Reporting and Analytics Aggregates User's Guide

RAA 9.0.0

2/8/2022

Table of Contents

Reporting and Analytics Aggregates 9.0 User’s Guide

Welcome to the *Reporting and Analytics Aggregates User's Guide*. This document introduces you to Reporting and Analytics Aggregates (RAA), which is the aggregation layer of Genesys Info Mart—how it functions, how to invoke and stop it, how to configure custom user data, and how to troubleshoot it. This guide is valid only for the 9.0.x releases of RAA.

About Reporting and Analytics Aggregates

RAA 9.0 provides the mechanism for creating, maintaining, and populating a subset of tables and views in a Genesys Info Mart 8.5 database that provide aggregated data of contact center operations for reporting and analytical purposes. This aggregation layer is both an optional component of the Genesys Info Mart 8.5 product and a necessary and transparent component of Genesys Customer Experience Insights (GCXI) 9.0.

New In This Release

This section describes the changes that have been incorporated within this guide since the 9.0.0 release of RAA.

For information about what's new in the *software*, see the *Reporting and Analytics Aggregates Release Notes*.

9.0.001.03

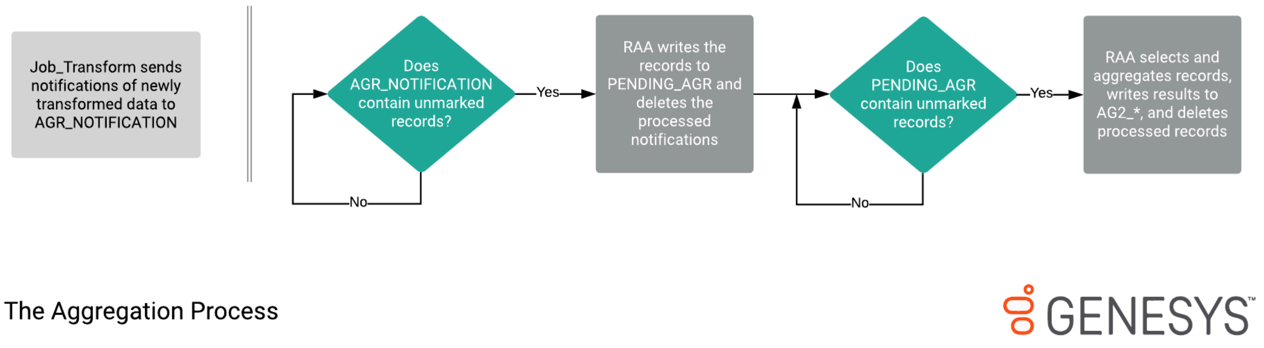
This is the initial 9.0 release of this document.

What is Aggregation and How Do I Enable It?

This page describes how to enable Reporting and Analytics Aggregates (RAA) and how it interfaces with the Genesys Info Mart Server to aggregate contact center data for reporting purposes. The RAA option of Genesys Info Mart must be installed before aggregation can occur. Unlike most other Genesys products though, RAA is not configured within the Genesys Configuration Server. Refer to the *Reporting and Analytics Aggregates Deployment Guide* for installation instructions.

How Does The Aggregation Process Work?

A Genesys Info Mart job, **Job_TransformGIM**, is responsible for sending notifications about newly transformed factual data that is ready for aggregation. If this job is not running, the Genesys Info Mart Server cannot send notifications. As part of its extraction, transformation, and loading (ETL) cycle and before it commits updates, the Genesys Info Mart Server determines the start and end DATE_TIME keys for which the updates apply and sends notification of the updates along with the corresponding time range to an intermediate queue—a table named AGR_NOTIFICATION. After some asynchronous processing, the aggregation interface timestamps and grabs these notifications and writes them to the PENDING_AGR internal queue. The aggregation process is illustrated in the Figure, **The Aggregation Process**.


Job_Transform sends notifications of newly transformed data to AGR_NOTIFICATION
Does AGR_NOTIFICATION contain unmarked records?
Yes
No
RAA writes the records to PENDING_AGR and deletes the processed notifications
Does PENDING_AGR contain unmarked records?
Yes
No
RAA selects and aggregates records, writes results to AG2_*, and deletes processed records
GENESYS

The Aggregation Process

After processing the notifications, the aggregation engine aggregates low-level details (sourced from Info Mart *_FACT tables) and writes the results to aggregate tables (with the prefix AGT_*) in the Info Mart database, and presented through views with the prefix AG2_*. When the aggregation engine is enabled, it constantly polls both AGR_NOTIFICATION and PENDING_AGR for newly added notifications. In this fashion, aggregated data becomes available for reporting in near-real time.

How Do I Enable Aggregation?

Because RAA is an optional component of Genesys Info Mart 8.5, aggregation is not automatically enabled. To start aggregation, you must enable the aggregation engine, and invoke the aggregation process.

To enable the aggregation engine and cause the Genesys Info Mart Server to start sending notifications to the internal queue, you must assign the aggregation class within the Genesys Info Mart application by setting the aggregation-engine-class-name configuration option (described in the "Disable Aggregation in the GIM Application" section of the *Reporting and Analytics Aggregates Deployment Guide*).

Invoke the aggregation process in one of two modes, which launches the activity that routinely creates, modifies, and updates the aggregation tables:

- **Autonomous mode**—Requiring no direct involvement with the Genesys Info Mart Server.
  Invoking the aggregation process in Autonomous mode runs the aggregation engine stand-alone, from the command line, without referencing configuration settings of the Genesys Info Mart Application object in Configuration Server or invoking the **Job_AggregateGIM** job (invoking this job is equivalent to operating aggregation in Integrated mode). In fact, the Genesys Info Mart Server need not even be running or sending notifications to the AGR_NOTIFICATION queue—although this mode of operation is not particularly useful for capturing ongoing contact center activity. Operating RAA in Autonomous mode, however, is recommended for those circumstances where you want to reaggregate data (see Reaggregating Data over a Certain Time Range) or when you want to migrate 7.6 data to 8.5/9.0 (described in the *Genesys Migration Guide*). To learn how to invoke aggregation in Autonomous mode, see How Do I Configure Continuous Aggregation?.
- **Integrated mode**—Where the Genesys Info Mart Server drives aggregation activity.
  Operating the aggregation process in Integrated mode relies on Genesys Info Mart internal processes to manage all aspects of aggregation. The aggregation engine is driven by **Job_AggregateGIM**, a job that is managed by the Genesys Info Mart Manager. With respect to aggregation, this console respects the values of aggregation-related configuration options that are defined in the Genesys Info Mart Application object. Other Genesys Info Mart jobs, including **Job_MaintainGIM**, are also managed using Genesys Info Mart Manager; however, this user interface is beyond the scope of this document. Refer to the *Genesys Info Mart Operations Guide* to learn how to start, stop, and manage jobs.

In What Order is Data Aggregated?

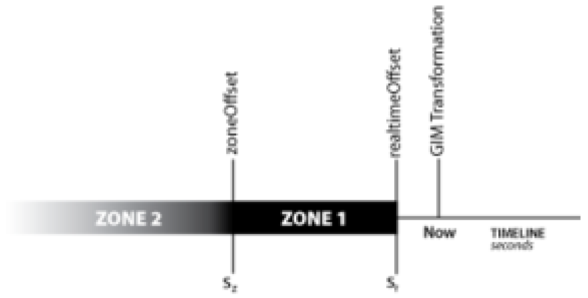
The RAA 9.0 aggregation engine processes chunks of data according to priority, using a simple distribution algorithm that processes chunks in the order in which they appear in the PENDING_AGR queue (beginning with the oldest data), and without giving priority to any particular hierarchy or aggregation level. (Hierarchies and aggregation levels are described in What is an Aggregation Hierarchy?)

RAA 9.0 determines priority by the time at which RAA registers notifications. Older notifications receive higher priority than newer notifications and are generally processed before newer notifications are processed for a given aggregate set. RAA rounds notification timestamps to nearest 15 minutes. For purposes of priority comparison, these are grouped into larger units. For instance, data about which notifications were registered between 2–4 hours ago are considered to have higher priority than data notifications that were registered between 0–2 hours ago.

RAA 9.0 attributes all data to one of two sliding zones:

- **Zone 1** contains notifications about more recent pending aggregation requests;
- **Zone 2** contains notifications about all other older data.

You configure the boundary that divides the zones by setting the **zone-offset** option in the Genesys Info Mart Application object for aggregation run in Integrated mode, or by issuing the zoneOffset runtime parameter for aggregation run in Autonomous mode. (The *Reporting and Analytics Aggregates Deployment Guide* describes both the option and the runtime parameter. How Do I Manage the Aggregation Process? describes both modes of running aggregation.) Use the **realtime-offset** option (**realtimeOffset** runtime parameter) to delay RAA processing of notifications that are received up to two hours after Genesys Info Mart transformation. The Figure **Two Zones Contain All Notifications** illustrates the two zones and their boundaries.



Two Zones Contain All Notifications

RAA 9.0 is able to process data concurrently in both zones. Data within each zone is processed in order of priority.

The number of writers allocated across the two zones establishes which zone is dominant and which is recessive. When you assign more writers to Zone 1, it becomes the dominant zone and Zone 2 becomes the recessive zone. When there is an idle writer (one that has just finished processing a chunk of data, for example), RAA first tries to allocate it to the dominant zone. If the allocation fails, RAA allocates the idle writer to the recessive zone. Depending on the degree of pliability (strict or flex), RAA will borrow writers, as necessary, between zones in order to handle the workload of the zone with the higher resource demands. The **writer-schedule** configuration option (**writerSchedule** runtime parameter) defines how many writers are initially allocated to each zone for a particular set of hours. Refer to the *Genesys Interactive Insights Deployment Guide* for more information.

For any chunk of data, the aggregation engine performs aggregations first for the lowest table node of the aggregation hierarchy—the *_SUBHR or *_HOUR level depending on the model (for a discussion of models, see What is an Aggregation Hierarchy?)—then moves up the line to the highest table node of the hierarchy—the *_MONTH level. As aggregation gets propagated up a particular aggregation hierarchy, all aggregation levels within that hierarchy inherit the priority of the original notification. The higher nodes (quarter and year levels) are views that are based on the *_MONTH tables, so no further aggregation processing is necessary beyond the month level. When the aggregation process completes aggregating data at certain stages, it deletes the corresponding row(s) from the PENDING_AGR table as part of the transaction and then commits the transaction.

