Kubernetes workloads to use a wide variety of block and file storage to persist data.

You can use the GCXI Helm chart to set up your own PVs, or you can configure PV Dynamic Provisioning in your cluster so that Kubernetes automatically creates PVs.

Volumes Design

GCXI installation uses the following PVC:

Mount Name	Mount Path (inside container)	Description	Access Type	Approximate Size	Default Mount Point on Host (You can change the mount point using values.) The local provisioner requires that the specified directory pre-exists on your host.	Must be Shared across Nodes?	Required Node Label (applies to default Local PV setup)
gcxi-backup	/genesys/gcxi_shared/backup	Backup files Used by control container / jobs.	RWX	Depends on backup frequency. 5 GB+	/genesys/gcxi/backup You can override this setting using Values.gcxi.local.pv.backup.path	Only in multiple concurrent installs scenarios.	gcxi/local-pv-gcxi-backup = "true"
gcxi-log	/mnt/log	MSTR logs Used by main container. The GCXI Helm chart allows log volumes of legacy `hostPath`	RWX	Depends on rotation scheme. 5 GB+	/mnt/log/gcxi subPathExpr: $(POD_NAME) You can override this setting using	Not necessarily.	gcxi/local-pv-gcxi-log = "true" If you are using hostPath volumes for logs, you don't need

Mount Name	Mount Path (inside container)	Description	Access Type	Approximate Size	Default Mount Point on Host (You can change the mount point using values.) The local provisioner requires that the specified directory pre-exists on your host.	Must be Shared across Nodes?	Required Node Label (applies to default Local PV setup)
		type. This scenario is the default and used in examples in this document.			Values.gcxi.local.pv.log.path		node label.
gcxi-postgres	/var/lib/postgresql/data (if using Postgres in container) or disk space in Postgres RDBMS	Meta DB volume Used by Postgres container, if deployed.	RWO	Depends on usage. 10 GB+	/genesys/gcxi/shared You can override this setting using Values.gcxi.local.pv.postgres.path	Yes, unless you tie the Postgres container to some particular node.	gcxi/local-pv-postgres-data = "true"
gcxi-share	/genesys/gcxi_share	MSTR shared caches and cubes Used by main container.	RWX	Depends on usage. 5 GB+	/genesys/gcxi/data subPathExpr: $(POD_NAME) You can override this setting using Values.gcxi.local.pv.share.path	Yes.	gcxi/local-pv-gcxi-share = "true"

Preparing the environment

To prepare your environment, complete the following steps:

1. To log in to the cluster, run the following command:

- For AKS:

  ```
  az aks get-credentials --resource-group  --name
  ```

- For GKE:

  ```
  gcloud container clusters get-credentials gke1
  ```

- For OpenShift:

  ```
  oc login --token  --server
  ```

  - To check the cluster version on OpenShift deployments, run the following command:

    ```
    oc get clusterversion
    ```

2. To create a new project, run the following command:
   GKE or AKS:

   1. Edit the **create-gcxi-namespace.json**, adding the following values:

      ```
      {
        "apiVersion": "v1",
        "kind": "Namespace",
        "metadata": {
          "name": "gcxi",
          "labels": {
            "name": "gcxi"
          }
        }
      }
      ```

   2. To apply the changes, run the following command:

      ```
      kubectl apply -f apply create-gcxi-namespace.json
      ```

   OpenShift:

   ```
   oc new-project gcxi
   ```

3. For GKE or AKS, to confirm namespace creation, run the following command:

   ```
   kubectl describe namespace gcxi
   ```

4. Create a secret for docker-registry to pull images from the Genesys JFrog repository:

   ```
   kubectl create secret docker-registry  --docker-server= --docker-username= --docker-
   password= --docker-email= -n gim
   ```

5. Create the file **values-test.yaml**, and populate it with appropriate override values. For a simple deployment using PostgreSQL inside the container, you must include PersistentVolumes named **gcxi-log-pv**, **gcxi-backup-pv**, **gcxi-share-pv**, and **gcxi-postgres-pv**. You must override **GCXI_GIM_DB** with the name of your Genesys Info Mart data source.

Network requirements

Ingress

Ingress annotations are supported in the **values.yaml** file (see line 317). Genesys recommends session stickiness, to improve user experience.

```
ingress:
    # http path and annotations may be overriden for external and internal access separately
    annotations:
      kubernetes.io/ingress.class: "nginx"
      nginx.ingress.kubernetes.io/affinity: cookie
      nginx.ingress.kubernetes.io/affinity-mode: persistent
      nginx.ingress.kubernetes.io/proxy-body-size: "50m"
      nginx.ingress.kubernetes.io/proxy-buffer-size: "8k"
      nginx.ingress.kubernetes.io/ssl-redirect: "false"
    domain:
    external:
      annotations:
      host:
      tls:
        enabled: false
        secretName:
    internal:
      annotations:
      host:
      tls:
        enabled: false
        secretName:
    path: /
```

Allowlisting is required for GCXI.

WAF Rules

WAF rules are defined in the **variables.tf** file (see line 245).

SMTP

The GCXI container and Helm chart support the environment variable `EMAIL_SERVER`.

TLS

The GCXI container does not serve TLS natively. Ensure that your environment is configured to use proxy with HTTPS offload.

Browser requirements

MicroStrategy Web is the user interface most often used for accessing, managing, and running the Genesys CX Insights reports. MicroStrategy Web certifies the latest versions, at the time of release, for the following web browsers:

- Apple Safari
- Google Chrome (Windows and iOS)
- Microsoft Edge
- Microsoft Internet Explorer (Versions 9 and 10 are supported, but not certified)
- Mozilla Firefox

To view updated information about supported browsers, see the MicroStrategy ReadMe.

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.
Microsoft Edge (Legacy)	Current release	
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.

Genesys dependencies

GCXI requires the following services:

- Reporting and Analytics Aggregates (RAA) is required to aggregate Genesys Info Mart data.
- Genesys Info Mart and / or Intelligent Workload Distribution (IWD) Data Mart. GCXI can run without these services, but cannot produce meaningful output without them.
- GWS Auth/Environment service
- Genesys Platform Authentication thru Config Server (GAuth). Alternatively, GCXI includes a native internal login, which you can use to authorize users, instead of GAuth. This document assumes you are

using GAuth (the recommended solution), which gives ConfigServer users access to GCXI.

- GWS client id/client secret

GDPR support

GCXI can store Personal Identifiable Information (PII) in logs, history files, and in reports (in scenarios where customers include PII data in reports). Genesys recommends that you do not capture PII in reports. If you do capture PII, it is your responsibility to remove any such report data within 21 days or less, if required by General Data Protection Regulation (GDPR) standards.

