Reporting and Analytics Aggregates User's Guide

How do I manage the aggregation process?

4/15/2025

How do I manage the aggregation process?

This page describes how to manage the aggregation process apart from Genesys Info Mart Manager. To learn how to manage the aggregation process within the Genesys Info Mart Manager—operating in Integrated mode—refer to the *Genesys Info Mart Operations Guide*.

Should I Run the Aggregation Process?

Reporting and Analytics Aggregates (RAA) provides an aggregation engine that must be run in order to populate the aggregation tables on which Genesys CX Insights (GCXI) relies. *However*, if your deployment does not include GCXI, running this process is optional, unless you have some other reason to require aggregated data.

Tip

Running the aggregation process creates the AG2_* / AGT_* (and supporting) tables and database constructs—if they do not already exist—within Info Mart database.

Configuring continuous aggregation

You can invoke aggregation so that it will run on an ongoing basis.

Genesys recommends that you save the aggregation command in the Genesys Info Mart root directory (though it is possible to execute it from another location). All instructions in this document pertaining to invoking this process presume that the command is within the Genesys Info Mart root directory.

Running Continuous Aggregation

To invoke aggregation in Integrated mode, refer to the discussion about **Job_AggregateGIM** in the *Genesys Info Mart Operations Guide*.

To invoke the aggregation process in Autonomous mode from the Genesys Info Mart root directory and have it run continuously until it is stopped, open a console window, and then issue the appropriate command:

- **On UNIX Platforms:** from the Genesys Info Mart root directory, issue the following command:

```
java -jar agg/GIMAgg.jar -user=<dbo> -pass=<password> -jdbcurl=<URL>
        [OtherParams]
```

- **On Microsoft Windows platforms**: from the Genesys Info Mart root directory, issue the following command:

```
java -jar agg\GIMAgg.jar -user=<dbo> -pass=<password> -jdbcurl=<URL>
        [OtherParams]
```

- where:
- `GIMAgg.jar` is the name of the Java archive that contains the aggregation engine.

  and `user`, `pass`, and `jdbcurl` are mandatory runtime parameters:
- <dbo> is the user name of the database owner.
- <`password`> is the password of the database owner.
- <URL> is the jdbc URL, including the host, port, system ID (SID, for Oracle), and database name (for Microsoft SQL).
- `[OtherParams]` are optional runtime parameters that you can specify to affect aggregation results.

Refer to the *Reporting and Analytics Aggregates Deployment Guide* for descriptions of all runtime parameters and command-line syntax.

Using the -conf Runtime Parameter

Optionally, you can use the **-conf** runtime parameter to reference a configuration file that stores one or more runtime parameters. If you specify this one parameter, you don't have to specify every other parameter on the command line.

To invoke a continuous aggregation using this parameter, issue the following command from the Genesys Info Mart root directory:
`java -jar agg/GIMAgg.jar -conf <file>`
where `<file>` is the name of the file that contains the full listing of runtime parameters that are not specified at the command line. This specification must include the file's absolute path if the file is not located in the same directory as the aggregation java archive.

Aggregation Examples

The following are examples of how to invoke aggregation in Autonomous mode for two different platforms:

Oracle on UNIX:	`java -jar agg/GIMAgg.jar -user=Administrator -pass=Adm1n2046 -jdbcurl=jdbc:oracle:thin:@whale:1521:orcl -levelOfLog=.AGG:FINE`
Microsoft SQL Server on Microsoft Windows:	`java -jar agg\GIMAgg.jar -user=dbo -pass=Adm1n2046 -jdbcurl=jdbc:jtds:sqlserver://octopus:1433;DatabaseName=Widg -levelOfLog=.:FINE`

Before it runs, the aggregation engine checks to see whether another aggregation process is already running. If there is one running, instead of starting a new process, the aggregation engine logs an error similar to the following:

```
Failed to acquire lock ... ... Unable to get aggregation lock
```

Reaggregating data

You can submit a request in Autonomous mode for certain data chunks to be queued for aggregation.