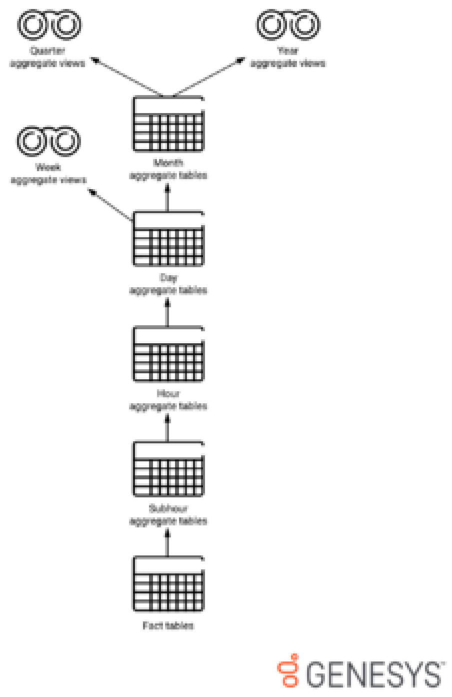
Note that data for all aggregation levels might not be available at the same point in time. The aggregation engine might propagate data to higher aggregation nodes in different transactions. For a short period of time, it is possible for users to be able to retrieve data from one aggregation level that is not available at the higher nodes. This momentary discrepancy is most evident with the subhour views and hour tables of the Disposition-Based Measures model when newly transformed data has not yet been aggregated. The subhour views retrieve data directly from the source Genesys Info Mart *_FACT tables and immediately reflect this newly transformed data in detail-level reports. Minutes could pass, however, before this data would be reflected in the hour and higher level aggregate tables.

Also playing a role to a lesser extent, RAA might process more than one data chunk simultaneously depending on how many threads it is instructed to use. This instruction comes from the value of the **write-schedule** configuration option or the **writerSchedule** runtime parameter.

When are Notifications Sent?

For voice interactions, the Genesys Info Mart Server sends notifications about completed interactions only. Stuck calls are not eligible for aggregation until they are cleared from queue; Genesys Info Mart does not extract from ICON information about active, in-process calls. For multimedia interactions that originate from an Interaction Server, **Job_TransformGIM** transmits notifications about active interactions as well as completed interactions.

What is an Aggregation Hierarchy?



The Aggregation Hierarchy

The hierarchies that are present in the aggregation layer are based on time—ranging from subhour table or view constructs to year views of contact center data.

Factual data (for example, from the MEDIATION_SEGMENT_FACT table) is gathered and grouped by a specific time interval (as well as by other dimensions) and then written to a time-based aggregate table or made available via aggregate views that are based on these FACT tables and lower-level aggregation tables.

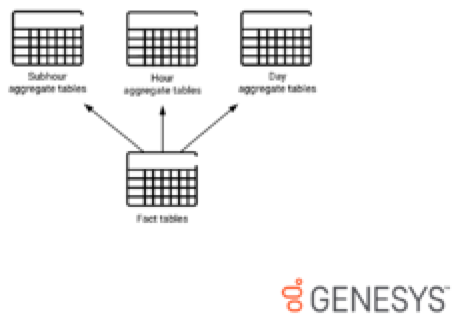
This page explains the hierarchies of Reporting and Analytics Aggregates (RAA):

Aggregation Intervals

RAA provides aggregation tables or views for the following seven time intervals: Subhour, Hour, Day, Week, Month, Quarter, Year

The definition of subhour is configurable using the **sub-hour-interval** Genesys Info Mart configuration option at either 15 minutes or 30 minutes. Each level derives its data by aggregating values from the preceding node in the hierarchy, as shown in the Figure **The Aggregation Hierarchy**.

The fact tables at the bottom of the Figure **The Aggregation Hierarchy** store the individual interaction-, session-, or state-related details. Fact tables are populated and maintained outside of RAA by **Job_LoadGIM**, **Job_MaintainGIM**, and other Genesys Info Mart jobs. The RAA Physical Data Model Documentation for your RDBMS describes the fact table(s) that serve as the source for a particular hierarchy from whence data is aggregated, and the relevant Genesys Info Mart Physical Data Model Documentation describes these fact tables in detail.



Release 7.x Interval-Based Measures Model

Unlike the initial release (shown in the Figure **Release 7.x Interval-Based Measures Model**), the Interval-Based Measures model in 8.x derives its values from the previous node in the hierarchy, instead of directly from the source fact table. This design enhancement puts less strain on database resources by improving the performance of queries run against these tables. The 8.x Interval-Based Measures model also includes week, quarter, and year views for each hierarchy, which the Release 7.6 model did not. The Disposition-Based Measures model is the same as was presented in the initial 7.6 release.

Hierarchies of the Aggregation Layer

RAA defines hierarchies for each set of aggregate tables and views; these hierarchies differ in composition only by their time interval of aggregation. Views AG2_QUEUE_HOUR and AG2_QUEUE_YEAR, for instance, share a hierarchy, whereas the AG2_QUEUE_HOUR and AG2_QUEUE_GRP_HOUR views do not.

Disposition-Measure Hierarchies

The RAA Disposition-Based Measures model consists of ten hierarchies:

- The H_AGENT hierarchy—Consisting of the following views:
    - The AG2_AGENT_SUBHR view
    - The AG2_AGENT_HOUR view
    - The AG2_AGENT_DAY view
    - The AG2_AGENT_WEEK view

    - The AG2_AGENT_MONTH view
    - The AG2_AGENT_QRTR view
    - The AG2_AGENT_YEAR view
- The H_AGENT_GRP hierarchy—Comprising rollups of values from the H_AGENT hierarchy and consisting of the following views:
    - The AG2_AGENT_GRP_SUBHR view
    - The AG2_AGENT_GRP_HOUR view
    - The AG2_AGENT_GRP_DAY view
    - The AG2_AGENT_GRP_WEEK view
    - The AG2_AGENT_GRP_MONTH view
    - The AG2_AGENT_GRP_QRTR view
    - The AG2_AGENT_GRP_YEAR view
- The H_AGENT_QUEUE hierarchy—Following the same pattern of tables and views as the H_AGENT hierarchy.
- The H_AGENT_CAMPAIGN hierarchy—Following the same pattern of tables and views as the H_AGENT hierarchy.
- The H_CAMPAIGN hierarchy—Following the same pattern of tables and views as the H_AGENT hierarchy.
- The H_ID hierarchy—Following the same pattern of tables and views as the H_AGENT hierarchy.
- The H_QUEUE hierarchy—Following the same pattern of tables and views as the H_AGENT hierarchy.
- The H_QUEUE_ABN hierarchy—Following the same pattern of tables and views as the H_AGENT hierarchy.
- The H_QUEUE_ACC_AGENT hierarchy—Following the same pattern of tables and views as the H_AGENT hierarchy.
- The H_QUEUE_GRP hierarchy—Following the same pattern of tables and views as the H_AGENT hierarchy., and comprising rollups of values from the H_QUEUE hierarchy.

Interval-Measure Hierarchies

The RAA Interval-Based Measures model consists of three hierarchies:

- The H_I_AGENT hierarchy—Consisting of the following views:
    - The AG2_I_AGENT_SUBHR view
    - The AG2_I_AGENT_HOUR view
    - The AG2_I_AGENT_DAY view
    - The AG2_I_AGENT_WEEK view
    - The AG2_I_AGENT_MONTH view
    - The AG2_I_AGENT_QRTR view

    - The AG2_I_AGENT_YEAR view

- The H_I_SESS_STATE hierarchy—Following the same pattern of tables and views as the H_I_AGENT hierarchy.
- The H_I_STATE_RSN hierarchy—Following the same pattern of tables and views as the H_I_AGENT hierarchy.

How Hierarchies Are Used Within RAA

The aggregation process references several **select-*.ss** files that are located inside the Java aggregation archive (**lib/meta-8.x.x.jar**). These files were written using the Scheme programming language. Each Scheme file identifies the hierarchy to which the file applies, along with instructions (that is, code) for generating that hierarchy's measures. The aggregation process follows these instructions to populate all of the tables that belong to the hierarchy. For information about editing the Scheme files, see How Do I Customize Queries and Hierarchies?.

Hierarchies are also referenced within the user-data mapping file, **user-data-map.ss**, which you prepare to extend the dimensions by which aggregated data can be partitioned. Refer to How Do I Configure User Data for Aggregation? for information about its use.

RAA also manages a number of parallel hierarchies exclusively for migration purposes. These migration hierarchies are temporary in nature and are not further described in RAA documentation.

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.

What Do I Need to Know About Managing Multi-Tenant Environments?

In environments configured with more than one tenant, the Genesys Info Mart Server enables the use of tenant aliases to control user access to data stored in a single Info Mart database. These aliases include a set of intermediate views (all prefixed AGR_ALIAS_) to the original source tables and views that restrict the data set that is returned to that data which pertains to the tenant only. Because Reporting and Analytics Aggregates (RAA) is an optional Genesys Info Mart component, the mechanism for updating tenant aliases for the aggregate tables is achieved apart from the Genesys Info Mart mechanism for updating tenant views.

This page describes how to run the RAA tenant alias update module.

How Often Should I Update Tenant Aliases?

Update tenant aliases whenever a tenant is added or deleted from Genesys Info Mart configuration or whenever there are changes to RAA queries (such as with the rollout of a hot fix, the inclusion of a query patch as described in How Do I Customize Queries and Hierarchies?, or in a development environment in which you are designing your own reports). If information about a tenant changes—such as the tenant moving to a different account— updating the aliases will also be necessary. Otherwise, existing aliases might become unusable, and the subset of reports (queries) based on the existing tenant views might not retrieve expected data.

After the aggregation process runs for the first time, run an update of tenant aliases. Genesys recommends that you also schedule an update of tenant aliases to occur regularly and automatically (for example, using a batch file that is called by an O/S scheduler). Running the RAA tenant alias update affects views of only those database objects that are controlled by the aggregation layer—namely:

- The AGT_* tables.
- The AGR_* tables (these are internal RAA entities).
- All GCXI–specific tables and views.

Whenever table structures change, you must also update tenant views for the dimension and fact tables (which are referenced by the Genesys CX Insights Detail reports); however, this update is apart from RAA. Refer to the section about creating read-only tenant views in the *Genesys Info Mart Deployment Guide* for further information.

Prerequisites and Logging

The following are the minimum required permissions for the alias (or tenant) account:

Oracle	Microsoft SQL
grant connect to <account> grant create view to <account> grant create table to <account> grant resource to <account>	grant create view to <account> grant create table to <account> The account (database user) must belong to the same database as the Genesys Info Mart account. A schema in the database must exist with the same name as database user. The database user must own the schema.

When the update is run, RAA connects to the specified database and begins creating intermediate views. RAA logs each operation—for example:

```
Creating view <AG2_QUEUE_ACC_AGENT_MONTH> in tenant schema
```

When all views are created, the update drops stale views and then exits. Any errors that the update encounters are also logged.