For more information and relevant procedures, see: Genesys CX Insights Support for GDPR and the suite-level **Link to come** documentation.

Configure GCXI

Contents

- 1 Override Helm chart values
- 2 Configure Kubernetes
- 3 ConfigMaps
- 4 Secrets
- 5 Configure security
  - 5.1 Arbitrary UID

Learn how to configure Genesys Customer Experience Insights (GCXI).

Related documentation:

-
-
-
-

RSS:

- For private edition

Override Helm chart values

Before you begin, download the latest yaml files from the repository, or examine the attached files: Sample GCXI yaml files . Helm values are described in **values.yaml**. See the comments accompanying each Helm value.

You can override values in the Helm charts to configure Private Edition. For more information, see the "suite-level" documentation about how to override Helm chart values: Overriding Helm chart values.

Configure Kubernetes

This section provides information about Kubernetes configuration.

ConfigMaps

Configuration information is stored in ConfigMap.

See the **gcxi-worker-statefulset.yaml** file:

```
envFrom:
        - configMapRef:
            name: gcxi-config{{ template "deploymentCode" . }}
            optional: true
        - configMapRef:
            name: gcxi-config-ext{{ template "deploymentCode" . }}
            optional: true
        {{- range $cm := .Values.gcxi.configMaps }}
        - configMapRef:
            name: {{ tpl $cm.name $ }}
            optional: true
```

Secrets

GCXI supports the following methods of secret injection:

- CSI driver
- Kubernetes secrets
- Environment Variables

See the **values.yaml** file:

```
secrets:
  - name: gcxi-secret-pg
```

See the **gcxi-worker-statefulset.yaml** file:

```
- name: gcxi-var
        projected:
          sources:
          - secret:
              name: gcxi-secret{{ template "deploymentCode" . }}
              optional: true
          - secret:
              name: gcxi-secret-ext{{ template "deploymentCode" . }}
              optional: true
          {{- range $secret := .Values.gcxi.secrets }}
          - secret:
              name: {{ tpl $secret.name $ }}
              {{- with $secret.items }}
              items:
              {{- range $item := $secret.items }}
              - key: {{ tpl $item.key $ }}
                path: {{ tpl $item.path $ }}
              {{- end }}
              {{- end }}
              optional: true
          {{- end }}
```

Configure security

GCXI is based on a 3rd-party product (MicroStrategy), and as result has some special considerations:

- The main container is about 12 GB.
- Genesys recommends that you enable hostIPC. However, hostIPC is disabled by default and in some deployments, GCXI can operate successfully without it.
  1. Enable hostIPC by setting:

     ```
     hostIPC: true
     ```

  2. Configure hostIPC at the node level as follows:

```
echo "kernel.sem = 250 1024000 250 4096" >> /etc/sysctl.conf
echo "vm.max_map_count = 5242880" >> /etc/sysctl.conf
sysctl -p
```

Arbitrary UID

- The deployment routine assigns an arbitrary user ID (UID) and group ID to pods during deployment. Default file ownership is **genesys:root (500:0)**.
- If your OpenShift deployment uses arbitrary UIDs, you must override the securityContext settings in the **values.yaml** file (see line 456) as follows:

```
secrets:
 podSecurityContext:
   fsGroup: null
   runAsUser: null
   runAsGroup: 0
   runAsNonRoot: true

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

The default values (user ID = 500) are suitable for many other deployment scenarios:

```
secrets:
  securityContext:
    control:
      fsGroup: null
      runAsUser: 500
      runAsGroup: 500
    worker:
      fsGroup: null
      runAsUser: 500
      runAsGroup: 500
```

Provision GCXI

- Administrator

Learn how to provision Genesys CX Insights Guide (GCXI).

Related documentation:

-
-
-
-

RSS:

- For private edition

In Agent Setup, add users to appropriate groups. See the following resources:

- Manage agents and other users using Agent Setup
- Configuration Server Access Groups
- Genesys CX Insights User Management

Deploy Genesys Customer Experience Insights

Contents

- 1 Assumptions
- 2 Prepare to deploy GCXI
  - 2.1 Creating PVCs
- 3 Install GCXI
  - 3.1 1. Prepare for deployment
  - 3.2 2. Deploy GCXI
- 4 Validate the deployment
- 5 Maintenance and troubleshooting

Learn how to deploy Genesys Customer Experience Insights (GCXI) into a private edition environment.

Related documentation:

-
-
-
-

RSS:

- For private edition

Assumptions

- The instructions on this page assume you are deploying the service in a service-specific namespace, named in accordance with the requirements on Creating namespaces. If you are using a single namespace for all private edition services, replace the namespace element in the commands on this page with the name of your single namespace or project.
- Similarly, the configuration and environment setup instructions assume you need to create namespace-specific (in other words, service-specific) secrets. If you are using a single namespace for all private edition services, you might not need to create separate secrets for each service, depending on your credentials management requirements. However, if you do create service-specific secrets in a single namespace, be sure to avoid naming conflicts.

> **Important**
> Review Before you begin deploying GCXI for the full list of prerequisites required to deploy GCXI.

Prepare to deploy GCXI

Before you begin, ensure that:

- You are logged in to your OpenShift, GKE, or AKS cluster.
- For HA deployments, at least two worker nodes are available for GCXI pods.
- Each Worker machine meets the following minimum requirements:
  - 64-bit compatible CPU architecture. (2 or more CPUs.)

  - 10 GB for each GCXI container, and 2 GB for the PostgreSQL container. Production deployments commonly reserve between 16 GB and 64 GB of RAM for each container.
  - 40 GB of available disk space, if you are loading images from a repository.
- Helm-3 is installed on the host where the deployment will run.
- Images **gcxi** and **gcxi_control** are tagged and loaded on the registry. During deployment, OpenShift pulls the images from the registry to each OpenShift worker node.
- On each worker node, values are set for `kernel.sem` and `vm.max_map_count`, as required by MicroStrategy. For example:

```
echo "kernel.sem = 250 1024000 250 4096" >> /etc/sysctl.conf echo "vm.max_map_count =
 5242880" >> /etc/sysctl.conf sysctl -p
```

PVCs required by GCXI

Mount Name	Mount Path (inside container)	Description	Access Type	Default Mount Point on the Host (you can change this using values; ensure that the indicated directory pre-exists on your host, to accommodate the local provisioner)	Shared across Nodes?	Required Node Label (applies to default Local PVs setup)
gcxi-backup	/genesys/gcxi_shared/backup	Backup files Used by control container / jobs.	RWX	/genesys/gcxi/backup You can override this path using Values.gcxi.local.pv.backup.path	Not necessarily.	gcxi/local-pv-gcxi-backup = "true"
gcxi-log	/mnt/log	MSTR logs Used by main container. The Chart allows log volumes of legacy hostPath type. This scenario is the default.	RWX	/mnt/log/gcxi subPathExpr: $(POD_NAME) You can override this value using Values.gcxi.local.pv.log.path	Must not be shared across nodes.	gcxi/local-pv-gcxi-log = "true" If you are using hostPath volumes for logs, the Node label is not required.
gcxi-postgres	/var/lib/postgresql/data	Meta DB volume Used by Postgres	RWO	/genesys/gcxi/shared You can override this	Yes, unless you tie the PostgreSQL container to a particular	gcxi/local-pv-postgres-data = "true"

		container, if deployed.		value using Values.gcxi.local.pv.postgres.path	node	
gcxi-share	/genesys/gcxi_share	MSTR shared caches and cubes. Used by main container.	RWX	/genesys/gcxi/data subPathExpr: $(POD_NAME) You can override this value using Values.gcxi.local.pv.share.path	Yes	gcxi/local-pv-gcxi-share = "true"

Creating PVCs

Genesys recommends either of the following methods to create PVCs:

Create PVCs in advance

You can create PVCs in advance, and just configure the names in the values override file, for example:

```
pvc:
    log:
      volumeName: gcxi-log-pv
    backup:
      volumeName: gcxi-backup-pv
    share:
      volumeName: gcxi-share-pv
    postgres:
      volumeName: gcxi-postgres-pv
```

Create PVCs using Helm

You can configure the storage class name in the values override file, and let Helm create the PVCs, for example:

```
pvc:
    backup:
      capacity: 1Gi
      # if gcxi.deployment.deployLocalPV == true, this defaults to
gcxi.storageClass.local.name
      # otherwise must be set explicitly
      storageClassName: azure-files-retain #storage class name
      volumeName:
    log:
      capacity: 1Gi
      hostPath: /mnt/log/gcxi
      # if gcxi.deployment.deployLocalPV == true, this defaults to
gcxi.storageClass.local.name
      # otherwise must be set explicitly
      storageClassName: azure-files-retain #storage class name
      volumeName:
    postgres:
      capacity: 1Gi
      # if gcxi.deployment.deployLocalPV == true, this defaults to
gcxi.storageClass.local.name
```

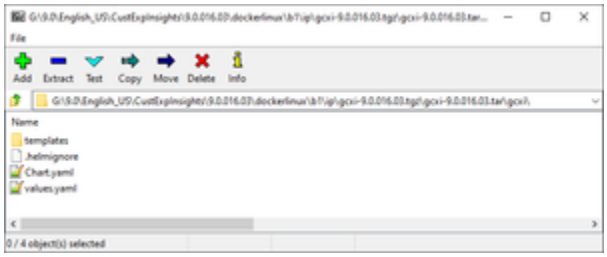
```
      # otherwise must be set explicitly
      storageClassName: azure-files
      volumeName:
    share:
      capacity: 1Gi
      # if gcxi.deployment.deployLocalPV == true, this defaults to
gcxi.storageClass.local.name
      # otherwise must be set explicitly
      storageClassName: azure-files-retain
      volumeName:
```

Install GCXI

The following procedures describe example steps to deploy GCXI with OpenShift, GKE, or AKS. The exact steps required vary depending on your environment.

1. Prepare for deployment

To prepare the environment and gather files needed for deployment, complete the steps in this section.



Contents of the Helm IP

Prerequisites

Within the Genesys Customer Experience Insights package for the Docker Linux OS, look for the Helm installation package (IP) — a small TGZ file (for example **gcxi-9.0.018.00.tgz**) that contains the Helm files. You require these files to complete this procedure.

1. On the host where the deployment will run, create a folder: **helm**.
2. Copy the Helm installation package (for example **gcxi-9.0.018.00.tgz**) into the **helm** folder, and extract the archive into a subfolder called **helm/gcxi**.
3. View the file **helm/gcxi/Chart.yaml**, and ensure that the appVersion is set to the appropriate GCXI version.
4. Open the file **helm/gcxi/values.yaml.** Follow the instructions provided in the file, and create a **values-test.yaml** file with content that is appropriate for your environment. Save the new file in the **helm** folder.

   Genesys provides the content in the following **values-test.yaml** file as an example. The example contains values that are appropriate for a simple deployment with key parameters, ingress, and the PersistentVolumes **gcxi-log-pv**, **gcxi-backup-pv**, and **gcxi-share-pv**. For information about PersistentVolumes, see PVCs required by GCXI.

```
gcxi:
     env:
         GCXI_GIM_DB:
              DSNDEF:
DSN_NAME=GCXI_GIM_DB;DB_TYPE=POSTGRESQL;DB_TYPE_EX=PostgreSQL;HOST=gim_db_host;PORT=5432;DB_NAME=gim_d
              LOGIN: gim_login
              PASSWORD: gim_password

