Reporting Technical Reference

8.0 Overview

About Genesys

Alcatel-Lucent's Genesys solutions feature leading software that manages customer interactions over phone, Web, and mobile devices. The Genesys software suite handles customer conversations across multiple channels and resources—self-service, assisted-service, and proactive outreach—fulfilling customer requests and optimizing customer care goals while efficiently using resources. Genesys software directs more than 100 million customer interactions every day for 4000 companies and government agencies in 80 countries. These companies and agencies leverage their entire organization, from the contact center to the back office, while dynamically engaging their customers. Go to www.genesyslab.com for more information.

Each product has its own documentation for online viewing at the Genesys Technical Support 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, the Genesys logo, and T-Server are registered trademarks of Genesys Telecommunications Laboratories, Inc. All other trademarks and trade names referred to in this document are the property of other companies. 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.

Technical Support from Genesys

If you have purchased support directly from Genesys, please contact Genesys Technical Support at the regional numbers provided on page 7. For complete contact information and procedures, refer to the ***Genesys Technical Support Guide***.

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.genesyslab.com

Document Version: 80rtr_overview_10-2010_v8.0.001.00

Table of Contents

Preface

Welcome to the *Reporting Technical Reference Overview.* This document is the first book in the *Reporting Technical Reference* series. This document introduces you to the concepts, terminology, and procedures relevant to reporting within a Genesys environment.

This guide is valid only for the Reporting 8.0 release(s).

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

This preface provides an overview of this guide, identifies the primary audience, introduces document conventions, and lists related reference information:

For information about related resources and about the conventions that are used in this document, see the supplementary material starting on page 136.

The Reporting Technical Reference Series

This *Overview* is the first of five books in the *Reporting Technical Reference* series. The other books are:

- *Reporting Technical Reference 7.6 Report Generation Assistant*
- *Reporting Technical Reference 8.0 Solution Reporting Templates*
- *Reporting Technical Reference 8.0 Customization*
- *Reporting Technical Reference 7.6 Data Mart Conceptual Data Model*

Some components of Reporting (such as RGA and Data Mart) are associated with the 7.6 release while others (such as CCPulse+ and CCPulse+ reporting templates) are 8.0—hence, the difference in the release numbers that appear in the titles.

This book provides an in-depth discussion of the Genesys Call Model, the Genesys Multimedia Interaction Model, and the Genesys Statistical Model. While Overview is mostly theory, the other books of the series show practical applications describing how to run and customize the Genesys-provided reports, their definition, and descriptions of the tables and columns from which the reports are sourced.

Intended Audience

This document, primarily intended for advanced contact center and database administrators, assumes that you have a basic understanding of:

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

You should also be familiar with database technology.

Chapter Summaries

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

- Chapter 1, “Overview of Genesys Reporting,” on page 9, contains an overview of Genesys Reporting and explains key concepts.
- Chapter 2, “Sources of Solution Reporting Data,” on page 27, provides an in-depth discussion of the sources of data for Genesys Reporting and explains the Genesys Call Model, the Genesys Multimedia Interaction Model, and the Genesys Statistical Model.
- Chapter 3, “Historical Reporting,” on page 101, discusses Genesys historical reporting in detail, including descriptions of the various components and their relationships.
- , “Acronym List” on page 131, defines the acronyms used in this guide.

Making Comments on This Document

If you especially like or dislike anything about this document, please feel free to e-mail your comments to Techpubs.webadmin@genesyslab.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 information in this document only and to the way in which the information is presented. Speak to Genesys Technical Support 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 Technical Support

If you have purchased support directly from Genesys, contact Genesys Technical Support at the following regional numbers:

Region	Telephone	E-mail
North America and Latin America	+888-369-5555 (toll-free) +506-674-6767	support@genesyslab.com
Europe, Middle East, and Africa	+44-(0)-1276-45-7002	support@genesyslab.co.uk
Asia Pacific	+61-7-3368-6868	support@genesyslab.com.au
Malaysia	1-800-814-472 (toll-free) +61-7-3368-6868	support@genesyslab.com.au
India	000-800-100-7136 (toll-free) +91-(022)-3918-0537	support@genesyslab.com.au
Japan	+81-3-6361-8950	support@genesyslab.co.jp
Before contacting Technical Support, refer to the *Genesys Technical Support Guide* for complete contact information and procedures.		

New In This Release

This section lists topics that are new or that have changed significantly since the 7.2 release of this document.

- For your reference, the previous edition of this document is the *Reporting Technical Reference Guide for the Genesys 7.2 Release.* All of the books listed on page 5 (with the exception of RGA) comprised this single volume.
- A Session Initiation Protocol T-Server—or SIP Server—has been certified for use within the Reporting architecture. This release includes templates that measure the instant messaging activities of agents (and places) as detected by a SIP Server.
- All Genesys-provided templates have been updated to function with Hyperion Interactive Reporting 11.1.2.
- Templates for the Voice Callback option of Genesys Universal Routing have been discontinued beginning with the Reporting Templates 8.0 release.
- Call Concentrator, Interaction Concentrator, and Genesys Info Mart have been discontinued as data sources for Solution Reporting beginning with the Reporting Templates 8.0 release.

Chapter

1 Overview of Genesys Reporting

This chapter briefly describes the Genesys software environment and how the Reporting products fit into that structure. This chapter includes these sections:

The detailed discussion of Reporting structure and functionality is presented in subsequent chapters.

The Genesys CIM Platform

Genesys Solution Reporting is a part of the Genesys Customer Interaction Management (CIM) Platform. The CIM Platform consists of:

- Management Framework
- Solution Reporting (CC Analyzer, CCPulse+)
- Universal Routing
- Genesys eServices/Multimedia (optional)

Note: Genesys eServices/Multimedia is a group of components that are required if Universal Routing, Solution Reporting, and Management Framework are to handle interactions in any medium other than traditional telephony.

You must add at least one Media Channel—such as Inbound Voice, E-mail, or Chat (Web Media)—to the CIM Platform. Figure 1 shows an overview of the CIM Platform architecture and indicates where Genesys Solution Reporting fits within it.

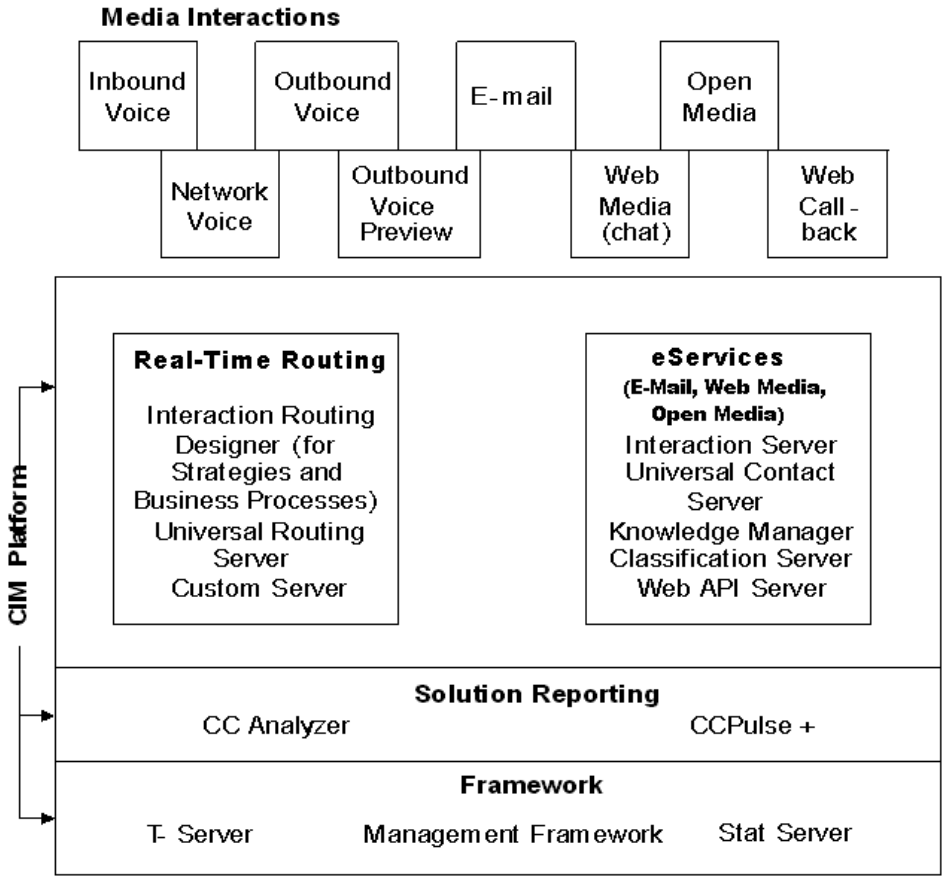

Figure 1: The Genesys CIM Platform

Figure 2 shows a more detailed view of a typical contact center environment, including the components from which Genesys Solution Reporting draws its data.

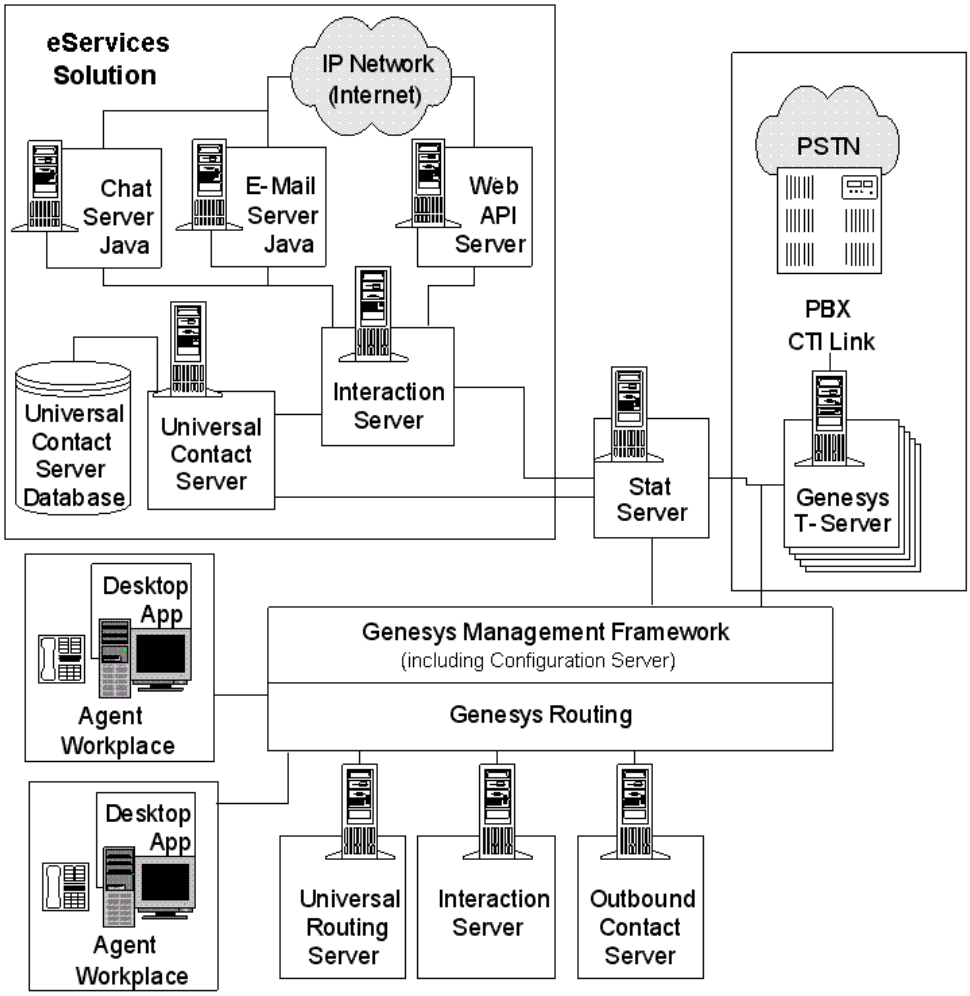

Figure 2: Genesys-Based Contact Center Environment

Now consider the major elements of the Genesys Solution Reporting environment in more detail.

T-Server/ SIP Server

The Genesys Telephony Server (T-Server) connects the switching telephony network and the contact center computing environment. In terms of computer telephony, T-Server is a CTI Server connecting the switching and computing domains. T-Server is connected with the PBX through a CTI link operating in correspondence with a stack of protocols. Through this link, T-Server monitors and controls processes within the switching domain. For example, if the PBX receives an incoming call and the corresponding telephone set starts ringing, T-Server receives a CTI protocol packet informing it about this operation. On the other end, T-Server is connected to Genesys contact center applications via T-Lib APIs to supply call-monitoring and DN-control functionality.

Note: References to T-Server and voice interactions throughout this document and series also apply to SIP Server and instant-message interactions.

Interaction Server, Universal Contact Server, and Web API Server

Contact centers that use Genesys eServices use Universal Contact Server (UCS) and Interaction Server to provide support for their multimedia interactions (for example, e-mail, chat), along with T-Server for telephony interactions. Interaction Server produces events for Internet-based interactions, enabling Genesys components and products, such as Solution Reporting, to use data on multimedia interactions.

The Universal Contact Server writes information about multimedia interactions to the Universal Contact Server database. Universal Contact Server stores interaction contents, such as e-mail contents and chat transcripts, as well as other information that identifies the interaction and how it has been processed.

Genesys eServices also uses the Web API Server, which enables customers to send e-mail using a web form. The Web API Server provides web callback functionality, enabling users to schedule callbacks from a company website.

Note: This release of the *Reporting Technical Reference Guide* does not discuss the Internet Contact Solution (ICS), an early predecessor to eServices/Multimedia. If you are using ICS, refer to the *Genesys Reporting Technical Reference Guide for the 6.5 Release*.

Configuration Server

The Genesys Configuration Server maintains and manages the contact center configuration data. For instance, it stores configuration data about all database objects of a contact center—agents, places, devices, tenants, and so on.

If you are using Open Media statistics, use Genesys Configuration Manager to configure the media type and the stat types for interactions processed by Interaction Server. Configuration Manager stores the stat types in Configuration Server.

You also use Configuration Manager or Genesys Administrator, rather than Data Modeling Assistant (DMA), to create real-time stat types. DMA is specifically designed to handle historical stat types. Refer to “Managing Statistical Parameters” in the *Customization* book of the *Reporting Technical Reference* series for a detailed explanation.

Stat Server

The Genesys Stat Server presents statistical data about contact center interactions and objects. This data is used by various Genesys applications. For instance, the Universal Routing Server uses information from Stat Server about the status of agents to determine agent availability. The Solution Reporting tools use Stat Server data to generate statistical indicators of contact center performance and status, such as average handle time for customer calls, the number of calls in a queue, and so on.

Certain multimedia reports require that you enable the Stat Server Java Extensions (SSJE) because the data for these reports is sourced through the SSJE. This added flexibility in Stat Server architecture, available in version 7.0.2 and higher, enables you to dynamically extend Stat Server functionality with new statistical types and have Stat Server supply them to Genesys applications.

The *Framework 8.0 Stat Server Deployment Guide* describes how to enable Java functionality in your Stat Server applications.

Note: Metrics derived from the SSJE are calculated only for those Interaction Server, Java instances that have the associated Stat Server included in their list of connections. The *Reporting Technical Reference 8.0 Solution Reporting Templates* indicates which metrics are derived from SSJE.

Agent Desktop Applications

Contact center agents use desktop applications in processing customer interactions. For instance, desktop applications present information about customers and incoming interactions, provide agents with options (such as transfer and conference) for handling interactions, and enable agents to attach data to interactions.

Genesys Products

Components of the Genesys platform that perform a particular function and depend on a dedicated server are called *products.* Genesys Solution Reporting provides pre-made (canned) layout templates for these products.

Universal Routing

Universal Routing, which includes Enterprise Routing and Network Routing, is based on the Universal Routing Server (URS), which is configured to process interactions and route them to a target. To do this, URS executes routing strategies, which are loaded on Routing Points. Routing strategies reflect each contact center's business logic. Based on routing strategy configuration, URS can route interactions based on agent skills or skill levels, information retrieved from a database, service levels, the value of a statistic, priority tuning, or other criteria. When executing a routing strategy, URS determines the most appropriate agent or target for the interaction, and routes the interaction accordingly.

Note: Universal Routing 7.2^{+} provides multimedia routing support and is compatible with Genesys eServices.

Outbound Contact

Genesys Outbound Contact (based on the Outbound Contact Server, as shown in Figure 2 on page 11) generates outbound calls to customers on contact-center calling lists. For instance, you can configure the software to run predictive dialing campaigns, during which it calls customers and connects the calls to agents if the customer answers the phone.

eServices

Genesys eServices tracks and processes multimedia interactions, such as e-mail and web-based chat, and eServices components also work with T-Server to process voice interactions. It enables coordinated handling of all interaction types you might be using. The Universal Contact Server identifies interaction threads and stores interaction history in the Universal Contact Server database. The database also stores information such as customer account and phone numbers and supplies the content for standard responses and screening rules.

Genesys eServices provides business process handling of multimedia interactions through Interaction Server, which receives and caches data about interactions, works with Universal Routing and its interaction workflow function to queue interactions and direct them to the appropriate targets, based on business processes and routing strategies.

Note: Universal Routing 7.2+ is designed to work with Genesys eServices. For documentation specific to the relevant release, refer to the *eServices/Multimedia* documentation set.

Genesys eServices includes e-mail and chat, SMS, and web co-browse interaction management. Using Genesys Open Media, you can route and report on any type of media. For example, you might want to track incoming faxes, route them, and report on how they are handled.

To use Genesys Open Media:

- Create custom media servers and custom media types using the software development kits (SDKs) provided by the Genesys Developer Program.
- Create routing strategies for your custom media types using Genesys Universal Routing.
- Report on custom media interaction processing by creating custom media statistics, used by both CCPulse+ and CC Analyzer.

The process for creating stat types for open media custom statistics differs in certain details from that for other stat types. However, after you have performed the initial configuration, data collection and reporting functions occur in essentially the same way as for other stat types.

Note: For a detailed explanation of how to create open media statistics, refer to the *Customization* book of the *Reporting Technical Reference* series.

