Reporting and Analytics Aggregates 8.1

User's Guide

About Genesys

Genesys is the world's leading provider of customer service and contact center software—with more than 4,000 customers in 80 countries. Drawing on its more than 20 years of customer service innovation and experience, Genesys is uniquely positioned to help companies bring their people, insights and customer channels together to effectively drive today's customer conversation. Genesys software directs more than 100 million interactions every day, maximizing the value of customer engagement and differentiating the experience by driving personalization and multi-channel customer service—and extending customer service across the enterprise to optimize processes and the performance of customer-facing employees. Go to www.genesys.com for more information.

Each product has its own documentation for online viewing at the Genesys Documentation website or on the Documentation Library DVD, which is available from Genesys upon request. For more information, contact your sales representative.

Notice

Although reasonable effort is made to ensure that the information in this document is complete and accurate at the time of release, Genesys Telecommunications Laboratories, Inc., cannot assume responsibility for any existing errors. Changes and/or corrections to the information contained in this document may be incorporated in future versions.

Your Responsibility for Your System's Security

You are responsible for the security of your system. Product administration to prevent unauthorized use is your responsibility. Your system administrator should read all documents provided with this product to fully understand the features available that reduce your risk of incurring charges for unlicensed use of Genesys products.

Trademarks

Genesys and the Genesys logo are registered trademarks of Genesys Telecommunications Laboratories, Inc. All other company names and logos may be trademarks or registered trademarks of their respective holders.

The Crystal monospace font is used by permission of Software Renovation Corporation, www.SoftwareRenovation.com.

Technical Support from VARs

If you have purchased support from a value-added reseller (VAR), please contact the VAR for technical support.

Customer Care from Genesys

If you have purchased support directly from Genesys, please contact Genesys Customer Care. Before contacting Customer Care, please refer to the ***Genesys Care Program Guide*** for complete contact information and procedures.

Ordering and Licensing Information

Complete information on ordering and licensing Genesys products can be found in the ***Genesys Licensing Guide.***

Released by

Genesys Telecommunications Laboratories, Inc. www.genesys.com

Document Version: 81ii_us-raa_09-2014_v8.1.402.00

Table of Contents

Preface

Welcome to the *Reporting and Analytics Aggregates 8.1 User's Guide*. This document introduces you to 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 8.1.*x* releases of Reporting and Analytics Aggregates (RAA).

This document is valid only for the 8.1 release(s) of this product.

Note: For versions of this document created for other releases of this product, visit the Genesys Documentation website, or request the Documentation Library DVD, which you can order by e-mail from Genesys Order Management at orderman@genesys.com.

This preface contains the following sections:

About Reporting and Analytics Aggregates

Reporting and Analytics Aggregates (RAA) 8.1 provides the mechanism for creating, maintaining, and populating a subset of tables and views in a Genesys Info Mart 8.1 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 8.1 product and a necessary and transparent component of the Genesys Interactive Insights 8.1.0 product.

Intended Audience

This guide serves primarily two audiences—namely, network, IT, and contact center administrators for:

- Genesys Info Mart
- Genesys Interactive Insights

It assumes that both audiences have a basic understanding of:

- Computer-telephony integration (CTI) concepts, processes, terminology, and applications.
- Network design and operation.
- Your own network configurations.

Both audiences should be familiar also with Genesys Info Mart configuration and Info Mart schema. In addition, Genesys Info Mart-only users should be familiar with operation of the Genesys Info Mart Administration Console (and, in release 8.1.4, with Genesys Info Mart Manager) and general Genesys Info Mart functionality.

Chapter Summaries

In addition to this preface, this guide contains the following chapters:

- Chapter 1, "Introduction," on page 13, which provides an overview of how the aggregation engine operates.
- Chapter 2, "Understanding the Aggregation Hierarchies," on page 19, which describes the models that are used to aggregate data based on the type of data that is being aggregated and different reporting intervals.
- Chapter 3, "Managing the Aggregation Process," on page 23, which describes how to run and stop aggregation for both continuous operation and one-time execution.
- Chapter 4, "Updating Tenant Aliases in Multi-Tenant Environments," on page 35, which describes why and how often to update tenant aliases as well as the format of the tenant alias file.
- Chapter 5, "Configuring User Data for Aggregation," on page 39, which describes how to set up the environment to aggregate data based on your own custom, attached-data dimensions in addition to the standard attached-data dimensions of Info Mart.
- Chapter 6, "Customizing Aggregation," on page 43, which demonstrates how to add new measures and table joins to the existing queries and how to have RAA aggregate them.
- Chapter 7, "Viewing the Aggregation Query," on page 55, which shows what runtime parameter you can set to view the underlying query that populates each aggregation hierarchy.

- Chapter 8, "Business View of Aggregation Subject Areas," on page 57, which provides several star-schema diagrams that illustrate how each aggregation hierarchy is dimensioned. This chapter also provides a bus matrix that summarizes relationships between aggregate and dimension tables.
- Chapter 9, "Troubleshooting Aggregation," on page 73, which helps you decipher aggregation-related entries in the Genesys Info Mart log and provides a listing of configuration mishaps that you can use to troubleshoot why Reporting and Analytics Aggregates (RAA) operates in a manner other than expected.

Making Comments on This Document

If you especially like or dislike anything about this document, feel free to e-mail your comments to Techpubs.webadmin@genesys.com.

You can comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this document. Please limit your comments to the scope of this document only and to the way in which the information is presented. Contact your Genesys Account Representative or Genesys Customer Care if you have suggestions about the product itself.

When you send us comments, you grant Genesys a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

Contacting Genesys Customer Care

If you have purchased support directly from Genesys, please contact Genesys Customer Care.

Before contacting Customer Care, please refer to the *Genesys Care Program Guide* for complete contact information and procedures.

New in This Release

This section describes the changes that have been incorporated within this guide since the 8.0.1 release of Reporting and Analytics Aggregates:

Changes Introduced in Release 8.1.4

- A new GUI, Genesys Info Mart Manager, is introduced, which you can use to manage Genesys Info Mart jobs.

- RAA now provides split-media aggregation for new installations, which significantly enhances database performance by separating media by *flavor*; that is, by separating connection-oriented (online) and connection-less (offline) media in aggregate tables.
  - To support this change, aggregate data formerly stored in tables with the prefix `AG2_*` is now stored in tables with the prefix `AGT_*`, and the data is presented through views with the prefix `AG2_*`.
- Queue attached data is now supported by default for new installations: queue hierarchies now provide key columns that you can configure to join to two custom user data Info Mart dimension tables of your choice.

Changes Introduced in Release 8.1.104

- Queue attached data is now supported (but is not enabled by default): queue hierarchies now provide key columns that you can configure to join to two custom user data Info Mart dimension tables of your choice. Contact Customer Care for information about enabling this feature.
- A new aggregation runtime parameter option, `-InsertPendingAgg`, is introduced. See page 55 for details.

Changes Introduced in Release 8.1.101

- The internal coding that determines the order of aggregation was improved in this release. This feature stems from a Genesys Info Mart improvement in the manner in which it sends notifications (with finer granularity) to RAA. See page 15 for details.
- RAA now accepts the value `all` when updating tenant aliases. See page 36 for details.
- Instead of a Boolean value, the `allowDestructiveDDL` runtime parameter accepts a `date` argument in this release. See page 52 for details.
- The troubleshooting example describing RAA's casting of `REVENUE` and `SATISFACTION` user data to integer fields, and what to do if RAA encounters errors, has been clarified on page 79.

Changes Introduced in Release 8.1.100

- On page 29, information was added about the need to both truncate and reaggregate data when the time zone of your environment has changed.
- The format of the tenant alias file now includes the optional time-zone parameter which is shown in the syntax on page 36.
- Additional database permissions are required for updating tenant aliases. The full listing of permissions is provided in Table 2, "Required Permissions for Alias Account," on page 37.

- Additional troubleshooting examples were added to Chapter 9, beginning on page 73.

Changes Introduced in Release 8.1.000

- Three chapters were added to facilitate your understanding of RAA:
  - "Customizing Aggregation" on page 43
  - "Viewing the Aggregation Query" on page 55
  - "Troubleshooting Aggregation" on page 73

Other Changes

Other changes, which describe the changes to database schema, deployment, and migration of 7.6 aggregate data to 8.1 are provided in the *Reporting and Analytics Aggregates 8.1 Reference Manual*, the *Reporting and Analytics Aggregates 8.1 Deployment Guide,* and the *Genesys Migration Guide* respectively.

Chapter

1 Introduction

This chapter 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. It contains the following sections:

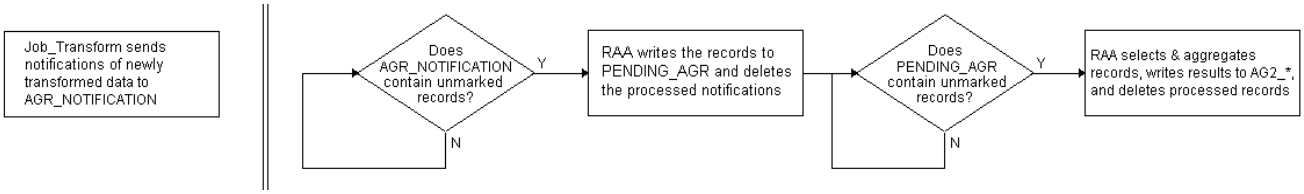
The RAA option of Genesys Info Mart must be installed before aggregation can occur. Unlike most other Genesys products though, RAA is not configured within the Genesys Configuration Server. Refer to the *Reporting and Analytics Aggregates 8.1 Deployment Guide* for installation instructions.

What is the Aggregation Process?

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.

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 `AG2_*`) in the Info Mart database (releases 8.1.3 and earlier). In release 8.1.4 and later, the results are written to aggregate tables with the prefix `AGT_*`, and the data is presented through views with the preface `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.