Reference: Format of the Tenant Alias File

The **-updateAliases** runtime parameter (described in the *Reporting and Analytics Aggregates Deployment Guide*) requires that you specify a file named *AliasFile*, which must contain one line for each tenant alias account, formatted as described below. For any tenant alias not listed in the file, **-updateAliases** clears all views.

Specifying the Genesys Info Mart schema

The **-updateAliases** runtime parameter supports GIM schemas that have a name different from the GIM user name. However, if the GIM schema name differs from GIM user name, and the GIM schema is not either `public` (on PostgreSQL) or `dbo` (on MS SQL), you must complete the following steps:

1. Specify the GIM schema in the *AliasFile*:

   ```
   (gim-user-schema <schema_name>)
   ```

   where ‹`schema_name`› is the name of the GIM schema. This is required when the GIM schema name is not the same as the GIM user name.

2. Before running the -updateAliases tool, ensure that the default schema matches the GIM user in database:

   - **PostgreSQL** — Use the following SQL statement to view the search path:

     ```
     show search_path
     ```

     The GIM schema is expected to be first in the `search_path` variable. If it is not, then execute the following SQL statement:

     ```
     alter role <GIM user> set search_path = <GIM schema>, public;
     ```

   - **MSSQL** — Use the following SQL statement to view the default schema:

     ```
     SELECT SCHEMA_NAME()
     ```

The GIM schema is expected to be the default schema for the GIM user. If its not, then execute the following SQL statement:

```
ALTER USER <GIM user> WITH DEFAULT_SCHEMA = <GIM schema>;
```

3. Specify the GIM schema (`gim-user-schema "<schema_name>"`) in the Tenant Alias File.

Formatting subsequent lines

Format each line, other than the first line, as follows:

```
(aliases-for-account name: <UName> login: "<lname>" password: "<pwd>" (tenant[s]: <ID#>)
(time-zone: <TZcode>))
```

where:

- `‹UName›` is the name of the tenant user account.
- `‹lname›` is the tenant user account (should now be the same as the tenant schema name).
- `‹pwd›` is the password of the tenant user account.
- `‹ID#›` is the ID of the tenant as specified in the TENANT_KEY field of the TENANT Info Mart database table. This value, incidentally, matches the tenant’s DBID in Configuration Server. You can specify more than one tenant for this parameter—for example: `(tenants: 4 5)` or the value all for all tenants.
- `‹TZcode›` is the code for an additional `time-zone`—one other than the Genesys Info Mart default time zone. This code must be declared in the **time-zones.ss** file and placed in the Genesys Info Mart root directory. Note that specifying an additional `time-zone` parameter and setting up the **time-zones.ss** file are optional. See Aggregation in Multiple Time Zones for more information.

Depending on the RDBMS type and the number of tenant accounts that are specified in the alias file, the update completes in a matter of seconds after it has been run.

How Do I Configure User Data for Aggregation?

This page describes how to optionally configure Reporting and Analytics Aggregates (RAA), so that data is aggregated based on user-defined dimensions. Note, however, that aggregated measures (which are dimensioned by some aspects of user data) are already prestructured within those hierarchies that include a key (INTERACTION_ DESCRIPTOR_KEY) to the INTERACTION_DESCRIPTOR Info Mart table. This preconfigured dimension table allows contact center data to be classified by four predefined user-specified business attributes:

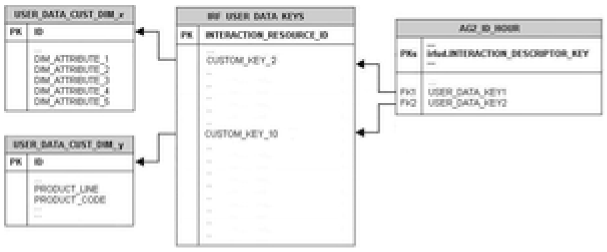
- Business result
- Customer segment
- Service type
- Service subtype

Use the information on this page only if you want to classify and aggregate interactions based on business attributes other than these predefined attributes. For example, based on product/ product line, business importance (Gold Star, Premium, and so on), or tier (such as different technical-support levels of experience).

Overview

The aggregate tables and views of the H_AGENT, H_AGENT_CAMPAIGN, H_AGENT_ QUEUE, H_CAMPAIGN, H_ID, H_QUEUE, H_QUEUE_ABN, H_QUEUE_ACC_AGENT, and H_QUEUE_GRP hierarchies provide two key columns that you can configure to join to two custom user data Info Mart dimension tables of your choice. (Values for these columns in the H_QUEUE_GRP hierarchy are inherited from the H_QUEUE hierarchy). These columns are the following:

- USER_DATA_KEY1—A key that points to one dimension table that stores five dimensions.
- USER_DATA_KEY2—A key that points to another dimension table (or the same table) that stores another five dimensions.



Mapping User Data Keys in the Aggregate Tables/ Views to User Data Dimensions

These two columns provide access to two hierarchies, or a total of 10 attached-data dimensions, as illustrated the Figure **Mapping User Data Keys in the Aggregate Tables/Views to User Data Dimensions**. In this figure, USER_DATA_KEY1 in the H_ID hierarchy of tables and views serves as a foreign key pointing to a mapping key in the IRF_USER_DATA_KEYS table, which joins one-to-one to the primary key in the custom user data table (*x*). USER_DATA_KEY2 is a foreign key to IRF_USER_DATA_KEYS.CUSTOM_KEY_ 10 which joins to the custom user data table (*y*). (*x* and *y* can be the same table.) You can configure the aggregation job to aggregate data based on these dimensions.

Important

These custom user data fields are not available within other hierarchies.

Refer to the deployment guides of the Genesys Info Mart and Interaction Concentrator documentation sets for instructions on how to configure the user-data tables on which these aggregates are based and how to map them so that the Genesys Info Mart Server recognizes user data keys and populates their values along with other pertinent data about interactions.

The User-Data Mapping File

The mapping that defines which user data keys in the aggregate hierarchies point to which custom user data dimensions occurs in a flat file that is named `user-data-map.ss`. Prepare this file and place it in the Genesys Info Mart root directory. The aggregation process recognizes and processes the contents of this file if it is formatted properly:

Format of the user-data-map.ss File

To begin, the `user-data-map.ss` file should contain one—and only one—line for every hierarchy and for each key within that hierarchy that you want to map to a custom user data dimension table. Next, all of the lines in the file should follow this format:

```
(map-user-data-key (hierarchy: HName) (dimension: HCol) (expression: irfud.MappingCol))
```

where:

- HName is the name of the hierarchy, such as H_ID (for a full listing of hierarchies, see What is an Aggregation Hierarchy?).
- `HCol` is the name of the user data key column within that hierarchy. This value is either USER_DATA_KEY1 or USER_DATA_KEY2.
- `MappingCol` is the column name of the key that you configured in the IRF_USER_DATA_KEYS (irfud) table. This table stores mappings to all user data keys.

To map both user data keys in all aggregation tables to custom user data dimensions, you must include independent lines in the user-data-map.ss file—one line for each of the following:

- USER_DATA_KEY1 in the H_AGENT hierarchy
- USER_DATA_KEY2 in the H_AGENT hierarchy
- USER_DATA_KEY1 in the H_AGENT_QUEUE hierarchy
- USER_DATA_KEY2 in the H_AGENT_QUEUE hierarchy
- USER_DATA_KEY1 in the H_CAMPAIGN hierarchy
- USER_DATA_KEY2 in the H_CAMPAIGN hierarchy
- USER_DATA_KEY1 in the H_AGENT_CAMPAIGN hierarchy
- USER_DATA_KEY2 in the H_AGENT_CAMPAIGN hierarchy
- USER_DATA_KEY1 in the H_ID hierarchy
- USER_DATA_KEY2 in the H_ID hierarchy
- USER_DATA_KEY1 in the H_QUEUE hierarchy*
- USER_DATA_KEY2 in the H_QUEUE hierarchy*
- USER_DATA_KEY1 in the H_QUEUE_ABN hierarchy*
- USER_DATA_KEY2 in the H_QUEUE_ABN hierarchy*
- USER_DATA_KEY1 in the H_QUEUE_ACC_AGENT hierarchy*
- USER_DATA_KEY2 in the H_QUEUE_ACC_AGENT hierarchy*

*These user data keys are supported in the H_QUEUE_* hierarchies only in releases that support split-media (online/offline) aggregation, and then only for aggregation tables that were created in RAA release 8.1.400.23 or later.

Note that one user data key maps to the same user data dimension throughout a particular hierarchy. For example, you cannot both map USER_DATA_KEY1 in the AG2_AGENT_MONTH view to `irfud.x` and map USER_DATA_KEY1 in the AG2_AGENT_HOUR view to `irfud.y`. However, you could define these keys differently for different hierarchies (where neither depends on the other.)

Example Mapping File

```
(map-user-data-key (hierarchy: H_ID) (dimension: USER_DATA_KEY1) (expression: irfud.CUSTOM_KEY_q))
(map-user-data-key (hierarchy: H_ID) (dimension: USER_DATA_KEY2) (expression: irfud.CUSTOM_KEY_r))
(map-user-data-key (hierarchy: H_AGENT) (dimension: USER_DATA_KEY1) (expression: irfud.CUSTOM_KEY_s))
(map-user-data-key (hierarchy: H_AGENT) (dimension: USER_DATA_KEY2) (expression: irfud.CUSTOM_KEY_t))
(map-user-data-key (hierarchy: H_AGENT_QUEUE) (dimension: USER_DATA_KEY1) (expression: irfud.CUSTOM_KEY_u))
(map-user-data-key (hierarchy: H_AGENT_QUEUE) (dimension: USER_DATA_KEY2) (expression: irfud.CUSTOM_KEY_v))
(map-user-data-key (hierarchy: H_AGENT_CAMPAIGN) (dimension: USER_DATA_KEY1)
                   (expression: irfud.CUSTOM_KEY_w))
(map-user-data-key (hierarchy: H_AGENT_CAMPAIGN) (dimension: USER_DATA_KEY2)
                   (expression: irfud.CUSTOM_KEY_x))
(map-user-data-key (hierarchy: H_CAMPAIGN) (dimension: USER_DATA_KEY1) (expression: irfud.CUSTOM_KEY_y))
(map-user-data-key (hierarchy: H_CAMPAIGN) (dimension: USER_DATA_KEY2) (expression: irfud.CUSTOM_KEY_z))
```

For other examples that demonstrate attached data configuration starting from Interaction Concentrator (ICON) and extending to GCXI with report customization that provides results dimensioned by your selected user data, see the *Genesys CX Insights User’s Guide*.