Reporting in a Typical Contact Center

The processing of customer interactions within a contact center can be distributed over several CTI and Internet-based components. For example, a typical inbound call from a customer might first be connected to an IVR to

collect information about the interaction and/or customer. Then the call might be directed to a Universal Routing Server, which finds the most appropriate agent to handle the call and routes the call accordingly. Each component involved in call processing can be interesting from the reporting perspective. Figure 3 presents the general schema of Solution Reporting in a Genesys environment.

Note: Genesys offers several reporting applications that, taken together, form the Genesys Reporting Layer.

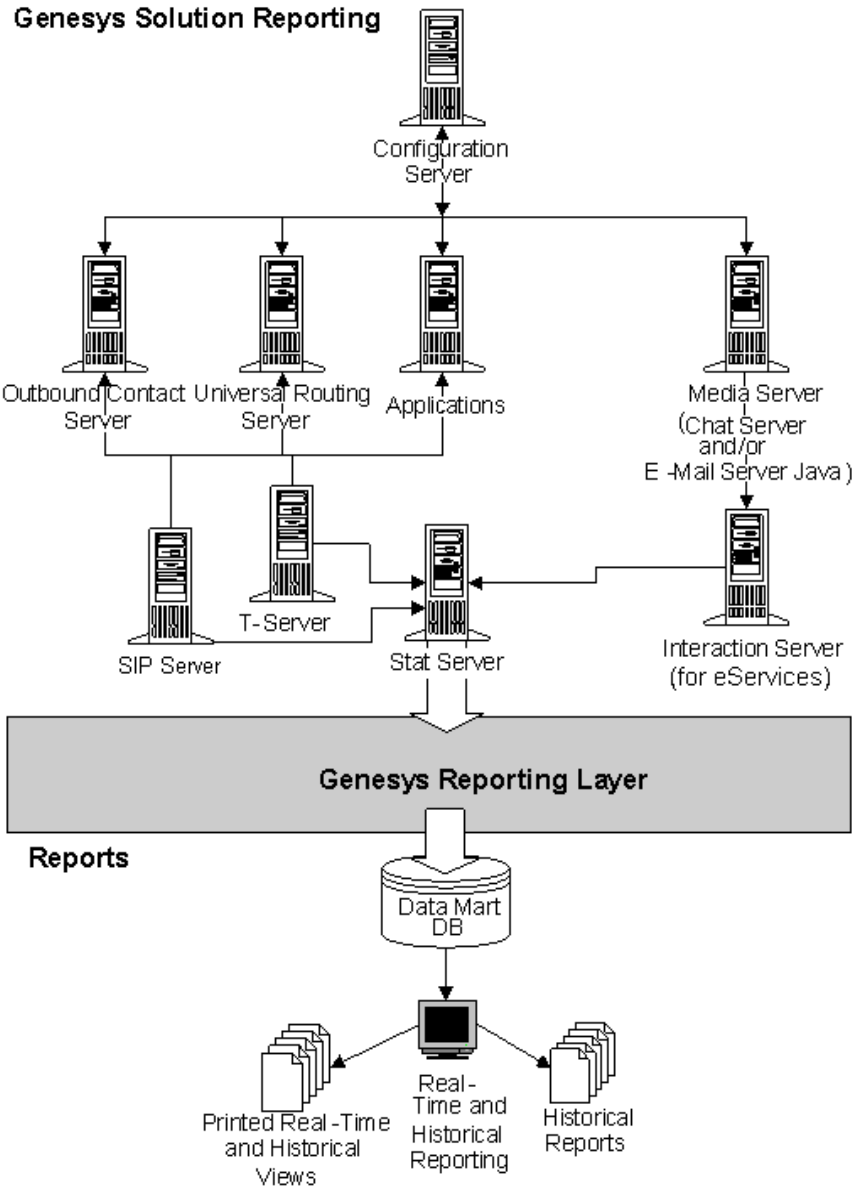

Figure 3: Genesys Solution Reporting

Reporting Layer Elements

The Reporting Layer includes three major product families:

- Solution Reporting, consisting of CC Analyzer and CCPulse+ and their common, historical, back-end components, which create object-centered historical and real-time reports based on Stat Server data.
- Interaction Concentrator (ICON), which collects and stores detailed data about interactions and resources in customer interaction networks that use Genesys Framework (contact center, enterprise-wide, or multi-enterprise telephony and computer networks). ICON serves as a staging area for Genesys Info Mart.
- Genesys Info Mart (GIM), which stores data about configuration, agent, interaction, and campaign details in a database. A series of star schemas, together with corresponding aggregate tables (available in Info Mart 7.2^{+}) speeds the retrieval of the stored data. Querying the data helps you uncover trends, chart heavy usage times, and reveal patterns in your contact center.
- Genesys Interactive Insights (GI2), which accesses Genesys data stores like Genesys Info Mart to provide historical reporting and analysis of contact center activities with business context and related business results. It also allows easy drilling and ad hoc querying to support root cause analysis.

Note: This document focuses on Solution Reporting. For additional information:

- About ICON, refer to the Interaction Concentrator 8.0 documentation set.
- About GIM, refer to the Genesys Info Mart 8.0 documentation set.
- About GI2, refer to the Interactive Insights 8.0 documentation set.

Solution Reporting consists of the following services:

- The Data Collection Services, which gather information about interactions and resources from Configuration Server and Stat Server.
- The Data Mart Services, which organize collected information into more usable forms, and aggregate metrics to months, quarters, and years.
- The Information Delivery Services, consisting of the CCPulse+ and Hyperion Interactive Reporting applications.
  - Contact center managers use the information-delivery components of CC Analyzer, which are powered by Hyperion Interactive Reporting, to construct and view reports that provide information about the performance of various contact center objects over time. Managers can filter these performance reports based on business rules.

- Managers, supervisors, and administrators use CCPulse+ to create custom real-time and historical views of contact center objects, to facilitate the analysis of staffing, business, and contact-routing strategies.

Genesys Data Sources for Solution Reporting

To collect and report business data, Solution Reporting components communicate with other pieces of the Genesys platform. Solution Reporting collects Configuration Server data about contact center objects such as agents' places, groups of agents, telephony devices, and so forth.

Stat Server supplies Solution Reporting with statistical data about the processing of interactions and the performance of contact center objects. Stat Server collects and processes the information from T-Server and Configuration Server. If you are using Genesys eServices, Stat Server also draws data from Interaction Server.

Stat Server collects primary information about the progress of calls and other types of interactions from these servers, processes it, and presents it as aggregated values or more basic numeric or status information. Upon receiving interaction events from T-Server, SIP Server, or Interaction Server, Stat Server determines the actual states of objects affected by these events and recalculates corresponding aggregates or other numeric or status parameters of related objects.

Solution Reporting Classifications

Solution Reporting functionality in the Genesys products covers many aspects of contact center performance. It is useful to classify reporting from different perspectives.

- **By time:**
  - Real-time Solution Reporting displays values of metrics in real time. For example, real-time Solution Reporting can present data such as the current number of calls in the contact center, average time of processing calls, current number of available agents, and so forth.
  - Historical Solution Reporting maintains records of past contact center activity and reports against it. The total time an agent spent processing calls during the past month is one such metric.
- **By type of subject:**
  - Object-centric Solution Reporting focuses on the statistical values of contact center objects, such as an agent's call-handling time, number of calls in queue, and so on.

- **By owner of information:**
  - Systems Solution Reporting gathers and presents general information about system elements, for example, contact center objects such as agents, groups, and places.
  - Product-specific Solution Reporting collects and presents product-specific information, such as the number of records agents process during a specific outbound campaign.
  - Business Solution Reporting collects and presents business-specific information, such as the revenue that agents generate during conversations.

Metrics and Statistics for Solution Reporting

The term *metric* is widely used in different areas of computer science and everyday life. Regardless of its application, a *metric* defines:

- The kind of subject to be measured.
- The characteristics of the subject to be measured.
- How measuring is performed.

The results of those measurements in a specific instance are *statistics.*

An example is a checking account statement. The statement displays preset categories of information, such as account balance, deposits, withdrawals, and so on. The specific information that appears in each column depends on whether the information comes from your bank account or the bank account of your college-age child or your rich aunt. The categories (metrics) stay the same. The results when the categories are applied to a specific data source at a specific time (the statistics) change.

Metrics collected for contact centers must illuminate various performance aspects of the contact center, which management can then use to formulate additional strategies that improve performance. The Solution Reporting provides just this sort of information. They are also flexible enough that you can use them to create your own metrics to fill your own business-specific needs.

The interrelation between the major statistical concepts used in Genesys Solution Reporting is illustrated in Figure 4.

The figure shows that a metric defines not only *what* should be measured, but also *where*, *how,* and *when* it should be measured. The *what* component is defined on the basic actions that characterize processes in the contact center. *Where* points to the objects of the contact center—which ones should be considered. *How* defines the manner of calculation—the algorithm (statistical category). *When* specifies time constraints of calculation. For instance, the time profile defines an interval from which statistics are to be calculated.

When metrics are applied to a specific contact center object, they produce a *statistic*. Data obtained from analyzing statistics are called *statistical values*.

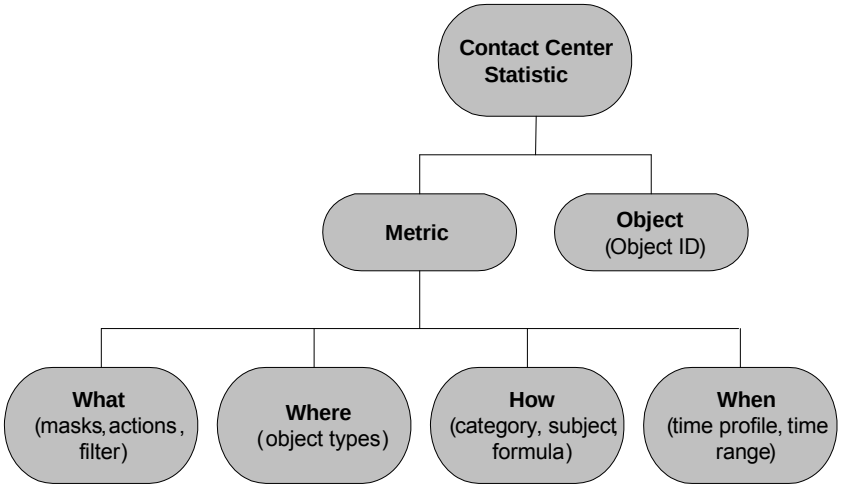

Figure 4: Statistics and Metrics for Solution Reporting

Report Layout and Layout Template

Layout templates and report layouts identify what information should be gathered about which objects and over which time period. It is important to understand how each one functions in the process of creating a report.

- *Report layouts* define which contact center objects and what data about those objects are of interest. Report layout content includes information about contact center objects, statistics for the objects, the time frames in which the statistics should be gathered, and so forth. A report layout is the application of a layout template to a specific data object(s).
- *Layout templates*, which are an abstract version of a report layout, simplify the process of report layout creation. Layout templates specify a set of defined metrics that are to be applied to a particular object type. They outline the content of a report layout but do not refer to any actual contact center object.

  Note: All of the metrics in a layout template use the same time profile.

- *Reports* emerge from the use of a report layout to collect data on specific objects over a particular time. They present actual events, whereas report layouts and layout templates represent increasing levels of abstraction.

Note: The report layout defines only report content. Report layout appearance is controlled by the Information Delivery Services tool you

use, such as CCPulse+ or CC Analyzer's Hyperion Intelligence Designer.

The most convenient way to create a report layout is to use a layout template to define the content of the report layout. Figure 5 shows the relationship between a report layout and a layout template.

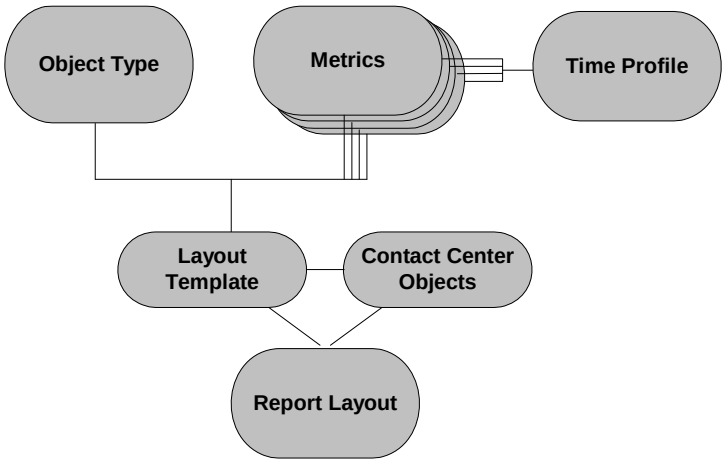

Figure 5: Relation of Report Layout to Layout Template

Genesys Reporting Layer Products

The Genesys Reporting Layer provides contact center administrators with real-time and historical views of the performance metrics of contact center objects and how these metrics change over time. The Genesys Reporting Layer offers the following set of reporting tools:

- CCPulse+—a GUI application that enables users to monitor real-time and historical statistical values of contact center objects and application-specific objects.
- CC Analyzer—an engine for Historical Reporting and analysis of the performance of contact center objects. CC Analyzer draws data from the Data Collection and Data Mart services, which also supply historical reporting data to CCPulse+.

Figure 6 presents a general schema of the Genesys Reporting Layer. In addition to data shown in Figure 6, all server components receive contact center configuration data from the Configuration Layer. Certain products also store some configuration data. Most of the components in this figure have already been discussed; the following sections describe Genesys Solution Reporting products.

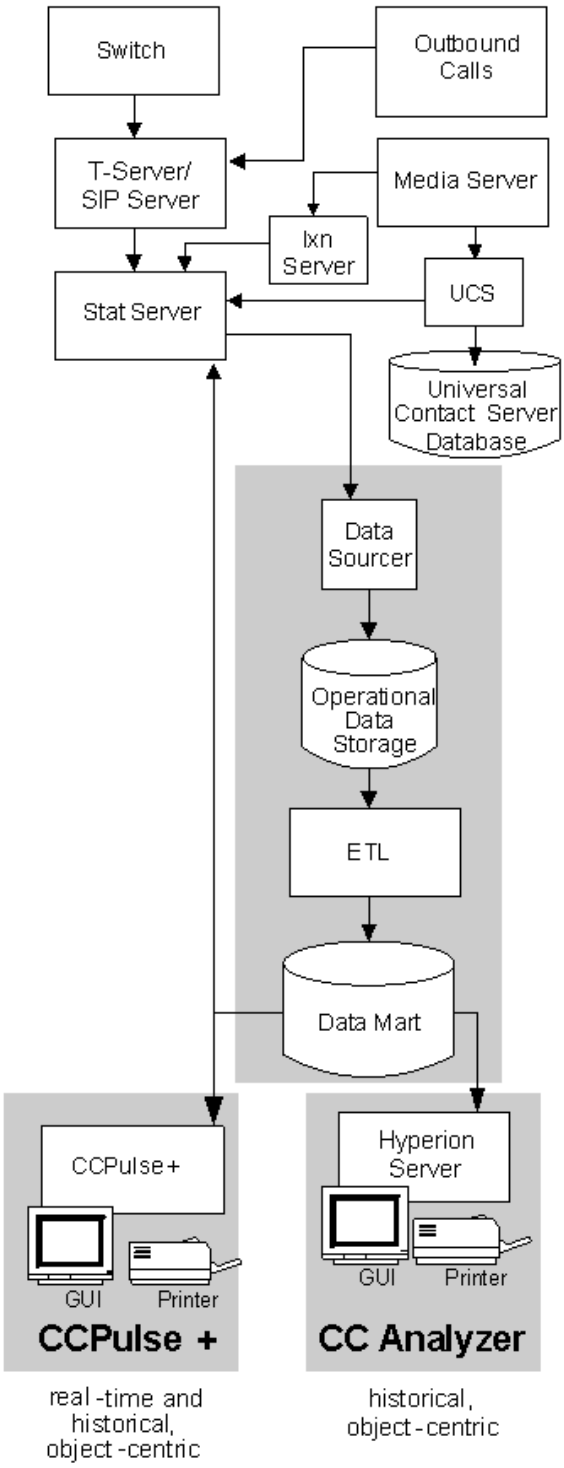

Figure 6: Reporting Information Flow through CCPulse+ and CC Analyzer

Data Collection and Processing

The Data Collection and Data Mart services provide Historical Reporting information to CCPulse+ and CC Analyzer, which includes Hyperion Interactive Reporting for data-presentation functionality. You can customize these statistics-gathering and transforming functions. You can:

- Specify which statistics should be collected for which contact center objects and how often.
- Define new statistic types (stat types), complete with retrieval parameters such as time ranges and filters.
- Propagate the information collected into a database.
- Preaggregate information into hours, days, weeks, months, quarters, and years.

CCPulse+—Real-Time and Historical Views

CCPulse+ is a desktop GUI application that displays real-time and historical status and statistical data about agents, agent groups, places, queues, and more. CCPulse+ is bundled with Genesys Universal Routing, Outbound Contact, and eServices products and supplies pre-made view report templates that best fit each product's focus of activity. CCPulse+ displays real-time and historical statistical data about agents, agent groups, places, queues, and more. CCPulse+ is bundled with Genesys Universal Routing, Outbound Contact, and eServices products and supplies premade view templates that best fit the focus of activity for each product.

CCPulse+ can display statistics for open media interactions for which you have created custom media types and statistical types.

CCPulse+ Features and Functions

- CCPulse+ monitors activity within the contact center across all media types. For example, a supervisor might monitor the number of chat sessions currently in the queue, the average handle time of e-mails, and so on.
- CCPulse+ monitors the operational behavior of a contact center, such as the number of agents logged in, the number on calls, and average call-handling time. By combining business data with operational data, a contact center manager can also obtain an up-to-the-second view of the contact center.
- CCPulse+ includes wizards that guide you in creating customized views of real-time and historical data. Genesys eServices, Outbound Contact, and Universal Routing also provide their own out-of-box views tailored to monitor the effectiveness of a various product functions; for example, the effectiveness of an outbound campaign or routing strategy.