Enabling Aggregation

Because RAA is an optional component of Genesys Info Mart 8.1, aggregation does not occur automatically. Assigning the aggregation class, that is deployed with RAA installation, within the Genesys Info Mart application enables the aggregation engine and rouses the Genesys Info Mart Server to start sending notifications to the internal queue. This awareness is accomplished by appropriately setting the aggregation-engine-class-name configuration option (described in the *Reporting and Analytics Aggregates 8.1 Deployment Guide*).

Setting this definition alone, however, does not launch the activity that routinely updates the aggregation tables. To do this, you must separately invoke the aggregation process in one of two modes:

- Autonomous mode—Requiring no direct involvement with the Genesys Info Mart Server
- Integrated mode—Where the Genesys Info Mart Server drives aggregation activity

Each mode is described in the following subsections.

Invoking Aggregation in Autonomous Mode

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 tantamount 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 (described on page 25) or when you want to migrate 7.6 data to 8.1 (described in the *Genesys Migration Guide*).

Refer to "Running Continuous Aggregation" on page 24 to learn how to invoke aggregation in autonomous mode.

Invoking Aggregation in Integrated Mode

Operating the aggregation process ited 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 Scheduler—the Genesys Info Mart Administration Console (all 8.1 releases) or Genesys Info Mart Manager (release 8.1.4 and later). 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 Administration Console or Genesys Info Mart Manager; however, this user interface is beyond the scope of this document. Refer to the *Genesys Info Mart 8.1 Operations Guide* to learn how to start, stop, and manage jobs via this console.

Order of Aggregation

The 8.1.1 aggregation engine processes chunks of data according to priority. This is different from how the 8.1.0 aggregation engine processes data chunks —specifically, the 8.1.0 aggregation engine uses a simple distribution algorithm that processes chunks in the order in which they appear in the `PENDING_AGR` queue (from older data towards newer data), and without giving priority to any particular hierarchy or aggregation level. (Hierarchies and aggregation levels are described in the next chapter, beginning on page 19.)

RAA 8.1.1 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 8.1.1 attributes all data to one of two sliding zones: Zone 1 or Zone 2. 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 8.1 Deployment Guide* describes both the option and the runtime parameter. Chapter 3, "Managing the Aggregation Process," beginning on page 23 describes both modes of running aggregation.) Use the `realtime-offset` option (`realtimeOffset` runtime parameter) to delay RAA's processing of notifications that are received up to two hours after Genesys Info Mart transformation. Figure 1 illustrates the two zones and their boundaries.

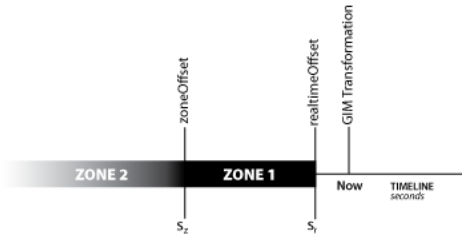


Figure 1: Two Zones Contain All Notifications

RAA 8.1.1 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 *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 page 20)—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 imme-diately 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. For 8.1.0 and prior releases ($8.1.0^{-}$), this instruction comes via the value of the `number-of-writers` configuration option (for RAA operation in integrated mode) or the `numberOfWriters` runtime parameter (for operation in autonomous mode). For $8.1.1^{+}$ releases, this instruction comes via the value of the `write-schedule` configuration option or the `writerSchedule` runtime parameter.

When Notifications Are 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 and Genesys Info Mart does not even 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.

Chapter

2 Understanding the Aggregation Hierarchies

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.

In the following sections, this chapter 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 Figure 2.

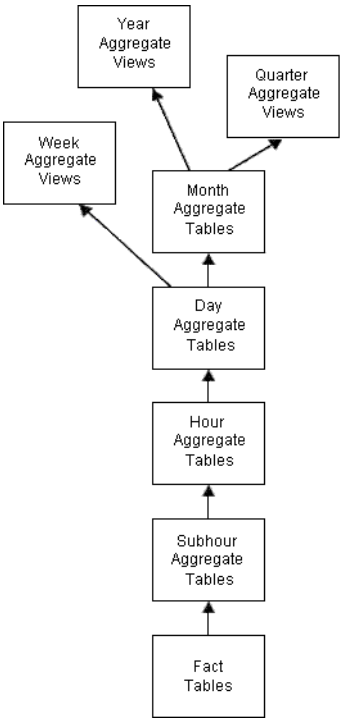
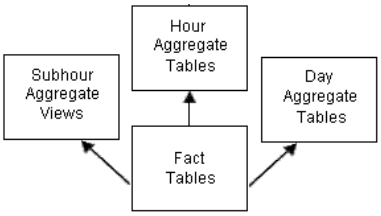


Figure 2: Aggregation Hierarchy

The fact tables at the bottom of the figure store the individual interaction-, session-, or state-related details. Fact tables are populated and maintained outside of RAA by the `Job_LoadGIM`, `Job_MaintainGIM`, and other Genesys Info Mart jobs. The *Reporting and Analytics Aggregates 8.1 Reference Manual* lists the fact table(s) that serve as the source for a particular hierarchy from whence data is aggregated, and the *Genesys Info Mart 8.1 Reference Guide* describes these fact tables in detail.

Interval-Based Measures Model in Release 7.6



Unlike the initial Interactive Insights 7.6 release, 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 (as shown to the left). 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 it previously 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. Tables* `AG2_QUEUE_HOUR` and `AG2_QUEUE_YEAR`, for instance, belong to the same

hierarchy whereas the AG2_QUEUE_HOUR and AG2_QUEUE_GRP_HOUR tables* do not (*views, in new deployments of release 8.1.4 and later).

For new deployments of release 8.1.4 and later, many AG2_* tables are replaced by views of the same name, which represent underlying tables with the prefix AGT_*. For example, the AG2_AGENT_HOUR table is replaced by AGT_AGENT_HOUR table and AG2_AGENT_HOUR view.

Disposition-Measure Hierarchies

Ten hierarchies comprise RAA's Disposition-Based Measures model:

- The H_AGENT hierarchy—Consisting of the following tables and views:
  - The AG2_AGENT_SUBHR view
  - The AG2_AGENT_HOUR table*
  - The AG2_AGENT_DAY table*
  - The AG2_AGENT_WEEK view
  - The AG2_AGENT_MONTH table*
  - 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 tables and views:
  - The AG2_AGENT_GRP_SUBHR view
  - The AG2_AGENT_GRP_HOUR table*
  - The AG2_AGENT_GRP_DAY table*
  - The AG2_AGENT_GRP_WEEK view
  - The AG2_AGENT_GRP_MONTH table*
  - 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
- The H_CAMPAIGN hierarchy—Following the same pattern
- The H_ID hierarchy—Following the same pattern
- The H_QUEUE hierarchy—Following the same pattern
- The H_QUEUE_ABN hierarchy—Following the same pattern
- The H_QUEUE_ACC_AGENT hierarchy—Following the same pattern
- The H_QUEUE_GRP hierarchy—Following the same pattern, and comprising rollups of values from the H_QUEUE hierarchy

*For new deployments of release 8.1.4 and later, this table is replaced by a view of the same name (and an underlying AGT_* table).

Interval-Measure Hierarchies

Three hierarchies comprise RAA's Interval-Based Measures model:

- The H_I_AGENT hierarchy—Consisting of the following tables and views:
  - The AG2_I_AGENT_SUBHR table*
  - The AG2_I_AGENT_HOUR table*
  - The AG2_I_AGENT_DAY table*
  - The AG2_I_AGENT_WEEK view
  - The AG2_I_AGENT_MONTH table*
  - 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

*For new deployments of release 8.1.4 and later, this table is replaced by a view of the same name (and an underlying AGT_* table).

How Hierarchies Are Used Within RAA

The aggregation process references several select-*.ss files that are located inside the Java aggregation archive (Genesys Info MartAgg.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. The 8.1.1 release opens up these Scheme files to Genesys users for limited editing. Refer to Chapter 6, "Customizing Aggregation," on page 43 for instructions.

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 "Configuring User Data for Aggregation" on page 39 for information about its use.

RAA also manages a number of parallel hierarchies exclusively for migration purposes. These migration hierarchies (defined in 76migrate-*.ss files) are temporary in nature and are not further described in RAA documentation.

Chapter

3 Managing the Aggregation Process

There is more than one way in which to invoke the Reporting and Analytics Aggregates (RAA) engine and more than one way in which to stop it from running. In the following sections, this chapter describes how to manage the aggregation process apart from the Genesys Info Mart Administration Console or Genesys Info Mart Manager:

To learn how to manage the aggregation process within the Genesys Info Mart Administration Console or Genesys Info Mart Manager—operating in integrated mode—refer to the *Genesys Info Mart 8.1 Operations Guide.*

Overview

The aggregation engine must run in order to populate the aggregation tables that are so heavily referenced by Genesys Interactive Insights. However, running this process for use of the Genesys Info Mart product alone is optional, especially if you are not interested in referencing aggregated data. Running the aggregation process creates the `AG2_*` or `AGT_*` (and supporting) tables and database constructs—if they do not already exist—within Info Mart.

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 that is similar to the following:

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

Running Continuous Aggregation

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 from one of the following subsections. Refer to the *Reporting and Analytics Aggregates 8.1 Deployment Guide* for descriptions of all runtime parameters and command-line syntax.

To invoke aggregation in integrated mode, refer to the discussion about `Job_AggregateGIM` in the *Genesys Info Mart 8.1 Operations Guide.*

Note: If properly configured, the aggregation process can operate from any location. However, for simplicity, Genesys recommends that it be positioned within the Genesys Info Mart root directory. All instructions in this document pertaining to invoking this process presume this placement.

