Genesys Customer Experience Insights Private Edition Guide

Configure RAA

8/2/2025

Contents

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
```