- CCPulse+ enables you to easily customize the objects monitored, the presentation format (for example, graph type), the color coding, and so forth, to best fit your needs.
- CCPulse+ includes wizards that enable contact center managers and supervisors to define thresholds and associated actions, such as alarms, so that a supervisor can be notified when an agent reaches a certain revenue goal, for example, or when a queue is backlogged.

Figure 7 shows a snapshot of a view created within the CCPulse+ application.

The left pane shows monitoring functionality. The right pane provides historical and real-time Solution Reporting.

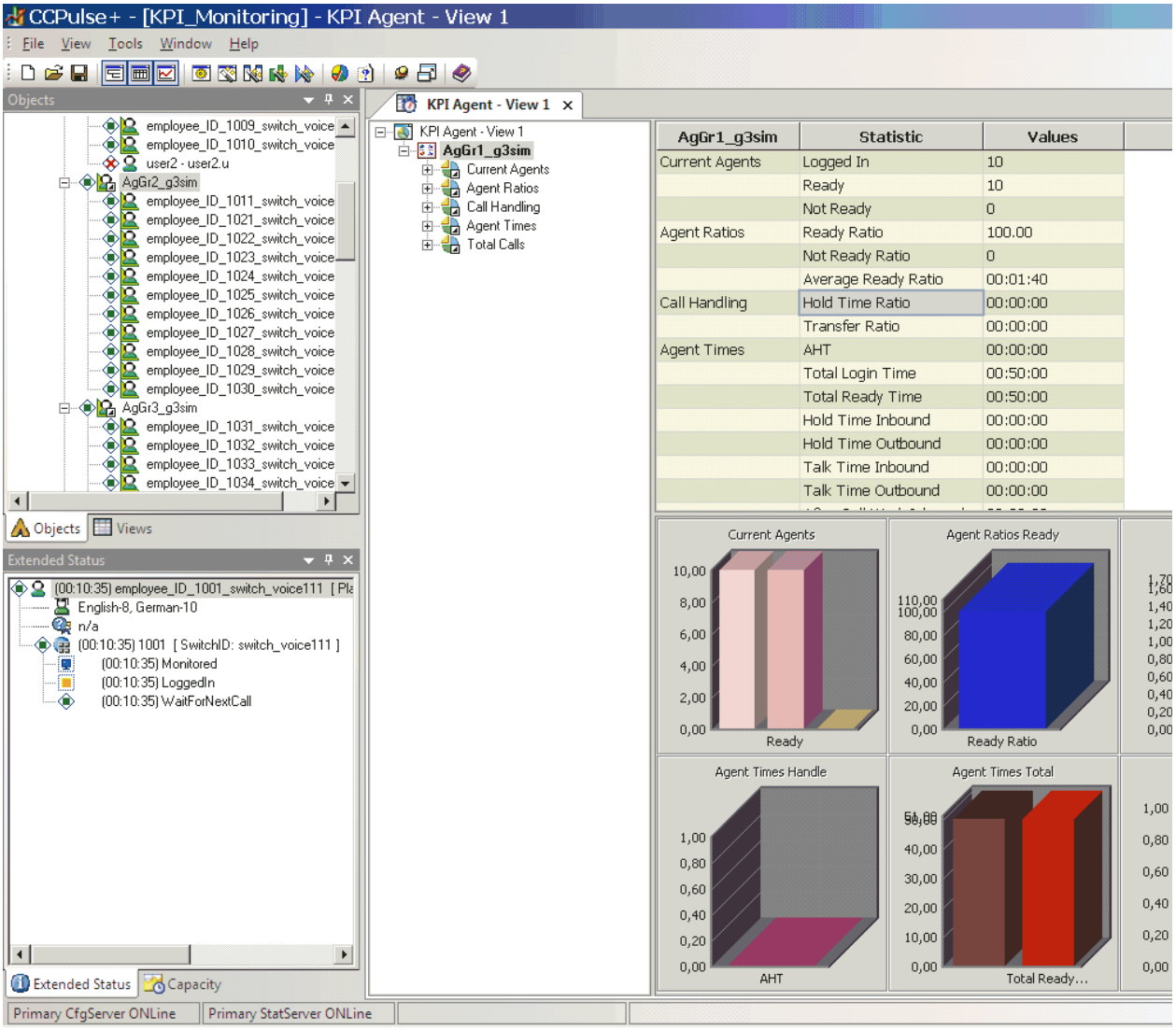

Figure 7: CCPulse+ View

CC Analyzer—Examining Historical Performance

CC Analyzer draws on the collected and aggregated statistical values for selected contact center objects. More specifically, CC Analyzer enables users to:

- Generate canned reports, that is, reports based on pre-made layout templates that are included with CC Analyzer.
- Design custom reports and custom metrics.
- Drill down data to the 15-minute level.
- Use Hyperion Intelligence Server to publish reports on the Web and schedule report generation.

CC Analyzer uses the Oracle EPM System 11.1.2 suite (specifically, Hyperion Interactive Reporting within that suite) to provide GUI access to the data and to customize data presentation.

Hyperion Clients—CC Analyzer Report Generator GUIs

Solution Reporting-optimized Data Mart content enables administrators to use Hyperion Interactive Reporting to either automatically generate reports from canned templates or build custom reports (see Figure 8) using both client/server based tools and web-based clients.

The Solution Reporting Databases

The Data Collection and Data Mart services use two databases:

- Operational Data Storage (ODS)—for statistics collection
- Data Mart—for report generation

Note: *At a minimum*, position these databases on separate physical hard drives or, better yet, on separate computers. Separating the read and write processes to and from each of the databases improves Solution Reporting performance.

Refer to the *Reporting 8.0 Deployment Guide* for additional important deployment considerations.

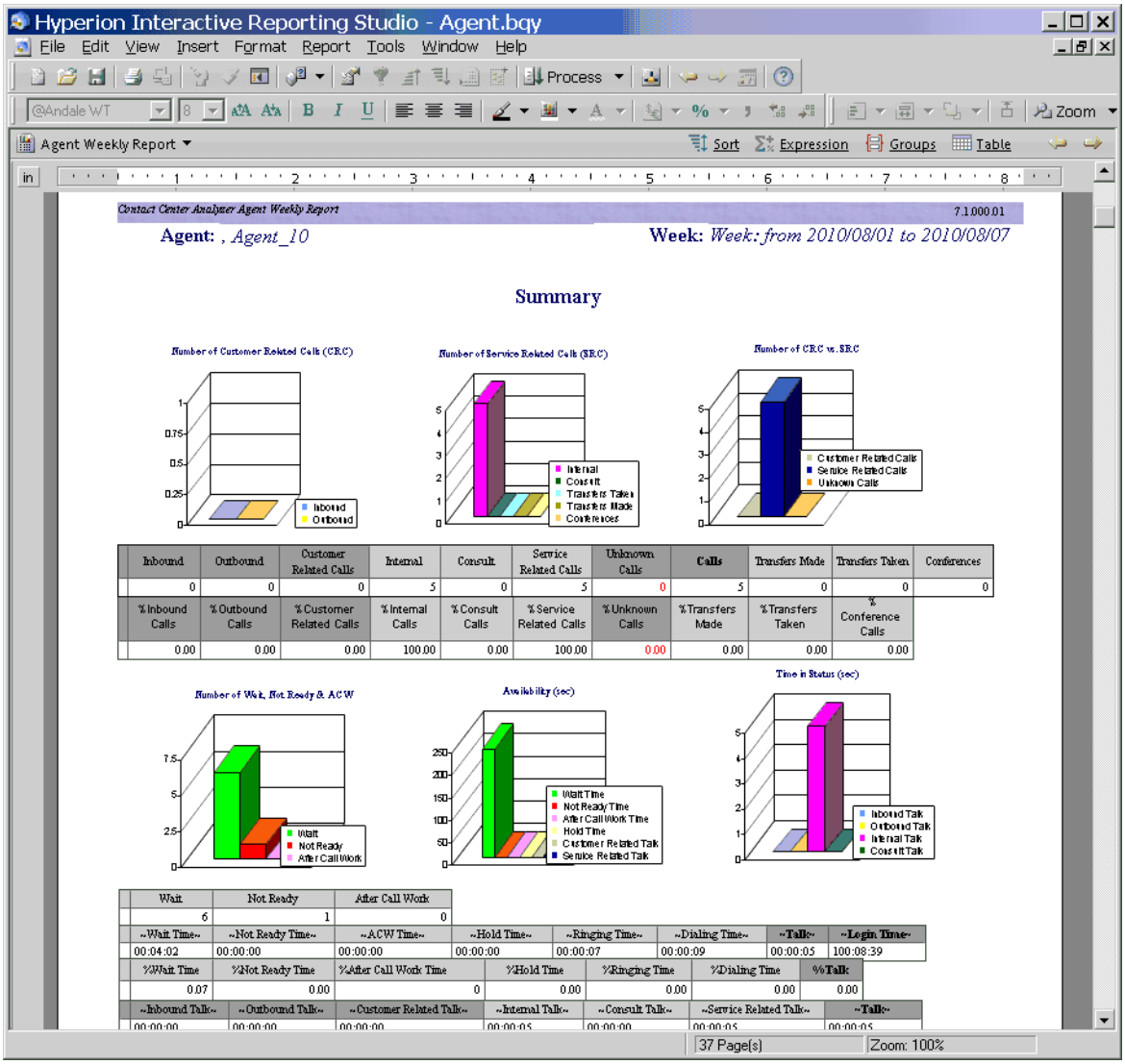

Figure 8: Hyperion Solution Reporting View

Chapter

2 Sources of Solution Reporting Data

This chapter provides an in-depth discussion of the sources of information for Genesys Solution Reporting and contains these sections:

Introduction

Genesys Solution Reporting collects, processes, organizes, and presents information about the behavior and performance of contact centers. More specifically, Solution Reporting collects information about interaction processing, agent performance, and behaviors of other contact center objects.

Information about interaction processing might include data about how many interactions pass through a contact center, arrival time at the contact center, their routing from one resource (for instance, a device) to another, their changing properties, and their termination.

Information about contact center objects might include the performance of resources and groups of resources, the most interesting of which (and most expensive) is the agent. Administrators might be interested in agent performance metrics such as average call-processing time, total time in the not-ready state, and so forth.

Physical Structure of Solution Reporting Sources

Figure 9 depicts the physical environment that enable this information to be gathered.

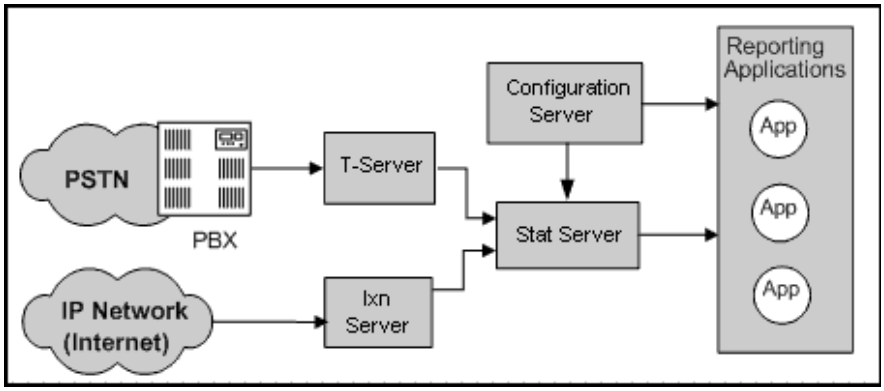

Figure 9: Sources of Solution Reporting Data

The main source of information about telephony interactions is Genesys T-Server, which communicates with telephony switching networks. T-Server tracks all telephony interactions and uses telephony events (TEvents) to report elementary interaction behavior to its clients. For object-centered data, Solution Reporting depends on combining events and producing metrics. Therefore, to report on telephony interactions in your contact center, you must deploy a Stat Server as well as a T-Server.

Note: Interaction Concentrator uses TEvents to store interaction-centered rather than object-centered reporting data.

Stat Server receives and processes raw information that it receives from T-Server, to meet the data requirements of object-centered Solution Reporting applications. By combining elementary events, Stat Server compiles a picture of the behavior of interactions and objects within a contact center and provides metrics to analyze performance.

Stat Server pulls information about the existence of objects such as agents, places, and groups from the Configuration Server. Then, using TEvents, Stat Server reconstructs the behavior for each object and makes that information available to the Reporting Layer. The Reporting Layer, in turn, reads information about the existence of objects from Configuration Server and about the status and behavior of those objects from Stat Server.

eServices-Based Solution Reporting

Genesys eServices/Multimedia is yet another source of interaction information. The eServices reports provide details about all interactions, including nontelephony interactions such as chat sessions, SMS, and e-mail. Along with the conventional telephony information and instant messaging processed by

Stat Server, Genesys eServices provides Internet-specific information that is stored in the Universal Contact Server database.

Genesys eServices uses the Interaction Server, to track e-mail, chat, SMS, etc. interactions. Stat Server uses data about these interactions to generate metrics on eServices interaction behavior.

Certain eServices reports require that you enable the Stat Server Java Extensions (SSJE) functionality. The extensions calculate data that is then supplied to Genesys Solution Reporting through Stat Server. The *Framework 8.0 Stat Server Deployment Guide* describes how to enable Java functionality in Stat Server applications.

Note: Internet Contact Solution and the inbound templates previously provided for Genesys Info Mart are not documented in this release of the *Reporting Technical Reference* series. For information about either, refer to previous releases of this document; namely, the *Reporting Technical Reference Guide for the Genesys 6.5 (or 7.2) Release*.

Logical Structure of Solution Reporting Sources

In contrast with Figure 9 on page 28, which illustrates the physical sources of Solution Reporting information, Figure 10 shows the logical structure of these sources. The latter, more abstract representation captures data models, data flows, and so forth.

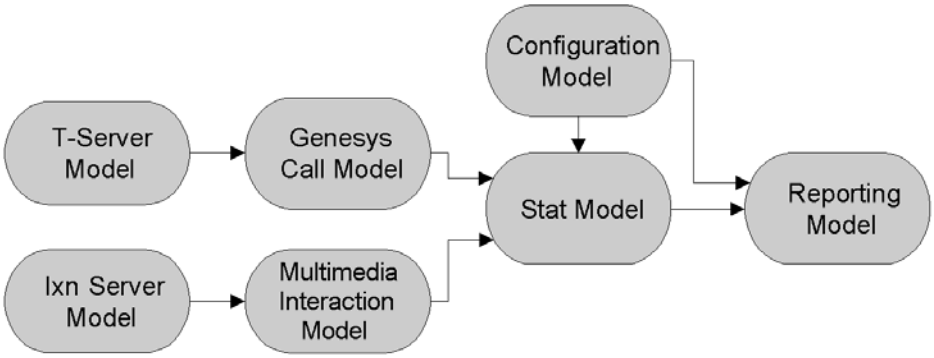

Figure 10: Logical Structure of Sources for Solution Reporting

The *Genesys Call Model*—an abstract representation of a switching network constructed and maintained by T-Server—is one foundation for these information sources. The Genesys Call Model draws on the T-Server Model and the Configuration Model, a representation of configuration data flow that is not described in this document. The Multimedia Interaction Model is another foundation of Solution Reporting information sources, with data flows that are specific to eServices data sources, such as e-mail. The Multimedia Data Model uses Interaction Server rather than T-Server, and, like the Call Model, also draws on configuration data.

Stat Server takes the data provided from the Genesys Call Model and Genesys Multimedia Data Model sources and operates as described in the Statistical Model—an abstract representation of statistical information in a contact center. The Solution Reporting Model then is constructed on the basis of the Statistical Model and the Configuration Model.

The Genesys Call Model

The Genesys Call Model is, in essence, a telephony model based on TEvents generated by T-Server. (References to T-Server include a Session Initiation Protocol T-Server, or SIP Server.) Genesys T-Server, a key element of the Genesys platform, serves as a gateway between the switching and computing environments by monitoring and controlling switching functions for different contact center applications. Figure 11 illustrates the T-Server environment.

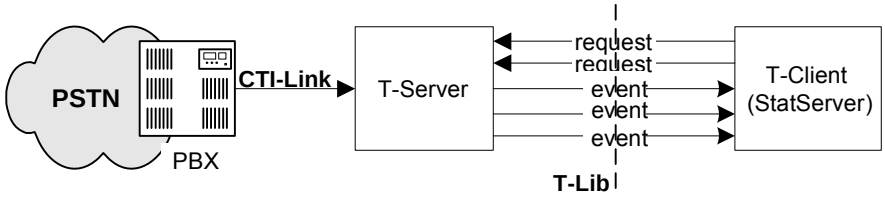

Figure 11: T-Server Environment

T-Server is connected to a switch (for instance, a PBX) through a CTI link and a communications protocol stack (with CTI at the top). Typically, the CTI link is vendor-specific. Genesys has implemented a separate T-Server for each CTI link. The T-Server veils vendor-specific switch features and provides access to information on the telephony side that is transparent to other Genesys applications. In other words, T-Server provides its clients with a unified interface based on a common telephony call model and implemented within the Telephony Library (T-Library). T-Library serves as a media-independent interface between a T-Server client, such as Stat Server, and a T-Server.

Note: This discussion assumes a telephony-based system. For a discussion of how Genesys eServices handles multimedia interactions, see “The Multimedia Interaction Model” on page 48.

T-Server sends events to its clients to inform them about processes within the switching domain. Likewise, T-Server clients send requests to T-Server to control switching functions. These requests are not relevant for Solution Reporting, because you need only be concerned with monitoring switching domain behavior. For more information on controlling functions, refer to the *Genesys 7 Events and Models Reference Manual.*

You must understand the call model to understand how telephony networks operate. From the technical point of view, think of a call model as an abstract,

virtual state machine representing users, terminals, and/or network behavior during the establishment, processing, and ending of an interaction.

This document considers only the basic principles of the Genesys Call Model. You can find a more detailed specification in the Framework T-Server documentation.

Call Model Structure

The Genesys Call Model is specified using objects that form a call model structure. This specification describes the evolution (or behavior) of the structure.

This structure is described using five types of objects:

- Device object
- Call object
- Party object
- Agent object
- Attached data object

Figure 12 depicts an example of the basic call model.

