This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Genesys Info Mart Private Edition Guide

Configure GSP

2/28/2024

Contents

Learn how to configure GIM Stream Processor (GSP).

Related documentation:

-
-
-
-

RSS:

- For private edition

GSP Helm chart overrides

The GSP requires some configuration for deployment that must be made by modifying the GSP's default Helm chart. You do this by creating override entries in the GSP's **values.yaml** file.

Download the GSP Helm charts from your image registry, using the appropriate credentials.

For information about how to download the Helm charts, see Downloading your Genesys Multicloud CX containers. To find the correct Helm chart version for your release, see Helm charts and containers for Genesys Info Mart for the Helm chart version you must download for your release. For general information about Helm chart overrides, see Overriding Helm chart values in the *Genesys Multicloud CX Private Edition Guide*.

At minimum, you must create entries in the **values.yaml** file to specify key system information, as described in the following sections.

> **Important**
>
> Treat your modified **values.yaml** file as source code, which you are responsible to maintain so that your overrides are preserved and available for reuse when you upgrade.

Image registry and pull secret

Image registry

Create an entry in the GSP's **values.yaml** file to specify the location of your image registry. This is

the repository from which Kubernetes pulls images.

The location of the image registry is defined when you set up the environment for the GSP. It is represented in the system as the `docker-registry`. In the GSP Helm chart, the repository is represented as `image: registry`, as shown in the following example. You can optionally set a container version for the image, as in the following example.

```
image: # The repository from which Kubernetes pulls images
  registry:   # The default registry is pureengage-docker-staging.jfrog.io
  tag:  # The container image tag/version
```

Pull secret

When you set up your environment, you define a pull secret for the image registry (`docker-registry`). You must include the pull secret in the GSP's **values.yaml** file in order for Kubernetes to be able to pull from the repository.

```
imagePullSecrets:
  docker-registry: {} # The credentials Kubernetes will use to pull the image from the
registry
```

Note that other services use a different syntax than this to configure the repository pull secret, as follows:

```
imagePullSecrets:
  name: docker-registry
```

Genesys Info Mart, GIM Stream Processor, and GIM Configuration Adaptor helm charts all support advanced templating that allow the helm to create the pull secret automatically; hence the variation in syntax.

Kafka

Kafka secret

If Kafka is configured with authentication, you must configure the Kafka secret so GSP can access Kafka. The Kafka secret is provisioned in the system as `kafka-secrets` when you set up the environment for GSP. Configure the Kafka secret by creating a Helm chart override in the **values.yaml** file.

```
kafka:
  password:  # Credentials for accessing Kafka. This secret is created during deployment.
```

Kafka bootstrap

To allow the Kafka service on GSP to align with the infrastructure Kafka service, make a Helm override entry with the location of the Kafka bootstrap.

```
kafka:
```

```
  bootstrap:  # The Kafka address to align with the infrastructure Kafka
```

Custom Kafka topic names

Some of the Kafka topics used by the GSP support customizing the topic name. If any topic name has been customized, ensure it is represented as a Helm chart override entry, using the `kafka:topic` parameter.

For a list of the Kafka topics that GSP produces and consumes, including which of those support customized naming, see Before you begin GSP deployment.

S3-compatible storage

S3 storage credentials

When you set up the environment for GSP, you provision S3-compatible object storage for GSP to use as a persistent data store. In the **values.yaml** file, record the credentials needed by GSP to access this storage.

```
gsp-s3:  # Credentials for accessing S3-compatible storage
```

Enable S3-compatible storage

When you set up your environment for the GSP, you provision S3-compatible object storage for the GSP's persistent data store. You must enable this storage with override entries in the **values.yaml** file.

By default, GSP is configured to use Azure Blob Storage as the persistent data store. If you have provisioned Azure Blob Storage in your deployment, modify the following entries in the **values.yaml** file:

```
job:
  storage:
    gspPrefix:  # The URI path prefix under which GSP savepoints, checkpoints, and high
availability data will be stored
    gcaSnapshots:  # The URI path under which GCA snapshots will be stored
```

To enable other types of S3-compatible storage, modify the following entries in the **values.yaml** file:

```
azure:
  enabled: false
job:
  storage:
    gspPrefix:  # The bucket where GSP savepoints, checkpoints, and high-availability data
will be stored
    gcaSnapshots:  # The bucket where the GCA snapshots will be stored.
    s3:  # The applicable details defined for the OBC or GCP bucket.
```

Note: The `host` parameter is ignored.