Reaggregating Data over a Certain Time Range

To reaggregate data over a specified time range, add the **-insertPendingAgg** runtime parameter to the command line. This command does not invoke aggregation. Instead, the queued request is processed if aggregation is already running (in either Integrated or Autonomous mode) or the next time that the process is started.
The following formats are supported:

```
java -jar agg\GIMAgg.jar -user=<dbo> -pass=<password> -jdbcurl=<URL>
        -insertPendingAgg <AGR_SET>:<START>:<END>
```

OR

```
java -jar agg\GIMAgg.jar -user=<dbo> -pass=<password> -jdbcurl=<URL>
        -insertPendingAgg <AGR_SET>:DATES_FROM:<FACT_TABLE>
```

Where:

- <AGR_SET> indicates what set to aggregate (ALLSETS, or an aggregate set name). Aggregate set name is formatted as follows:

  ```
  <HIERARCHY_NAME>-<AGG_LEVEL>[.Flavor].
  ```

  Where:

  - <HIERARCHY_NAME> is the name of the hierarchy to be aggregated.
  - <AGG_LEVEL> is the aggregation level (SUBHOUR, HOUR, DAY, MONTH, QUARTER, YEAR).
  - `[.Flavor]` indicates what data to include (Online or Offline).
- <START> is a value in the format YYYY-MM-DD
- <END> is a value in the format YYYY-MM-DD
- <FACT TABLE> is the name of the fact table from which to retrieve start and end time values. The start and end values are retrieved from the MIN and MAX values of the START_DATE_TIME_KEY field in the specified fact table.

Examples:

- To reaggregate all available data for a one-year period:

```
java -jar GIMAgg.jar -conf XXX.properties -insertPendingAgg ALLSETS:2017-01-01:2017-12-31
```

- To reaggregate a particular aggregate set for a particular day:

```
java -jar GIMAgg.jar -conf XXX.properties -insertPendingAgg H_ID-DAY:2017-01-01:2017-01-01
```

- To reaggregate all available data over a period corresponding to the MIN and MAX values form START_DATE_TIME_KEY field of the INTERACTION_RESOURCE_FACT table :

```
java -jar GIMAgg.jar -conf XXX.properties -insertPendingAgg
ALLSETS:DATES_FROM:INTERACTION_RESOURCE_FACT
```

- To reaggregate only one flavor (online- or offline- data):

```
java -jar GIMAgg.jar -conf XXX.properties -insertPendingAgg H_I_AGENT-
SUBHOUR.Online:2017-01-01:2017-01-01
```

> **Important**
> A request to reaggregate data for a specific time range first deletes aggregated data from that time range (to prevent duplicate data from being written to Info Mart). Before you issue such a request, make sure that the Genesys Info Mart facts for your selected time range exist and have not been purged. Otherwise, you could be left with no data at all for that time range.

Reaggregating Data when DATE_TIME Changes

When you first initialize the Genesys Info Mart database, set a time zone for the DATE_TIME table, and avoid changing it thereafter. If your environment time zone changes after data has been aggregated and written to the Info Mart database, then in order to maintain consistency (between data written to Info Mart before and after the time zone change) within the aggregate tables, you must truncate existing data from all AGT_* tables before requesting reaggregation. This procedure is described in the *Changing DATE_TIME Options* section of the *Reporting and Analytics Aggregates Deployment Guide*. Note that the time range for reaggregation must include the beginning of the month in which the time-zone switch occurred plus two more days as illustrated in the following example:

Example: On January 10, 2015, you change the time zone in your contact center environment from Eastern Standard Time to Pacific Standard Time. To set reaggregation to repopulate AGT_* tables properly, you must set the reaggregation interval from December 30, 2014 to January 11, 2015, inclusive. This is in addition to executing all of the requisite steps in Genesys Info Mart to effect the time-zone change.