On UNIX Platforms

To run continuous aggregation 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]
```

where:

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

On Microsoft Windows Platforms

To run continuous aggregation on Microsoft Windows platforms, open a command window and at the Genesys Info Mart root directory, issue the following command:

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

The parameters are the same as described in the previous section, "On UNIX Platforms".

Using the -conf Runtime Parameter

The `-conf` runtime parameter enables you to reference a configuration file that stores one or more runtime parameters. You specify this one parameter so that 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.

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=Widgets
   -levelOfLog=.:FINE
```

Reaggregating Data over a Certain Time Range

You can also submit a request in autonomous mode for certain data chunks to be queued for aggregation. This is accomplished by adding the `-insertPendingAggRaw` (or, in release 8.1.4 or later [or 8.1.1 releases beginning with 8.1.104],`-insertPendingAgg`) runtime parameter to the command line.

This command exits upon request submission and does not actually invoke aggregation. 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.

For example:

- `java -jar agg\GIMAgg.jar -user=<dbo> -pass=<password> -jdbcurl=<URL> -insertPendingAggRaw ALLTENANTS:ALLSETS:<startKey>:<endKey>`

  Refer to the Determining Start and End DATE_TIME Keys section to learn how to get the values for start and end date_time keys.

  This command records a request to reaggregate data and then exits. Then, run aggregation as described in the previous section, "Running Continuous Aggregation", or from the Genesys Info Mart Administration Console or Genesys Info Mart Manager.

- in release 8.1.4 or later [and 8.1.1 releases beginning with 8.1.104]: `java -jar agg\GIMAgg.jar -user=<dbo> -pass=<password> -jdbcurl=<URL> -insertPendingAgg <AGR_SET>:<START>:<END>`

Both parameters return the same result, but the `-insertPendingAgg` runtime parameter:

- Accepts the start and end time in a simplified format (`YYYY-MM-DD`), whereas the `-insertPendingAggRaw` parameter requires that you enter `DATE_TIME_KEY values` from the `DATE_TIME table`.
- Enables you to specify the set `<AGR_SET>: ALLSETS`, or an aggregate set name.

Therefore, if your deployment has release 8.1.4 or later (or 8.1.104 or a later 8.1.1 release), Genesys recommends that you use `-insertPendingAgg`.

Warning! 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.

Determining Start and End DATE_TIME Keys

You can determine which start and end `DATE_TIME` keys to specify with the `-insertPendingAggRaw` runtime parameter by looking up the corresponding records in the `DATE_TIME` Info Mart table. To retrieve all data within a specific time range, you must select the lowest `DATE_TIME_KEY` for the beginning of the time range and the highest value for the end of the time range.

Example: Determining Keys for a Specific Date Range

The following two SQL statements determine which key values from the DATE_TIME table should be selected for the July 1–August 31, 2011 time range:

```
SELECT MIN(DATE_TIME_KEY)
FROM DATE_TIME
WHERE LABEL_YYYY_MM_DD='2011-07-01';

SELECT MAX(DATE_TIME_KEY)
FROM DATE_TIME
WHERE LABEL_YYYY_MM_DD='2011-08-31';
```

Example: Determining Keys to Reaggregate All Data

To obtain the start and end DATE_TIME keys for the purpose of reaggregating all existing data in the Info Mart, issue the following command. (In this example, the DATE_TIME_KEY, 1767224700, represents 2025-12-31 23:45:00.000, which is the date and time that Genesys Info Mart uses for unfinished activities.)

Warning! Running this query can be a time-consuming and resource-expensive operation for a large Info Mart.

```
SELECT MIN(mins) startKey,
       MAX(maxe) endKey
FROM   ( SELECT MIN(START_DATE_TIME_KEY) AS mins,
                MAX(END_DATE_TIME_KEY)    AS maxe
       FROM    SM_RES_STATE_REASON_FACT
       WHERE   END_DATE_TIME_KEY   < 1767224700
       AND     END_DATE_TIME_KEY   > 0
       AND     START_DATE_TIME_KEY > 0

       UNION ALL

       SELECT MIN(START_DATE_TIME_KEY),
              MAX(END_DATE_TIME_KEY)
       FROM   SM_RES_STATE_FACT
       WHERE  END_DATE_TIME_KEY   < 1767224700
       AND    END_DATE_TIME_KEY   > 0
       AND    START_DATE_TIME_KEY > 0

       UNION ALL

       SELECT MIN(START_DATE_TIME_KEY),
              MAX(END_DATE_TIME_KEY)
       FROM   SM_RES_SESSION_FACT
       WHERE  END_DATE_TIME_KEY   < 1767224700
       AND    END_DATE_TIME_KEY   > 0
       AND    START_DATE_TIME_KEY > 0
```

```
UNION ALL

SELECT MIN(START_DATE_TIME_KEY),
       MAX(END_DATE_TIME_KEY)
FROM   INTERACTION_RESOURCE_FACT
WHERE  END_DATE_TIME_KEY   < 1767224700
AND    END_DATE_TIME_KEY   > 0
AND    START_DATE_TIME_KEY > 0

UNION ALL

SELECT MIN(START_DATE_TIME_KEY),
       MAX(END_DATE_TIME_KEY)
FROM   INTERACTION_FACT
WHERE  END_DATE_TIME_KEY   < 1767224700
AND    END_DATE_TIME_KEY   > 0
AND    START_DATE_TIME_KEY > 0

UNION ALL

SELECT MIN(START_DATE_TIME_KEY),
       MAX(END_DATE_TIME_KEY)
FROM   IXN_RESOURCE_STATE_FACT
WHERE  END_DATE_TIME_KEY   < 1767224700
AND    END_DATE_TIME_KEY   > 0
AND    START_DATE_TIME_KEY > 0

UNION ALL

SELECT MIN(START_DATE_TIME_KEY),
       MAX(END_DATE_TIME_KEY)
FROM   MEDIATION_SEGMENT_FACT
WHERE  END_DATE_TIME_KEY   < 1767224700
AND    END_DATE_TIME_KEY   > 0
AND    START_DATE_TIME_KEY > 0

UNION ALL

SELECT MIN(START_DATE_TIME_KEY),
       MAX(END_DATE_TIME_KEY)
FROM   CONTACT_ATTEMPT_FACT
WHERE  END_DATE_TIME_KEY   < 1767224700
AND    END_DATE_TIME_KEY   > 0
AND    START_DATE_TIME_KEY > 0

UNION ALL

SELECT MIN(START_DATE_TIME_KEY),
       MAX(END_DATE_TIME_KEY)
FROM   CAMPAIGN_GROUP_SESSION_FACT
```

```
WHERE  END_DATE_TIME_KEY   < 1767224700
AND    END_DATE_TIME_KEY   > 0
AND    START_DATE_TIME_KEY > 0
)
t ;
```

Reaggregating Data when DATE_TIME Changes

You must set a time zone for the DATE_TIME table upon Info Mart initialization and avoid changing it after you have set it. If, for some reason, however, your environment's time zone does change after data has been aggregated and written to Info Mart, then in order to maintain consistent data (data written to Info Mart before and after the time zone change) within the aggregate tables, all existing data from all AG2_* (AGT_*, in 8.1.4) tables must be truncated before requesting a re-aggregation of data. This procedure is described in "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, 2012, you change the time zone in your contact center environment from Eastern Standard Time to Pacific Standard Time. To set reaggregation to repopulate AG2_* (or AGT_*) tables properly, you must set the reaggregation interval from December 30, 2011 to January 11, 2012, inclusive. This is in addition to executing all of the requisite steps in Genesys Info Mart to effect the time-zone change.

Configuring More than One Time Zone to RAA

Genesys Info Mart enables users to create custom calendars for dimensioning data. You can configure RAA to recognize these calendars and aggregate data accordingly. This procedure entails declaring the different calendars to RAA, running aggregation, creating different schemas within Info Mart to restrict user access to the appropriate data, and updating tenant aliases. The complete procedure with details is provided in the "Reporting Outside the GMT Time Zone" section of the *Interactive Insights User's Guide.*

Stopping the Aggregation Process

The method by which to stop the aggregation process depends on the mode of its operation.

Stopping Aggregation in Integrated Mode

When it has started, the aggregation process, operating in integrated mode, runs continuously, aggregating new facts until the process (`Job_AggregateGIM`) is *scheduled* to be stopped. You must explicitly reset or stop its schedule to halt aggregation. If you attempt to stop aggregation manually within the Genesys Info Mart Administration Console or Genesys Info Mart Manager only, and without changing its schedule, the Genesys Info Mart Server will restart the job so long as it is scheduled to be running. Conversely, the Genesys Info Mart Server will not permit the job to be started within the Genesys Info Mart Administration Console or Genesys Info Mart Manager if the job is scheduled to be not running.

This schedule is controlled by the values of options in the `[schedule]` configuration section of the Genesys Info Mart `Application` object. Refer to the *Genesys Info Mart Deployment Guide* or the *Genesys Info Mart Deployment Procedure* for a description of this section, the options pertinent to aggregation, and the procedure for stopping aggregation that is operating in integrated mode.

Stopping Aggregation in Autonomous Mode

Although the aggregation process operating in autonomous mode will stop upon command, the Genesys Info Mart Server will restart the process in integrated mode if the process is scheduled to be running. Therefore, you first must deactivate the scheduler, as described in the *Genesys Info Mart Deployment Guide.* Then, you can stop aggregation from being invoked in autonomous mode by either:

- Clicking `^C` within the console window in which aggregation was invoked.

  or

- Killing the aggregation process.

Purging Aggregate Data

The initial 8.0.*x* and 8.1.*x* releases of Reporting and Analytics Aggregates provide no general purge capability of the aggregate (`AG2_*` or `AGT_*`) tables. `Job_MaintainGIM`, which is described in the *Genesys Info Mart 8.1 Operations Guide*, purges operational data and data from Genesys Info Mart core fact tables only. A limited purge functionality is provided with the 7.6–to–8.1 RAA

migration utility (described in the *Genesys Migration Guide*); however, this capability is reserved for use within that utility only.