         IWD_DB:
              DSNDEF:
DSN_NAME=IWD_DB;DB_TYPE=POSTGRESQL;DB_TYPE_EX=PostgreSQL;HOST=iwd_db_host;PORT=5432;DB_NAME=dm_gcxi;LO
            LOGIN: iwd_login
            PASSWORD: iwd_password
     deployment:
        deployLocalPV: false
        useDynamicLogPV: false
        hostIPC: false
     replicas:
        worker: 2
     pvc:
         log:
            volumeName: gcxi-log-pv
        backup:
             volumeName: gcxi-backup-pv
        share:
            volumeName: gcxi-share-pv
     ingress:
       # http path and annotations may be overriden for external and internal access
separately
       annotations:
         nginx.ingress.kubernetes.io/affinity: cookie
         nginx.ingress.kubernetes.io/affinity-mode: persistent
         nginx.ingress.kubernetes.io/proxy-body-size: "50m"
         nginx.ingress.kubernetes.io/proxy-buffer-size: "8k"
         nginx.ingress.kubernetes.io/ssl-redirect: "false"
      domain:
        external:
          annotations:
          host: gcxi.
          tls:
            enabled: true
            secretName: gcxi-ingress-ext
        internal:
          annotations:
          host: gcxi-int.
          tls:
            enabled: true
            secretName: gcxi-ingress-int
        path: /
```

2. Deploy GCXI

This procedure provides steps you can use to deploy GCXI in environments without LDAP. For environments that include LDAP or other features not supported in **values.yaml,** you can pass container environment variables such MSTR_WEB_LDAP_ON=true using the **gcxi.envvars** file, for example: `--set-file gcxi.envext=gcxi.envvars`.

1. Log in to the OpenShift, GKE, or AKS cluster from the host where you will run deployment.

2. For debug purposes, run the following command to render templates without installing:

   ```
   helm template --debug -f values-test.yaml gcxi-helm gcxi/
   ```

   Kubernetes descriptors appear. The values you see are generated from Helm templates, and based on settings from **values.yaml** and **values-test.yaml**. Ensure that no errors appear; you will later apply this configuration to your Kubernetes cluster.

3. To deploy GCXI, execute the following command:

   ```
   helm install --debug --namespace gcxi --create-namespace -f values-test.yaml gcxi-oc
   gcxi/
   ```

   Genesys recommends that you avoid using helm install with the `--wait` option when deploying GCXI. If you use `--wait`, the Helm architecture post-install hook (which in this case is the gcxi_init job) won't be triggered properly. For more information, see the Helm documentation.

   This process takes several minutes. Wait until the installation routine has created and allocated all of the objects, and the Kubernetes descriptors that are applied to the environment appear.

4. To check the installed Helm release, run the following command:

   ```
   helm list –n gcxi
   ```

5. To check GCXI OpenShift objects created by Helm, run the following command:

   ```
   kubectl get all -n gcxi
   ```

6. This step is applicable only to OpenShift deployments. Complete this step only if you have not enabled ingress in the values override file to make GCXI accessible from outside the cluster, using the standard HTTP port. For production environments, Genesys recommends that you create secure routes as discussed on the OpenShift website. For testing or development environments, perform the following steps:

   1. To make available the gcxi service, run the following command:

      ```
      oc expose service gcxi --port web --name web
      ```

   2. To verify that the new route exists in the gcxi project, run the following command:

      ```
      oc get route -n gcxi
      ```

      Route information appears, similar to the following:

      ```
      NAME      HOST/PORT                              PATH   SERVICES   PORT
      TERMINATION   WILDCARD
      web       web-gcxi.                               gcxi
      web                     None
      ```

      where is the host name generated by OpenShift.

7. Verify that you can now access GCXI at the following URL:

   ```
   http://web-gcxi./MicroStrategy/servlet/mstrWeb
   ```

Validate the deployment

Check to ensure that you can now view reports and dashboards.

Viewing Historical Reports

To view (or edit) the GCXI historical reports, open MicroStrategy Web by pointing your web browser to `http://:/MicroStrategy/servlet/mstrWeb*`, where `:` are the server name and port provided by your administrator.

Verify that an assortment of report folders are present, such as Agents, Business Results, and Callback, and that each one contains one or more reports. Before you can view a report, you must run the report to populate data. For more information, see Generate historical reports.

Maintenance and troubleshooting

Use the instructions in this section only if you encounter errors or other difficulties. Problems with the deployment are most often associated with the following three kinds of objects:

- PVs
- PVCs
- pods

Use the following steps to examine events associated with these objects:

1. To list the objects that can cause problems, run the following commands:

   ```
   kubectl get pv -o wide

   kubectl get pvc -o wide -n gcxi

   kubectl get po -o wide -n gcxi
   ```

2. Examine the output from each **get** command. If any of the objects have a non-ready state (for example, **Unbound** (PVCs only), **Pending**, or **CrashLoop**) run the following command to inspect the object more closely using **kubectl describe**:

   ```
   kubectl describe   For example: kubectl describe po gcxi-0
   ```

3. In the **describe** output, inspect the section **Events**.

Upgrade, roll back, or uninstall GCXI

Contents

- 1 Supported upgrade strategies
- 2 Timing
  - 2.1 Scheduling considerations
- 3 Monitoring
- 4 Preparatory steps
- 5 Back up the meta database
  - 5.1 Back up using the container variable
  - 5.2 Back up using the Kubernetes job
  - 5.3 Restoring from a backup
- 6 Rolling Update
  - 6.1 Rolling Update: Upgrade
  - 6.2 Rolling Update: Verify the upgrade
  - 6.3 Rolling Update: Rollback
  - 6.4 Rolling Update: Verify the rollback
- 7 Uninstall
  - 7.1 GCXI example
  - 7.2 RAA example

Learn how to upgrade, roll back, or uninstall GCXI.

Related documentation:

-
-
-
-

RSS:

- For private edition

> **Important**
> The instructions on this page assume you have deployed the services in service-specific namespaces. If you are using a single namespace for all private edition services, replace the namespace element in the commands on this page with the name of your single namespace or project.

Supported upgrade strategies

Genesys Customer Experience Insights supports the following upgrade strategies:

Service	Upgrade Strategy	Notes
Genesys CX Insights	• Rolling Update	
Reporting and Analytics Aggregates	• Rolling Update	

The upgrade or rollback process to follow depends on how you deployed the service initially. Based on the deployment strategy adopted during initial deployment, refer to the corresponding upgrade or rollback section on this page for related instructions.

For a conceptual overview of the upgrade strategies, refer to Upgrade strategies in the Setting up Genesys Multicloud CX Private Edition guide.

In scenarios where you have previously deployed GCXI using Helm, use the instructions on this page to upgrade to a newer GCXI release.

Timing

A regular upgrade schedule is necessary to fit within the Genesys policy of supporting N-2 releases, but a particular release might warrant an earlier upgrade (for example, because of a critical security fix).

If the service you are upgrading requires a later version of any third-party services, upgrade the third-party service(s) before you upgrade the private edition service. For the latest supported versions of third-party services, see the Software requirements page in the suite-level guide.

Scheduling considerations

Genesys recommends that you upgrade the services methodically and sequentially: Complete the upgrade for one service and verify that it upgraded successfully before proceeding to upgrade the next service. If necessary, roll back the upgrade and verify successful rollback.

Monitoring

Monitor the upgrade process using standard Kubernetes and Helm metrics, as well as service-specific metrics that can identify failure or successful completion of the upgrade (see Observability in Genesys Customer Experience Insights).

Genesys recommends that you create custom alerts for key indicators of failure — for example, an alert that a pod is in pending state for longer than a timeout suitable for your environment. Consider including an alert for the absence of metrics, which is a situation that can occur if the Docker image is not available. Note that Genesys does not provide support for custom alerts that you create in your environment.

Preparatory steps

Ensure that your processes have been set up to enable easy rollback in case an upgrade leads to compatibility or other issues.

Each time you upgrade a service:

1. Review the release note to identify changes.
2. Ensure that the new package is available for you to deploy in your environment.
3. Ensure that your existing **-values.yaml** file is available and update it if required to implement changes.

The Helm installation package (IP) has a file name in the format gcxi-.tgz, for example **gcxi-100.0.029+0000.tgz**.

The following steps provide detailed information about preparing to upgrade GCXI:

1. On the Control Plane node, set the current directory to the helm folder, and run the following command to create a subfolder to differentiate this release from others:

```
mkdir helm_
```

where is the folder in which to extract the Helm package. For example: **018.00**.

2. To extract the Helm package, run the following command:

```
tar xvzf  -C helm_
```

where:
 is the folder you created in the previous step.
 is the name of the gcxi Helm package you are installing.

For example:

```
mkdir helm_018.00; tar xvzf gcxi-9.0.018.00.tgz -C helm_018.00
```

3. From the folder where you deployed the previous release of GCXI using Helm, copy the file **values-test.yaml** into the new folder.

Back up the meta database

The GCXI upgrade process automatically backs up your meta database before upgrading to the new release of GCXI. However, GCXI also provides additional options to back up the contents of your meta database, including:

- Scheduled backup using the container option BACKUP_HOUR.
- Immediate backup using a Kubernetes job.
- Immediate backup using PostgreSQL tools.

Backup files are stored in the gcxi-shared mount point, which is found in /genesys/gcxi/shared, unless you configure a different location. (By default, backups are stored in a subfolder called backup in /genesys/gcxi/shared).

Back up using the container variable

Genesys recommends that you configure the container option to automatically back up your data. To schedule automatic backups, set the following container variable:

```
BACKUP_HOUR=
```

Where digits in 24-hr format instruct the container to perform meta db backup in corresponding hours, for example to back up at 22:00, 6:00, and 15:00 hours:

```
BACKUP_HOUR=22,6,15
```

Back up using the Kubernetes job

You can perform a backup at any time using the included Kubernetes job, gcxi-backup, as follows:

```
kubectl delete job "gcxi-backup" || true
kubectl apply -f gcxi-backup.yaml
```

This creates two backup files (that is, service databases): mstr_meta_.pgdump and mstr_hist_.pgdump, where is a timestamp. You can optionally set another value for using the BACKUP_TAG variable in gcxi-backup job yaml.

Restoring from a backup

Ensure that backup files (pgdump) are present in the **gcxi-shared** mount location. If they are in another location, you must set the RESTORE_TAG variable in the **gcxi-restore.yaml** file.

You must stop the GCXI pods before performing a restore operation. Stop the pods using the following command:

```
kubectl scale --replicas=0 statefulset/gcxi
```

Execute the following commands to restore the meta database:

```
kubectl delete job "gcxi-restore" || true
kubectl apply -f gcxi-restore.yaml
```

Use the following command to restart the pods:

```
kubectl scale --replicas=1 statefulset/gcxi
```

Rolling Update

Rolling Update: Upgrade

Execute the following command to upgrade :

```
helm upgrade --install  -f -values.yaml  -n
```

Tip: If your review of Helm chart changes (see Preparatory Step 3) identifies that the only update you need to make to your existing **-values.yaml** file is to update the image version, you can pass the image tag as an argument by using the `--set` flag in the command:

```
helm upgrade --install  -f -values.yaml  --set .image.tag=
```

GCXI example

Set the command context by navigating to the new folder you created (see above) and run the following command:

```
helm upgrade --debug gcxi-helm gcxi/ --namespace gcxi --create-namespace -f values-test.yaml
```

RAA example

Run the following command:

```
helm upgrade gcxi-raa-lab helmrepo/gcxi-raa --values myvalues.yaml
```

Rolling Update: Verify the upgrade

Follow usual Kubernetes best practices to verify that the new service version is deployed. See the information about initial deployment for additional functional validation that the service has upgraded successfully.

Rolling Update: Rollback

Execute the following command to roll back the upgrade to the previous version:

```
helm rollback
```

or, to roll back to an even earlier version:

```
helm rollback
```

Alternatively, you can re-install the previous package:

1. Revert the image version in the `.image.tag` parameter in the **-values.yaml** file. If applicable, also revert any configuration changes you implemented for the new release.
2. Execute the following command to roll back the upgrade:

   ```
   helm upgrade --install  -f -values.yaml
   ```

   Tip: You can also directly pass the image tag as an argument by using the `--set` flag in the command:

   ```
   helm upgrade --install  -f -values.yaml  --set .image.tag=
   ```

RAA example

```
helm rollback gcxi-raa-lab --recreate-pods
```

GCXI example

> **Important**
>
> Do not use the `rollback` command to roll back a GCXI upgrade. You must re-install a previous version.

Rolling Update: Verify the rollback

Verify the rollback in the same way that you verified the upgrade (see Rolling Update: Verify the upgrade).

Uninstall

> **Warning**
> Uninstalling a service removes all Kubernetes resources associated with that service. Genesys recommends that you contact Genesys Customer Care before uninstalling any private edition services, particularly in a production environment, to ensure that you understand the implications and to prevent unintended consequences arising from, say, unrecognized dependencies or purged data.

Execute the following command to uninstall :

```
helm uninstall  -n
```

GCXI example

Execute the following command to remove GCXI:

```
helm uninstall gcxi-helm -n gcxi
```

RAA example

```
helm uninstall gcxi-raa-lab
```

Observability in Genesys Customer Experience Insights

Contents

- 1 Monitoring
  - 1.1 About monitoring in GCXI
  - 1.2 About monitoring in RAA
  - 1.3 Enable monitoring
  - 1.4 Enable GCXI monitoring
  - 1.5 Enable RAA monitoring
  - 1.6 Monitoring summary
  - 1.7 Configure metrics
- 2 Alerting
  - 2.1 Configure alerts
- 3 Logging

Learn about the logs, metrics, and alerts you should monitor for Genesys Customer Experience Insights.

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 Customer Experience Insights metrics, see:

- Genesys CX Insights metrics
- Reporting and Analytics Aggregates metrics

See also System metrics.

The metrics described in this document pertain to the internal monitoring of Genesys CX Insights (GCXI) and Reporting and Analytics Aggregates (RAA) performance and containers, and are intended for administrator use only. These metrics are distinct from the metrics which GCXI provides for monitoring contact center activity, which are described in Work with Genesys CX Insights Reports

About monitoring in GCXI

GCXI container provides TCP metrics endpoint (Prometheus format) on port 8180, and HTTP endpoint at: **/gcxi/monitor/metrics**, for example https://gcxi-22-30.genhtcc.com/gcxi/monitor/metrics.

About monitoring in RAA

In RAA, the **Health port** retrieves the health status. It can take as much as three minutes for RAA to analyze Health status.

In addition to health metrics, RAA provides other metrics that capture statistical information and other data extracted using the **Metrics port**. The first time RAA scrapes this metric data, it aggregates data beginning from the the earliest data available. On subsequent occasions, RAA aggregates the data over the time period that has elapsed since the previous scrape. RAA reads this data from a local file that contains only two or three scrape intervals, which speeds the process; the aggregation of this statistical information typically takes only a few seconds.

Enable monitoring

Monitoring is not enabled by default for GCXI or RAA.

Enable GCXI monitoring

The GCXI Helm Chart provides a standard ServiceMonitor object for integration with Prometheus Operator, in gcxi-service-monitor.yaml.

The following Helm Chart values indicate whether to deploy ServiceMonitor object for integration with Prometheus Operator:

```
gcxi.deployment.deployServiceMonitor = false|true



gcxi.ports.worker.metrics = 8180 (default)
container port for metrics endpoint
```

Enable RAA monitoring

RAA makes metrics available with the help of a sidecar container.

1. To start the sidecar container, you specify the **raa.statefulset.containers.monitor** element, including ports for metrics and/or health endpoints, in the **values.yaml** file. Ensure that the value of **raa.env.STAT_SCRAPE_INTERVAL** (in seconds) exceeds the scrape interval configured in Prometheus for the metrics port.
   For example:

```
raa: ...
  env:
    STAT_SCRAPE_INTERVAL: 15
  ...
  statefulset:
  ...
    containers:
    ...
      monitor:
        name: "{{$.Chart.Name }}-monitor"
        ...
        metrics:
          portName: "metrics"
```

```
                    containerPort: "9100"

                  health:
                    portName: "health"
                    containerPort: "9101"
                ...
```

2. If Prometheus Operator is configured in your Kubernetes environment, then you can configure Prometheus using the parameters **PodMonitor** and **PrometheusRule** in your **values.yaml** file. When configuring **interval** and **scrapeTimeout** for the health port, keep in mind that health check usually takes around a minute.
   For example, set the following values:

```
raa:
  ...
  podMonitor:
    name : "{{ tpl $.Values.raa.serviceName $ }}-monitor"
    podMetricsEndpoints:
      - port: "{{ $.Values.raa.statefulset.containers.monitor.metrics.portName }}"
        path: "/{{ tpl $.Values.raa.statefulset.containers.monitor.metrics.portName
$ }}"
        interval: "{{ $.Values.raa.env.STAT_SCRAPE_INTERVAL }}s"
      - port: "{{ tpl $.Values.raa.statefulset.containers.monitor.health.portName $
}}"
        path: "/{{ tpl $.Values.raa.statefulset.containers.monitor.health.portName $
}}"
        interval: 4m
        scrapeTimeout: 3m
    podTargetLabels:
      - service
      - servicename

  prometheusRule:
    name : "{{ tpl $.Values.raa.serviceName $ }}-alerts"
    alerts:
      health:
        for: "30m"
        labels:
          service: "{{ $.Values.raa.namespace }}"
          component: "{{ $.Chart.Name }}"
          release: "{{ tpl $.Values.raa.serviceName $ }}"
          severity: "severe"
        annotations:
          summary: RAA is unhealthy for more than hour
          description: |-
            Pod {{ "{{ $labels.pod }}" }} reports about unhealthy RAA for more than
hour already.
      error:
        labels:
          service: "{{ $.Values.raa.namespace }}"
          component: "{{ $.Chart.Name }}"
          release: "{{ tpl $.Values.raa.serviceName $ }}"
          severity: "warning"
        annotations:
          summary: RAA error detected
          description: |-
            Pod {{ "{{ $labels.pod }}" }} reports about new RAA errors detected.
      longAggregation:
        thresholdSec: 300
        labels:
          service: "{{ $.Values.raa.namespace }}"
          component: "{{ $.Chart.Name }}"
          release: "{{ tpl $.Values.raa.serviceName $ }}"
```