Checking the health of the aggregate process

RAA includes a *healthCheck* tool, which you can use to check the state of aggregation, based on the working directory associated with the process running aggregation (Genesys Info Mart or standalone RAA).
For example:

```
# java -jar ./GIMAgg.jar -healthCheck <work dir>
```

Where <work dir> is the directory path, for example:

- Typical Linux deployments where RAA is running from Genesys Info Mart use this path: `/usr/local/genesys/gim`
- Typical Linux deployments where RAA is running standalone use this path: `/usr/local/genesys/raa`

If it finds any problem, the healthCheck tool outputs an error code indicating the type of error, and prints details about the problem to the console.

The tool checks the following files:

- **.agglaunched** — created at launch of aggregation process (inside GIM or standalone).
- **.aggrelaunched** — created at relaunched of aggregation process (inside GIM only).
- **.aggerror** — created when error happens.
- **.aggheartbeat** — created during each writers hear-beat event (once per 5 minutes).
- **.aggdispatch** — created during each dispatch event (once per 15 seconds).

Genesys recommends that you do not move, delete, or edit these files manually.

The healthCheck tool also accepts database parameters, allowing the tool to check a dispatcher of a local or remote aggregation by database.
For example:

```
# java -jar ./GIMAgg.jar -conf=agg.properties -levelOfLog=.:SEVERE -healthCheck
```

Stopping the aggregation process

You may have to perform multiple steps to stop the aggregation process, depending on whether it is scheduled.

To stop aggregation

Job schedules are controlled by the values of options in the **[schedule]** configuration section of the Genesys Info Mart Application object. If aggregation is scheduled:

- To halt aggregation, you must explicitly reset or stop the job schedule. If you attempt to stop aggregation manually within the Genesys Info Mart Manager only, and without changing the job schedule, the Genesys Info Mart Server automatically restarts the job so long as it is scheduled to be running.
- Conversely, if the job is scheduled to be not running, the Genesys Info Mart Server will not permit you to start the job from the Genesys Info Mart Manager.
  The method you use to stop the aggregation process depends on the mode of its operation:
  - **Stopping Aggregation in Integrated Mode**: Once started in Integrated mode, the aggregation process runs continuously, aggregating new facts until the process (**Job_AggregateGIM**) is

scheduled to be stopped. Refer to the *Genesys Info Mart Deployment Guide* or the *Genesys Info Mart Deployment Procedure* for more information about the options pertinent to aggregation, and the procedure for stopping aggregation that is operating in Integrated mode.

- **Stopping Aggregation in Autonomous Mode**: Although you can manually stop the aggregation process when it is operating in Autonomous mode, the Genesys Info Mart Server will restart the process in Integrated mode if the process is scheduled to be running. Therefore, to stop the aggregation process:
  - Deactivate the job schedule as described in the *Genesys Info Mart Deployment Guide*.
  - Stop the aggregation job by typing `^C` within the console window in which aggregation was invoked, or by stopping the aggregation process.

Purging aggregate data

RAA provides the ability to purge aggregate tables on a scheduled basis. To use this functionality, you must configure purging rules, and schedule and enable purging:

Configuring Purging Rules

You create purging rules in a file (**purge.ss**), which you must store in the folder where aggregation or Genesys Info Mart is running. Each line of the **purge.ss** file must consist of a single purge rule, consisting of the purge statement with the following syntax:
`(purge <agg-set-selector> keep <value> <unit> delay <value> <unit>)`

RAA evaluates the rules as follows:

- Rules are evaluated in order, beginning from the top of the file.
- If more than one rule applies to an aggregate set, the first rule that matches is the effective purging rule.
- If more than one aggregate set shares storage (such as sub-hour level aggregates for compatible time zones), and are impacted by separate purging rules, the most conservative rule is used to purge the shared storage. If there is at least one such aggregate set which matches no purging rule, the shared storage is not purged.
- Purging rules are evaluated in the time zone of the hierarchy to which the aggregate sets belong.
- Purging rules are applicable only to aggregates that are stored as tables (where AGT tables exist). For aggregates that are defined only as Views, purging rules do not apply and are ignored. Aggregates that are defined only as Views inherit their purging from underlying aggregates that are stored as tables.