The call consists of two `Device` objects, identified by directory numbers **a** and **b.** `Device b` is associated with the `Agent` object, which represents an actual agent in a contact center. `Device a` is associated with a customer. So this call structure contains two participants.

The Call object, identified by **A,** is the core element uniting all other object types. For instance, call **A** might link to user data, represented in the figure by the `Attached Data` object. This call structure has two `Party` objects, identified by **(a, A)** and **(b, A).** These `Party` objects represent the combination of a `Device` object with its user and the `Call` object. The Party objects store information about states of the participants in the call.

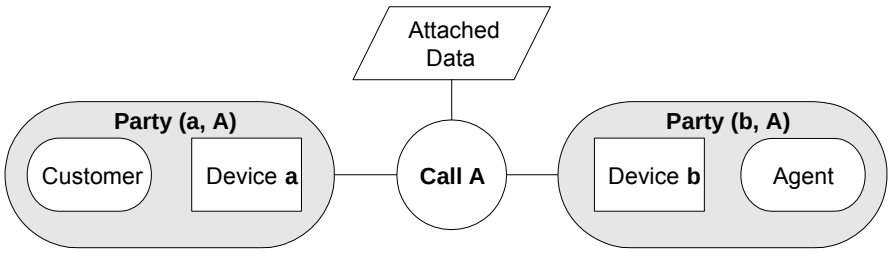

Figure 12: Structure of Simple Call

Call structure might vary over time. Object states might change during the progress of a call without changing the call's overall structure. For example, one participant can press the `Hold` button on the telephone device, putting the corresponding party into a `Held` state. Or call evolution might change the call structure. For example, transferring the call to another device deletes one `Party` object and creates a new one.

Apart from the simple call structure (depicted in Figure 12), the Genesys Call Model accommodates other, more sophisticated, call configurations. Figure 13

depicts a different call structure, a consultation call where `Device b` is connected to two separate calls: **A** and **B.** `Call A` represents the typical call between a customer and `Agent 1`. `Call B` represents a consulting call that connects `Agent 1` to Device **c** of `Agent 2`. Attached data might be available for both normal and consultations.

> Note: In general, attached data objects associated with different calls contain different information.

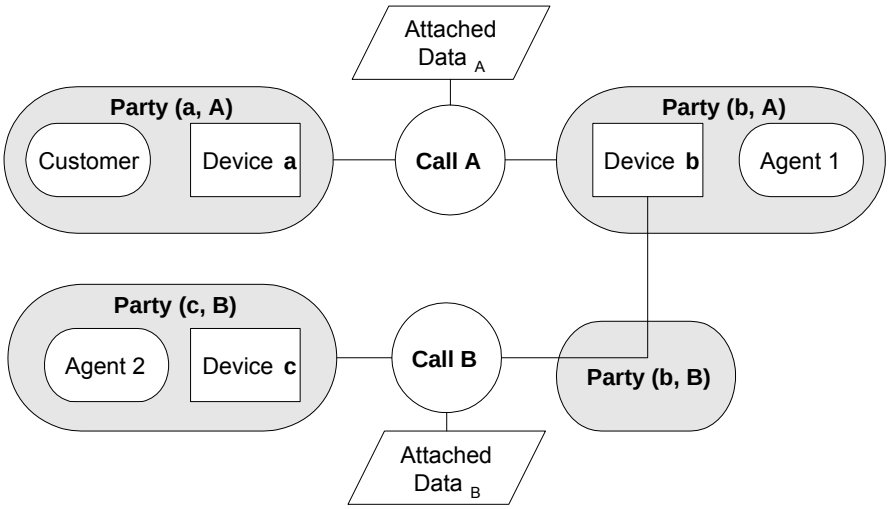

Figure 13: Consultation Call

Figure 14 demonstrates another sophisticated call configuration: a conference call. This structure occurs when two or more parties are simultaneously participating in a single call. This example also has three parties: `Agent 1`, `Agent 2`, and `Customer`.

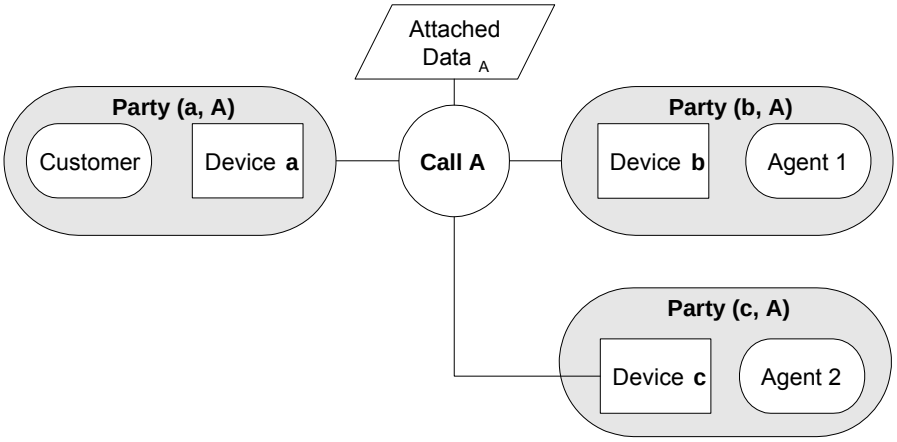

Figure 14: Conference Call

Triggering events, known as TEvents, track a call's evolution. Typically, TEvents signal changes in the status of a specific `Call` object. The relation of

events and objects is discussed in more detail later in this chapter. Specific event descriptions are provided during a discussion of `Call Model` objects in "T-Server Model" on page 43.

Telephony Events and Their Structure

T-Server uses TEvents to notify its clients about object changes. To receive TEvents, a client must be registered on a directory number (DN). The client receives events related to this DN. If several clients are registered on the same DN, T-Server distributes events related to this DN to all registered clients. Table 1 lists some fundamental event attributes.

`Note:` For the full set of event attributes, see the *Genesys 7 Events and Models Reference Manual*.

Table 1: T-Server Event Attributes

Attribute	Description
ThisDN	The directory number, or identifier, of the device to which this event is related.
OtherDN	The attribute indicating the other device involved in a call. For example, for a two-party call, `OtherDN` might indicate the called device.
ConnID	The current call identifier to which this event is related.
ANI	The Automatic Number Identification indicating the calling party's directory number. For example, for an inbound call, ANI indicates the customer's number. Once established, this attribute cannot be changed.
DNIS	The Dialed Number Information Service indicating the directory number to which the inbound call was made. For example, for an inbound call, DNIS indicates the 800 number that the customer calls. Once established, this attribute cannot be changed.
CustomerID	The attribute indicating the tenant ID in a multi-tenant environment.
CallType	One of the following five types of calls: `inbound`, `outbound`, `internal`, `consulting`, and `unknown`. The call-type attribute is set at the creation of the call and does not change during the duration of the call. The only exception is the call of type `unknown`, which might be changed once another call type is established

Table 1: T-Server Event Attributes (Continued)

Attribute	Description
PreviousConnID	The attribute linking two associated calls. For example, events relating to consultation calls might have `PreviousConnID` indicating the originating call.
AgentID	The parameter uniquely identifying an agent registered in the Automatic Call Distribution (ACD) queue.
CallState	An attribute that refines the reason for changing a party's state. An example is the `Redirected` attribute of the `EventReleased` TEvent, which indicates that a call has been redirected in accordance with a forwarding service.
WorkMode	The attribute indicating the agent's current work mode. For example, work mode can be `AfterCallWork`, signaling that the agent is still working on the call (for example, updating customer records) following call termination.
UserData	The attribute indicating user-related data (Attached User Data).

All TEvents are classified into one of the following groups:

- **Network Status Events**—TEvents indicating the status of the CTI link and connections between T-Server and its clients. If either status changes, the corresponding TEvents are sent to all T-Server clients. These events relate neither to a specific device nor to a specific call.
- **Call-Related Events—**TEvents indicating how a particular call is processed; therefore, the events must have `ThisDN` and `ConnID` attributes.
- **Device Feature Events—**Device-specific TEvents not relating to, or in the absence of, a particular call. For example, the `EventDNDOn` TEvent is invoked when the corresponding button is pressed on the telephony set. This group of events must have the `ThisDN` attribute but need not contain the `ConnID` attribute.
- **Agent-Status Events—**TEvents reporting agent behavior in using the special capabilities of a telephone set or soft phone. All events in this group contain the `ThisDN` attribute but not the `ConnID` attribute.
- **Special Event—**The `EventUserEvent` TEvent initiated by one client and distributed to all registered clients. This event is used for data exchange and to synchronize T-Server clients with T-Server.

Device Object

A Device object is an object representing a physical or virtual device. The three types of `Device` objects are Regular device, ACD Queue device, and

Routing Point device. The `Device` object is identified by a unique directory number (DN), which calls for three corresponding types of DNs: Regular DN, Queue DN, and Routing Point DN. In the following discussion, `Device` objects are sometimes referred to as DNs.

A terminal or physical device, such as a telephone on an agent desktop, is a `Regular Device` object. Regular DNs might also represent other types of devices such as chat DNs, e-mail DNs, IVR DNs, and so forth.

An ACD queue device corresponds to the ACD queue of a switch, which holds interactions waiting to be distributed to other devices. A built-in switch mechanism performs distribution. Sometimes, an ACD queue is simply referred to as a queue.

Routing to the Correct Device

A routing algorithm, or strategy, loaded on or applied to a Routing Point DN determines the most suitable target or destination DN for distribution out of a queue. Genesys Universal Routing, which includes both Enterprise Routing and Network Routing, supports this functionality. You can customize a routing strategy to account for many factors, such as agent availability, agent skills, cost, time of day, load, and so forth. The main difference between the ACD queue and Routing Point devices is *where* the routing algorithm is executed, whether by the switch or the Router.

Regular Device State Machines

A regular device emits device-specific events, whether or not the device participates in a call. These events are sometimes referred to as noncall-specific events.

Think of the structure of the regular device model as a device state machine (see Figure 15) that consists of three independent state machines working in parallel (see Figure 16).

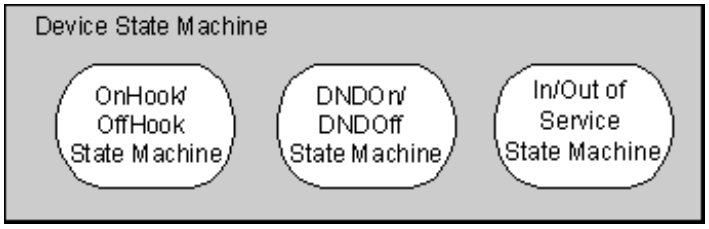

Figure 15: Device State-Machine Structure

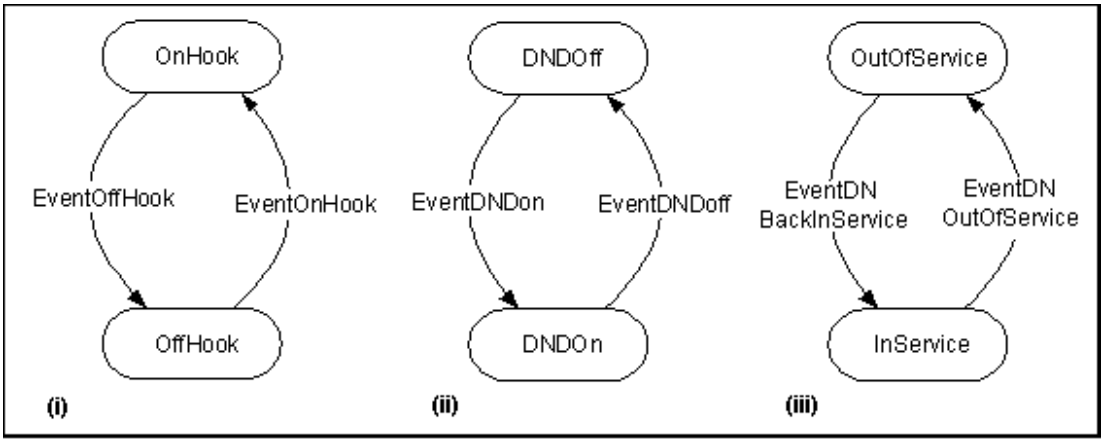

Figure 16: Regular Device-State Machines

The first machine **(i)** represents `On/Off Hook` functionality and corresponds to the state of a receiver, if any. When an agent hangs up the receiver, the state machine changes from the `OffHook` state to the `OnHook` state and triggers the `EventOnHook` TEvent. When the agent picks up the receiver, the state machine returns to an `OffHook` state and triggers the `EventOffHook` TEvent.

The second state machine **(ii)** represents do-not-disturb (DND) functionality triggered when the agent presses the Do Not Disturb button on a telephone set. Two events, `EventDNDOn` and `EventDNDOff`, report state changes. The device cannot receive any calls while in a `DNDOn` state.

Agent Object

The *Agent object* is associated with a contact center operator (a human being) using a regular device. However, the agent appears to the Genesys environment only when operating equipment attached to a device, such as when the agent presses the `Ready`, `NotReady`, `LogIn`, and `LogOut` buttons.

`Note:` Genesys SoftPhone, a third-party desktop application that provides agents with telephone functionality by way of a graphical user interface, also emulates this functionality.

In order to cover a wide range of vendor ACD models, the generic T-Server model is represented as a collection of independent state machines running concurrently. Two such machines appear in Figure 17. Figure 18 presents these state machines in more detail.

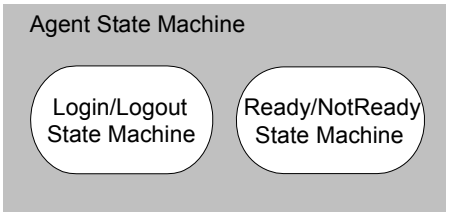

Figure 17: Structure of Agent State Machines

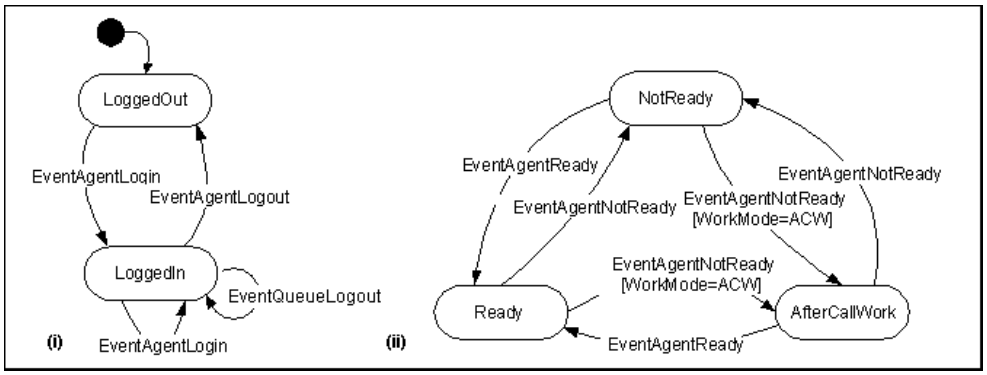

Figure 18: Agent State Machines

Login/Logout State Machine

The first state machine **(i)** in Figure 17 illustrates login behavior. When the agent performs a login operation, it triggers the `EventAgentLogin` TEvent and the agent's state machine changes to a `LoggedIn` state. While in this state, the agent can perform other login operations to another ACD Queue illustrated by loop transitions. A logout operation triggers the `EventAgentLogout` TEvent, and the state machine returns to a `LoggedOut` state. Note that as a rule, while the agent is in the `LoggedOut` state, the device cannot receive calls from ACD queues and might receive only direct calls (including those routed by URS).

Ready/Not Ready State Machine

The second state machine **(ii)** depicts the readiness of an agent—usually triggered by pressing the `Ready/NotReady` button on the telephone set. The current state is indicated by a special marker on the set, such as the presence, or absence, of a blinking arrow. The state machine in Figure 18 shows three states: `NotReady`, `Ready`, and `AfterCallWork`. Each transition from one state to another triggers a TEvent. For example, a transition to an `AfterCallWork` state triggers the `EventAgentNotReady` TEvent and sets the `WorkMode` attribute to `AfterCallWork` (ACW).

Note: The second state machine **(ii)** has neither initial nor final states (represented in **(i)** by a black circle). That means that during switch and/or T-Server initialization, the state can be set randomly.

Keep in mind that representation of the `Agent` object at T-Server level is very limited. The `Agent` object in the Genesys environment is merely an extension of the regular device. Indeed, knowing that an agent pressed the `Ready` button does not truly indicate that agent's availability to accept calls. The agent might subsequently step away from his/her desk or otherwise be unable to accept the next call.

Figure 19 depicts the agent model. Suppose that an agent has two telephone sets. That means two DNs are associated with that agent. Each DN might be associated with one or more parties.

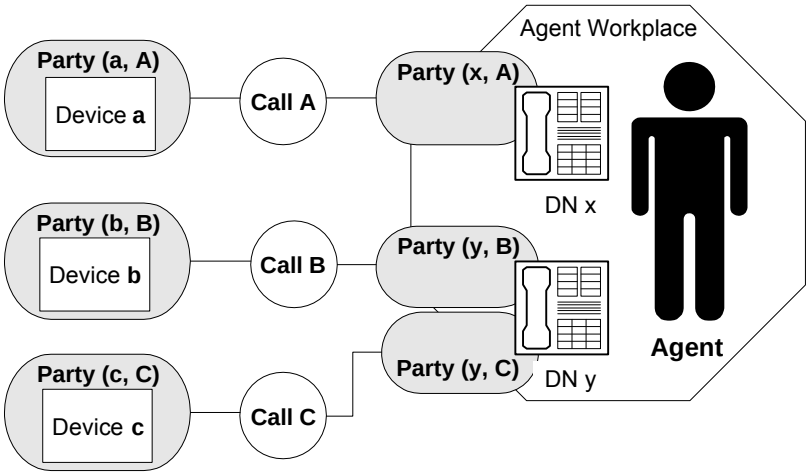

Figure 19: Agent's Environment

In our example, DN *x* participates in one call and DN *y* participates in two calls, one of those being a consultation. The real status of the agent (`Ready` or `NotReady`) is constructed based on three Party state machines, two Device state machines, and two Agent state machines. The `Ready/NotReady` Agent state machine describes agent behavior related to the corresponding DNs; in a case such as that shown in Figure 19, the state of the `Agent` object is constructed by compiling behavior for all three calls and presenting it as though it came from one DN.