If your environment is configured to record social-media data, note that in GCXI, UserDataKey1 is reserved exclusively in the H_ID, H_AGENT, H_AGENT_GRP, and H_AGENT_QUEUE hierarchies for aggregation of social-media data.

How Do I Customize Queries and Hierarchies?

This page demonstrates how to customize the existing Reporting and Analytics Aggregates (RAA) queries and hierarchies to:

- Replace the definitions of existing measures with custom definitions.
- Aggregate additional measures.
- Define other joins to Info Mart tables.
- Add a `WHERE` qualification to RAA queries.
- Rename the aggregation tables.
- Define the aggregation levels that RAA will aggregate.
- Define which aggregation levels should be tables and which should be views within Info Mart.

Building RAA Queries

The RAA aggregation archive (`GIMAgg.jar`) includes several data-definition files that create the aggregation queries—language files that define the identifiers and macros that are used, templates, and other supporting files—all of which have been created using the Scheme programming language (a high-level dialect of the Lisp programming language). To customize aggregation, you must have a clear understanding of Scheme:

Using Scheme to Build RAA Queries

Understand that editing the files in the aggregation archive is not a supported feature. Nonetheless, this page provides some fundamental information on how the RAA queries are written. From there, you can extrapolate how to tailor them—at your own discretion—to better meet a specific business need.

Out of the box, RAA includes the following query families:

• AGENT	• I_AGENT	• SDR_SESS_MILESTONE
• AGENT_GRP	• I_SESS_STATE	• SDR_SESSION
• AGENT_CAMPAIGN	• I_STATE_RSN	• SDR_SESS_BLOCK
• AGENT_QUEUE	• ID	• SDR_SURVEY
• BGS_SESSION	• QUEUE	• SDR_SURVEY_ANS

• CALLBACK • CAMPAIGN • CHAT_STATS	• QUEUE_GRP • QUEUE_ABN • QUEUE_ACC_AGENT	• ID_FCR

These query families correspond directly to the hierarchies that are described in Disposition Measure Hierarchies, and share similar names. (Hierarchies, however, are all prefaced with “H_”.)

Internal coding for each query family further differentiates family members along the seven time intervals (see Aggregation Intervals).

To modify a query’s family, you should be familiar, at minimum, with the query’s top-level definition, which is stored in a `select-*.ss` Scheme file. Depending on the release of RAA installed in your environment, `select-*.ss` is found either inside the aggregation Java archive, or in the `\agg\lib\meta*.jar` file. There is one such file for each query family; for example:

- `select-AGENT.ss` stores the top-level definition for the AGENT query.
- `select-AGENT_CAMPAIGN.ss` stores the top-level definition for the AGENT_CAMPAIGN query.

These files reference macros that are defined in other, lower levels of query definition that reside in language and other ss files. RAA queries are complex.

Customize Aggregation Using the Patch-Aggregation File

For aggregation that operates in integrated mode, you can customize aggregation by creating / editing a Patch-Aggregation file (`patch-agg.ss`), which you must place in the Genesys Info Mart folder in which the `gim_etl_server` batch script is located. For aggregation that operates in autonomous mode, place the `patch-agg.ss` file in the Genesys Info Mart working directory of the java process that runs standalone aggregation—the Genesys Info Mart root directory.

See the following sections for more information:

- Loading the Patch
- Editing the patch-agg.ss file
- Format of the patch-agg.ss file

Loading the patch-agg.ss file

The following procedure describes the steps you must follow to load the patch-agg.ss file.

Procedure: Loading the Patch

Purpose: Use this procedure to load the patch.

Steps

1. Create (or update) the patch file—**patch-agg.ss**— and specify the desired changes to one or more queries. Create the file either in ASCII format, or UTF8, as follows:
   - **UTF-8 format** — UTF-8 specific characters are defined in `patch_agg.ss` (or in another custom schema file). Create or update the file in UTF-8 format and define JVM option `-Dfile.encoding=UTF-8` during Genesys Info Mart start up, as follows:
     - When running RAA in a standalone mode, execute the following command:{

       ```
       java -Dfile.encoding=UTF-8 -jar GIMAgg.jar
       ```

       OR
     - When running with Genesys Info Mart, follow the instructions for modifying the JVM Startup Parameters in the Genesys Info Mart Deployment Guide when adding the `-Dfile.encoding=UTF-8` option, and then start Genesys Info Mart and RAA.
   - **ASCII format** — use the syntax described in Format of the patch-agg.ss file, below.
2. Place the patch file in the Genesys Info Mart root directory.
3. Restart aggregation. (The aggregation engine reads the patch file only upon start.)
4. Use Genesys Info Mart Manager to verify that the aggregation process is running successfully. Alternatively, check for the following message in the Genesys Info Mart log:

   ```
   Agg.Main     processing: patch-agg.ss
   ```

   Tip

   The aggregation process will not run at all if `patch-agg.ss` contains errors.
5. For multi-tenant environments, run an update of tenant aliases as discussed in What Do I need to Know About Managing Multi-Tenant Environments?.

Editing the patch-agg.ss file

In the `patch-agg.ss` file, you specify the query modifications that will override or add to the existing Genesys-provided query definitions. Use the instructions in this patch file to add, customize, or correct metric definitions.

- **Sample Patch** The following code in the `patch-agg.ss` file provides one example that updates the ID query. This example relies on a contact-center configuration that attaches agent cost to each interaction and records this information in the AGENT_COST column of the IRF_USER_DATA_GEN_1 table. AGENT_COST is a custom attached-data measure that you must configure before you can add it to the aggregation query.

```
(alter-query ID
         (add                                    (~metric AGENT_COST
 "max(irfug.AGENT_COST)"))
         (replace                                    (~metric INVITE
 "sum(irf.CUSTOMER_RING_COUNT)"))
         (add                                    (inner-join IRF_USER_DATA_GEN_1 irfug
                                                     "irf.INTERACTION_RESOURCE_ID =
 irfug.INTERACTION_RESOURCE_ID"))
 )
```

This code accomplishes the following:

  - Adds the AGENT_COST metric to the ID query.
  - Replaces the definition of the existing INVITE metric to a new definition that tallies ringing interactions only (instead of ringing and dialing interactions).
  - Adds a join to a user-defined, attached-data Info Mart table that is named IRF_USER_DATA_GEN_1 (from which the AGENT_COST metric is sourced in this example).

When this file is loaded, the aggregation engine will follow these instructions and regularly populate the new and changed columns in the AGT_ID_* group of tables and views, as notifications of newly transformed data are sent to the RAA internal queue.

- **Special Runtime Parameters for Customization** The parameters that are listed in the Table *Special Runtime Parameters for Customization and Debugging* are not described elsewhere within the RAA documentation. They are provided to aid in the construction and debugging of your customized aggregation queries.

Special Runtime Parameters for Customization and Debugging

Runtime Parameter	Description

allowDestructiveDDL	Specifies whether the aggregation engine can use destructive data-definition language (DDL)—such as drop table—to delete and recreate database objects as needed. This parameter uses a date range to limit RAA issuance of destructive operations to two days before and after the date that you specify with this parameter. You must specify the date value in YYYY-MM-DD format.
	For example:
	`java -jar ./agg/GIMAgg.jar`
	`-user=<GIMDBUser> -pass=<passwd>`
	`-jdbcurl=<GIMDBURL>`
	`-allowDestructiveDDL=2013-07-31`
	Dropping and recreating aggregation tables would be

	necessary, for instance, if you customized a hierarchy to add new dimensions in a database created with release 8.1.103 or earlier. Setting this option is unnecessary for standard aggregation operations.

checkQuery	Validates the specified query against Info Mart. The query can be a Genesys-provided query or a custom query.
	
	For example:
	
	`java -jar ./agg/GIMAgg.jar`
	`-user=<GIMDBUser> -pass=<passwd>`
	`-jdbcurl=<GIMDBURL>`
	`-checkQuery AGENT_GRP`

evaluateFile	Evaluates the specified Scheme file within the context of aggregation, enabling you to perform debugging without the use of any tools and without connection to Info Mart.
	
	For example:
	
	`java -jar ./agg/GIMAgg.jar`
	`-evaluateFile my-Agg.ss`
	
	Tip
	You can also use Scheme `printf` and `display` statements to output results for debugging purposes.

- **Limitations of the patch-agg File** By following this format, you can add additional alter-query statements to the patch file. If you include more than one alter-query statement for the same query, make sure that their instructions do not conflict; otherwise, RAA will use the last statement's definition. Also, you can include as many add and replace statements as needed within an alter-query statement to attain the desired level of query modification. Lastly, at this time, you *cannot* use this file to accomplish any of the following:
  - Remove metrics from the query (although you can replace their definition with a constant, such as "`0`", or build a new query altogether).
  - Restrict changes only to a particular member of a query family. For instance, you cannot specify to update the query definition for the AGT_I_SESS_STATE_SUBHR table without also simultaneously affecting the query definition for all other family members (AGT_I_SESS_STATE_HOUR and AGT_I_SESS_STATE_DAY).
  - Add another join to the DATE_TIME dimension; only one join is permitted and this one join is already used.

Format of the patch-agg.ss file

The commands that you write in the `patch-agg.ss` file must obey syntax rules and be written by using the identifiers that RAA recognizes. This section provides syntax for the following types of customizations, including examples for each type:

Adding measures to query definitions — Syntax

```
(alter-query queryName1
      (add     (~metric mNameX (mDefX)))
      (add     (~metric mNameY “expr1“ (expr2) “expr3“))
      ...
)

 (alter-query queryName2
      ... )
```

where:

- `queryName` is the name of any RAA-defined query.
- `mNameX` and `mNameY` are the names of measures that you want to add to the query definition.
- `mDefX` and `mDefY` (defined in the preceding syntax by the concatenation of three expressions: `expr1`, `expr2`, and `expr3`) are the measures’ definitions, written in the SQL format, and bound by double quotation marks or parentheses. In this example, `expr2` is a call to a procedure that returns a string value.

> Tip
>
> You can also use macros and functions as part of the SQL definition; however, this release does not support their use for customization purposes.

Example

```
(alter-query ID
      (add     (~metric OTHER_COST (max(ext.COLUMN))))
)
```

- **Joining other tables to those within the query definition — Syntax**

```
(alter-query queryName
     (add joinType TableX aliasX (joinCondition1)))
     (add joinType TableY aliasY "expr1“ (expr2) “expr3“)
)
```

Where:

- `joinType` describes the type of table join—either of the following:
    - `inner-join`
    - `left-outer-join`
- TableX and TableY are the names of the tables that are to be joined. These tables could be custom tables. For performance reasons, Genesys recommends that they exist within Info Mart.
- `aliasX` and `aliasY` are the aliases for TableX and TableY, respectively

> **Tip**
> You must provide an alias. This alias must differ from the reserved aliases that are used within the `select-*.ss` files if you are adding a new table join to the query or defining a second (or third) join to a table that already exists within the query.