The table *Purging Rule Parameters* describes the purging rules and parameters.

Purging Rule Parameters

Parameter	Description
<agg-set-selector>	This parameter specifies the hierarchy or query to purge, the aggregate level to urge, and (optionally) the flavor of media to purge, with the following syntax:

	`<HIERARCHY|QUERY>-<AGG_LEVEL>[.Flavor]` where: `<HIERARCHY|QUERY>` - the hierarchy or query to purge. Accepts the wildcard *: enter * to purge all heirarchies or queues, or *-* to purge all heirarchies and all queues. Usage Examples: • `H_QUEUE-HOUR` - selects hourly aggregation level from the H_QUEUE hierarchy. • `H_GMT_QUEUE-HOUR` - selects hourly aggregation level from the H_GMT_QUEUE hierarchy (if H_GMT_QUEUE exists). • `QUEUE-HOUR` - selects hourly aggregation levels from both hierarchies. • <AGG_LEVEL> - the aggregation level to purge (SUBHOUR, HOUR, DAY, WEEK, MONTH, QUARTER, YEAR). Enter * to purge all aggregation levels. • `[.Flavor]` - the flavor of media to purge: • Online to purge only data from online media (such as voice call or chat) • Offline to purge only data from offline media (such as email or tasks). To purge both online and offline data, exclude this attribute. Note that flavor applies to all aggregates, with the exception of `SESS_STATE`, `STATE_RSN`, `AGENT_CAMPAIGN`, and `CAMPAIGN` aggregates (which are said to be *plain*, or *flavorless*).
keep <value> <unit>	This parameter specifies the number of complete units of time for which to retain data, where: • `<value>` - an integer • `<unit> - DAY, WEEK, MONTH, QUARTER, or YEAR` The delay parameter must have a value lower than that of the keep parameter.
delay <value> <unit>	This parameter specifies the period of time to wait before purging, where: • `<value>` - an integer • `<unit> - HOUR, DAY, WEEK, MONTH, QUARTER, or YEAR` The `delay` parameter must have a value lower than that of the keep parameter.

Examples

The following examples show how you can use use purge rules in various ways:

Rule	Result
`(purge H_ID-* keep 1 YEAR delay 7 DAY)`	In this rule, the 1 year *keep* value causes purge to retain all data in the H_ID hierarchy for the entire year (January 1st to December 31st). The *delay* value works as follows: • If evaluated during the first week of 2018, retain data from 2016-01-01 to 2017-12-31; (two years of data, because the 7 day delay after the end of the keep period has not yet passed). • If evaluated after January 8th of 2018, retain data from 2017-01-01 to 2017-12-31; (one year of data, the minimum specified by the keep parameter, because the 7 day delay after the end of the keep period has passed).
`(purge H_ID-* keep 365 DAY delay 0 HOUR)`	This rule retains 365 days of all data in H_ID hierarchy, aligned to DAY boundaries. Delay is as near zero as possible. So, if evaluated on January 5th of 2018, this rule retains all data from 2017-01-05 to 2018-01-04
`(purge *-*.Online keep 3 WEEK delay 1 DAY)`	The rule retains at least 3 whole weeks of online data in all hierarchies. If evaluated on the first day of the week, then four weeks of data is retained.
`(purge QUEUE-*.Offline keep 3 YEAR delay 1 MONTH)`	This rule keeps 3 whole years of offline data in all hierarchies based on QUEUE query. If evaluated before February 1st in a given year, then four years of data is retained.

Enabling and Scheduling Purge

To enable purging, you must add the suffix <hour(X-Y)=purge> to the **writerSchedule** option of the aggregation command, and set the purging schedule using the hour parameter. For example, to enable purging, and schedule it to occur between 1 am and 2 am:

```
writerSchedule=default=flex(7:3),hour(1-2)=purge
```

Additionally, purging can occur only if aggregation is scheduled to be is active during the time you schedule for purging so the time period specified for **aggregate-schedule** in the aggregate-schedule option must overlap with the time period specified for purging in writer-schedule. For more information about scheduling purging, see *Reporting and Analytics Aggregates Deployment Guide*.