A more adequate agent model can be built on top of these state machines by also considering the call-related states of the agent device(s). Later, you will see how the agent model becomes more sophisticated within Stat Server.

Call Object

The function of the Call object is to group all devices (participants) in the call. The Call object is uniquely identified within T-Server by the `ConnID` identifier, which might be present in events. The ConnID might not have a counterpart in the switching domain. The Call object is not associated with any particular state machine, so it has no states. However, the total state of the call can be determined by the states of all associated parties.

The Call object has some inherent attributes such as call type—inbound, outbound, internal, consulting, or unknown. A `Call` object is labeled *inbound* when the originating device of the call falls outside the domain controlled by T-Server. If the originator of the call is within T-Server's domain and the destination is outside, the call type is *outbound*. A two-party call with both parties falling within T-Server's domain is *internal*. And a call originating from a device already participating in an existing call is termed *consulting*. If T-Server cannot identify the call type as one of these four, its attribute is set to *unknown*.

The Call object might also have ANI and DNIS attributes (see Table 1 on page 33). If it does, these attributes are set at call creation and usually do not change for the duration of the call.

Party Object

The Party object represents the association between a call and a device. The `Party` object exists only if the corresponding `Call` and `Device` objects exist within T-Server. Therefore, the `Party` object is entirely identified by the pair `ConnID` and DN.

Three types of state machines can show `Party` object behavior: regular device, ACD queue, and Routing Point.

State Machine for a Party Object on a Regular DN

The simplified version of a state machine corresponding to a party on a regular DN is depicted in Figure 20. In the initial state, no relationship exists between a `Device` object and a `Call` object; that is, a `Party` object does not exist. A `Party` object is created when either the `EventDialing` or `EventRinging` TEvent is triggered; the state machines assume the `Dialing` or `Ringing` state respectively.

Party-object transitions from state to state trigger the TEvents shown in Figure 20.

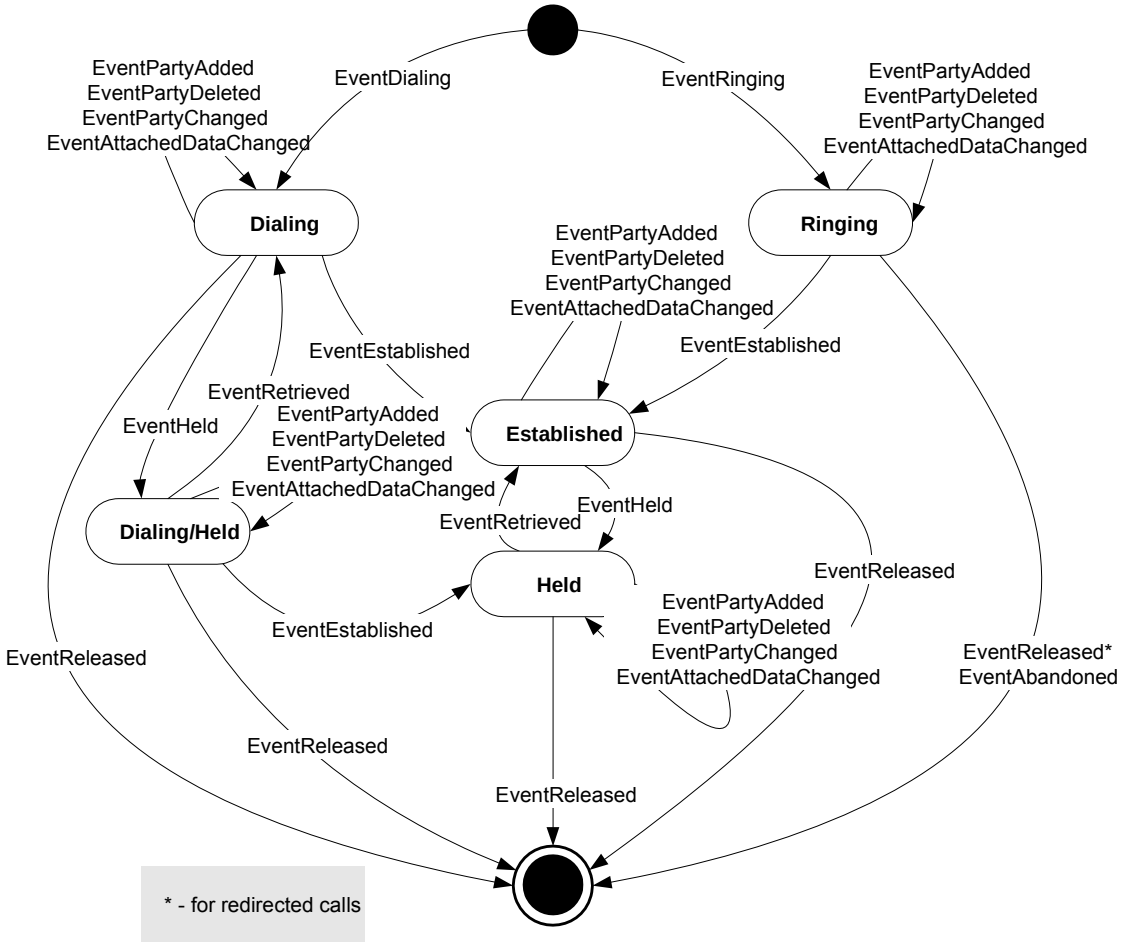

Figure 20: State Diagram for a Regular Party

Note: In Figure 20, one transition labeled by several events actually represents several different transitions with the corresponding events. For example, the transition labeled:

```
EventPartyAdded
EventPartyDeleted
EventPartyChanged
EventAttachedDataChanged
```

in the upper right-hand corner for the `Ringing` state should actually be depicted as four separate transitions. Likewise for the other five multilabeled transitions in this figure.

- A `Dialing` state occurs when the device requests a connection service and waits for a reply.
- In the `Ringing` state, the device alerts (rings) the agent that an attempt to connect a call to the device is being made.

- The `Established` state occurs when the device actively participates in the call; that is, the device is physically connected to a voice stream.
- The `Held` state occurs when the device is inactive, not participating in the call, has no voice stream.
- Similar to the `Held` state, the `Dialing/Held` state occurs when the transition to `Held` state happens during dialing (before a connection is established).

Note: The loop transition labeled by the `EventPartyChanged` TEvent might terminate one party and add another party on another call. This might happen for consultations in a two-step transfer or in a conferencing procedure.

State Machine for a Party Object on a Queue

The state machine for a `Party` object connected to an ACD queue is shown in Figure 21. The state machine exists in the `Queued` state as well as in initial and final states. A `Party` object is created when the `EventQueued` TEvent triggers and the state machine enters the `Queued` state, indicating that an interaction (a call, for example) will wait in queue for distribution to another device. Successful distribution triggers the `EventDiverted` TEvent, and the `Party` object ceases to exist (that is, it enters its final state—nonexistence). The call might also leave the ACD queue abnormally, if, for example, the caller hangs up the receiver, which triggers `EventAbandoned`.

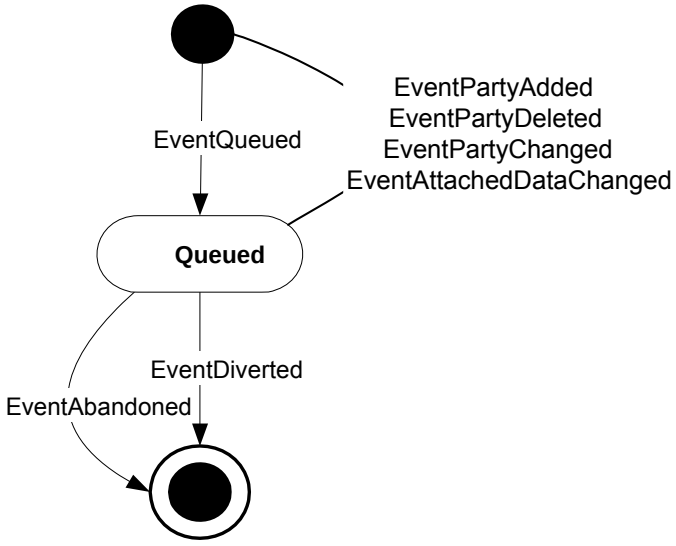

Figure 21: ACD Queue Party State Diagram

State Machine for a Party Object on a Routing Point

You can see a simplified version of a state machine for a `Party` object related to a Routing Point device in Figure 22. It resembles the state machine for `Party` objects related to ACD queues in Figure 21; however, Figure 22 illustrates

different TEvents. Furthermore, the `Queued` state now has its own state machine that indicates when treatments (playing a music file while the call is waiting, for example) are applied or removed.

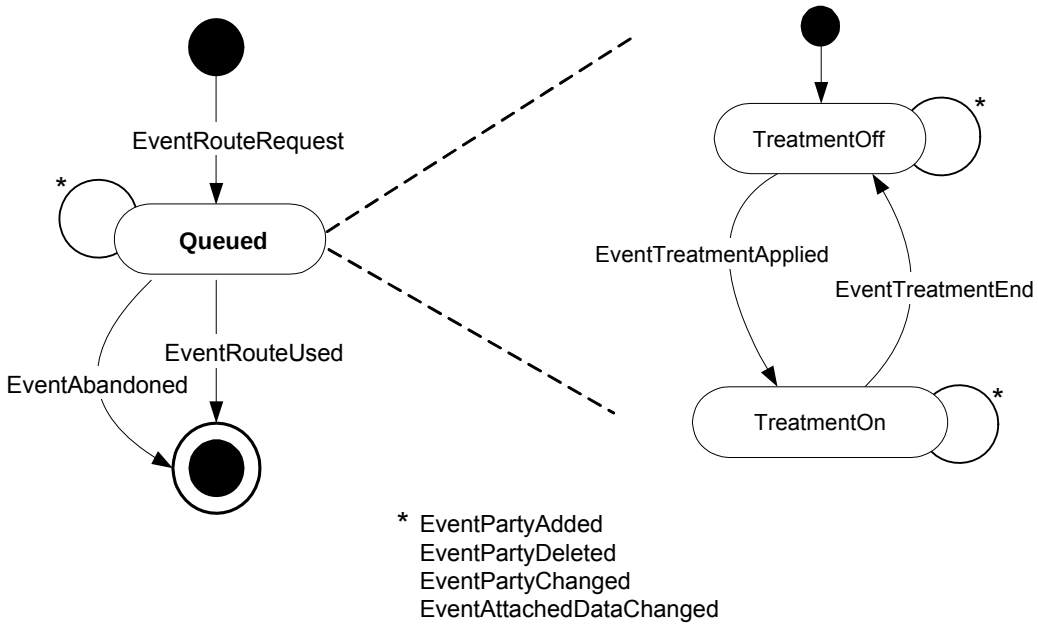

Figure 22: Party State Diagram for a Routing Point

Attached Data Object

The `Attached Data` object, which is associated with a `Call` object, collects and/or manipulates user data during call processing. The `Attached Data` object is created when the `Call` object is created. Initially, it might contain a null value.

User data is organized into a list of key-value pairs called a TKV-List. Each key-value pair structure consists of a character key plus a character, integer, list, or binary value. The following is a sample TKV-List with three TKV pairs:

```
("CS","Platinum")
("Service","E-Mail")
("Revenue","2459.29")
```

The first pair identifies the customer segment as Platinum; the second identifies the current interaction as e-mail; the third presents this interaction's revenue.

Any participating party might add or modify data through T-Server and a third-party control technique. When one participant modifies attached data, all other call participants receive the `EventAttachedDataChanged` TEvent. More precisely, the event is received by clients registered on participating DNs.

Network Object

T-Server conveys information about the status of the CTI link and connections between T-Server and its clients when certain system events are triggered. These events stem from a `System` object called the Network object, which operates according to two independent state machines (see Figure 23).

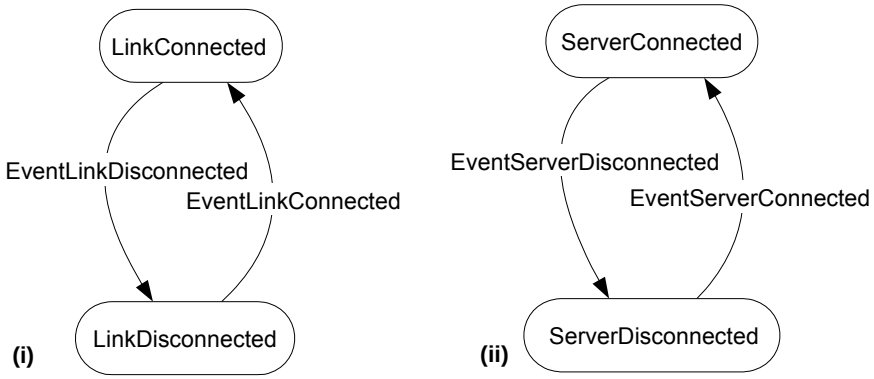

Figure 23: Network State Machines

The first state machine **(i)** models a CTI link that might exist either in connected or in disconnected state. The status of a link between T-Server and a switch (CTI link) is reported by the `EventLinkConnected` and `EventLinkDisconnected` T-Events.

The second state machine **(ii)** depicts the status of a link between T-Server and its client: either in connected or in disconnected state. The `EventServer Connected` and `EventServerDisconnected` TEvents report status changes.

T-Server Model

Conceptually, the whole T-Server environment is as a container for objects and object configurations (see Figure 24).

T-Server contains configurations comprising Call Model objects (`Call`, `Party`, `Device`, and `Attached Data` objects). For instance, in Figure 24, `Call` object **A** represents a typical two-party call with regular devices. `Call` object **B** represents a conference call with three participants and attached data. `Call` objects **C** and **D** share the same regular device, which occurs when one `Call` object, say **D**, is a consultation and the party associated with `Call` object **C** and its associated device is placed in `Held` state.

`Call` objects **E**, **F**, and **G** are waiting in an ACD queue device labeled **Q.** `Call` objects **H** and **J** wait in a queue on a Routing Point device labeled **RP.**

The `Network` object can trigger events about the status of connections; namely, `EventLinkConnected`, `EventLinkDisconnected`, `EventServerConnected`, and `EventServerDisconnected`.

Figure 24 also shows that T-Server monitors DNs that are not currently involved in calls and sends events concerning them to its clients. For instance, regular devices not participating in calls might send out agent-related events (`EventAgentReady`, `EventAgentNotReady`, `EventAgentLogin`, `EventAgentLogout`), device-related events (`EventDNDOn`, `EventDNDOff`, `EventOnHook`, `EventOffHook`), and the special user event, `EventUserEvent`.

Note: Any of these calls can also contain attached data. For simplicity, Figure 24 does not show attached data.

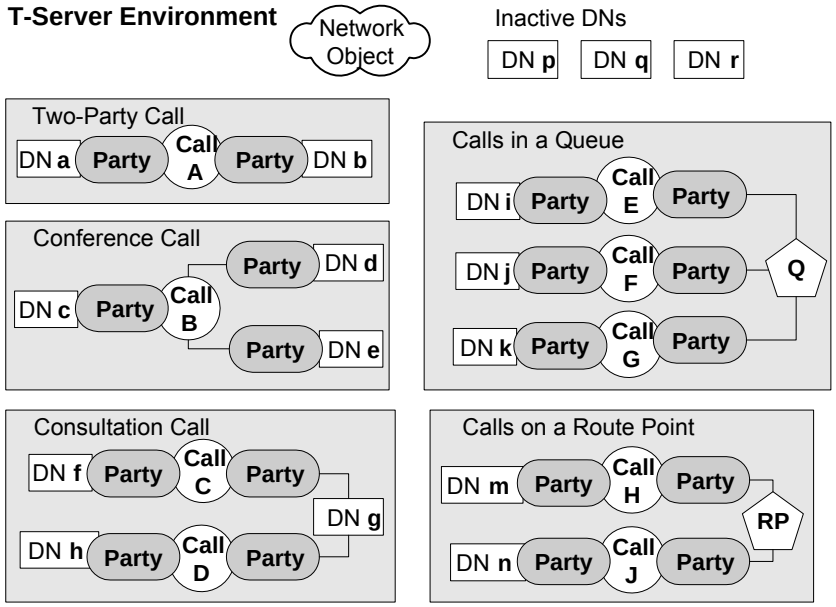

Figure 24: T-Server as a Container of Objects

Clients "see" all these call structures through events sent by T-Server. More precisely, they observe changes to the structures, but not the structures themselves. For instance, a client receiving the `EventEstablished` TEvent is informed that the corresponding party has changed its state from a `Dialing`, `Ringing`, or `Ringing/Held` state to an `Established` state, but the exact state change is not identified.

The following section presents some examples of these call structures in action.

Example 1 Figure 25 demonstrates how a connection is established between two devices.