- `joinCondition` describes how two tables are joined, written in SQL format and bound by double quotation marks or parenthesis.

Example

```
(alter-query ID
     (add (inner-join IRF_USER_DATA_GEN_1 irfug
        "irf.INTERACTION_RESOURCE_ID = irfug.INTERACTION_RESOURCE_ID"))
)
```

> **Tip**
> The irf alias that is used in this example is already defined in the `select-ID.ss` file.

- **Adding WHERE qualifications to queries — Syntax**

```
(alter-query queryName
     (add-predicate ( whereQualification ))
)
```

Where:

- `whereQualification` is the expression to be added to the WHERE clause of the query. Place the expression in double quotation marks to code advanced SQL in free form.

You cannot delete or modify expressions in the WHERE clause using add-predicate.

Example

```
(alter-query ID
     (add-predicate (itf.tenant_key=1))
)
```

- **Altering the definitions of measures within queries — Syntax**

```
(alter-query queryName
        (replace (~metric mName1 “mDefX”))
        (replace (~metric mName2 “expr1“ (expr2) “expr3“))
        ...
)
```

Example

```
(alter-query ID
     (replace (~metric AGENT_COST "sum(case when " (is-agent?) " then
              irf.CUSTOMER_HANDLE_DURATION*r_.AGENT_HOURLY_RATE else 0 end)"))
)
```

- **Editing joins within queries — Syntax**

```
(alter-query queryName
        (replace joinType Table alias (joinCondition))
        (replace joinType Table alias "joinCondition"))
        ...
)
```

Example

```
(alter-query CAMPAIGN
     (replace (left-outer-join IRF_USER_DATA_KEYS irfud
         (irf.INTERACTION_RESOURCE_ID = irfud.INTERACTION_RESOURCE_ID
          and irf.START_DATE_TIME_KEY = irfud.START_DATE_TIME_KEY)))
)
```

- **Adding dimensions to queries — Syntax** Exercise caution when you add new dimensions to an

existing aggregation hierarchy that is populated. To keep the hierarchy's data homogeneous, you must delete and reaggregate all of the data for all aggregation levels of the hierarchy, thereby ensuring that existing records are dimensioned by the new addition. RAA requires that you specify special-permission within the Scheme code to perform this destructive operation. In addition, consider setting the allow **DestructiveDDL** runtime parameter (described in Special Runtime Parameters for Customization) to `true`.
The syntax for adding dimensions to a query is:

```
(special-permission ‘add-dimension)
 ...
 (alter-query queryName
      (add     (~dimension dimNameX))
      (add     (~dimension dimNameY Tbl.Col))
         ...
      ...
 )
```

where:

- `dimNameX` and `dimNameY` are two dimensions that you want to add to the query definition.
- `Tbl.Col` are the table (or alias) and column, respectively, that hold the dimension. If the name of the dimension and its schema location are identical, you do not have to include `Tbl.Col`.
- `TableX` and `TableY` are the names of the tables that are to be joined. These can be custom tables. For performance reasons, Genesys recommends that you enter names of tables that exist within Info Mart.

Example

```
(special-permission ‘add-dimension)
 ...
 (alter-query AGENT
      (add     (~dimension r_.CREATE_AUDIT_KEY))
      (add     (~dimension r_.Updated  “r_.UPDATE_AUDIT_KEY”))
 )
```

- **Modifying hierarchies — Syntax**
  The following syntax provides only some of the identifiers that are used with the alter-hierarchy command:

```
(alter-hierarchy hierName1
      (agg-levels:  AgLv1 AgLv2 .. AgLvX)
      (materialize: AgTb1 AgTb2 .. AgTbX)
      (query:        baseQuery)
      (template:    “CustomText”))

 (alter-hierarchy hierName2
      ... )
```

where:

- `hierName` is the name of the aggregation hierarchy that you want to alter. For information about the Genesys-provided hierarchies, see What is an Aggregation Hierarchy?.
- `AgLv1 AgLv2 .. AgLvX` are the levels that you want RAA to aggregate. For example, HOUR, DAY, MONTH.
- `AgTb1 AgTb2 .. AgTbX` are those aggregation levels that RAA will materialize as table objects in Info Mart, instead of views. For example, `(materialize: HOUR DAY)` indicates that only the HOUR and DAY levels will exist as tables. All other levels will exist as views. You can also specify none.

  Not all combinations of views and tables will work within the materialize statement. Within a hierarchy, table creation cannot be dependent on a view; instead, table creation must be based directly from data that is pulled from FACT tables. Otherwise dependencies (not covered in this document) must also be updated. So, for instance, the following alter-hierarchy statement will not yield results, if this is the only change that is made to Scheme files:

  ```
  (alter-hierarchy H_I_SESS_STATE
        (agg-levels:   SUBHOUR, HOUR, DAY)
        (materialize: HOUR DAY)
  ```

  For the interval-based hierarchies, hour results are derived from subhour results. In this example, the SUBHOUR aggregation level is defined to exist as a view to the detriment of this customization example.

- `baseQuery` defines what entity the hierarchy is based on—either data from an Info Mart FACT table (in which case you specify fact) or the name of a defined query. The H_AGENT_GRP hierarchy, for example, is based on the AGENT query. CustomText defines the template by which RAA names the aggregation tables and views. Genesys recommends that this template include `%QUERY%` and `%LEVEL%` somewhere within the custom text. For example: `KJM_%QUERY%_%LEVEL%_72099`

Example
Views typically have slower performance than tables. Perhaps you would prefer that the subhour views within Info Mart be tables. The following example makes the subhour entity for the H_AGENT hierarchy a table that contains subhour aggregations. This coding also changes the template by which aggregation entities are named and reduces the number of aggregation levels processed. The subhour table that RAA creates and populates, for example, is AG3_AGENT_SUBHR_TEST, instead of AG2_AGENT_SUBHR. Notice also that WEEK is omitted from agg-levels.

```
(alter-hierarchy H_AGENT
 (agg-levels: SUBHOUR HOUR DAY MONTH QUARTER YEAR)
      (materialize: SUBHOUR HOUR DAY MONTH)
      (template:    “AG3_%QUERY%_%LEVEL%_TEST”)
 )
```

- **Adding SQL strings to Scheme aggregate definition** — **Syntax** The following syntax examples illustrate how you can use the Scheme macro `inline-sql` to add, drop, or replace specific SQL statements in Scheme aggregate definitions.
  - To specify an aggregator Scheme file description without alias:
    `(inline-sql <SQL statement>)`
  - To specify an aggregator Scheme file description with alias:
    `(inline-sql left-outer-join-alias <SQL statement>)`
  - To specify a drop statement in the patch-agg.ss Scheme patch file (if you've specified an alias):
    `(drop (inline-sql left-outer-join-alias))`
  - To specify a replace statement in the patch-agg.ss Scheme patch file (if you've specified an alias):
    `(replace (inline-sql left-outer-join-alias <SQL statement>))`

Examples

```
(inline-sql "inner join media_type mt2 on mt2.MEDIA_TYPE_KEY = irf.MEDIA_TYPE_KEY")
 (inline-sql testtenant "inner join tenant t2 on t2.tenant_key = irf.tenant_key")
 (inline-sql media3 "inner join media_type mt3 on mt3.MEDIA_TYPE_KEY =
 irf.MEDIA_TYPE_KEY")
 (inline-sql tenant3 "inner join tenant t3 on t3.tenant_key = irf.tenant_key")
```

```
(alter-query MYAGENT
        (drop (inline-sql media3))
        (replace (inline-sql tenant3 "inner join tenant t3 on t3.tenant_key =
 irf.tenant_key and t3.tenant_key > 0"))
```

Important

General Syntax Note: Any commas within the measure definitions or join conditions must be escaped by using a backward slash (\) or bound by double quotation marks (”). When more than one element makes up the expression, use double quotation marks. Single quotation marks (’) must be bound by double quotation marks. For example: `it.INTERACTION_SUBTYPE_CODE` in `('INTERNALCOLLABORATIONREPLY','INBOUNDCOLLABORATIONREPLY')` can be presented as any of the following in Scheme code:

- `it.INTERACTION_SUBTYPE_CODE in "('INTERNALCOLLABORATIONREPLY', 'INBOUNDCOLLABORATIONREPLY')"`
- `it.INTERACTION_SUBTYPE_CODE in (\'INTERNALCOLLABORATIONREPLY\'\, \'INBOUNDCOLLABORATIONREPLY\')`
- `it.INTERACTION_SUBTYPE_CODE in ("'INTERNALCOLLABORATIONREPLY'"\, "'INBOUNDCOLLABORATIONREPLY'`

How Do I View the Aggregation Query?

The Scheme files for the Genesys-provided hierarchies include high-level constructions that employ macros. These macros simplify development of SQL queries but make it difficult to see the actual queries that are passed to the RDBMS. This page describes where and how to view these queries.

Using LogLevel=FINEST Logs Database Queries

To view the actual SQL queries for the interval-based aggregates (in which subhour data is stored in tables rather than views), you must submit a request to reaggregate data for any range of time and then run aggregation with the finest log level of detail:

Procedure: Viewing SQL Queries for Interval-Based Aggregates

Purpose: Use this procedure to view the SQL queries for the interval-based aggregates.

Steps

1. Using the **-insertPendingAgg** runtime parameter (described in Reaggregating Data over a Certain Time Range), submit a request to reaggregate an existing range of data. The following command, for example, accomplishes this for all aggregates:

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

Tip

Your Info Mart does not have to contain contact center data, but it must be initialized in such a way that the DATE_TIME table is populated.

2. Run aggregation (in either integrated or autonomous mode) with the log level set to FINEST:

```
java -jar agg\GIMAgg.jar -user=<name> -pass=<password> -jdbcurl=<url>
                  -log=<filename> -levelOfLog=.:FINEST
```

RAA outputs the results of this request and the SELECT statements issued for *all* database queries—including those for the interval-based aggregate tables.

printQuery Logs RAA Queries

In autonomous mode, you can also specify the **-printQuery** runtime parameter on the command line:

- to view a particular query:

  ```
  java -jar ./agg/GIMAgg.jar -printQuery <queryName>
  ```

  where:

  - `<queryName>` is any query that is known to the aggregation engine.