GKE example

```
azure:
  enabled: false
...
job:
  storage:
    host: gspstate{{.Values.short_location}}{{.Values.environment}}.blob.core.windows.net
    #gspPrefix: wasbs://gsp-state@{{ tpl .Values.job.storage.host . }}/{{ .Release.Name }}/
    gspPrefix: "s3p://test-example-bucket-one/{{ .Release.Name
}}/"
    #gcaSnapshots: wasbs://gca@{{ tpl .Values.job.storage.host . }}/
    gcaSnapshots: "s3p://test-example-bucket-one/
gca/"
    checkpoints: '{{ tpl .Values.job.storage.gspPrefix . }}checkpoints'
    savepoints: '{{ tpl .Values.job.storage.gspPrefix . }}savepoints'
    highAvailability: '{{ tpl .Values.job.storage.gspPrefix . }}ha'
    s3:
      endpoint: "https://storage.googleapis.com:443"
      accessKey: ""
      secretKey: ""
      pathStyleAccess: "true"
```

Kubernetes API

GSP uses Apache Flink for stateful stream processing, with communications handled via the Kubernetes API. To use the Kubernetes API, GSP must have the permissions shown in the following example:

Verbs	On Resource	API Group	Comment
get list watch delete	jobs	batch	GSP uses these commands during upgrade and for a pre-upgrade hook to ensure that the previous version of GSP is stopped before upgrading to the new version.
create update patch get list watch delete	configmap	general ("")	GSP uses these commands to: • Support Flink's Kubernetes high availability (HA) services • Record the path to the savepoint (periodically taken by the **take-savepoint** cron job and by the upgrade

Verbs	On Resource	API Group	Comment
			hook)

Configure GSP behavior

You can specify values in the **values.yaml** file to override the default values of configuration options that control GSP behavior and to customize user data and Outbound field mappings.

You can override aspects of the default configuration to modify GSP behavior and customize the way data is stored in the Info Mart database.

You can customize the following:

- Configuration options, to modify data-related aspects of GSP behavior
- User data mappings, to map custom key-value pairs (KVPs), which are attached to event and reporting protocol data, to tables and columns in the Info Mart database
- Outbound record field mappings, to map custom CX Contact record field data to Outbound-related tables and columns in the Info Mart database

Customize configuration option settings

For full information about the options you can configure, including the default and valid values, see GSP configuration options.

To configure options, edit the GSP **values.yaml** file. Under the **cfgOptions** object, specify the option and value in JSON format, noting the following:

- Options are separately configurable by tenant and, where applicable, by media type or even at the level of individual queues (DNs or scripts).
- Where an option can be configured at various levels, you can override a value set at a higher level (for example, for a particular media type in general) to set a different value for a particular lower-level object (for example, for that media type for an individual DN).
- See the note about configuration levels for information about the available configuration levels for certain options.

The entries in the **values.yaml** file are structured as follows:

```
cfgOptions:
    "": |
      standard:

:
      media:
        :

:

:
```

```
        :

:

:
      dn:
        :
          media:
            :

:
      script:
        :
          media:
            :

:
      ...
```

For an example, see example.

Customize user data mapping

Genesys Info Mart uses a wide range of user-data KVPs from a number of upstream services to populate data in the Info Mart database. For full information about user data in Genesys Info Mart, see [[PEC-REP/Current/GIMPEGuide/UserData|]].

As described on [link TBD], you can extend storage of user data in the Info Mart database to include additional user-data KVPs you want to capture as custom user-data facts or dimensions.

To configure user-data mapping, edit the GSP **values.yaml** file. Under the **udeMapping** object , specify the mapping between your custom KVPs and the custom user-data database table(s) and column(s), noting the following:

- The mapping, which is specified in JSON format, is configured separately by tenant.
- In addition to specifying the database table and column in which to store the KVP value, you also specify the *propagation rule* that Genesys Info Mart uses to determine what value to store if more than one value is extracted for the same key in the same interaction. See Propagation rules for more information.
- You can specify a default value to use if the KVP value is null. You must provide a default value for dimensions.
- You map custom user-data dimensions via the IRF_USER_DATA_KEYS table, where you specify the foreign key reference(s) for the user-data dimension table(s).

Genesys Info Mart provides the following sample tables in the Info Mart database schema for storage of custom KVPs:

- USER_DATA_CUST_DIM_1 — for low-cardinality user data to be stored as dimensions
- IRF_USER_DATA_CUST_1 — for high-cardinality user data to be stored as facts

The structure of the entries to specify in the **values.yaml** file is:

```
udeMapping:
  ": |
    IRF_USER_DATA_KEYS:
      columns:
        :
          type: dim
          table:
          attributes:
            :
              kvp:
              rule:
              default:
            ...
    :
      columns:
        :
          type: value
          kvp:
          rule:
          default:
        ...
```

See the example.

Customize Outbound field mapping

Genesys Info Mart stores data about every outbound contact attempt, based on Record Field data it receives from the CX Contact (CXC) service. As described on [[PEC-REP/Current/GIMPEGuide/Outbound|]], some of the mapping between Field data and the Info Mart database tables and columns is predefined, and some is custom.