Beginning in release 8.1.4, 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

Beginning in release 8.1.4, you can 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>)
[default=[flex(<writers:writers>)],[hour(<start:end>)]=[purge]]
```

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.

Table 1 describes the purging rules and parameters.

Table 1: 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 e-mail 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*).

Table 1: Purging Rule Parameters (Continued)

Parameter	Description
[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
[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 If both keep and delay parameters are present, the delay parameter must have a value lower than that of the keep parameter.

For example:

To perform a purge every February 1st, keeping 3 whole years of offline data in all QUEUE-query based hierarchies, enter the following line:

```
(purge QUEUE-*.Offline keep 3 YEAR delay 1 MONTH)
```

Or, to purge all hourly data more than 60 days old, without delay, enter the following line:

```
(purge *-HOUR keep 60 DAY delay 0 HOUR)
```

If you combine more than one such rule, they are evaluated top-down. For example:

To purge all hourly data more than 60 days old, and also perform an annual purge, keeping 3 whole years of offline data in all QUEUE-query based hierarchies, enter the following lines:

```
(purge *-HOUR keep 60 DAY delay 0 HOUR)
(purge QUEUE-*.Offline keep 3 YEAR delay 1 MONTH)
```

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
```

For more information, see *Reporting and Analytics Aggregates 8.1 Deployment Guide*.

Chapter

4 Updating Tenant Aliases in 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 prefaced `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 Analytics and 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.

In the following sections, this chapter describes the RAA tenant alias update module and how to run it:

How Often Should You 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 [described on page 43], 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, you should also run an update of tenant aliases. Thereafter, consider scheduling the update to occur regularly and automatically. This can be accomplished, for example, in 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 `AG2_*` (or `AGT_*`) tables
- The `AGR_*` tables (these are internal RAA entities)
- All Interactive Insights–specific tables and views—such as `GI2_CONSTANTS`, `RELATIVE_RANGE`, `TODAY`, and all `*_GI2` views

Whenever table structures change, you must also update tenant views for the dimension and fact tables (which are referenced by the Interactive 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 8.1 Deployment Guide* for further information.

Format of the Tenant Alias File

The `-updateAliases` runtime parameter (described in the *Reporting and Analytics Aggregates 8.1 Deployment Guide*) requires that you specify a flat file that defines which tenant accounts should be updated. The file should contain one line for each tenant alias account to be updated. The format of this line is 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.
- *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. Refer to "Reporting Outside the GMT Time Zone" in the *Interactive Insights User's Guide* for more information on the subject.

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.

Prerequisites and Logging

The minimum required permissions for alias (or tenant) account are listed in Table 2:

Table 2: Required Permissions for Alias 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.

Genesys

Chapter

5 Configuring User Data for Aggregation

This chapter describes how to configure Reporting and Analytics Aggregates (RAA), so that RAA aggregates data that is based on such user-defined dimensions.

Aggregated measures that 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 dimension table allows contact center data to be classified by four predefined user-specified business attributes:

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

You might, however, want to classify interactions and have them aggregated for reporting by additional or other business attributes, such as by product/product line, by business importance (Gold Star, Premium, and so on), or by tier (such as different technical-support levels of experience).

This chapter contains the following sections:

This chapter does not further address the aggregation of data that is based on preconfigured user data that is available in `INTERACTION_DESCRIPTOR`.

Overview

The aggregate tables and views of the H_AGENT, H_AGENT_CAMPAIGN, H_AGENT_QUEUE, H_CAMPAIGN, and H_ID, hierarchies provide two key columns that you can configure to join to two custom user data Info Mart dimension tables of your choice. 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

In release 8.1.3 and earlier, the H_AGENT_GRP hierarchy also provides two such columns, but their values are inherited from the H_AGENT hierarchy. For new installations of release 8.1.4 or later, H_QUEUE, H_QUEUE_ABN, H_QUEUE_ACC_AGENT, and H_QUEUE_GRP also provide these columns (values for these columns in the H_QUEUE_GRP hierarchy are inherited from the H_QUEUE hierarchy).

These two columns provide access to two hierarchies, or a total of 10 attached-data dimensions, as illustrated in Figure 3. 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.

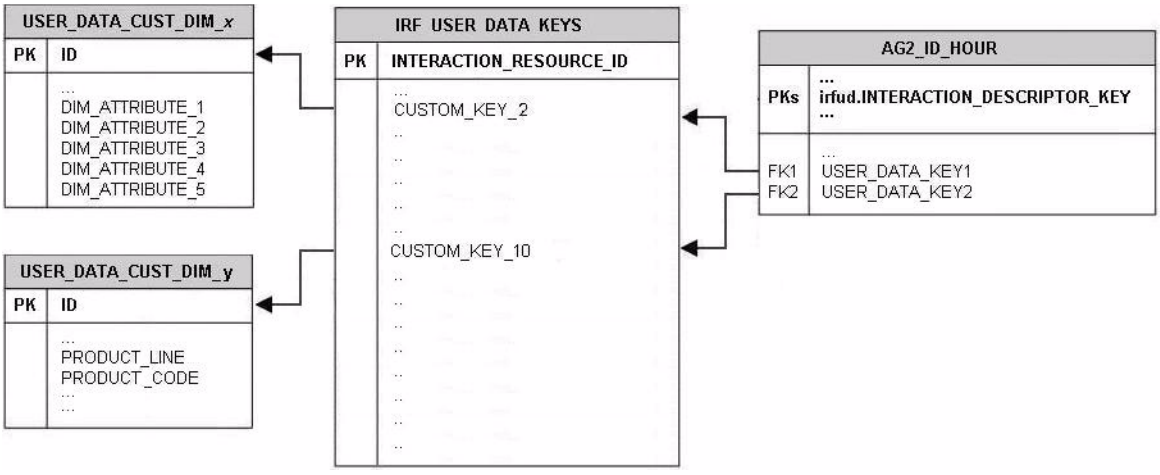


Figure 3: Mapping User Data Keys in the Aggregate Tables/Views to User Data Dimensions

Note: 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` (see page 21 for a full listing of hierarchies).
- *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

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 table (or view, in release 8.1.4) to irfud.x and map USER_DATA_KEY1 in the AG2_AGENT_ HOUR table (or view) to irfud.y. However, you could define these keys differently for different hierarchies (where neither depends on the other.)

Example of a 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))
```

Refer also to the *Genesys Interactive Insights 8.1 User's Guide* other examples that demonstrate attached data configuration starting from Interaction Concentrator (ICON) and extending to Interactive Insights with report customization that provides results dimensioned by your selected user data.

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

Genesys

Chapter

6 Customizing Aggregation

Whereas the previous chapter enabled you to customize aggregation for the reserved fields to the user-data dimensions, this chapter demonstrates how to customize the existing Reporting Analytics and 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.

In the following sections, this chapter describes how to customize RAA queries:

Using Scheme to Build 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. Scheme is a high-level dialect of the Lisp programming language. Genesys has chosen

to use Scheme because of its flexibility. To customize aggregation, you should have a solid understanding of this programming language.

Understand that editing the files in the aggregation archive is not a supported feature. Nonetheless, this chapter 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 comprises 13 query families. They correspond directly to the hierarchies that are described on page 21, and share similar names. (Hierarchies, however, are all prefaced with "H_".) The names of these queries are as follows:

- AGENT
- AGENT_CAMPAIGN
- AGENT_GRP
- AGENT_QUEUE
- CAMPAIGN
- I_AGENT
- I_SESS_STATE
- I_STATE_RSN
- ID
- QUEUE
- QUEUE_ABN
- QUEUE_ACC_AGENT
- QUEUE_GRP

Internal coding for each query family further differentiates family members along the seven time intervals (see page 19).

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 inside the aggregation Java archive. 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.

And so forth.

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.

The Patch-Aggregation File

For aggregation that operates in integrated mode, you can customize aggregation by editing a patch file named `patch-agg.ss`, which you 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.

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

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:

- Adding measures to a query definition
- Joining other tables to those within the query definition
- Adding `WHERE` qualifications to the query
- Altering the definitions of measures within a query
- Replacing joins within a query
- Adding dimensions to a query
- Adding custom aggregation levels
- Altering hierarchies

Each subsection includes examples.

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, the following SQL fragment:

```
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'")
```

Adding Measures to Queries

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.

> **Note:** 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))))
)
```

Adding Joins to Queries

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

> **Note:** 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"))
)
```

Note: The `irf` alias that is used in this example is already defined in the `select-ID.ss` file.

Adding a WHERE Qualification

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.

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

Example

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

Altering Measure Definitions 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)"))
)
```

Altering 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

You should exercise caution when you add new dimensions to an existing aggregation hierarchy that is populated. To keep the hierarchy's data homogeneous, you have no other option but to delete and reaggregate all of its data for all aggregation levels of the hierarchy. This is so that existing records can be 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 on page 52) to `true`.

Syntax

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 could be custom tables. For performance reasons, Genesys recommends that they 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. (The Genesys-provided hierarchies are described on page 21.)
- *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`.