```
          severity: "warning"
        annotations:
          summary:  Aggregation query is pretty slow
          description: |-
            Pod {{ "{{ $labels.pod }}" }} reports that query {{ "{{
$labels.hierarchy }}-{{ $labels.level }}--{{ $labels.mediaType }}" }}
            was running more than {{
$.Values.raa.prometheusRule.alerts.longAggregation.thresholdSec }} seconds.
```

Monitoring summary

Service	CRD or annotations?	Port	Endpoint/ Selector	Metrics update interval
Genesys CX Insights	ServiceMonitor	8180	See selector details on the Genesys CX Insights metrics and alerts page	15 minutes
Reporting and Analytics Aggregates	PodMonitor and PrometheusRule	metrics: 9100, health: 9101	See selector details on the Reporting and Analytics Aggregates metrics and alerts page	metrics: several seconds, health: up to 3 minutes

Configure metrics

The metrics that are made available by these services are available by default. No further configuration is required to define or make available these metrics. You cannot define your own custom metrics.

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 Customer Experience Insights alerts, see:

- Genesys CX Insights alerts
- Reporting and Analytics Aggregates alerts

Configure alerts

Private edition services define a number of alerts by default (for Genesys Customer Experience Insights, 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 Customer Experience Insights does not support customizing the alerts or defining additional **PrometheusRule** objects to create alerts based on the service-provided metrics.

Set the severity levels and some thresholds for alerts by specifying various **raa.prometheusRule.alerts.*** parameters in the **values.yaml** file. See RAA alerts for details.

Logging

For more information, see Logging

GCXI metrics and alerts

Contents

- 1 Metrics
- 2 Alerts

Find the metrics GCXI exposes and the alerts defined for GCXI.

Related documentation:

-

Service	CRD or annotations?	Port	Endpoint/Selector	Metrics update interval

GCXI	ServiceMonitor	8180	`selector:`
			`  matchLabels:`
			`  gcxi/deployment-code: gcxi{{ template "deploymentCode" . }}`
				15 minutes

See details about:

- GCXI metrics
- GCXI alerts

Metrics

Metric and description	Metric details	Indicator of
gcxi__projects__info General information about currently deployed GCXI projects. The labels of this metric provide important information. The metric value itself is always **1**, and does not provide any information.	**Unit:** **Type:** Gauge **Label:** ***host** = hostname of the container • **tenant_id** = tenant ID • **gcxi_active** = is project 'CX Insights' active • **gcxi_version** = project 'CX Insights' version • **iwd_active** = is project 'CX Insights for iWD' active • **iwd_version** = project 'CX Insights for iWD' version Important This metric considers projects to be active only if deployed and	Health check

Metric and description	Metric details	Indicator of
	configured so that reports in the project are expected to return data. In scenarios where only the CX Insights project or only the CX Insights for iWD project is deployed, reports in non-active projects are not expected to work and this metric does not reflect them in monitoring results. Default xxx_active values: • **true** for gcxi • **false** for iwd **Sample value:** gcxi__projects__info{host="gcxi-1-22-30",tenant_id="22-30",gcxi_active="true",gcxi_version="9.0.018.00 1 1614350887562	
gcxi__projects__status Indicates whether all active projects are healthy. Values: • **0** = all active projects are healthy. • nonzero = at least one active project failed.	**Unit:** **Type:** Gauge **Label:** * **host** = hostname of the container • **tenant_id** = tenant ID • **gcxi_tested** = was 'CX Insights' health status accounted in the overall health value?** • **iwd_tested** = was 'CX Insights for iWD' health status accounted in the overall health value?** The xxx_tested label is logically related to the xxx_active label. Important This metric considers projects to be active only if deployed and configured so that reports in the project are expected to return data. In scenarios where only the CX Insights project or only the CX Insights for iWD project is deployed, reports in non-active projects are not expected to work and this metric does not reflect them in monitoring results. While xxx_tested and xxx_active are implemented as separate variables, for purposes of monitoring they can be considered the same. **Sample value:** gcxi__projects__status{host="gcxi-1-22-30",tenant_id="22-30",gcxi_tested="true",iwd_tested="false"} 0 1614350887562	Health check

Metric and description	Metric details	Indicator of
gcxi__projects__gcxi__status Indicates whether the CX Insights project is healthy. Values: • **0** = project ok • **1** = url not available • **2** = test login failed • **3** = test run of the report failed (unable to parse expected pattern in response)	**Unit:** **Type:** Guage **Label:** • **host** = hostname of the container • **tenant_id** = tenant ID • **pattern** = the text pattern in the test report response to parse out • **project** = project name • **report** = test report name • **tested** = whether the CX Insights project is active and tested. This value matches the xxx_tested values in the metric gcxi__projects__status. **Sample value:** gcxi__projects__gcxi__status{host="gcxi-1-22-30",tenant_id="22-30",pattern="Genesys Info Mart",project="CX Insights",report="CX Insights Schema Version",tested="true"} 0 1614350887562	Health check
gcxi__projects__iwd__status Indicates whether the CX Insights for iWD project is healthy. Values: • **0** = project ok • **1** = url not available • **2** = test login failed • **3** = test report run failed (unable to parse expected pattern in response)	**Unit:** **Type:** Gauge **Label:** • **host** = hostname of the container • **tenant_id** = tenant ID • **pattern** = the text pattern in the test report response to parse out • **project** = project name • **report** = test report name • **tested** = whether the CX Insights for iWDproject active / tested. Matches the xxx_tested values in the metric gcxi__projects__status. **Sample value:** gcxi__projects__iwd__status{host="gcxi-1-22-30",tenant_id="22-30",pattern="com/genesys/iwd/dm/etl",project="CX Insights for iWD",report="iWD DB Schema Report",tested="false"} 0 1614350887562	Health check

Metric and description	Metric details	Indicator of
gcxi__cluster__info The number of nodes in the MicroStrategy cluster. Value: Either **1** or **2**, depending on configuration.	**Unit:** **Type:** Guage **Label:** ***host** = hostname of the container • **tenant_id** = tenant id (string) • **nodes** = current members of the cluster (hostnames) **Sample value:** gcxi__cluster__info{host="gcxi-1-22-30",tenant_id="22-30",nodes="gcxi-0-22-30,gcxi-1-22-30"} 2 1614350887562	Health check
MicroStrategy Performance Counters GCXI exports some MicroStrategy performance metrics (sometimes called Performance Counters). Not served by HTTP endpoint. For more information about performance metrics, see Performance Counters for Specific MicroStrategy Features in the MicroStrategy documentation.	**Unit:** **Type:** Counter **Label:** **Sample value:**	

Alerts

The following alerts are defined for GCXI.

Alert	Severity	Description	Based on	Threshold
gcxi__projects__status		If the value of cxi__projects__status is greater than 0, this alarm is set, indicating that reporting is not functioning properly.	cxi__projects__status	
gcxi__cluster__info		This alert indicates problems with the cluster states. Applicable only if you have two or more nodes in a cluster.	gcxi__cluster__info	

RAA metrics and alerts

Contents

- 1 Metrics
- 2 Alerts

Find the metrics RAA exposes and the alerts defined for RAA.

Related documentation:

-

Service	CRD or annotations?	Port	Endpoint/Selector	Metrics update interval

RAA	PodMonitor and PrometheusRule	metrics: 9100, health: 9101	RAA forms matched labels from raa.statefulset.selector.matchLabels values, specified in values.yaml.
			The default contains a single raa-app item with raa.serviceName variable as a value.
			The element raa.serviceName is a concatenation of parameters.
			
			`....`
			`  statefulset:`
			`    ## pod selector`
			`    selector:`
			
			`matchLabels:`
			`          raa-app: "{{ tpl $.Values.raa.serviceName $ }}"`
			
			`    template:`
			`      ## a map of pod specific labels to add to common labels`
			`      labels:`
			`          raa-app: "{{ tpl $.Values.raa.serviceName $ }}"`
				metrics: several seconds,
				health: up to 3 minutes

See details about:

- RAA metrics
- RAA alerts

Metrics

Metric and description	Metric details	Indicator of

gcxi_raa_health_level
A health status metric extracted using a dedicated port on the monitor container. The metric value is a sum of values from two different health-checks:
• A database-based health-check. Results:
  • A result of **2** indicates that RAA is working or in maintenance (has received a STOP command from Genesys Info Mart, and will restart only after receiving the START command).
  • A result of **0** indicates that RAA is not working according to this check.
• A local health-check.
  • A result of **1** indicates that RAA is processing aggregation requests according to local health files.
  • A result of **0** indicates that RAA is not processing aggregation requests according to local health files.
	Unit:
	Type: Gauge
	Label:
	Sample value: 0
		Health check

gcxi_raa_command_count The number of commands received from Genesys Info mart since the previous scrape. Label reflects the name of the command. The supported commands are: START, QUIT, EXIT, UPDATE_CONFIG, REAGGREGATE	**Unit:** **Type:** Counter **Label:** cmd **Sample value:** 10	Traffic
gcxi_raa_dispatch_count The number of dispatch events (moving aggregation requests from AGR_NOTIFICATION to PENDING_ARG) since the previous scrape. Dispatch events typically occur every 15 seconds. Such events are used for aggregation health check based on local files.	**Unit:** **Type:** Counter **Label:** **Sample value:** 100	Health check
gcxi_raa_heartbeat_count	**Unit:**	Health check

Metric and description	Metric details	Indicator of
The number of heartbeats since the previous scrape. Heartbeat is normally performed once every five minutes, and is used for health check based on local files. The label is the current RAA version.	**Type:** Counter **Label:** version **Sample value:** 10	
gcxi_raa_relaunched_count The number of times RAA was relaunched since the previous scrape. The aggregation process can exit when an error occurs. Genesys Info Mart sends a START command every 15 minutes during the aggregation period, which causes RAA to relaunch.	**Unit:** **Type:** Counter **Label:** **Sample value:** 1	Error
gcxi_raa_launched_count The number of times RAA launched since the previous scrape.	**Unit:** **Type:** Counter **Label:** version **Sample value:** 1	Error
gcxi_raa_error_count The number of errors registered since the previous scrape.	**Unit:** **Type:** Counter **Label:** **Sample value:** 1	Error
gcxi_raa_notification_count The number of fact change notifications received from Genesys Info Mart since the previous scrape.	**Unit:** **Type:** Counter **Label:** fact **Sample value:** 10	Latency
gcxi_raa_notification_period_ms The total amount of time attributed to changed fact periods in notifications received from Genesys Info Mart since the previous scrape.	**Unit:** milliseconds **Type:** Counter **Label:** fact **Sample value:**	Latency
gcxi_raa_notification_delay_ms The total amount of time attributed to fact notification delays in notifications received from Genesys Info Mart since the previous scrape. Notification delay is calculated as the difference between the moment of notification and the start of the changed period.	**Unit:** milliseconds **Type:** Counter **Label:** fact **Sample value:**	Latency
gcxi_raa_aggregated_count The number of aggregations completed by RAA since the previous scrape. RAA groups the data by aggregation hierarchy name, materialized level (usually SUBHOUR, HOUR, DAY, MONTH), and media type (Online, Offline).	**Unit:** **Type:** Counter **Label:** hierarchy, level, mediaType **Sample value:**	Traffic
gcxi_raa_aggregated_period_ms	**Unit:** milliseconds	Traffic

Metric and description	Metric details	Indicator of
The total number of periods aggregated by RAA since the previous scrape.	**Type:** Counter **Label:** hierarchy, level, mediaType **Sample value:** 10	
gcxi_raa_aggregated_duration_ms The total duration of time periods aggregations completed by RAA since the previous scrape.	**Unit:** milliseconds **Type:** Counter **Label:** hierarchy, level, mediaType **Sample value:**	Traffic
gcxi_raa_aggregated_delay_ms The total duration of delays for aggregations completed by RAA since the previous scrape. Aggregation delay is calculated as the difference between the moment aggregation competes, and the start of the aggregation range.	**Unit:** milliseconds **Type:** Counter **Label:** hierarchy, level, mediaType **Sample value:**	Latency
gcxi_raa_purged_count The number of records purged by RAA since the previous scrape. RAA groups the data by purged table name.	**Unit:** seconds **Type:** Counter **Label:** table **Sample value:**	Traffic
gcxi_raa_purged_duration_ms The total amount of time spent on purging since the previous scrape.	**Unit:** milliseconds **Type:** Counter **Label:** table **Sample value:**	Traffic

Alerts

Various *raa.prometheusRule.alerts.** parameters in the **values.yaml** file specify the severity of alerts and some thresholds.

The following alerts are defined for RAA.

Alert	Severity	Description	Based on	Threshold
raa-health	**Specified by**: raa.prometheusRule.alerts.raa-health.labels.severity **Recommended value:** severe	A zero value for a recent period (several scrape intervals) indicates that RAA is not operating.	gcxi_raa_health_level	Specified by: raa.prometheusRule.alerts.health. **Recommended value**: 30m
raa-errors	**Specified by**: raa.prometheusRule.alerts.raa-errors.labels.severity in values.yaml. **Recommended value**: warning	A nonzero value indicates that errors have been logged during the scrape interval.	gcxi_raa_error_count	>0

Alert	Severity	Description	Based on	Threshold
raa-long-aggregation	**Specified by**: raa.prometheusRule.alerts.longAggregation.labels.severity in values.yaml. **Recommended value**: warning	Indicates that the average duration of aggregation queries specified by the hierarchy, level, and mediaType labels is greater than the deadlock-threshold.	gcxi_raa_aggregated_duration_ms / gcxi_raa_aggregated_interval	Greater than the value (seconds) of raa.prometheusRule.alerts.longAg duration in values.yaml. **Recommended value**: 300

Logging

Contents

- 1 GCXI logging
  - 1.1 Log Level
  - 1.2 Redirect to stdout
  - 1.3 Configure the Fluent Bit version
  - 1.4 Log Retention
- 2 RAA logging
  - 2.1 Retrieving Logs

Learn how to access logs for Genesys Customer Experience Insights (GCXI).

Related documentation:

-
-
-
-

RSS:

- For private edition

GCXI logging

GCXI writes logs in the following locations:

- **pod stdout** -- General GCXI logs, which you can use for troubleshooting.
- **log** folder, mounted at `/mnt/log` -- MicroStrategy logs, which MicroStrategy support sometimes requests.

Log Level

Configure the following log-related container variables if necessary:

- **DEV_XTRACE_LEVEL** — Enables / disables Bash xtrace level in container stdout. It accepts the following values:
  - **OFF** = bash xtrace off (default)
  - **DEBUG** = partial bash xtrace with concealed secrets (sometimes needed for troubleshooting).
  - **ALL** = full bash xtrace. Caution: everything, including passwords, is visible in the console log. This log level can be necessary in some troubleshooting scenarios.
- **DEV_ERR_EXIT** — Enables / disables container exit on error. This variable is not directly related to logging, but is sometimes useful for troubleshooting. It accepts the following values:
  - **true** / **false** (default is true).

Redirect to stdout

You can optionally redirect logs to stdout (with Fluent Bit sidecar and No Shared Log Volumes). Use this option to view logs that are not sent to console output by default. To enable this option, set the following parameters in **values.yaml**:

```
#deploy fluentbit sidecar to redirect file logs to stdout?
```

```
deployFluentbitSidecar: true
# use emptyDir volume for logs?
useEmptyDirLogPV: true
```

Configure the Fluent Bit version

You can optionally set the Fluent Bit version to match the one available in your environment. To modify the default image location and version, set the following parameters in **values.yaml**:

```
images:
   # full image notation, consumed as is
   fluentbit: fluent/fluent-bit:1.8
```

Log Retention

Many GCXI logs are written as files to a local volume (for example, by MicroStrategy, GCXI Utils, or Tomcat). GCXI does not clean up these files; Genesys recommends that you configure appropriate log retention policies.

RAA logging

RAA writes logs to console output by default, which provides integration with tools such as Fluent Bit, FluentD, LogStash, or New Relic collector. The default Log level is .:INFO.

Retrieving Logs

Optionally, you can:

- Change the level of log by setting the **raa.env.LEVEL_OF_LOG** Helm value.
- Write logs to a file inside the container by overriding the **raa.env.LOG** value.

For example:

```
raa:
  ...
  env:
    ...
    LOG: "/genesys/raa_health/aggregator.log"
    LEVEL_OF_LOG: ".:FINEST"
  ...
```

For more information, see the option level-of-log, and the Logging overview and approaches documentation.