Suppose you have two devices (DNs), **a** and **b.** Device **b** is associated with an agent who performs a login operation. Device **a** initiates a connection to device **b.** In the initial state (1), there are only `Device` objects—no `Call` objects and no `Party` objects. Device **a** dials the number for device **b**. The structure transitions to state (2) where `Call` object **A** with `Attached Data` object **AD**, and `Party` object **(a, A)** have been created. The party is set in `Dialing` state. An `EventDialing` TEvent triggers this transition (i), with `ThisDN` equal to **a.**

In the next step (3), a `Party` object **(b, A)** between call **A** and device **b** is created and set to a `Ringing` state. The `EventRinging` TEvent triggers this transition (ii) and `ThisDN` is now set to **b.**

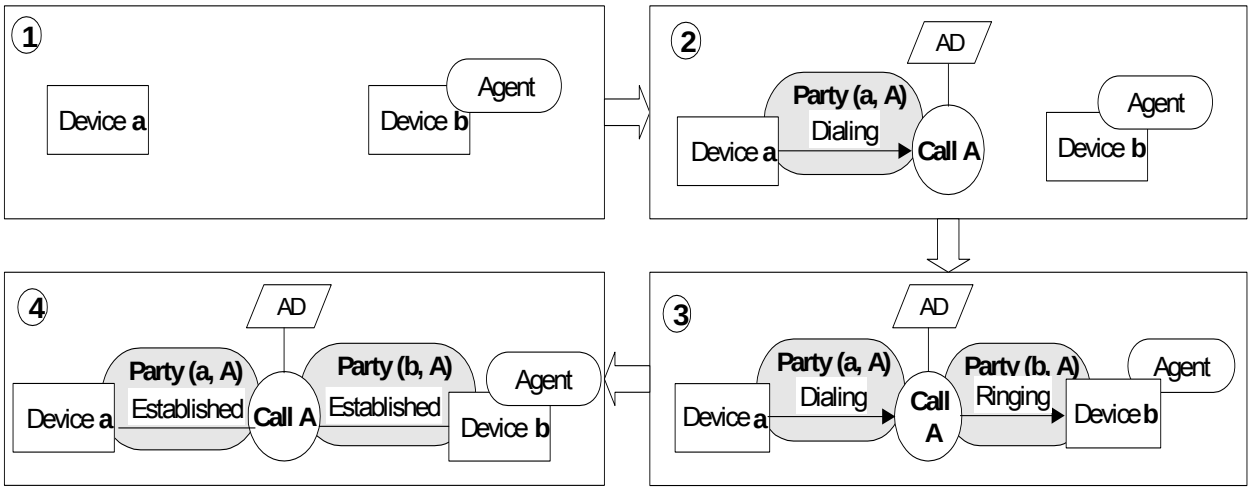

Figure 25: Connecting Two Devices

Finally, after device **b** answers, the configuration is transferred into its final state (4) representing an established connection and a voice stream between devices **a** and **b.** Two `EventEstablished` TEvents are triggered to change both parties to an `Established` state.

Example 2 Figure 26 illustrates the completion of a two-step transfer.

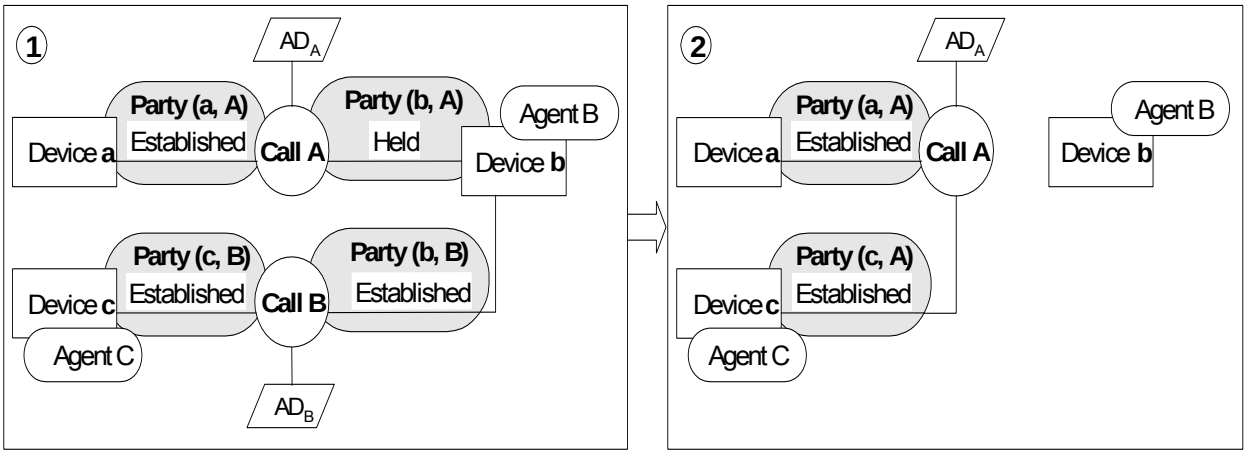

Figure 26: Completing a Two-Step Transfer

In the initial state (1), Agent **B** handles two calls—one of them a call **(A)** to a customer, the other **(B),** a consultation with Agent **C.** `Call` object **A** is on hold (its party in `Held` state) as Agent **B** is currently consulting with Agent **C.** During consultation, Agent **B** decides to connect the customer to Agent **C** and leave the conversation. To accomplish this, s/he must transfer the call. During this transition (2), `Call` object **B** is deleted along with its attached data (**AD_B**). Parties **(b, A)** and **(b, B)** are also terminated when two triggered `EventReleased`

TEvents and corresponding `ThisDN` parameters are sent. The **(c, B)** party was terminated and a new party **(c, A)** was created with a new `ConnID` parameter when T-Server sent the `EventPartyChanged` TEvent.

Example 3 Figure 27 illustrates how a consultation evolves into a conference call. It resembles the previous example except that agent **B** does not leave the call; therefore, the **(b, A)** party is not removed.

The TEvents triggered also differ. For parties **(a, A)** and **(b, A)** an `EventPartyAdded` TEvent is triggered indicating that another participant has joined the conversation.

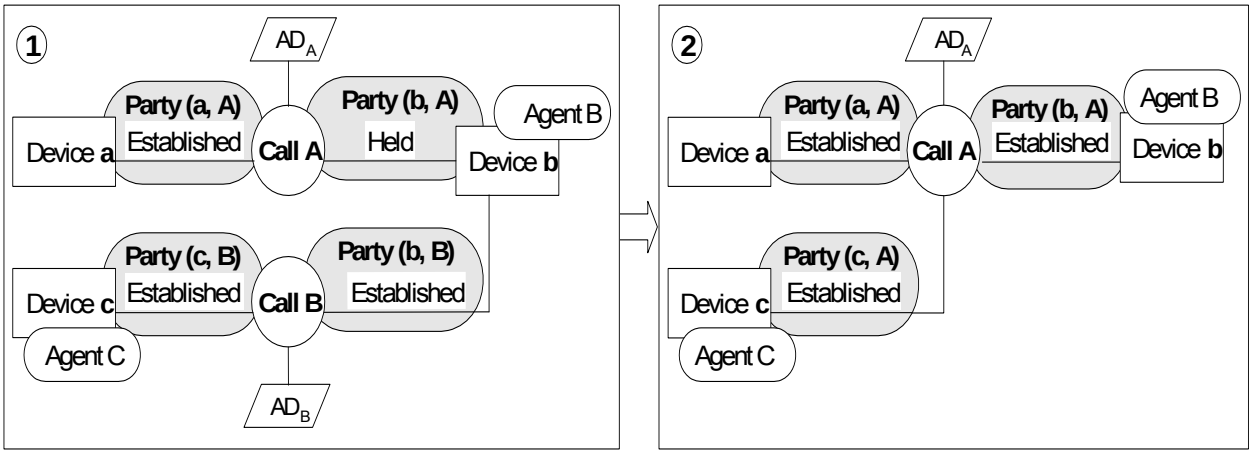

Figure 27: Evolving from a Consultation to a Conference Call

Example 4 Figure 28 illustrates routing a call. In the initial state **(1),** the call is connected to ACD queue device **b.** The **(a, A)** party is in `Dialing` state and the **(b, A)** party is in `Queued` state, waiting for distribution to an agent.

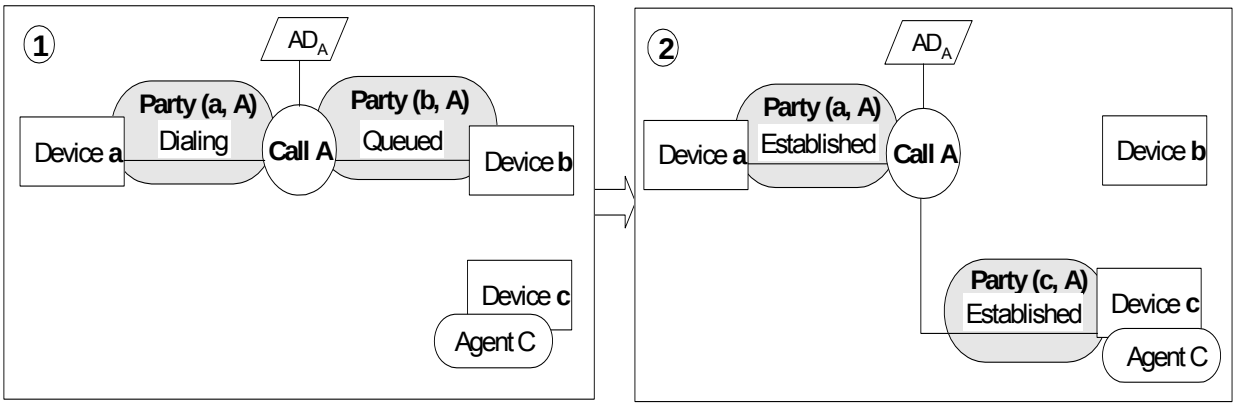

Figure 28: Call-Routing Scenario

Suppose that Agent **C** with device **c** is logged in to ACD queue device **b.** Further suppose that the ACD queue device distributes the call to device **c.** In final state **(2),** the call is connected to device **c** and both parties are in the `Connected` state. The transformation is triggered by the following events:

- The **(b, A)** party triggers an `EventDiverted` TEvent and disappears.

- The **(c, A)** party is created and enters `Ringing` state, triggering the `EventRinging` TEvent.
- After the call is answered on device **c**, parties **(a, A)** and **(c, A)** are transferred to `Connected` state and both trigger `EventEstablished` T-Events.

The routing via Routing Point device is similar to routing via ACD except that the disappearing party **(b, A)** triggers the `EventRouteUsed` TEvent.

Note: For more call scenarios, see the *Genesys 7 Events and Models Reference Manual*.

Are TEvents Suitable for Solution Reporting?

As you have seen, the main purpose of T-Server is to send TEvents to registered clients that make visible and controllable the processes occurring within a switching network. T-Server generates TEvents in accordance with the Call Model, an abstract representation of the switching network objects and processes that are of interest to the clients.

TEvents are an excellent data source for call-centered Solution Reporting, as done by Interaction Concentrator (ICON) and Genesys Info Mart (GIM). But does this functionality provide data usable for object centered Solution Reporting? In fact, TEvents are raw material that must be refined and processed before it is suitable for Solution Reporting on objects instead of calls.

For example, TEvents themselves do not say much about processes in a switching network. One TEvent might correspond to any number of different actions within a network. Moreover, an event often tells about only the part of a network action that is linked to one device or party. (Note that a party is determined and accessed by its DN.)

For example, a transfer is reported by four events related to the various parties involved. The four events can be combined to recreate the call history, as in ICON/GIM-based reporting, or they can be synthesized and interpreted to recreate the actions and statuses of the contact center objects involved.

Reporting on actions and states of objects such as agents, workplaces, groups, and so forth, require such synthesized information so they can then present it in natural terms. For example, to understand agent activity, a report on agent after-call work status is more useful than a sequence of `EventReleased` and `EventAgentReady` TEvents. In addition, the synthesis required for such object-centered Solution Reporting also enables you to create reports on the number and duration of certain object actions or statuses—that is, metrics.

Clearly, there is a significant gap between what T-Server provides and what object-centered Solution Reporting applications require. Genesys Stat Server fills this gap. For a discussion of Stat Server use of TEvents, see “The Statistical Model” on page 68.

The Multimedia Interaction Model

This section provides an in-depth discussion of the sources of multimedia interaction information for Solution Reporting. This section describes Genesys eServices from a Solution Reporting point of view. For more information on Genesys eServices capabilities and components, refer to the eServices documentation set.

This section covers these topics:

Note: This section covers the multimedia interaction types provided by Genesys eServices. If you are using the Open Media functionality to create custom reporting on additional media types, refer to the *Customization* book of the *Reporting Technical Reference* series.

Multimedia Interaction Model Overview

Genesys eServices enables users to route, track, and report on multimedia interactions. Because these interactions do not arrive through a telephony switch and, notably in the case of e-mail, are not necessarily handled as they arrive, they require a different interaction model than the telephony interactions analyzed in the preceding sections.

Multimedia Interaction Environment

The environment in which eServices operates is shown in Figure 23 on page 43. The eServices environment contains several media servers connected to an IP network. Figure 23 shows the Web API Server, which sends multimedia on to one of two media servers, the E-mail Server Java or the Chat Server. E-mail also enters Multimedia through POP3 servers and is sent to E-mail Server Java. These media servers are connected to Interaction Server, which processes all interactions in a unified way. The E-mail Server Java and the Chat Server are also connected, both directly and through Interaction Server, to Universal Contact Server (UCS), which stores all working information in the Universal Contact Server database. For example, UCS and the Universal Contact Server database store all chat transcripts.

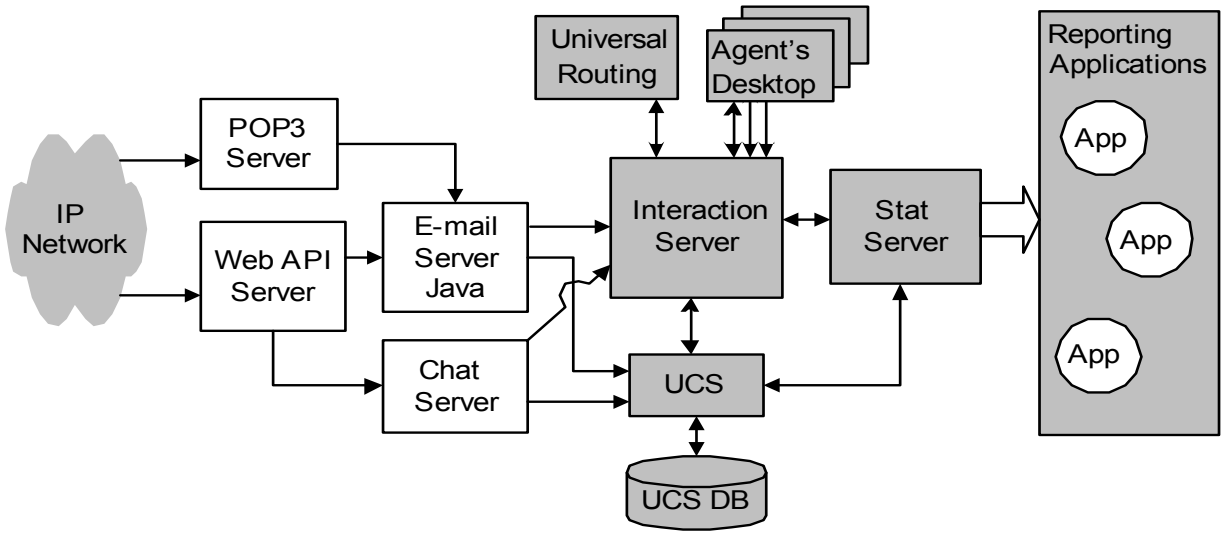

Figure 29: Genesys Multimedia Environment

Interaction Server

Interaction Server is also connected with Universal Routing 7.0.1 or higher, which determines the destination of incoming interactions. Agents in Genesys eServices are represented by their desktops, which are connected to Interaction Server.

Note: Releases of Universal Routing prior to 7.0.1 are not compatible with Genesys eServices. For more information on using Universal Routing with eServices, refer to the *Universal Routing 8.0* documentation set.

Interaction Server reports on all its interaction processing, in the form of Events, to Stat Server using a special reporting link. After some preprocessing, this reporting information is transferred to the Solution Reporting applications.

Universal Contact Server

UCS writes information about multimedia interactions to the Universal Contact Server database. The data includes:

- Data about each interaction, such as the InteractionID and the various objects with which the interaction has been associated
- The category to which each interaction belongs to (chat or e-mail, inbound or outbound, and so on)
- A library of possible auto response messages, suggested or previously used responses for specific situations, and so on
- E-mail threads, chat transcripts, replay/response links, and so on

The following subsections describe how interactions are processed for both e-mail and chat.

Processing of E-Mail Interactions

Figure 30 illustrates the typical processing of inbound e-mail.

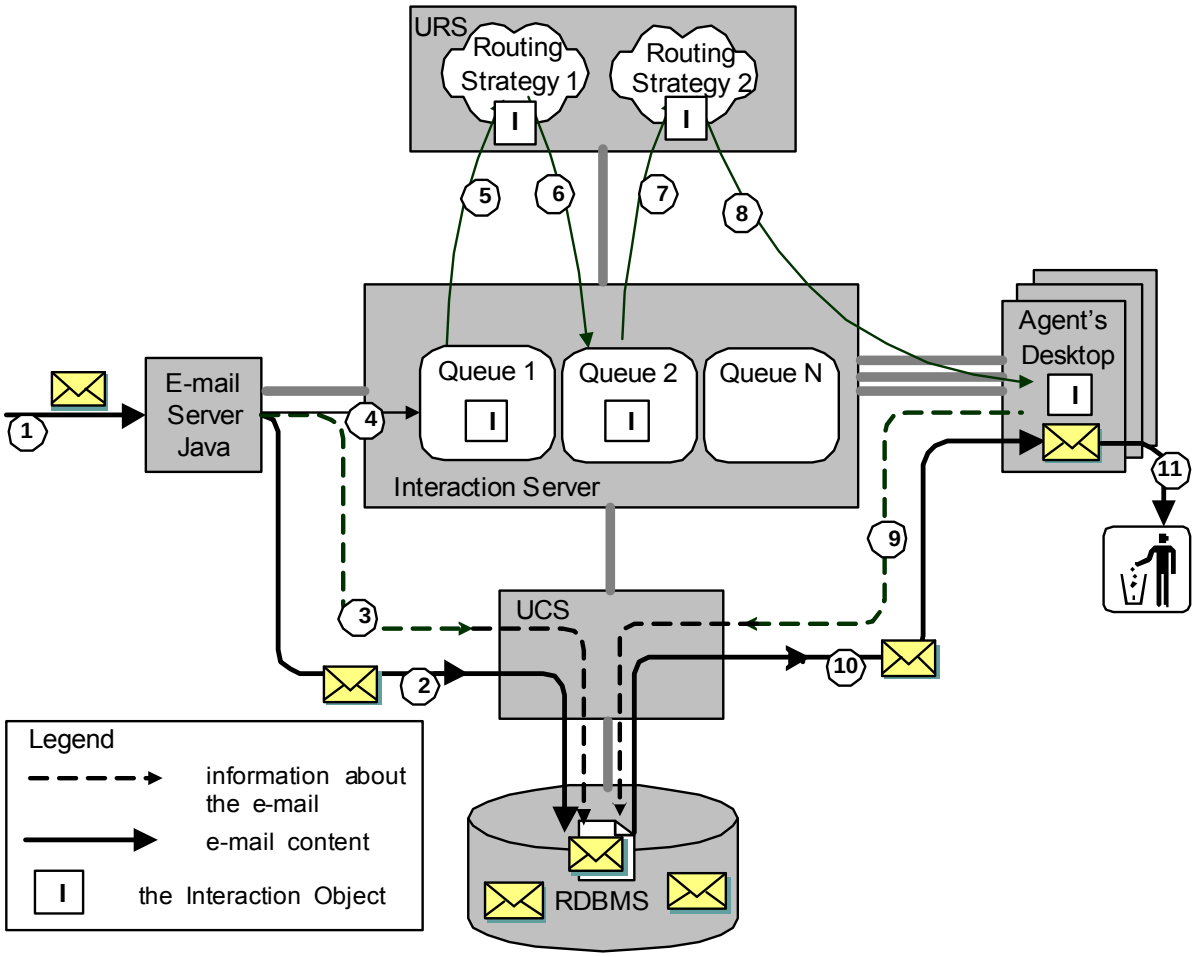

Figure 30: Processing of E-Mail Interactions

1. E-mail Server Java takes a new e-mail from the Internet POP server.
2. UCS records the e-mail in the Universal Contact Server database.