Note: 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")
)
```

Use 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 AG2_I_SESS_STATE_SUBHR table without also simultaneously affecting the query definition for all other family members (AG2_I_SESS_STATE_HOUR* and AG2_I_SESS_STATE_DAY) (AGT_* tables, in 8.1.4).
- Add another join to the DATE_TIME dimension; only one join is permitted and this one join is already used.

Loading the Patch

The procedure for loading the patch is as follows:

1. Create (or update) the patch file—patch-agg.ss—in ASCII format and specify the desired changes to one or more queries, following the expected syntax (see page 45).
2. Place this file in the Genesys Info Mart root directory.
3. Restart aggregation. (The aggregation engine reads this file only upon start.)
4. Verify that the aggregation process is running successfully. The following message should appear in the Genesys Info Mart log:

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

   Note: 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. This procedure is described beginning on page 35.

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 AG2_ID_* (AGT_*, in 8.1.4) group of tables and views, as notifications of newly transformed data are sent to the RAA internal queue.

Special Runtime Parameters for Customization

The runtime parameters that are listed in Table 3 are not described elsewhere within the RAA documentation. They are provided to aid in the construction and debugging of your customized aggregation queries.

Table 3: 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. Beginning with release 8.1.101, 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`
	For release 8.1.0 and prior releases (8.1.0-), this parameter takes a Boolean value. If destructive DDL is allowed, specify `true`; if not, specify `false`.
	For example:
	`java -jar ./agg/GIMAgg.jar -user=<GIMDBUser> -pass=<passwd> -jdbcurl=<GIMDBURL> -allowDestructiveDDL=true`
	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.

Table 3: Special Runtime Parameters for Customization and Debugging (Continued)

Runtime Parameter	Description
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:** Incidentally, you can also use Scheme `printf` and `display` statements to output results for debugging purposes.

Chapter

7 Viewing 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. In the following sections, this chapter 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:

1. Using the `-insertPendingAggRaw` (or, beginning in release 8.1.4 [or 8.1.1 releases beginning with 8.1.104] the `-insertPendingAgg`) runtime parameter (page 25), 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`*
     `-insertPendingAggRaw ALLTENANTS:ALLSETS:`*`startKey`*`:`*`endKey`*

- or, in release 8.1.4 or later [or 8.1.1 releases beginning with 8.1.104]:
  ```
  java -jar agg\GIMAgg.jar -user=<name> -pass=<password>
  -jdbcurl=<URL>
  -insertPendingAgg <AGR_SET>:<START>:<END>
  ```

Note: 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 will both output 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 output a particular query to the log:

```
java -jar ./agg/GIMAgg.jar -printQuery queryName -log logfile
```

where *queryName* is any query that is known to the aggregation engine. 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 8.1 Deployment Guide.*

Chapter

8 Business View of Aggregation Subject Areas

The subject areas in this chapter pictorially depict the Reporting and Analytics Aggregates (RAA) hierarchies that are described beginning on page 20. For a more technical discussion of each subject area, refer to the *Reporting and Analytics Aggregates 8.1 Reference Manual* and the *Genesys Info Mart 8.1 Reference Manual* for descriptions of the dimension tables that are depicted throughout this chapter.

This chapter contains the following sections:

Five of these subject areas allow for configuration using user data dimensions. Refer to the instructions in Chapter 5, “Configuring User Data for Aggregation,” on page 39, for more information.

Subject Area for Business Attribute Aggregates

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, shown in Figure 4, supports the H_ID hierarchy.

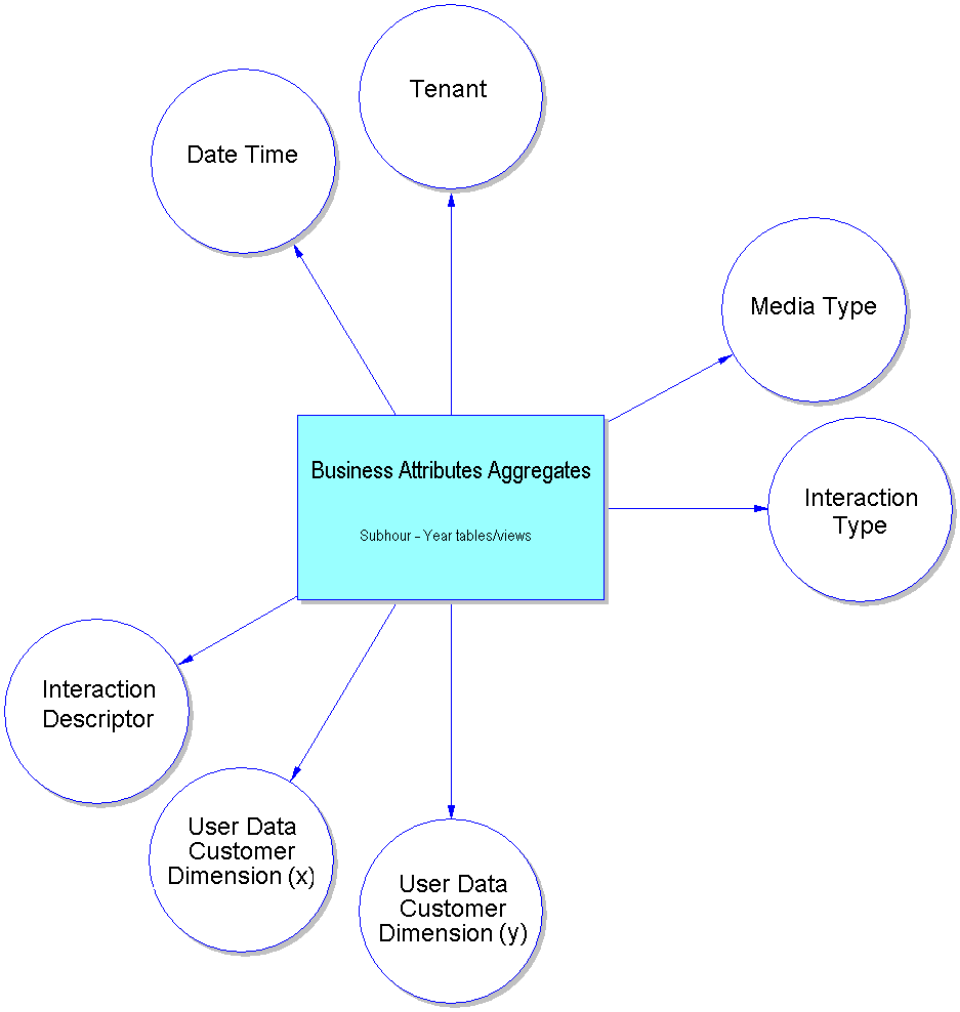


Figure 4: H_ID Star Schema

Subject Area for Session State Aggregates

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, shown in Figure 5, supports the `H_I_SESS_STATE` hierarchy.

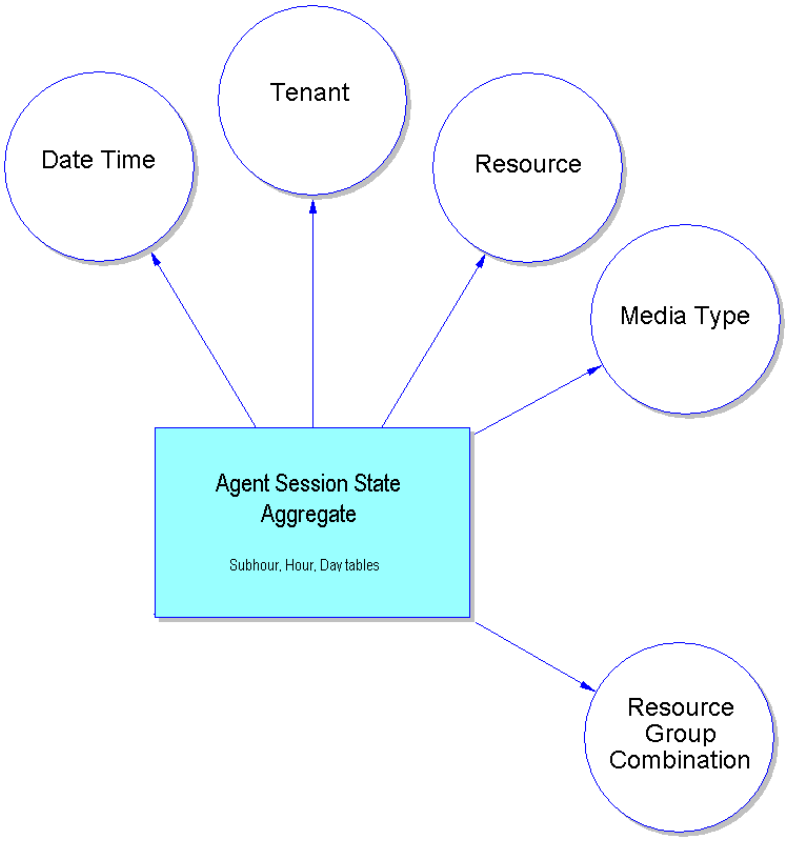


Figure 5: H_I_SESS_STATE Star Schema

Subject Area for State Reason Aggregates

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, shown in Figure 6, supports the `H_I_STATE_RSN` hierarchy.

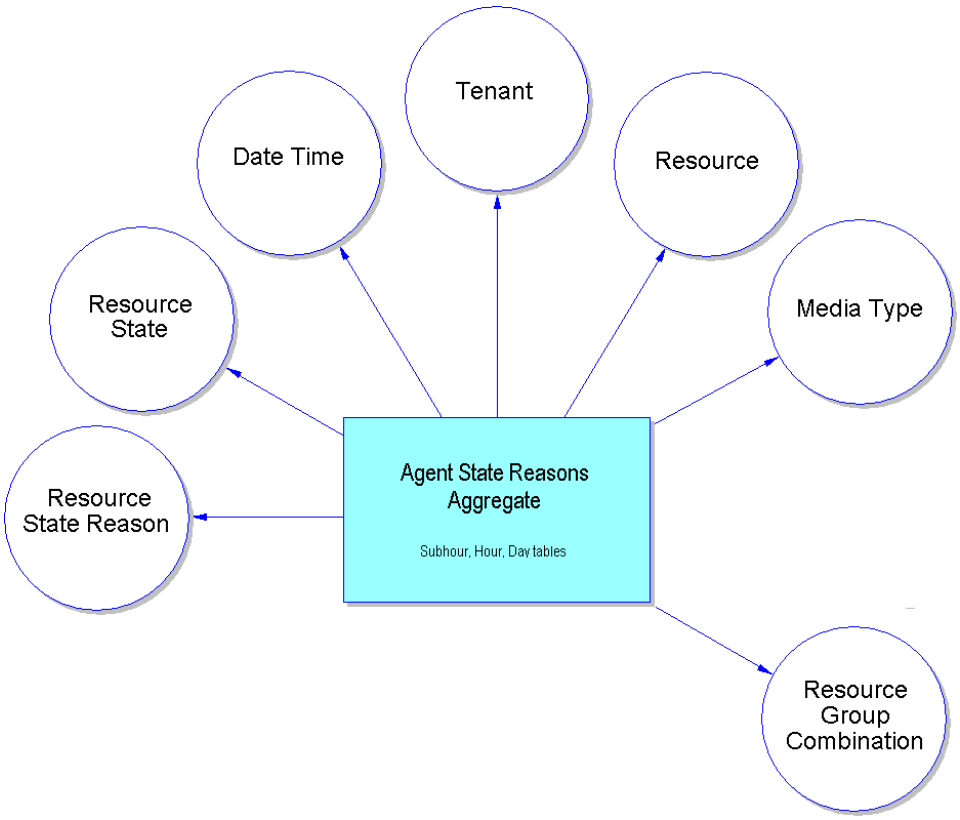


Figure 6: H_I_STATE_RSN Star Schema

Subject Area for Queue Aggregates

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, shown in Figure 7, supports the `H_QUEUE` hierarchy.

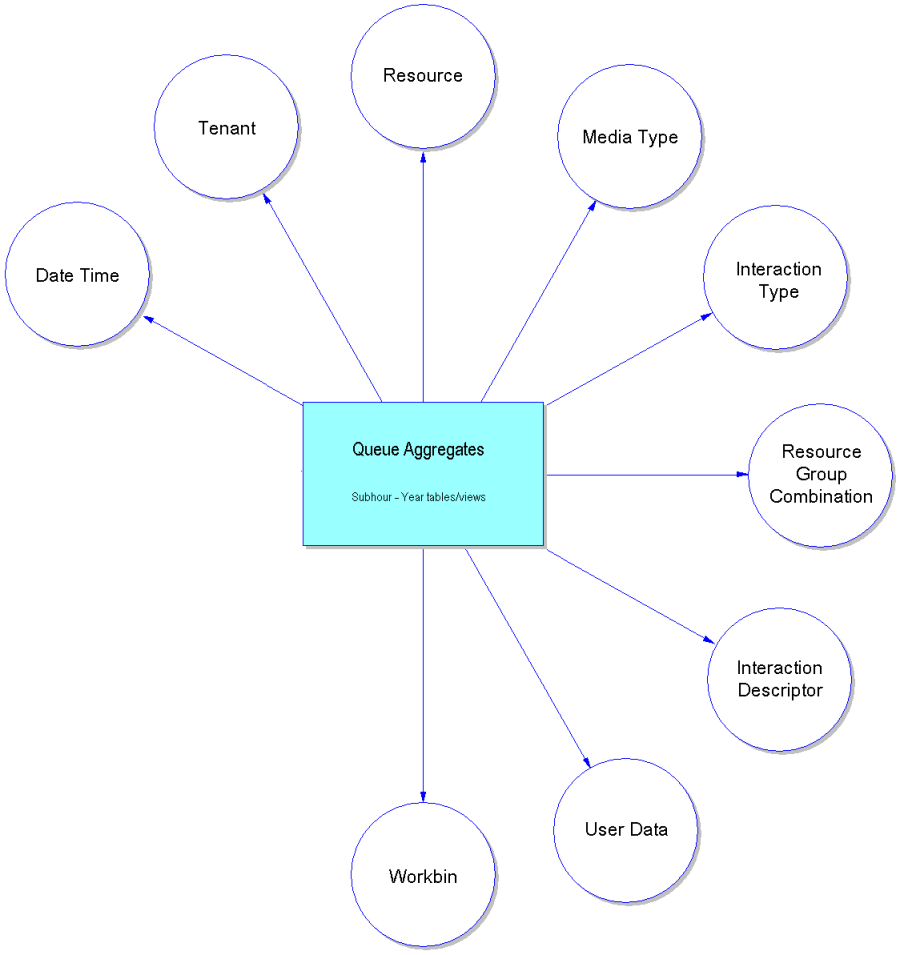


Figure 7: H_QUEUE Star Schema

Subject Area for Queue Group Aggregates

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, shown in Figure 8, supports the `H_QUEUE_GRP` hierarchy.

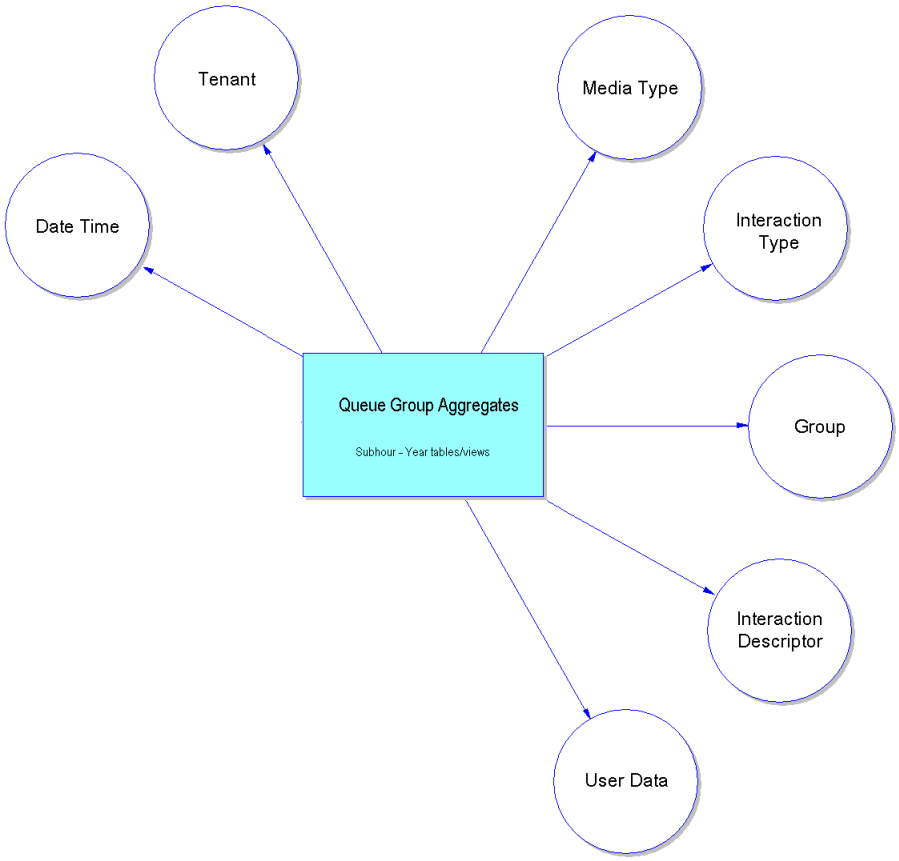


Figure 8: H_QUEUE_GRP Star Schema

Subject Area for Abandoned-in-Queue Aggregates

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, shown in Figure 9, supports the `H_QUEUE_ABN` hierarchy.

Figure 9: H_QUEUE_ABN Star Schema

Subject Area for Speed-of-Accept Aggregates

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, shown in Figure 10, supports the `H_QUEUE_ACC_AGENT` hierarchy.

Figure 10: H_QUEUE_ACC_AGENT Star Schema

Subject Area for Agent Queue Aggregates

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, shown in Figure 11, supports the `H_AGENT_QUEUE` hierarchy.

Figure 11: H_AGENT_QUEUE Star Schema

Subject Area for Agent Campaign Aggregates

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, shown in Figure 12, supports the `H_AGENT_CAMPAIGN` hierarchy.