Configuring aggregation across more than one time zone

Genesys Info Mart enables you to create custom calendars for dimensioning data. You can configure RAA to recognize these calendars and aggregate data accordingly.

A standard Genesys Info Mart deployment using the default **DATE_TIME** calendar yields reporting in the Genesys Info Mart default time zone only. There are, however, other supported deployments allowing:

- One tenant reporting across multiple time zones
- Multiple tenant reporting within one common time zone
- Multiple tenant reporting using a different time zone for each tenant

To configure RAA to recognize custom calendars, four general steps are required:

- Declare each calendar to RAA.
- Run aggregation.
- Create schemas within Info Mart to restrict user access to the appropriate data.
- Update tenant aliases.

The following procedure provides detailed steps for a common scenario: configuring Genesys Info Mart to aggregate data across multiple time zones in scenarios where Genesys CX Insights is configured with one project, and multiple connections:

Recognizing Custom Calendars

1. Configure additional calendars in Genesys Info Mart; for example, **DATE_TIME_CNT** and **DATE_TIME_AET**. See Creating Custom Calendars section in the *Genesys Info Mart Deployment Guide* for further instructions. Note that **DATE_TIME** tables for the base and tenant schema must be initialized such that both start from the same year. To specify the calendar start year, set the Genesys Info Mart **date-time-start-year** option in **date-time** and **date-time-<tz>** sections (where **date-time-<tz>** is a section created for a configured time zone, for example, **date-time-aet**).
2. Identify the created calendars to RAA:
    a. Create an ASCII file that contains the following code (substitute the AET and CNT time zones and their offsets with your desired time zone(s) and their corresponding offsets):

```
;This code identifies time zones to RAA
(~time-zone CNT "DATE_TIME_CNT" -12600 -9000)
(~time-zone AET "DATE_TIME_AET" +36000 +39600)
;This code instructs RAA to use the AET time zone when
;populating data for only those aggregation hierarchies that
;are listed
(add-other-tz AET
   (hierarchies: H_AGENT H_QUEUE
      H_AGENT_QUEUE H_QUEUE_ACC_AGENT H_QUEUE_ABN H_ID
      H_I_AGENT H_I_SESS_STATE H_I_STATE_RSN
      H_AGENT_CAMPAIGN H_CAMPAIGN))
;This code make all hierarchies CNT-time zone aware
(add-standard-hierarchies-in-tz CNT)
```

   b. Save this file in the Genesys Info Mart root folder with the file name `time-zones.ss`. The aggregation process must be able to locate this file from the location where aggregation is run.

3. Invoke aggregation. RAA creates a separate set of database objects for each calendar and names the objects with the time zone's abbreviation (AG2_AET_AGENT_SUBHR). RAA manages these objects within the main schema.
4. Create a schema in Genesys Info Mart for each tenant. Users should not directly reference objects in the main schema, so you must create aliases to control the access that users have to Info Mart data.
5. Create an alias file, (for example: `tenant-tz-alias.ss`), following the instructions in "Format of the Tenant Alias File" in the *Reporting and Analytics Aggregates User's Guide*. For example:

```
(aliases-for-account name: <tenant1_schema_name> login: "<tenant1_user>" password:
 "<tenant1_pwd>"
 (tenants: <tenant1_key>) (time-zone: PST))
 (aliases-for-account name: <tenant2_schema_name> login: "<tenant2_user>" password:
 "<tenant2_pwd>"
 (tenants: all) (time-zone: EST))
```

6. To update tenant aliases, invoke aggregation in standalone mode by issuing the following command:
   `java -jar agg/GIMAgg.jar -conf runagg -updateAliases tenant-tz-alias.ss`
   RAA creates views in the specified schema(s) that employ standard names. Therefore, no change to the definitions of metrics, attributes, or conditions is required in the GCXI project. Each tenant account now sees data in their own time zone.
7. Create connection parameters for each tenant account.