   Some contact information taken from the e-mail header is stored in the database, such as names, addresses, and phone numbers. If the contact already exists in the database, this information can be matched to a contact history that already exists.
3. At the same time, Interaction Server creates a corresponding interaction object and places it in `Queue_1`.

4. Universal Routing Server (URS) takes the interaction from the queue and processes it with the aid of `Routing Strategy_1`. The strategy processes the interaction. It might invoke some external processor, such as Classification Server.
5. Based on the results of the routing strategy, URS places the interaction into `Queue_2`.
6. URS takes the interaction from `Queue_2` and processes it using `Routing Strategy_2`. The processing also might involve external processing, such as sending an acknowledgement or an automatic reply, or applying a screening rule.
7. URS, guided by `Routing Strategy_2`, transfers the call to an agent desktop.
8. The agent receives information about the incoming e-mail and requests the e-mail content from UCS.
9. UCS retrieves the e-mail content and delivers it to the agent's desktop.
10. The agent works with the e-mail and, after processing, decides to stop processing this interaction. The interaction and the corresponding e-mail content are cleared from the agent's desktop.

> Note: The e-mail content and corresponding control information are still kept in the Universal Contact Server database.

This is one of many possible scenarios for e-mail processing. For example, the agent might decide to return the e-mail interaction to a queue for further processing, or the agent might create an outbound e-mail interaction to reply to a customer. For a detailed example, see "E-Mail Processing Example" on page 64.

You can configure some e-mail processing scenarios, such as acknowledgement and autoresponse, that do not require agent involvement. These e-mails are generated using routing strategies.

Processing of Chat Interactions

The processing of a chat interaction is similar to processing an e-mail. However, some differences exist because of the nature of the chat medium. Figure 31 shows a typical scenario for chat processing.

1. Chat Server receives a new chat session from the Internet.
2. Interaction Server creates a new interaction of the chat type and places it in a queue, here `Queue_1`.
3. UCS stores control information about the interaction, such as the address of Chat Server and the chat Session ID, in the Universal Contact Server database.

4. URS takes the interaction from the queue and processes it using `Routing Strategy_1`.
5. The routing strategy determines which agent should handle the interaction and transfers the interaction to the agent's desktop.

   The agent receives all information related to the interaction. The chat invitation supplies the Chat Server address and Session ID.

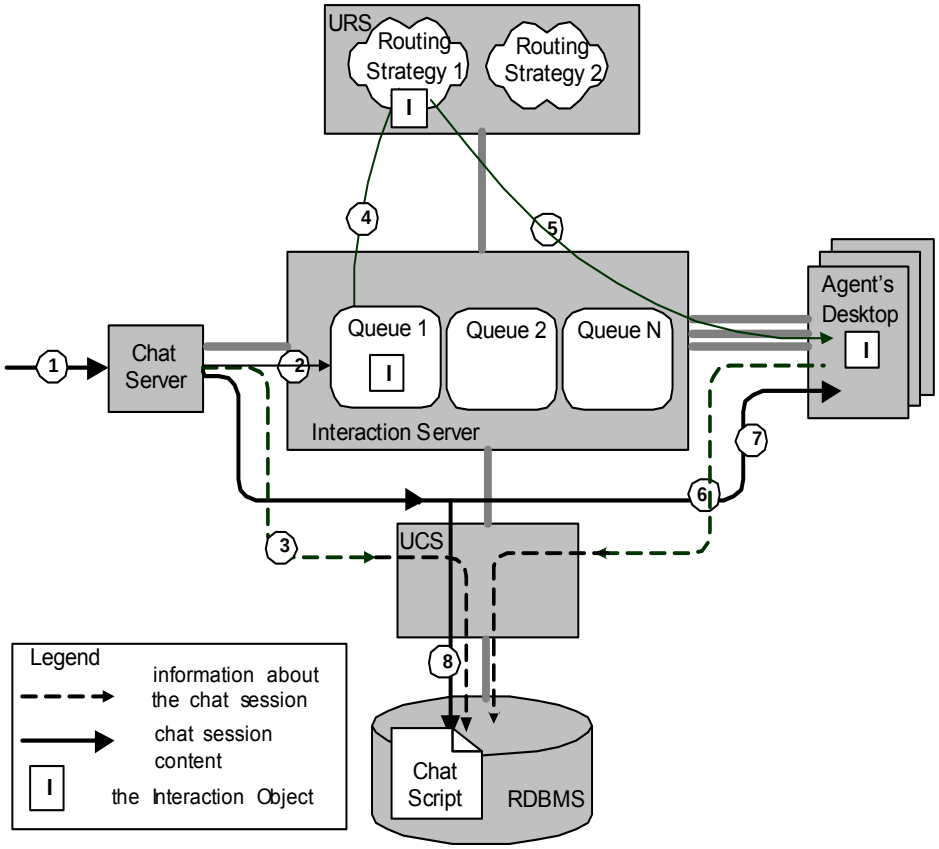

Figure 31: Processing of Chat Interactions

6. The agent accepts the interaction, and this results in the establishment of a chat session with Chat Server.
7. During the chat session, information about the dialog (the chat transcript) is stored in the Universal Contact Server database.

Structure of the Interaction Model

This section contains a more abstract view of the Genesys Multimedia interaction model. Organizing Solution Reporting on the handling of interactions in Genesys eServices requires a clear, well-defined understanding of multimedia interactions in their most abstract form. Like a telephony call (see “Telephony Events and Their Structure” on page 33), a multimedia interaction is built from abstract objects.

There are three major object types, each of which is discussed in the following subsections:

- **Endpoint Object**—An *endpoint object* is an abstract representation of a participant in the interaction, such as a customer or a routing strategy. This object is stateless in the context of the multimedia interaction model.
- **Interaction Object**—An *interaction object* is the abstract representation of the interaction as a whole. This object is stateless in the context of the multimedia interaction model.
- **Party Object**—A *party object* represents the processing of an interaction object by an endpoint object. Party objects are dynamic and can be represented in state machine diagrams.

Endpoint Objects

Figure 32 shows the four major endpoint object types.

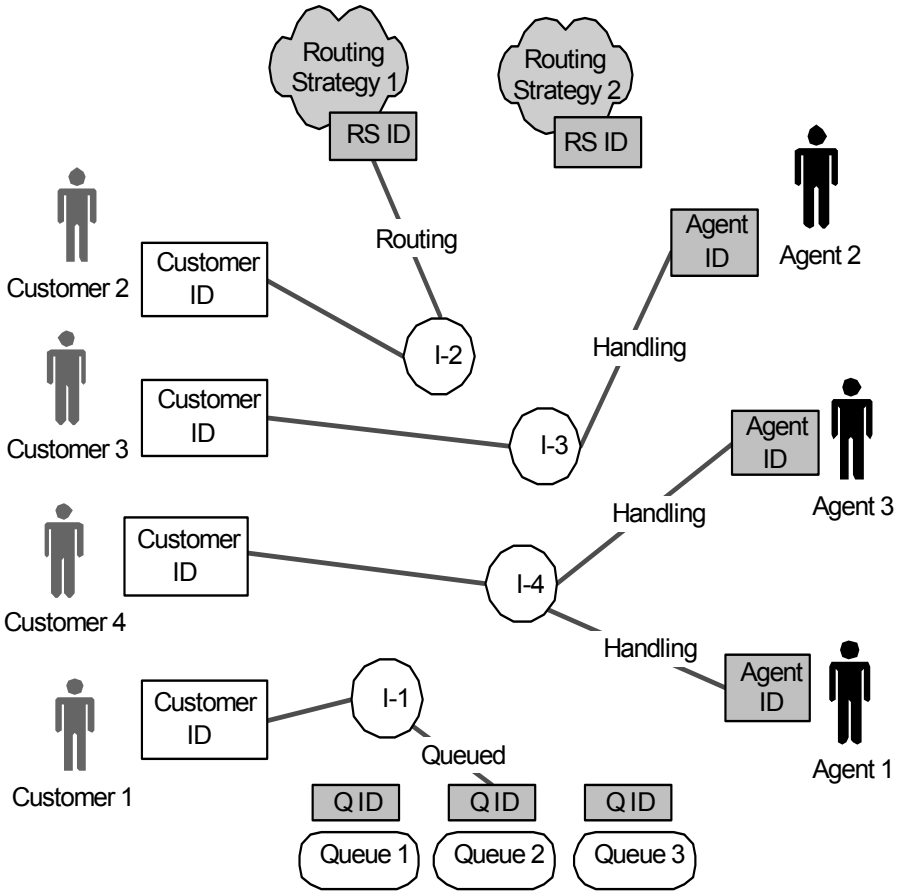

Figure 32: Multimedia Objects

The endpoint objects for interactions represent participants in the interactions. The Multimedia interaction model uses these four types of Endpoint Objects:

- **Customer**—Many interactions are strongly associated with a customer who initiated this interaction (in the case of inbound interactions) or to whom the interaction is directed (in the case of outbound interactions). A customer is uniquely identified by a Customer ID that can be implemented as an e-mail address or a chat address. Internal interactions might have no Customer Object associated with them.
- **Queue**—Interactions might wait in a queue for processing. A queue is identified by a Queue ID that is unique in Interaction Server.
- **Routing Strategy**—A *routing strategy* is a processing object that determines the next endpoint in the processing of an interaction. Each routing strategy is identified by a unique Strategy ID.
- **Agent**—An *agent* can be viewed as a processing object that handles an interaction. Each agent is identified by a unique Agent ID.

Endpoint Objects are stateless. That is, an Endpoint Object has no state machine associated with it.

The Interaction Object

In addition to the objects described above, an interaction uses an Interaction Object, which associates the endpoints (that is, the participants) involved in handling the interaction. The Interaction Object is also stateless. Each interaction is identified by a unique Interaction ID.

The Party Object

A Party Object is an object that represents involvement of an Endpoint Object with an Interaction Object. That is, the Party Object connects the Endpoint Object and the Interaction Object. Party Objects that connect Customer and Interaction Objects represent only the association between the interaction and the customer. Therefore, they are stateless. Party Objects that connect the other Endpoint Objects can be in different states at different times and can therefore be represented by state machines. The state of a Party Object indicates the current nature of the involvement between the Endpoint Object and the Interaction Object.

Figure 32 shows four situations in which an interaction might be found. The first one, `I-1`, is an interaction in a queue waiting for processing. The state of the corresponding Party Object is `Queued`, indicating that the Party Object connects the interaction with a Queue Object and the interaction is waiting to be processed.

Situation `I-2` shows a Party Object that connects an Interaction Object to the routing strategy that is determining the next Endpoint Object to which the

interaction should be sent. The state of the Party Object is `Routing`, indicating that the Router is currently handling the interaction.

Situation `I-3` shows a Party Object that connects the interaction to an agent. This Party Object is in the `Handling` state. Party Objects that are connected to an agent might have other states as well, as shown in "The Party Object State Machine" on page 55.

Situation `I-4` shows a Party Object that connects the interaction to multiple agents (two, in this case). This arrangement is known as a conference. The interaction has two Agent Endpoint Objects, the Party Object associated with each being in the `Handling` state.

The Party Object State Machine

As mentioned in the previous section, a Party Object has states that can be represented in a state machine diagram. Figure 33 shows this state machine.

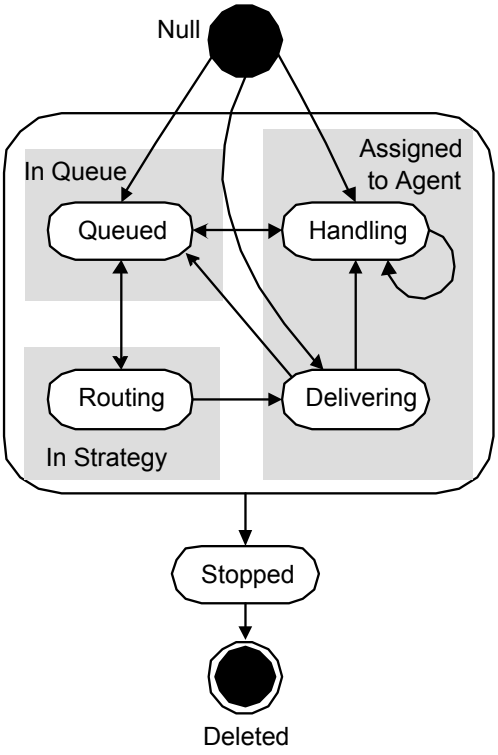

Figure 33: Party Object State Machine

This state machine contains several states and transitions between them. Interaction Server signals each transition from state to state by generating a corresponding event that is sent to Stat Server.

The progress of an interaction can be understood as the movement of the Party Object connected to the interaction from one to another Endpoint Object. As this happens, the Party Object take on the appropriate state, as shown in the Party Object state machine. "Typical Interaction Scenarios" on page 56

describes a number of interaction processing scenarios that follow the Party Object through different states.

The `Null` state corresponds to a case when the Party Object does not exist yet.

When the Party Object is created, it starts in the `Queued` or the `Handling` state. In the first case, the interaction is placed in a queue to wait for further processing. In the second case, the new interaction is created by an agent, such as outbound e-mail.

The `Routing` state indicates that a routing strategy is processing the interaction to determine the next endpoint, such as an agent or a queue.

An interaction that is connected to an agent is represented by a Party Object that is in one of two states, `Delivering` or `Handling`. If the Party Object is in the `Delivering` state, the interaction has been distributed to the agent but the agent has not yet accepted it for handling. If the agent accepts the interaction, the Party Object takes the `Handling` state. If the agent rejects the interaction (or the timeout period expires), the interaction returns to a queue.

When the Party Object is in a `Queued`, `Routing`, `Delivering`, or `Handling` state, the party can proceed to the `Stopped` state, which indicates that there is no more processing to be done to the interaction. After that, the Party Object takes the `Deleted` state that corresponds to elimination of the Party Object.

Transferring and Conferencing

A new Party Object created as a result of a transfer or conference behaves in accordance with the Party Object state machine shown in Figure 33 on page 55.

Transferring This section describes how a transfer proceeds. Suppose that `Agent A` is handling an interaction, creating a Party Object with the `Handling` state. Then `Agent A` decides to transfer the interaction to `Agent B`. As a result, a new Party Object is created with `Agent B` as the endpoint. This corresponds a state transition from the `Null` state to the `Delivering` state. If Agent B accepts the interaction, this Party Object proceeds from the `Delivering` to the `Handling` state. At the same time, the Party Object that represents the connection between `Agent A` and the interaction is destroyed.

Conferencing Conferencing proceeds in similar way. The difference is that the original Party Object (or Party Objects) is not destroyed when joining new the participant. Note that no participant's Party Object can be queued until it is the only remaining Party Object in the conference.

Typical Interaction Scenarios

In this section we present a set of scenarios that illustrate typical stages of interaction processing.

Initiating an Interaction

This scenario shows what happens when someone—whether customer or, in the case of an outbound interaction, an agent—initiates an interaction (see Figure 34). When the new interaction arrives, the receiving server (E-Mail Server or Web Media Server) creates a new Interaction Object. An initial record of the interactions is written to the Universal Contact Server database and the receiving server passes control of the interaction to Interaction Server.

The new interaction is associated with a Party Object that connects the interaction with the initiator, in this case, as is most typical, a customer. In addition, the Party Object takes the `Queued` state, which indicates that Interaction Server has placed the interaction in a queue.

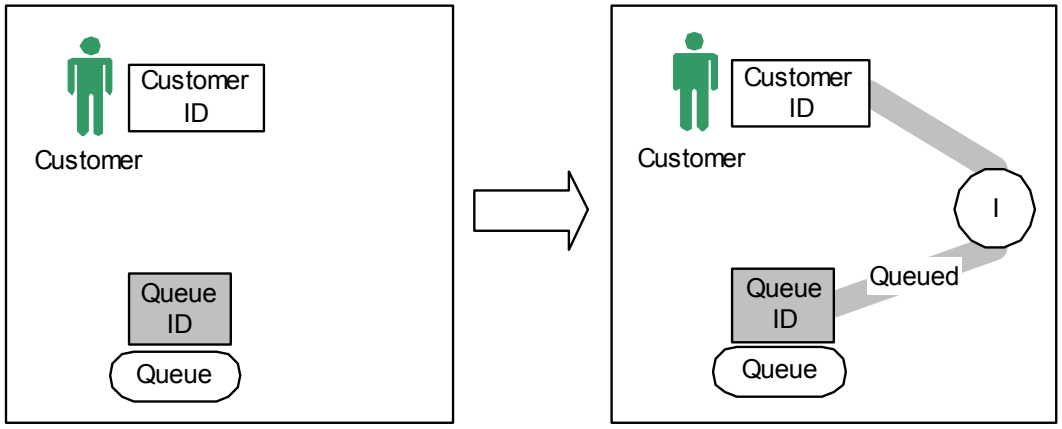

Figure 34: A Customer Initiates an Interaction

Routing of the Interaction

This scenario shows how the interaction is picked up by a routing strategy, which processes it to determine the next endpoint. See Figure 35.