Figure 12: H_AGENT_CAMPAIGN Star Schema

Subject Area for Campaign Aggregates

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, shown in Figure 13, supports the `H_CAMPAIGN` hierarchy.

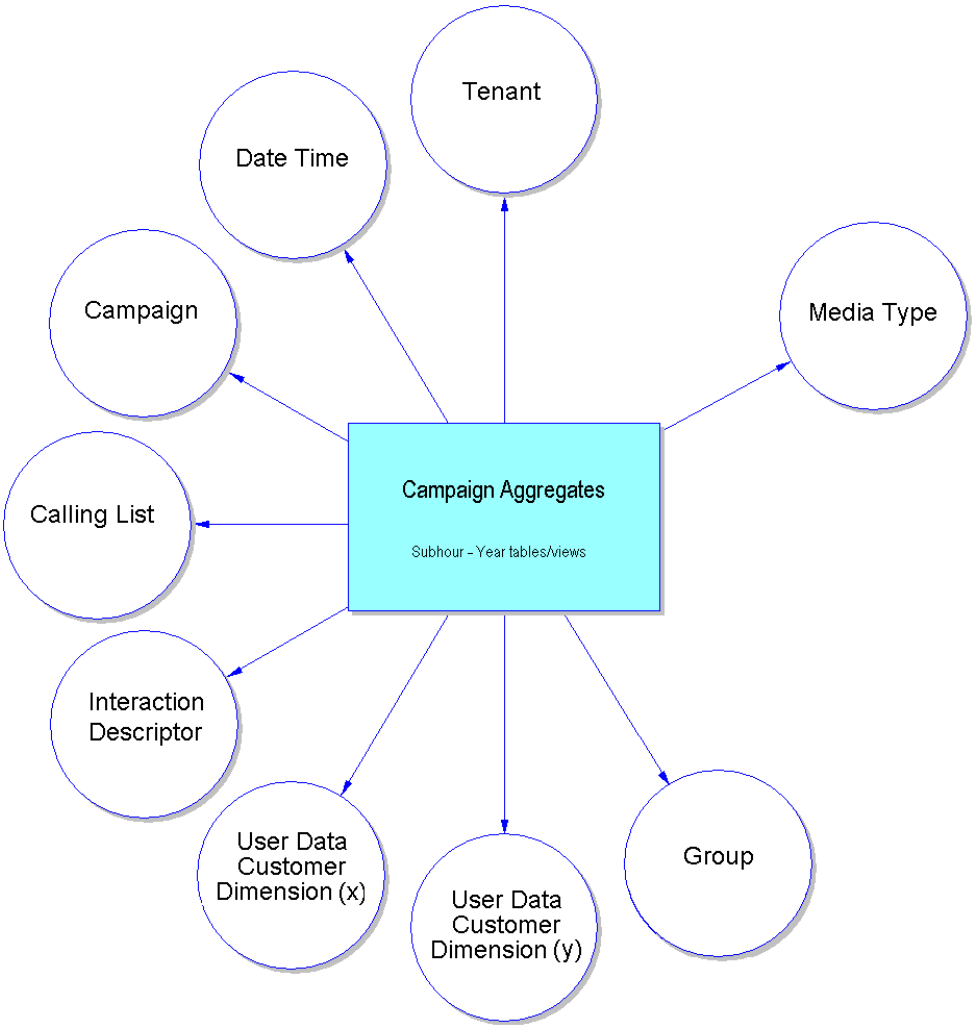


Figure 13: H_CAMPAIGN Star Schema

Subject Area for Agent Aggregates

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, shown in Figure 14, supports the H_AGENT hierarchy.

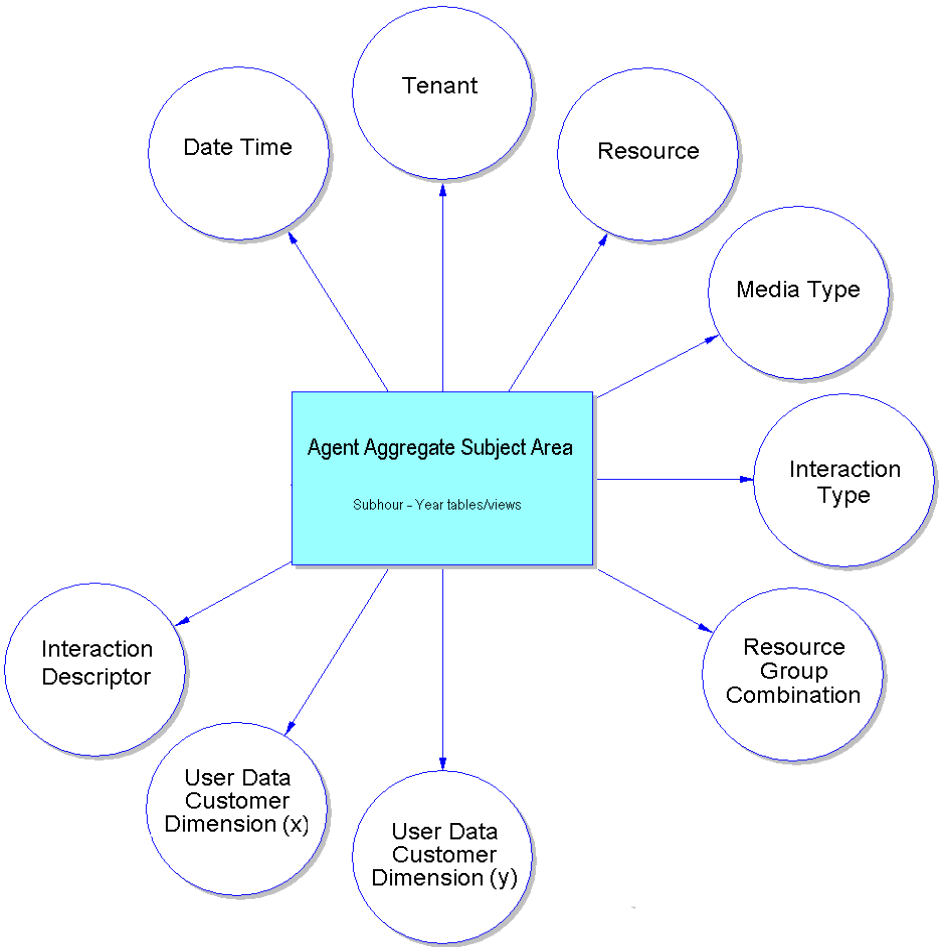


Figure 14: H_AGENT Star Schema

Subject Area for Agent Group Aggregates

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, shown in Figure 15, supports the `H_AGENT_GRP` hierarchy.

Figure 15: H_AGENT_GRP Star Schema

Subject Area for Agent Interval Aggregates

Alike the Agent Aggregates subject area, described on page 68, 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, shown in Figure 16, supports the `H_I_AGENT` hierarchy.

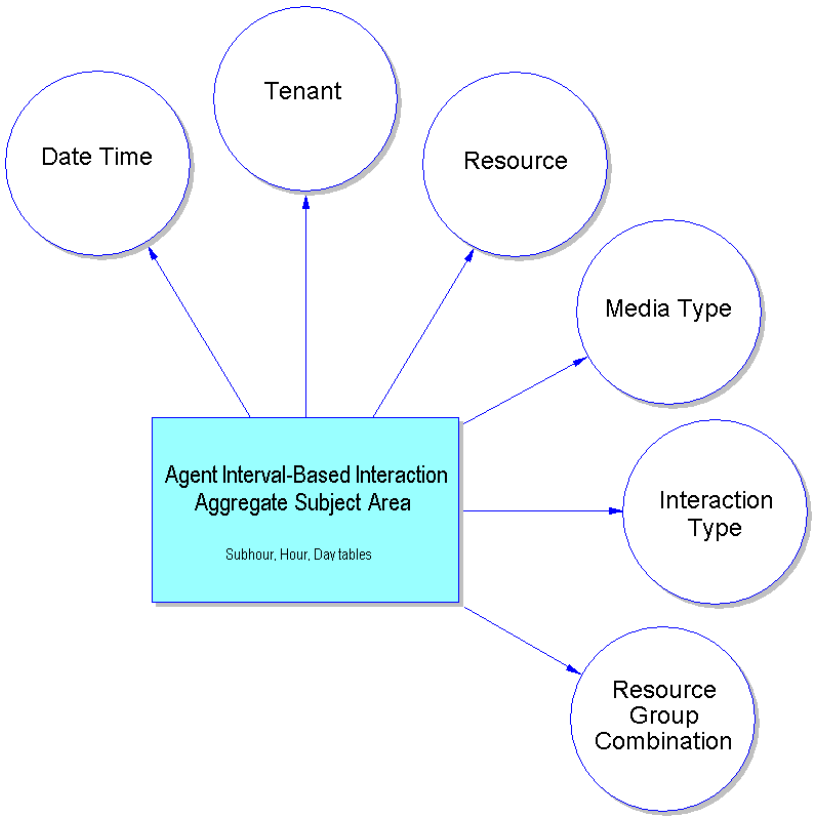


Figure 16: H_I_AGENT Star Schema

Bus Matrix

Table 4 summarizes the dimension tables that join to the aggregates in a bus matrix.

Table 4: Aggregate Tables Bus Matrix

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

Genesys

Chapter

9 Troubleshooting Aggregation

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

This chapter contains the following sections:

Check for Aggregation Misconfiguration

Table 5 provides some configuration checks that you can perform to diagnose and correct problems with configuration.

Table 5: Common RAA 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. (This message appears in 8.1.1+ environments.)

Table 5: Common RAA Configuration Problems (Continued)

Symptom	Diagnosis
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 (e.g., a queue) for which thresholds are configured but Genesys Info Mart has not yet added this resource to its `RESOURCE_` table. (This message appears in 8.1.1^{+} environments.)
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.	You might have defined too many writers. Although 16 is a valid setting for the `numberOfWriters` runtime parameter and the `number-of-writers` configuration option for 8.1.0^{-} releases, and a total of 10 writers is valid for the `writer-schedule` configuration option (`writerSchedule` runtime parameter) for 8.1.1^{+} releases, you must keep in mind that Genesys Info Mart must also run on the same server. Configure this setting within the limitations of your database server to respect a shared processor. Check other obvious culprits, such as database performance, statistics on the tables themselves, and other systems vying for the same resources.
The aggregation process will not start at all.	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` environment variable points to the aggregation executable: `set GIM_EXT_LIBS=%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:
```