- to output a particular query to a file:

  ```
  java -jar ./agg/GIMAgg.jar -printQuery <queryName> > <sqlfile>
  ```

  where:

  - `<queryName>` is any query that is known to the aggregation engine, or `ALL` , to print all existing queries.
  - `<sqlfile>` is the name of the log file.

  For example:

  ```
  java -jar ./agg/GIMAgg.jar -printQuery QUEUE > logfile.sql
  ```

Executing this command with this parameter requires no connection to Info Mart. For more information about the **-printQuery** parameter, refer to its description in the *Reporting and Analytics Aggregates Deployment Guide*.

How Do Subject Area Hierarchies Work?

This page provides diagrams that depict the Reporting and Analytics Aggregates (RAA) hierarchies (described in What is an Aggregation Hierarchy?). For a more technical discussion of each subject area, refer to the *Reporting and Analytics Aggregates Reference Manual* and the relevant Genesys Info Mart Reference Manual for descriptions of the dimension tables that are depicted on this page.

Aggregation Hierarchies

Figures on this page are *thumbnails*. Click any of them to view a full-sized version.

Subject Area for Business Attribute Aggregates

H_ID Star Schema

This subject area provides aggregated measures for interactions that are assigned a specific predefined business attribute. Rollups are based on media type and interaction type and are attributed to the reporting interval in which the interactions entered or began within the contact center. You can also configure custom user data by which to dimension the aggregates of this subject area. This subject area supports the H_ID hierarchy.

Subject Area for Session State Aggregates

H_I_SESS_STATE
Star Schema

This subject area provides aggregated measures of summarized agent states based on the media type that is associated with the agent session. Summarized agent sessions consider the collective state of all devices to which the agent has logged in for a particular media type. Following the Interval-Based Measures model, measures from this subject area are attributed to all intervals in which the agent states were active within a session. This subject area supports the H_I_SESS_STATE hierarchy.

Subject Area for State Reason Aggregates

H_I_STATE_RSN
Star Schema

This subject area provides aggregated measures of summarized agent states that were ascribed a particular reason code. Rollups are based on the media type that is associated with the agent session. Following the Interval-Based Measures model, measures from this subject area are attributed to all intervals in which the reason codes for the agent states were active within a session. This subject area supports the H_I_STATE_RSN hierarchy.

Subject Area for Queue Aggregates

H_QUEUE Star
Schema

This subject area provides aggregated measures for interactions that pass through a specific queue, as viewed from the perspective of that queue. Supported queue types include: ACD queues, virtual queues, interaction queues, and interaction workbins. Rollups are based on media type and interaction type and are attributed to the reporting interval in which the interactions entered the queue. This subject area supports the H_QUEUE hierarchy.

Subject Area for Queue Group Aggregates

H_QUEUE_GRP
Star Schema

This subject area provides aggregated measures for interactions that pass through queues that belong to a specific queue group, as viewed from the perspective of those queues. Rollups are based on media type and interaction type and are attributed to the reporting interval in which the interactions entered the queue group. This subject area supports the H_QUEUE_GRP hierarchy.

Subject Area for Abandoned-in-Queue Aggregates

H_QUEUE_ABN
Star Schema

This subject area provides aggregated measures for interactions that were abandoned within a specific queue, sorting their duration in queue into 20 time-range buckets. Rollups are based on media type and interaction type and are attributed to the reporting interval in which the interactions entered the queue. Aggregates include interactions that were abandoned within the short-abandoned threshold and exclude those that were abandoned immediately following distribution, such as abandoned-while-ringing interactions. This subject area supports the H_QUEUE_ABN hierarchy.

Subject Area for Speed-of-Accept Aggregates

H_QUEUE_ACC_AGENT
Star Schema

This subject area provides aggregated measures for interactions that were distributed from a specific queue and accepted by agent resources, sorting their durations from distribution queue to acceptance into 20 time-range buckets. Rollups are based on media type and interaction type and are attributed to the reporting interval in which the interactions entered the queue. Note that these aggregates do not reflect the customer’s overall wait time, as they do not include the duration that interactions spent at other queue objects before they reached the specific queue from which they were distributed. This subject area supports the H_QUEUE_ACC_AGENT hierarchy.

Subject Area for Agent Queue Aggregates

H_AGENT_QUEUE
Star Schema

This subject area provides aggregated measures of the interaction-handling activities of a specific agent where the interactions were distributed from a specific queue from one of the supported queue-type objects. Rollups for this agent/queue combination are based on key business attributes, media type, and interaction type and are attributed to the interval in which the agent received contact center interactions. You can also configure custom user data by which to dimension the aggregates of this subject area. The model references the Resource dimension twice—once, to ascertain from which queue that interactions were distributed, and once more to ascertain which agents received the interactions. Likewise, the Resource Group Combination dimension is referenced twice—once, to identify the queue-resource groups to which the queue belonged when interactions enter the queue, and once more to identify the agent-resource groups to which the agents belonged when they received the interactions. This subject area supports the H_AGENT_QUEUE hierarchy.

Subject Area for Agent Campaign Aggregates

H_AGENT_CAMPAIGN
Star Schema

This subject area provides aggregated measures of the interaction-handling activities of a specific agent where the interactions originated from a specific Genesys Outbound Contact campaign. Rollups are based both on interaction resource facts and the disposition of the contact attempts to reach customers that were generated by the campaign. Rollups are attributed to the interval in which the contact attempts were initiated by or on the behalf the agent. You can also configure custom user data by which to dimension the aggregates of this subject area. This subject area supports the H_AGENT_CAMPAIGN hierarchy.

Subject Area for Campaign Aggregates

H_CAMPAIGN Star
Schema

This subject area provides aggregated measures of the various call results of interactions that are initiated by a specific Genesys Outbound Contact campaign. Rollups are based on the contact attempts and calling lists that are used to dial Outbound Contact voice interactions and are attributed to the interval in which the campaign group sessions began. You can also configure custom user data by which to dimension the aggregates of this subject area. This subject area supports the H_CAMPAIGN hierarchy.

Subject Area for Agent Aggregates

H_AGENT Star Schema

This subject area provides aggregated measures of the interaction-handling activities that are performed by a specific contact center agent. Rollups are based on key, predefined business attributes (business result, customer segment, service type, and service subtype), media type, and interaction type and are attributed to the reporting interval in which the agent received contact center interactions. You can also configure custom user data by which to dimension the aggregates of this subject area. This subject area supports the H_AGENT hierarchy.

Subject Area for Agent Group Aggregates

H_AGENT_GRP Star Schema

This subject area provides aggregated measures of the interaction-handling activities of all agents who belong to a particular agent group. Rollups are based on key business attributes, media type, and interaction type and are attributed to the interval in which group members received contact center interactions. You can also configure custom user data by which to dimension the aggregates of this subject area. This subject area supports the H_AGENT_GRP hierarchy.

Subject Area for Agent Interval Aggregates

H_I_AGENT Star Schema

Like the Agent Aggregates subject area the Agent-Interval subject area also provides aggregated measures of the interaction-handling activities of agents. Rollups are based on media type and interaction type but they are not based on key business attributes, as they are within the Agent Aggregates subject area. The more distinguishing characteristic that differentiates the two, however, is the perspective from which the aggregates are prepared. Aggregate measures in the Agent Interval subject area are attributed to all intervals in which the agent processed contact center interactions—not exclusively to the interval in which the agent received the interactions. Refer to the Genesys Info Mart documentation set for more information about disposition versus interval measures. This subject area supports the H_I_AGENT hierarchy.

Aggregate Tables Bus Matrix

The following table summarizes the dimension tables that join to the aggregates in a bus matrix.

		Dimension Tables														
Hierarchies		**Calling List**	**Campaign**	**Date and Time**	**Group**	**Interaction Descriptor**	**Interaction Type**	**Media Type**	**Reason Code**	**Resource**	**Resource Group Combination**	**Resource State**	**Tenant**	**Time Range**	**User Data**	**Workbin**
Business Attribute Activity	**Tenant**			✓		✓	✓	✓					✓		✓	
Sessions, States, State Reasons	**Session State**			✓				✓		✓	✓		✓			
	State Reason			✓				✓	✓	✓	✓	✓	✓			
Queue Activity	**Individual Queue**			✓		✓	✓	✓		✓	✓		✓		✓	✓
	Abandoned in Queue			✓		✓	✓	✓		✓	✓		✓	✓	✓	✓
	Speed of Accept			✓		✓	✓	✓		✓	✓		✓	✓	✓	✓
	Queue Group			✓	✓	✓	✓	✓					✓		✓	
Agent Activity	**Agent-Queue Combination**			✓		✓	✓	✓		✓	✓		✓		✓	
	Individual Agent			✓		✓	✓	✓		✓	✓		✓		✓	
	Agent Group			✓	✓	✓	✓	✓					✓		✓	
	Agent Interval			✓			✓	✓		✓	✓		✓			
Campaign Activity	**Agent**		✓	✓	✓	✓	✓	✓		✓	✓		✓		✓	

		Dimension Tables														
	Contact Attempt	✓	✓	✓	✓	✓		✓					✓		✓	

How Do I Troubleshoot Aggregation?

If you find that the data you expect is not present in the aggregation tables, perform the troubleshooting steps described on this page to ensure proper configuration of your aggregation environment. Correcting these common problems should resolve the majority of issues that you are likely to encounter:

Check for Aggregation Misconfiguration

The following table provides information about errors, and describes configuration checks that you can perform to diagnose and correct configuration problems.

Symptom	Diagnosis
The following error is logged: `SCfg.Err Mapping not found, map: media-code, key: SomeKey`	RAA has encountered a media of unknown type.
The following error is logged: `Agg.SCfg.Err Mapping not found, map: resource-queue, key: someKey`	RAA logs this error when it detects queue configuration on the wrong type of DN or configuration detects a valid resource (such as a queue) for which thresholds are configured, but Genesys Info Mart has not yet added this resource to its RESOURCE_ table.
Not all RAA hierarchies are populated.	Check configuration for disabled hierarchies. If any hierarchy value is specified for the default option in the `[agg-populate-disable]` section of your Genesys Info Mart **Application** object, the aggregation process will not populate that hierarchy. If the appropriate hierarchies are enabled, check that their subhour views (for those hierarchies that have SUBHR views) are able to retrieve data. If they cannot, check the configuration of underlying Genesys Info Mart FACT tables.
Aggregation consumes too much of Info Mart’s resources and performance is slowed.	Check how many writers are defined. Although **numberOfWriters** runtime parameter and the **number-of-writers** configuration option accept values up to 16, and the **writer-schedule** configuration option (**writerSchedule** runtime parameter) accepts values up to 10, Genesys Info Mart runs on the same server, and requires resources to do so. Configure these values within the limitations of available processing power. Check database performance, statistics on the tables themselves, and other systems that might compete for the same resources.

