Genesys Info Mart Private Edition Guide

Configure GIM

7/27/2024

Contents

Learn how to configure GIM.

Related documentation:

-
-
-
-

RSS:

- For private edition

GIM Helm chart overrides

Genesys Info Mart requires some configuration for deployment that you can make only by modifying the Helm chart, which you do by creating override entries in the GIM **values.yaml** file.

Download the **gim** and **gim-monitoring** Helm charts from your image repository, using the appropriate credentials.

To learn 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 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 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 repository and pull secret

Image registry

To specify the location of the image registry, create an entry in the GIM **values.yaml** file. This is the

repository from which Kubernetes will pull images.

The location of the image registry is defined when you set up the environment for the GIM, and is represented in the system as the `docker-registry`. In the GIM 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.

```
image: # The repository from which Kubernetes will pull 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 image registry (`docker-registry`). You must include the pull secret in the GIM **values.yaml** file 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
```

Other services use a different syntax to configure the repository pull secret, as follows:

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

Genesys Info Mart, GIM Stream Processor, and GIM Configuration Adapter 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 the GIM service can access Kafka. The Kafka secret is provisioned in the system as `kafka-secrets` when you set up the environment for Info Mart. 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 Info Mart 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 GIM 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.

Data export and S3-compatible storage

If the Genesys Info Mart Data Export feature is part of your deployment, you can export your data to S3-compatible object storage.

You provision the storage when you set up the environment for Genesys Info Mart, and make override entries in the **values.yaml** file to enable the storage.

```
s3_storage_enabled: true
s3_storage:
  account:
    accessKey:  # The access key created when you created the storage bucket
    secretKey:  # The secret created when you created the bucket
    region: # The region in which the bucket was created
    entryPoint: # The URL for accessing bucket storage
gim_export:
  output_directory: # The bucket name
```

The `s3_storage` parameters are used to construct the **s3_storage_secrets** secret.

GKE example

```
gim_export:
  ...
  output_directory:                       "test-example-bucket-one"
  ...
s3_storage_enabled: true
s3_storage:
  account:
    accessKey: ""
    secretKey: ""
    region: ""
    entrypoint: "storage.googleapis.com"
```

Configure GIM behavior

You can specify values in the **values.yaml** file to control options that override aspects of the default configuration, thereby modifying GIM behavior and customizing the way data is stored in the Info Mart database. For information about the options you can configure including the default and valid values, see GIM configuration options.

To configure options, edit the GIM **values.yaml** file. Under the **gim_config** object in the, 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:

```
gim_config: ""

log:
  level:                                "info"
  console_pattern_layout:               "%d{ISO8601} %-5p %-12t %m%n"
  appender:
    ConsoleLogger:
      Threshold:                         "info"

gim_etl:
  days_to_keep_active_facts:             "27"
  days_to_keep_deleted_annex:            "2"
  days_to_keep_discards_and_job_history: "60"
  days_to_keep_gidb_facts:               "27"
  days_to_keep_gim_facts:                "400"
  etl_start_date:                        ""
  max_chunks_per_job:                    "10"
  max_time_deviation:                    "30"
  memory_threshold:                      "0"
  partitioning_ahead_range:              "31"
  partitioning_interval_size_gidb:       "604800"
  partitioning_interval_size_gidb_mm:    "604800"
  partitioning_interval_size_gidb_ocs:   "604800"
  partitioning_interval_size_gim:        "2592000"
  purge_thread_pool_size:                "32"
  purge_transaction_size:                "100000"

date_time:
  date_time_max_days_ahead:              "400"
  date_time_min_days_ahead:              "183"
  date_time_start_year:                  "2020"
  date_time_tz:                          "GMT"
  first_day_of_week:                     "1"
  fiscal_year_start:                     ""
  fiscal_year_end:                       ""
  fiscal_year_week_pattern:              "none"
  min_days_in_first_week:                "1"
  simple_week_numbering:                 "true"

schedule:
  aggregate_duration:                    "23:00"
  aggregate_schedule:                    "30 0"
  etl_end_time:                          "23:30"
  etl_frequency:                         "1"
  etl_start_time:                        "00:00"
  export_schedule:                       "0/30 *"
  maintain_start_time:                   "23:40"
  run_aggregates:                        "true"
```

```
  run_export:                           "true"
  run_maintain:                         "true"
  run_scheduler:                        "true"
  run_update_stats:                     "true"
  on_demand_migration:                  "true"
  timezone:                             "UTC"
  update_stats_schedule:                "0/10 *"

gim_export:
  chunk_size_seconds:                   "86400"
  days_to_keep_output_files:            "30"
  max_retries:                          "3"
  output_directory:                     "gim-export"
  output_files_encoding:                "utf8"
  retry_delay_seconds:                  "30"
  start_date:                           ""
  thread_pool_size:                     "10"
  use_export_views:                     "true"
```

Custom calendars

GIM permits you to create custom calendars by creating a section in the values.yaml file that similar to the date-time section, and has the same options; name the section by using the *date-time-* prefix:

- **Job_InitializeGIM** populates data in all configured calendars when it initializes the Info Mart database.
- **Job_MaintainGIM** then maintains the calendars in accordance with options that are specified in the [date-time] and custom [date-time-*] configuration sections. The maintenance job automatically adjusts for special requirements such as daylight saving time (DST) and fiscal years that do not start on the same day every year (floating fiscal years).

Consider the settings for the **date-time** options carefully before the calendar dimension tables are populated for the first time. You can subsequently change the values of the **date-time-min-days-ahead** and **date-time-max-days-ahead** options at any time.

However, changing any of the other **date-time** options during runtime can introduce inconsistencies into the calendar data and affect reporting results adversely. For example, if you change the timezone option (**date-time-tz**) after Genesys Info Mart has been initialized, your reports can mix the results for different time zones within the same reporting interval.

Config Maps

Helm creates a number of Config Maps based on option values you specify in the **values.yaml** file (see Configure GIM Behavior). There are no Config Maps you can configure directly for Info Mart.