To customize outbound mapping, edit the GSP values.yaml file. Under the **ocsMapping** object, specify the mapping between your custom record fields and the tables and columns provided in the Info Mart database for custom record field data, namely:

- In the CONTACT_ATTEMPT_FACT table:
  - 10 floating-point numbers: `numeric(14,4)`
  - 20 integers: `integer`
  - 30 strings: `varchar(255)`
- In the RECORD_FIELD_GROUP_1 and RECORD_FIELD_GROUP_2 tables:
  - 10 strings each: `varchar(255)`

The mapping, which is specified in JSON format, is configured separately by tenant. The structure of the entries is:

```
ocsMapping:
  "": |
    CONTACT_ATTEMPT_FACT:
      columns:
        RECORD_FIELD_1:
          field:
          rpcValue:
          conversionValue:
        ...
        RECORD_FIELD_11:
```

```
          field:
          rpcValue:
          conversionValue:
        ...
        RECORD_FIELD_31:
          field:
          rpcValue:
          conversionValue:
        ...
    RECORD_FIELD_GROUP_1:
      columns:
        RECORD_FIELD_1_STRING_1:
          field:
          rpcValue:
          conversionValue:
        ...
    RECORD_FIELD_GROUP_2:
      columns:
        RECORD_FIELD_2_STRING_1:
          field:
          rpcValue:
          conversionValue:
        ...
```

See the example.

Example

```
cfgOptions:
    "eb0c9f3a-5dca-498f-98d5-7610f5fd1015": |
      standard:
        completed-queues: iWD_Completed,iWD_Processed_ext
        populate-workbin-as-hold: false
      media:
        voice:
          q-answer-threshold: 30
      media:
        email:
          q-short-abandoned-threshold: 20
      dn:
        eb0c9f3a-5dca-498f-98d5-7610f5fd1015_MM_VQ:
          media:
            email:
              q-answer-threshold: 90
              q-short-abandoned-threshold: 40
      script:
        eb0c9f3a-5dca-498f-98d5-7610f5fd1015_ixn_queue:
          media:
            chat:
              q-answer-threshold: 20
      script:
        eb0c9f3a-5dca-498f-98d5-7610f5fd1015_dev.IQ:
          standard:
            populate-ixnqueue-facts: false

udeMapping:
  "eb0c9f3a-5dca-498f-98d5-7610f5fd1015": |
    IRF_USER_DATA_KEYS:
      columns:
        CUSTOM_KEY_1:
          type: dim
```

```
        table: USER_DATA_CUST_DIM_1
        attributes:
          DIM_ATTRIBUTE_1:
            kvp: BusinessLine
            rule: CALL
            default: retail
          DIM_ATTRIBUTE_2:
            kvp: RULE_PARTY
            rule: PARTY
            default: none
          DIM_ATTRIBUTE_3:
            kvp: RULE_IRF
            rule: IRF
            default: none
          DIM_ATTRIBUTE_4:
            kvp: RULE_IRF_INITIAL
            rule: IRF_INITIAL
            default: none
          DIM_ATTRIBUTE_5:
            kvp: RULE_IRF_FIRST_UPDATE
            rule: IRF_FIRST_UPDATE
            default: none
    IRF_USER_DATA_CUST_1:
      columns:
        CUSTOM_DATA_1:
          type: value
          kvp: RULE_CALL
          rule: CALL
          default: none
        CUSTOM_DATA_2:
          type: value
          kvp: RULE_PARTY
          rule: PARTY
          default: none
        CUSTOM_DATA_3:
          type: value
          kvp: RULE_IRF
          rule: IRF
        CUSTOM_DATA_4:
          type: value
          kvp: RULE_IRF_FIRST_UPDATE
          rule: IRF_FIRST_UPDATE
        CUSTOM_DATA_5:
          type: value
          kvp: RULE_IRF_INITIAL
          rule: IRF_INITIAL
        CUSTOM_DATA_6:
          type: value
          kvp: RULE_IRF_ROUTE
          rule: IRF_ROUTE
    IRF_USER_DATA_GEN_1:
      columns:
        CASE_ID:
          type: value
          kvp: CaseID
          rule: Call
          default: none
        CUSTOMER_ID:
          type: value
          kvp: GIM_GRP
          rule: Call
          default: none
```

```
ocsMapping:
  "eb0c9f3a-5dca-498f-98d5-7610f5fd1015": |
    CONTACT_ATTEMPT_FACT:
      columns:
        RECORD_FIELD_1:
          field: GenRecordField1
          rpcValue:
          conversionValue:
        RECORD_FIELD_2:
          field: GenRecordField2
          rpcValue:
          conversionValue:
        ...
        RECORD_FIELD_60:
          field: GenRecordField60
          rpcValue:
          conversionValue:
    RECORD_FIELD_GROUP_1:
      columns:
        RECORD_FIELD_1_STRING_1:
          field: GenRecordField1String1
          rpcValue:
          conversionValue:
        RECORD_FIELD_1_STRING_2:
          field: GenRecordField1String2
          rpcValue:
          conversionValue:
        RECORD_FIELD_1_STRING_3:
        ...
    RECORD_FIELD_GROUP_2:
      columns:
        RECORD_FIELD_2_STRING_1:
          field: GenRecordField2String1
          rpcValue:
          conversionValue:
        RECORD_FIELD_2_STRING_2:
        ...
```