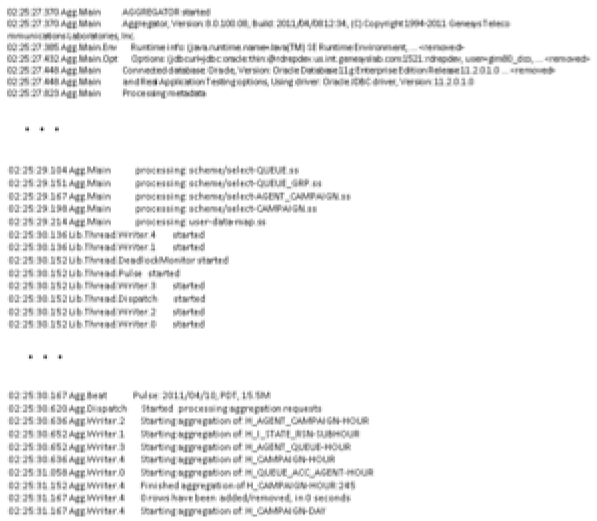
Symptom	Diagnosis
The aggregation process does not start.	Check the value of the **aggregation-engine-class-name** configuration option. If this option is not properly set, the aggregation process will not begin. Check the placement of the agg subdirectory and that the **GIM_ETL_PATHS** shell script points to the aggregation executable: `set GIM_EXT_LIBS=./agg/GIMAgg.jar`. When invoking aggregation in integrated mode, confirm that the aggregation schedule (controlled by the aggregate-schedule configuration option) permits the aggregation process to run. The **run-aggregates** configuration option must also be set to `true`.

Verify that Data Aggregation Has Begun

The aggregation process performs a number of operations before data aggregation begins. Among these operations are Info Mart connection verification; synchronization of tables and columns (if necessary, to ensure that the proper fields exist and are of the correct data type); and processing of the internal component Scheme files that define each aggregation hierarchy, software patches, language files, and user data configuration.

It is possible that the aggregation process will cease before initialization completes for any number of reasons—for example, lack of table space, RDBMS issues, and write-to-db problems, just to name a few. In this circumstance, data aggregation will not commence and an appropriate message, such as the following, will be logged:

```
16:48:39.745 Lib.Thread.AggManager     caught an exception. Monitor: Writer.2 Message:
Writer.2: unable to execute command: Agg
16:48:39.745 Lib.Thread.AggManager     SQLState: 23000
16:48:39.745 Lib.Thread.AggManager     Vendor:   1
16:48:39.745 Lib.Thread.AggManager     java.sql.BatchUpdateException: ORA-00001: unique
constraint (GIM_SG1_1.PK_AG2_I_STATE_RSN_SUBHR) violated
16:48:39.745 Lib.Thread.AggManager     Stack Trace:
```



```
02:25:27.370 Agg.Main        AGGREGATOR started
02:25:27.370 Agg.Main        Aggregator, Version 8.0.100.08, Build: 2011/04/08 12:34, (C) Copyright 1994-2011 Genesys Teleco
mmunications Laboratories, Inc.
02:25:27.385 Agg.Main.Env     Runtime info: {java.runtime.name=Java(TM) SE Runtime Environment, ... <removed>
02:25:27.432 Agg.Main.Opt     Options: {jdbcurl=jdbc:oracle:thin:@rdrepdev.us.int.genesyslab.com:1521:rdrepdev, user=gim80_dss, ... <removed>
02:25:27.448 Agg.Main        Connected database: Oracle, Version: Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 ... <removed>
02:25:27.448 Agg.Main        and Real Application Testing options, Using driver: Oracle JDBC driver, Version: 11.2.0.1.0
02:25:27.823 Agg.Main        Processing metadata

   . . .

02:25:29.104 Agg.Main        processing: schema/select-QUEUE.ss
02:25:29.151 Agg.Main        processing: schema/select-QUEUE_GRP.ss
02:25:29.167 Agg.Main        processing: schema/select-AGENT_CAMPAIGN.ss
02:25:29.198 Agg.Main        processing: schema/select-CAMPAIGN.ss
02:25:29.214 Agg.Main        processing: user-data-map.ss
02:25:30.136 Lib.Thread.Writer 4     started
02:25:30.136 Lib.Thread.Writer 1     started
02:25:30.152 Lib.Thread.DeadlockMonitor started
02:25:30.152 Lib.Thread.Pulse   started
02:25:30.152 Lib.Thread.Writer 3     started
02:25:30.152 Lib.Thread.Dispatch     started
02:25:30.152 Lib.Thread.Writer 2     started
02:25:30.152 Lib.Thread.Writer 0     started

   . . .

02:25:30.167 Agg.Beat        Pulse: 2011/04/10, PDT, 15.5M
02:25:30.620 Agg.Dispatch     Started  processing aggregation requests
02:25:30.636 Agg.Writer.2     Starting aggregation of: H_AGENT_CAMPAIGN-HOUR
02:25:30.652 Agg.Writer.1     Starting aggregation of: H_I_STATE_RSN-SUBHOUR
02:25:30.652 Agg.Writer.3     Starting aggregation of: H_AGENT_QUEUE-HOUR
02:25:30.636 Agg.Writer.4     Starting aggregation of: H_CAMPAIGN-HOUR
02:25:31.058 Agg.Writer.0     Starting aggregation of: H_QUEUE_ACC_AGENT-HOUR
02:25:31.152 Agg.Writer.4     Finished aggregation of H_CAMPAIGN-HOUR 245
02:25:31.167 Agg.Writer.4     0 rows have been added/removed, in 0 seconds
02:25:31.167 Agg.Writer.4     Starting aggregation of: H_CAMPAIGN-DAY
```

Excerpt from a Typical Genesys Info Mart Log
(Click to Enlarge)

You can confirm whether data aggregation has begun by viewing the Genesys Info Mart log (by default, named `gim_etl.log`) and looking for “`Thread.Writer ... started`” messages. The log snippet in the Figure **Excerpt from a Typical Genesys Info Mart Log**, for example, shows that five writer threads have been opened. These threads, numbered 0 through 4 in the snippet, correspond to the value defined by the **number-of-writers** configuration option or **numberOfWriters** runtime parameter depending on the mode of aggregation operation (integrated or autonomous).

Tip

The Figure **Excerpt from a Typical Genesys Info Mart Log** shows only aggregation-related log messages. If you run aggregation in integrated mode, other messages will be dispersed throughout the log to notify you about the behavior of other Genesys Info Mart jobs at key junctures.

Check the Content of Source FACT Tables

If data is not being written to aggregate tables:

- Verify that data aggregation has begun and ended for one or more hierarchies.
- Check the content of the source Genesys Info Mart FACT tables to ensure that content exists to be aggregated. Find out which FACT tables support the hierarchy. For information about how to view the SQL for RAA hierarchies, see How Do I View the Aggregation Query?.
- With an appropriate join on INTERACTION_ID to the STG_TRANSFORM_DISCARDS Info Mart table, determine if extraction was complete.

Isolate Aggregation-Related Messages in the Log

Aggregation runs asynchronously with extraction, transformation, and other Genesys Info Mart jobs that share the same processor and memory space. Log entries are directed toward the same output. You can view RAA logs using Genesys Administrator, or alternatively, you can isolate aggregation-related messages from messages written by other Genesys Info Mart jobs, by performing either of the following:

- **If aggregation is running in integrated mode, filter the log:** If Genesys Info Mart log output is directed to a file, run a filter against the log to extract aggregation-related messages. All aggregation-related messages are prefaced with a timestamp of *hh:mm:ss.ddd* format, for example: `2014-05-19 14:28:37,323 DEBUG Agg.DeadlockMonitor 35000 started` The following command creates a new output file, named `gim_agg.log`, that contains aggregation-related messages only:

  ```
  grep "[0–2][0–9]:[0–5][0–9]:[0–5][0–9]\.[0–9][0–9][0–9]"
                  gim_etl.log > gim_agg.log
  ```

  Note, however, that this command does not display log-event messages that are related to aggregation configuration, exceptions, connection and job status, or memory. Log-event messages that are generated by the Genesys Info Mart server have a predictable format. Refer to the Genesys Info Mart section of Framework Combined Log Events Help for further information.

- **Run aggregation in autonomous mode:** If aggregation is operating in integrated mode, disable it, and run it in autonomous mode from the command line. Subsequent output is related exclusively to the aggregation job. Also, issue the **-log** runtime parameter and log file, which directs all output to the specified file (otherwise, output is directed to the console).

Check for Congestion at Peak ETL Periods

By default, Genesys Info Mart maintenance begins daily at 3:00 AM. This is controlled by the values of the maintain-start-time and run-maintain configuration options. For large environments, Genesys recommends that you avoid running the aggregation process in autonomous mode during this period and during high loads.

Run updateAliases for Missing Tenant Data

Whenever the aggregation schema changes or you add tenants to your environment, you should update tenant aliases to modify and/or create new views of tenant data. Otherwise, existing aliases might become unusable, and the subset of queries that are based on the existing tenant views might not retrieve the data that you expect. Schema changes potentially occur with the deployment of hot fixes, upgrade to a new release, migration and, of course, your own database customizations (which Genesys does not support). You can update tenant aliases by running aggregation in autonomous mode and specifying the **-updateAliases** runtime parameter on the command line. For more information about the circumstances under which the update of tenant aliases must be run, and how to configure the accompanying tenant alias file, see What Do I need to Know About Managing Multi-Tenant Environments?. Should you encounter errors while running this alias update, check the log for any of the following and correct the problem:

- The specified tenant account might not exist.
- The account might have insufficient permissions to connect to the database.
- The account might lack permissions to create database objects (views).

Note that the update skips any problematic objects or accounts that it encounters, and proceeds to process the next object or account in the tenant alias file.

Check for Long-Running Interactions

Asynchronous interactions can be long-running—enduring on the order of several days, months, even years. This active interaction state can persist because of technical reasons—Genesys Info Mart might not terminate interactions that are stuck for some reason—or for legitimate business reasons, as in the case in which interactions should be kept active purposefully until a rather time-consuming process completes. Months could pass, for example, before a loan-processing interaction is funded. As described in What is Aggregation and How Do I Enable It?, Genesys Info Mart sends notifications about data that is ready for aggregation. RAA receives these notifications and performs aggregation for the entire length of time in which the interactions were active. For long-running interactions, this activity can generate problems that are manifested as:

- Arithmetic overflow in the Genesys Info Mart log. Most duration fields that RAA populates are measured in number of seconds. The number of elapsed seconds for long-running interactions can extend potentially beyond the field's data type. In this case, RAA logs an error that is similar to the following:

```
Arithmetic overflow error converting expression to data type int
```

- Slowed RDBMS performance. Specifically, the Genesys Info Mart log will show evidence of notifications that are sent about completed long-running interactions in which the interval between the first parameter and the second is huge (for example, in the tens of millions), such as the following:

```
17:59:01.264 Agg.NewData        Got addFactAvailNotification2: 1,267,438,500
 1,367,225,100 -1 INTERACTION_FACT
```

If you encounter either symptom, consider adjusting the value of the days-to-keep-active-facts configuration option to circumvent this option. This option is documented in the *Genesys Info Mart Deployment Guide*.