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 Figure 17, 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).

Note: Figure 17 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

You have verified that data aggregation has begun and ended for one or more hierarchies and still the expected data is not to be found. It is time to check the content of the source Genesys Info Mart `FACT` tables. After all, the aggregation process cannot aggregate data that was never written to these tables.

Find out which `FACT` tables support the hierarchy. Chapter 7, "Viewing the Aggregation Query," on page 55 demonstrates how to view the SQL for RAA hierarchies.

With an appropriate join on `INTERACTION_ID` to the `STG_TRANSFORM_DISCARDS` Info Mart table, determine if extraction was complete.

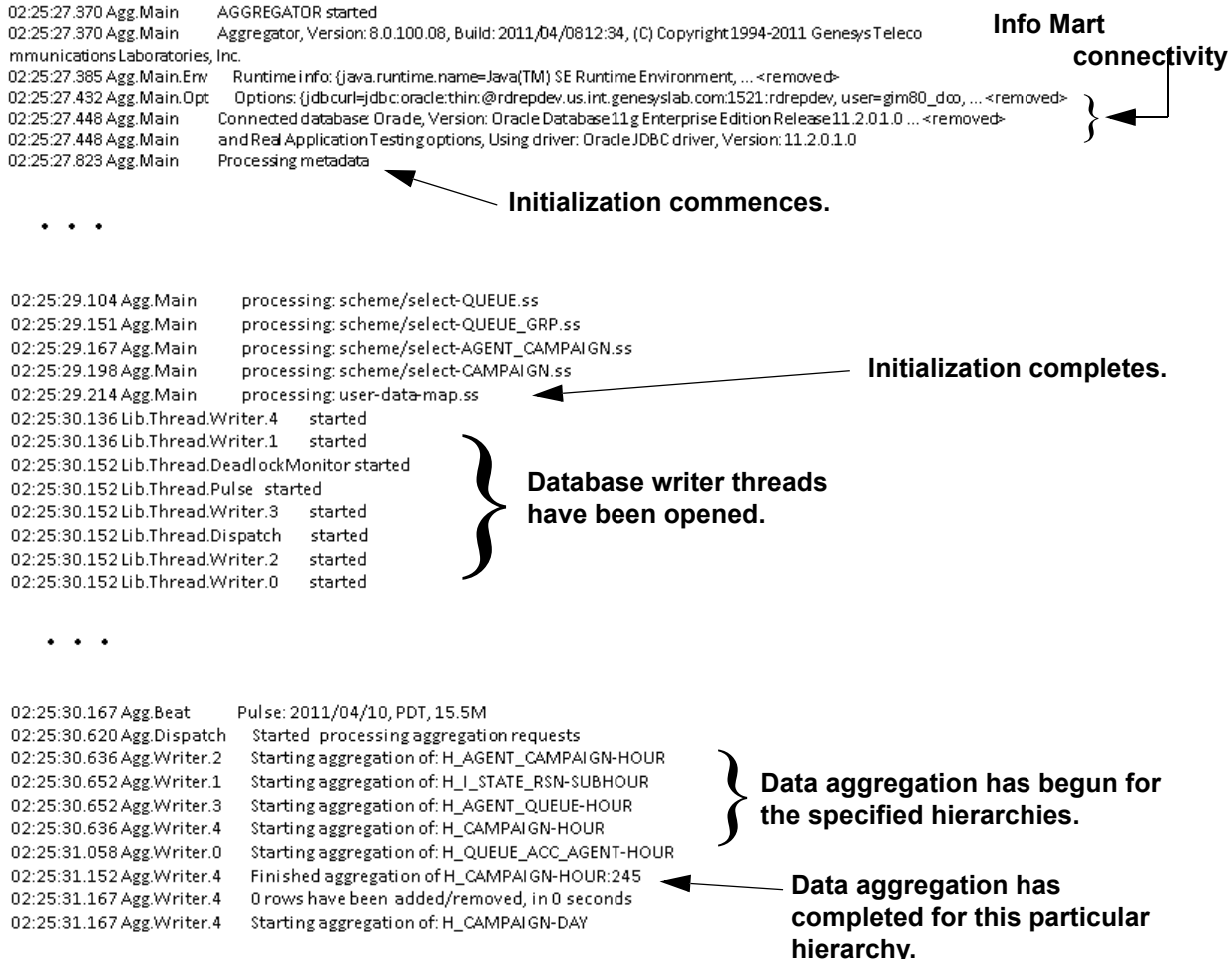


Figure 17: Snippet of a Typical Genesys Info Mart Log

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.

Beginning in release 8.1.4, you can view RAA logs using Genesys Solution Control Interface (SCI). Alternatively (or if you have release 8.1.3 or earlier) you can isolate aggregation-related messages from messages written by other Genesys Info Mart jobs, by performing either of the following:

- Filter the log, for aggregation that is running in integrated mode.

If Genesys Info Mart log output is directed to a file, you can run a filter against the log to extract aggregation-related messages. All aggregation-related messages—and only such messages—are prefaced with a timestamp of `hh:mm:ss.ddd` format, such as the following log message:

`02:25:30.152 Lib.Thread.DeadlockMonitor started` (in release 8.1.3 and earlier)

`2014-05-19 14:28:37,323 DEBUG Agg.DeadlockMonitor 35000 started` (in release 8.1.4 and later)

The following command, for instance, will create 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 will not grab 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.

  Disable aggregation if it is operating in integrated mode and run it in autonomous mode from the command line. When you do so, the output will be related exclusively to the aggregation job. Also, if you issue the `-log` runtime parameter and log file, all output will be directed to the specified file; otherwise, output will be 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, incidentally, 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. The section, "Updating Tenant Aliases in Multi-Tenant Environments" on page 35 further describes the circumstances under which the update of tenant aliases must be run and how to configure the accompanying tenant alias file.

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 will skip any problematic objects or accounts that it encounters and proceed in processing 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 on page 13, 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 (to `AG2_AGENT_HOUR.REVENUE,` (`AGT_AGENT_HOUR.REVENUE`, in 8.1.4) for example). 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
```

* Genesys Info Mart 8.1 supports numeric user data for custom-configured mapping only.

```
   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>);
```

Of course, a long term solution would be to institute procedures whereby the entry of nonnumeric characters is prohibited for the `REVENUE` and `SATISFACTION` fields. Genesys leaves this implementation to your own devices.

Related Documentation Resources

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

Reporting and Analytics Aggregates

- *Reporting and Analytics Aggregates 8.1 Deployment Guide,* which describes the runtime parameters and configuration options mentioned in this document.
- *Reporting and Analytics Aggregates 8.1 Reference Manual,* which describes the aggregate tables and subject areas.

Genesys Info Mart

- *Genesys Info Mart 8.1 Operations Guide,* for information about Genesys Info Mart jobs such as `Job_AggregateGIM` and the Genesys Info Mart Administration Console or Genesys Info Mart Manager for managing Genesys Info Mart jobs.
- *Genesys Info Mart 8.1 Deployment Guide,* for information about configuring the Genesys Info Mart and Interaction Concentrator servers to recognize user data.

Genesys Interactive Insights

- *Genesys Interactive Insights 8.1 User's Guide,* which includes a report-customization example that displays aggregated results that are sectioned by your own custom user data.

Genesys

- *Genesys Technical Publications Glossary,* available on the Genesys Documentation website, provides a comprehensive list of the Genesys and computer-telephony integration (CTI) terminology and acronyms used in this document.
- *Genesys Migration Guide*, available on the Genesys Documentation website and which ships on the Genesys Documentation Library DVD, provides documented migration strategies for Genesys product releases. Contact Genesys Customer Care for more information.

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

- *Genesys Supported Operating Environment Reference Guide*
- *Genesys Supported Media Interfaces Reference Manual*

Consult these additional resources, as necessary:

- *Genesys Interoperability Guide,* which provides information on the compatibility of Genesys products with various Configuration Layer Environments; Interoperability of Reporting Templates and Solutions; and Gplus Adapters Interoperability.
- *Genesys Licensing Guide,* which introduces you to the concepts, terminology, and procedures that are relevant to the Genesys licensing system.

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).

Genesys product documentation is available on the:

- Genesys Customer Care website at http://genesys.com/customer-care.
- Genesys Documentation site at http://docs.genesys.com/.
- Genesys Documentation Library DVD, which you can order by e-mail from Genesys Order Management at orderman@genesys.com.

Document Conventions

This document uses certain stylistic and typographical conventions—introduced here—that serve as shorthands for particular kinds of information.

Document Version Number

A version number appears at the bottom of the inside front cover of this document. Version numbers change as new information is added to this document. The following is a sample version number:

81ii_us-raa_03-2013_v8.1.102.00

You will need this number when you are talking with Genesys Customer Care about this product.

Screen Captures Used in This Document

Screen captures from the product graphical user interface (GUI), as used in this document, may sometimes contain minor spelling, capitalization, or grammatical errors. The text that accompanies and explains the screen captures corrects such errors, *except* when such a correction would prevent you from installing, configuring, or successfully using the product. For example, if the name of an option contains a usage error, the name would be presented exactly as it appears in the product GUI; the error would not be corrected in any accompanying text.

Type Styles

Table 6 describes and illustrates the type conventions that are used in this document.

Table 6: Type Styles

Type Style	Used For	Examples
Italic	• Document titles • Emphasis • Definitions of (or first references to) unfamiliar terms • Mathematical variables Used also to indicate placeholder text within code samples or commands, in the special case in which angle brackets are a required part of the syntax (see the note about angle brackets on page 84).	Please consult the *Genesys Migration Guide* for more information. Do *not* use this value for this option. A *customary and usual* practice is one that is widely accepted and used within a particular industry or profession. The formula: $x + 1 = 7$ where x stands for . . .

Table 6: Type Styles (Continued)

Type Style	Used For	Examples
Monospace font (Looks like `teletype` or `typewriter text`)	All programming identifiers and GUI elements. This convention includes: • The *names* of directories, files, folders, configuration objects, paths, scripts, dialog boxes, options, fields, text and list boxes, operational modes, all buttons (including radio buttons), check boxes, commands, tabs, CTI events, and error messages. • The values of options. • Logical arguments and command syntax. • Code samples. Used also for any text that users must enter manually during a configuration or installation procedure, or on a command line.	Select the `Show variables on screen` check box. In the `Operand` text box, enter your formula. Click `OK` to exit the `Properties` dialog box. T-Server distributes the error messages in `EventError` events. If you select `true` for the `inbound-bsns-calls` option, all established inbound calls on a local agent are considered business calls. Enter `exit` on the command line.
Square brackets ([ ])	A particular parameter or value that is optional within a logical argument, a command, or some programming syntax. That is, the presence of the parameter or value is not required to resolve the argument, command, or block of code. The user decides whether to include this optional information.	`smcp_server -host [/flags]`
Angle brackets (< >)	A placeholder for a value that the user must specify. This might be a DN or a port number that is specific to your enterprise. **Note:** In some cases, angle brackets are required characters in code syntax (for example, in XML schemas). In these cases, italic text is used for placeholder values.	`smcp_server -host <confighost>`

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z