Check for Incorrect Data Type

If, during aggregation, RAA encounters a string value where it expected an integer, aggregation will fail and log either of the following messages:

- `Conversion failed when converting the varchar value ... to data type int.`
- `ORA-01722: invalid number`

Info Mart stores revenue and satisfaction scores in character format (in the IRF_USER_DATA_GEN_1 table) because that is how the Genesys Info Mart Server receives the data from Interaction Concentrator (ICON). ICON reports all user data as strings, and Genesys Info Mart does not transform predefined user data to INTEGER. During the aggregation process, for certain fields, RAA converts this character data into numeric format and writes the aggregated results to INTEGER fields in the aggregate tables. RAA logs the error that is noted above if RAA encounters user data that it could not convert. To address the error, you must convert all problematic data—not their data type—into data that can be cast into the INTEGER data type. For example, the following value will generate an aggregation error on Oracle:

`IRF_USER_DATA_GEN_1.REVENUE=”$1,000.00”`

For RAA purposes, you should change this particular value to exclude the dollar sign, the comma, the decimal point, and the cents:

```
UPDATE IRF_USER_DATA_GEN_1 SET REVENUE=”1000”
   WHERE REVENUE=”$1,000.00”;
```

Upon resolving all problematic data, you must then reaggregate the time period in which aggregation failed. You can also resolve problematic data by setting it to **NULL**. Executing the following SQL statements will correct the error by setting REVENUE and SATISFACTION to NULL wherever these fields do not meet RAA standards.

Oracle Query:

```
UPDATE IRF_USER_DATA_GEN_1 SET REVENUE=NULL
WHERE REVENUE IS NOT NULL
   AND LENGTH(TRIM(TRANSLATE(REVENUE,' +-.0123456789',' ')))
           IS NOT NULL
 AND START_DATE_TIME_KEY IN
           (<values for interval in which aggregation failed>);

UPDATE IRF_USER_DATA_GEN_1 SET SATISFACTION=NULL
WHERE SATISFACTION IS NOT NULL
   AND LENGTH(TRIM(TRANSLATE(SATISFACTION,' +-.0123456789',' ')))
           IS NOT NULL
   AND START_DATE_TIME_KEY IN
           (<values for interval in which aggregation failed>);

COMMIT;
```

Microsoft SQL Server Query:

```
UPDATE IRF_USER_DATA_GEN_1 SET REVENUE=NULL
WHERE REVENUE IS NOT NULL
   AND ISNUMERIC(REVENUE)=0
   AND START_DATE_TIME_KEY IN
           (<values for interval in which aggregation failed>);

UPDATE IRF_USER_DATA_GEN_1 SET SATISFACTION=NULL
WHERE SATISFACTION IS NOT NULL
   AND ISNUMERIC(REVENUE)=0
   AND START_DATE_TIME_KEY IN
           (<values for interval in which aggregation failed>);
```

Alternatively, you can institute procedures whereby the entry of nonnumeric characters is prohibited for the REVENUE and SATISFACTION fields.

Check for connection problems

Use the information in this section to detect and troubleshoot problems with connections, which can be useful both to troubleshoot configuration problems, and for security forensic audits. RAA logs detailed information about the following events:

- Established connections — for example: `Agg.Connection successfully established`. These logs include the connection URL, connection name (if present), connection hash, and user name.
- Failed to establish connection — for example: `Agg.Connection failed to establish`.
- Problems when closing a connection.

You can match up 'established' log entries with the 'closed connection' log events by comparing the connection hash value in the log files.

Example 'connection established' and 'hash closed' log:

```
...
2019-05-20 17:38:37.552 I        Agg.Connection        successfully established
'AGR_NORMALIZER_2' connection hash_1254252690 to database jdbc:postgresql://localhost:5432/
gim as postgres
...
2019-05-20 17:38:37.561 I        Agg.Connection        jdbc connection hash_1254252690 closed
...
2019-05-20 17:38:37.753 I        Agg.Connection        successfully established
'Agg.Writer.0' connection hash_141460585 to database jdbc:postgresql://localhost:5432/gim
....
```

Example 'failed to establish a connection' attempt log:

```
...
2019-05-20 7:50:42.538 W        Agg.Connection        failed to establish 'Temp' connection
to database jdbc:postgresql://localhost:5432/gim as postgres
2019-05-20 17:50:42.538 F        ERROR                Exception happened:
2019-05-20 17:50:42.538 F        ERROR                SQLState: 08001
```

```
2019-05-20 17:50:42.538 F        ERROR               Vendor:   0
2019-05-20 17:50:42.538 F        ERROR               org.postgresql.util.PSQLException:
Connection refused. Check that the hostname and port are correct and that the postmaster is
accepting TCP/IP connections.
....
```

In some cases, a connection instance can be *wrapped* by another instance. In these scenarios, the original connection is closed when the wrapped connection is closed.

Example of a 'wrapped connection' log:

```
2019-05-23 20:46:50.896 I        Agg.Connection        jdbc connection hash_1246350906 was
wrapped by instance hash_1368316340
```

Additional Resources

The following resources provide additional information that is relevant to this software. Consult these additional resources, as necessary.

Genesys CX Insights

Documentation for Genesys Customer Experience Insights (CX Insights) is available on the Genesys Documentation website:

- *Genesys CX Insights Deployment Guide*, which will help you install, start, stop, and uninstall the Genesys-provided image of MicroStrategy and the CX Insights Project and reports.
- *Genesys CX Insights User's Guide*, which includes a report- customization example that displays aggregated results that are sectioned by your own custom user data.
- *Genesys CX Insights Projects Reference Guide*, which describes objects that are used in Genesys CX Insights projects and reports, focusing on metrics, attributes, and the folders that are used to organize them.
- *Genesys CX Insights Hardware Sizing Guide*, which provides information about hardware sizing for typical contact center scenarios.
- Genesys CX Insights Release Notes, Product Alerts, and What's New are available on the GCXI page of the Genesys documentation site.

MicroStrategy

Documentation for MicroStrategy software is available on the MicroStrategy Learning Center or Help page, or in an electronic format that you can download to your mobile device (QR codes).

Easy search for MicroStrategy topics

- MicroStrategy Community Search Page

Tip

On the Community Search Page, filter your search results by selecting the Document Version (such as **2020**).

Following are some popular topics, and where to find information about them on the MicroStrategy Wiki:

The latest information from MicroStrategy

- What's New in MicroStrategy
- Key information about MicroStrategy Web
- Key information about MicroStrategy Developer

Analyzing data in a MicroStrategy report or dashboard

- Basic Reporting Guide
- Mobile Analysis Guide

Creating dashboards and reports

- Enterprise Reporting
  - Document Creation Guide
  - Dashboard and Widgets Guide
- Slice and Dice Analysis
  - Basic Reporting Guide
  - Advanced Reporting Guide
- Advanced and Predictive Analysis
  - Advanced Reporting Guide
  - Function Reference Guide
- Alerts and Proactive Notification
  - System Administration Guide
  - Mobile Analysis Guide
- OLAP Analysis
  - In-memory Analytics Guide
- Integrate data reporting with Microsoft Office
  - MicroStrategy Office User Guide

Installing or upgrading MicroStrategy

- Installation and Configuration Guide
- Upgrade Guide

Modelling your data and designing a project

- Project Design Guide

- MDX Cube Reporting Guide

Configuring and Administering MicroStrategy

- System Administration Guide
- Timeout settings in MicroStrategy Web
- User Session Idle Timeout

MicroStrategy Quick Start

- Quick Start Guide

Docker

- About Docker

Kubernetes Installation

- Kubernetes Getting Started
- Installing kubeadm

OpenShift

- OpenShift documentation

Helm

- Helm documentation

Genesys Info Mart

Documentation for Genesys Info Mart is available on the Genesys Documentation website:

- *Genesys Info Mart Operations Guide*, for information about Genesys Info Mart jobs such as `Job_AggregateGIM` and the Genesys Info Mart Manager for managing Genesys Info Mart jobs.
- *Genesys Info Mart Deployment Guide*, for information about configuring the Genesys Info Mart and Interaction Concentrator servers to recognize user data.

Reporting and Analytics Aggregates

Documentation for Reporting and Analytics Aggregates (RAA) is available on the Genesys Documentation website:

- *Reporting and Analytics Aggregates Deployment Guide*, which describes the runtime parameters and configuration options mentioned in this document.
- *Reporting and Analytics Aggregates User's Guide*, which describes the different modes of running aggregation, the aggregation hierarchies, and how to configure Reporting and Analytics Aggregates (RAA) to aggregate data based on these user-defined dimensions.
- The Physical Data Model documentation for your RDBMS, which describes the aggregate tables and subject areas:
  - *Reporting and Analytics Aggregates Physical Data Model for a Microsoft SQL Server Database*
  - *Reporting and Analytics Aggregates Physical Data Model for an Oracle Database*
  - *Reporting and Analytics Aggregates Physical Data Model for a PostgreSQL Database*

Genesys

Additional documentation for Genesys products is available, as follows:

- The Genesys Glossary provides a comprehensive list of the Genesys and computer-telephony integration (CTI) terminology and acronyms.
- *Genesys Migration Guide*, available on the Genesys Documentation website, provides documented migration strategies for Genesys product releases. Contact Genesys Customer Care for more information.
- Release Notes and Product Advisories for each Genesys product, which are available on the Genesys Documentation website.

Information about supported hardware and third-party software is available on the Genesys Documentation website in the following documents:

- The Genesys CX Insights page in the *Genesys Supported Operating Environment Reference Guide*
- *Genesys Supported Media Interfaces Reference Manual*
- *Genesys Hardware Sizing Guide*, which provides information about Genesys hardware sizing guidelines for the Genesys 8.x releases. For additional system-wide planning tools and information, see the release-specific listings of System-Level Documents on the Genesys Documentation website (docs.genesys.com).

Other Genesys product documentation is available on the:

- Genesys My Support website (formerly Customer Care)
- Genesys Documentation website
- Genesys Documentation Library DVD, which you can order by email from Genesys Order Management at Genesys Order Management.