Note: Although it might sound as if the routing strategy removes the interaction from the queue, this is not the case. The interaction remains in queue during handling by the strategy. However, for clarity, an interaction's time in a queue and its time being processed by a routing strategy are discussed separately.

Before the routing strategy starts processing the interaction, the Party Object was in a `Queued` state and the interaction was in a queue, as shown in the left-hand pane of Figure 35. This queue might be associated with some routing strategy that processes interactions waiting in this queue. When the Router is ready, it begins to process this interaction. The right-hand pane of Figure 35 shows the Party Object in the `Routing` state, indicating that the interaction is being processed by the routing strategy.

Again note that, although the graphic shows the Party Object being changed from one that connects the customer to the queue to one that connects the customer to the routing strategy, the interaction is simultaneously in queue. However, the `Routing` state takes precedence over, and replaces, the `Queued` state.

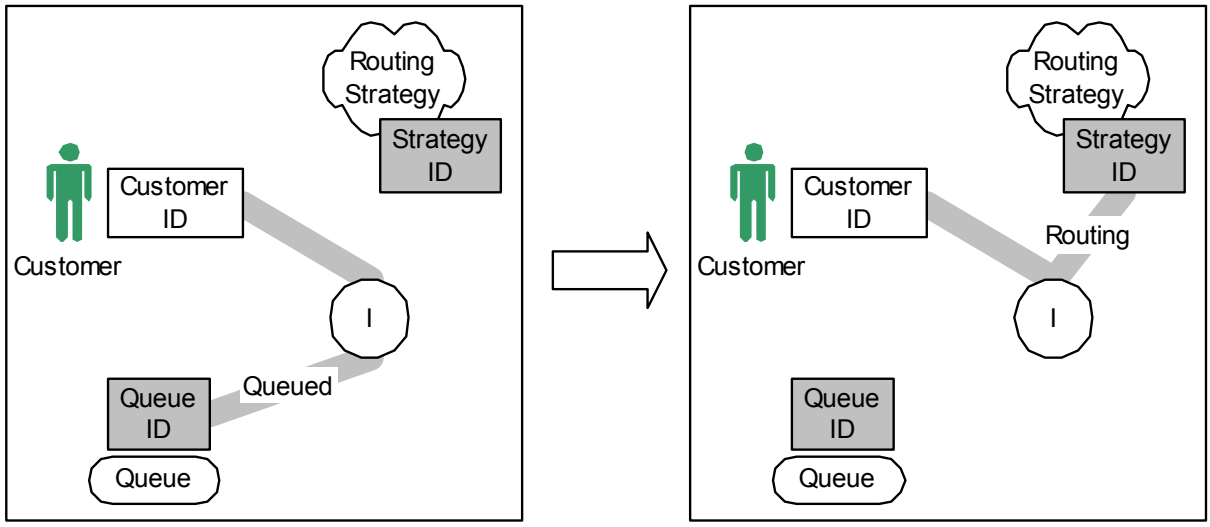

Figure 35: Routing the Interaction

The transition from the `Queued` state to the `Routing` state triggers a corresponding event, which is sent to Stat Server.

Distribution of the Interaction to an Agent

The routing strategy that processes the interaction might take any of a number of actions after completing interaction processing. One outcome is distribution of the interaction to an agent. See Figure 36.

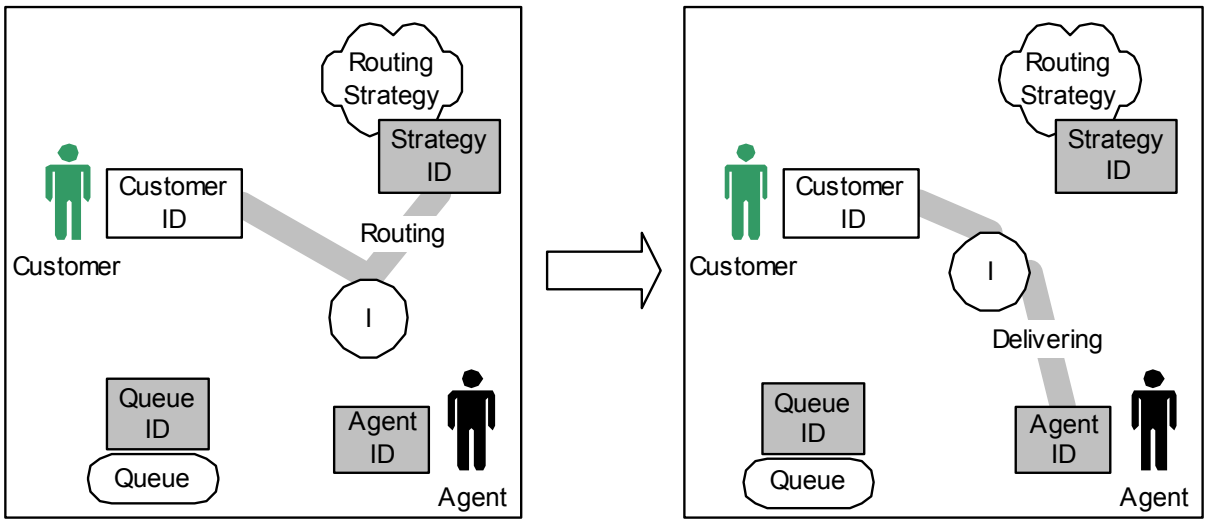

Figure 36: Distribution of the Interaction to an Agent

This results in a transition of the state of the Party Object from the `Routing` state to the `Delivering` state. This means that the interaction has been assigned to the agent.

This does not mean that the agent can now further process the interaction. To do so, the agent must first accept the interaction, which transfers the Party Object to the `Handling` state, as shown in the next section.

An Agent Accepts the Interaction

An agent to whom an interaction has been delivered might accept it for handling. See Figure 37. In this case the party changes from the `Delivering` state to the `Handling` state.

If the agent not does not accept the interaction, he or she might send it back to the queue or a specified timeout period might elapse, which transfers the interaction back to the queue automatically.

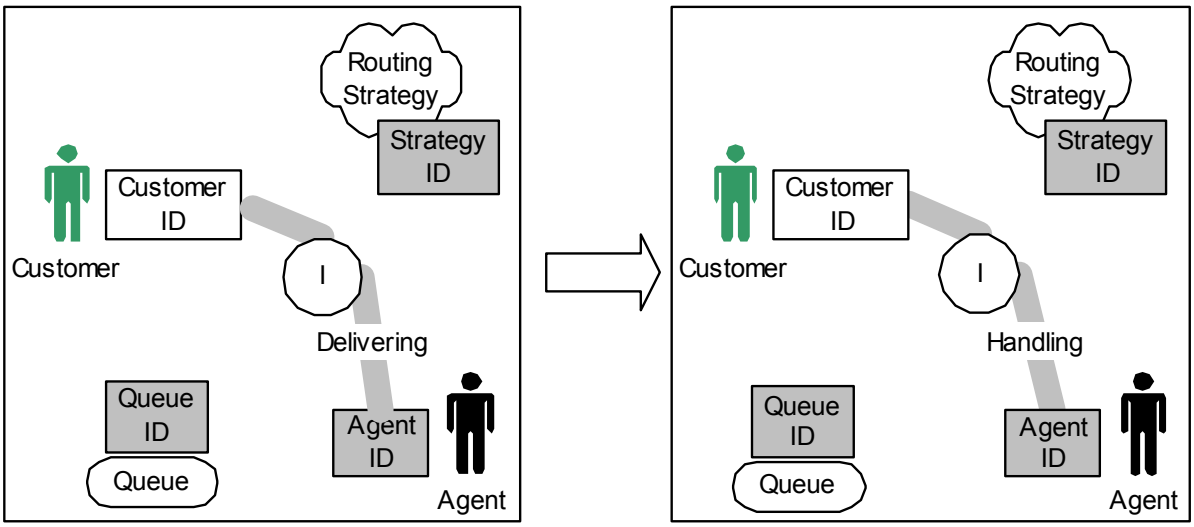

Figure 37: An Agent Accepts the Interaction

Placing Interaction into Queue by Agent

An agent handling an interaction might decide that a different agent should continue its processing. In this case, the agent returns the interaction to a queue. See Figure 38.

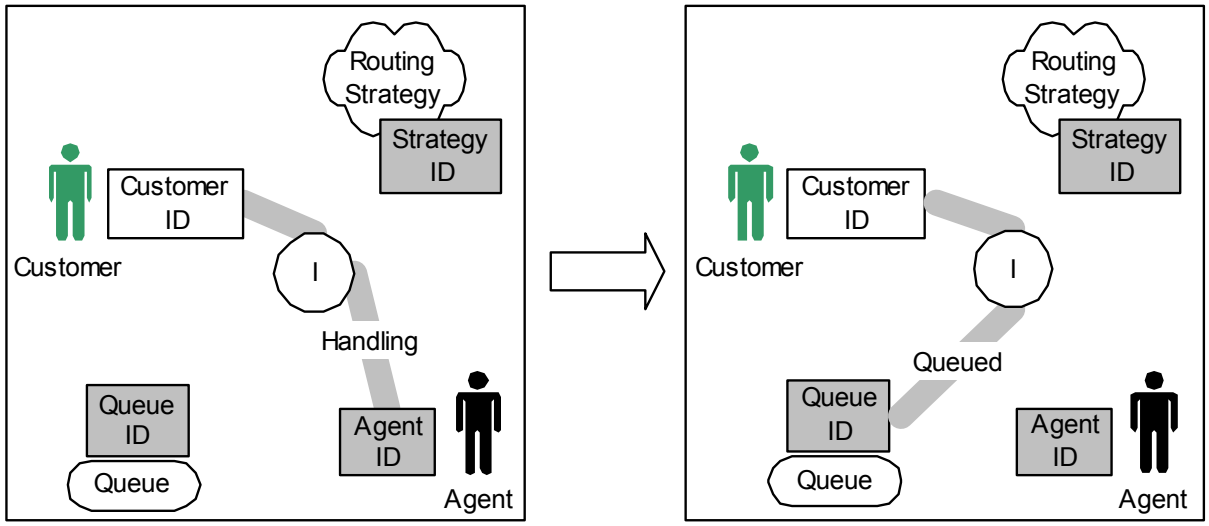

Figure 38: An Agent Placing an Interaction into a Queue

The Party Object changes state from the `Handling` state back to the `Queued` state.

Interaction Transfer: Initialization

An agent handling an interaction chooses to transfer the interaction directly to another agent without first placing it in a queue. The first stage of this interaction transfer scenario is shown in Figure 39.

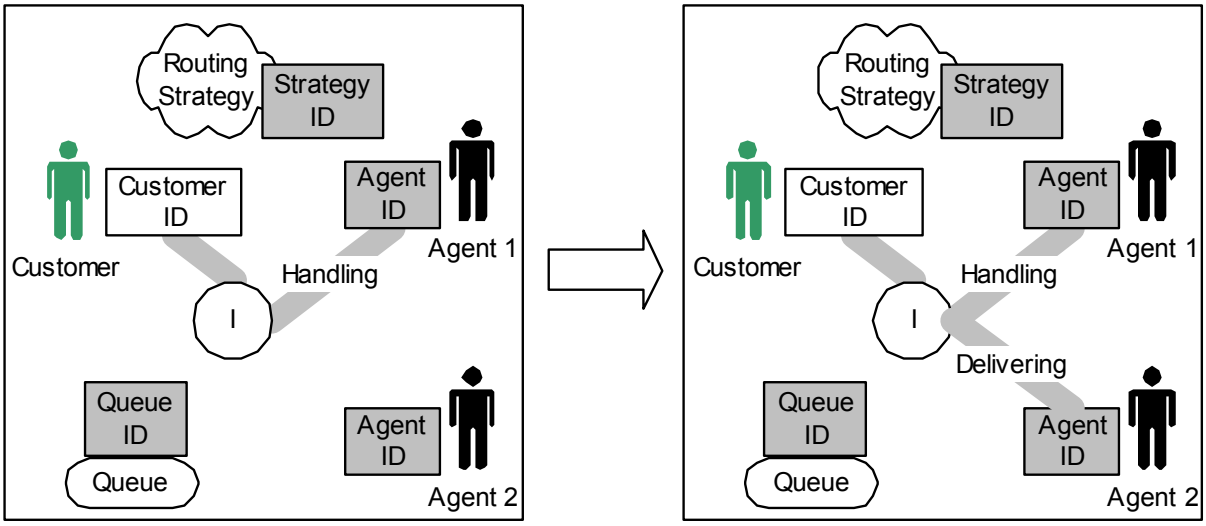

Figure 39: Transfer of an Interaction—Step One

Initiating the transfer creates a new Party Object with `Agent 2` as the endpoint. The new interaction takes the `Delivering` state, indicating that Agent 2 must accept (or reject) the transfer to continue interaction processing. At this point,

the original Party Object that connects to `Agent 1` still exists and is in the `Handling` state.

Interaction Transfer: Acceptance

When `Agent 1` has initiated the interaction transfer, there are two possible outcomes for the transfer. The normal case is when the second agent (here `Agent 2`) decides to accept the transferred interaction. This scenario appears in Figure 40.

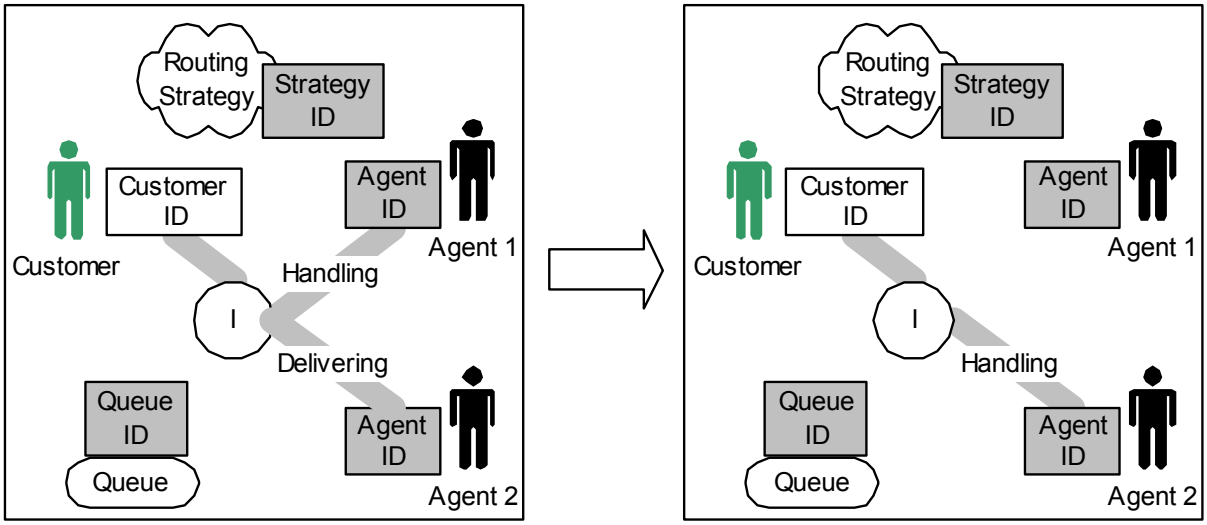

Figure 40: Interaction Transfer Accepted

After `Agent 2` accepts the interaction, the Party Object connected to `Agent 2` changes to the `Handling` state. The Party Object connected to `Agent 1` is destroyed. The transfer is complete.

Interaction Transfer: Rejection

If `Agent 2` does not want to participate in handling the interaction, he or she might reject it, as shown in Figure 41.

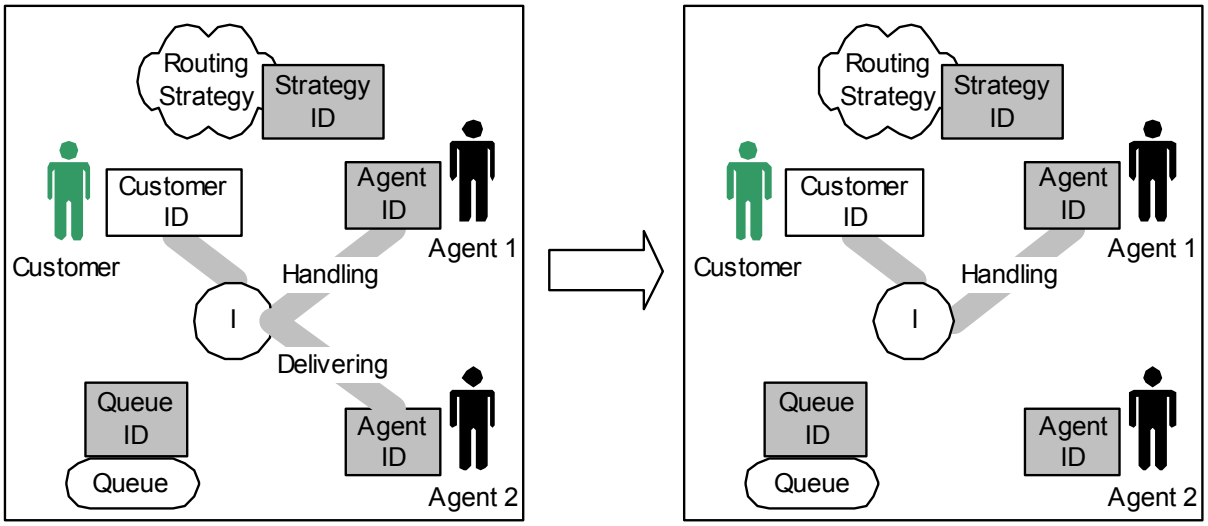

Figure 41: Interaction Transfer Rejected

In this case, the Party Object connected to `Agent 2` is destroyed, while the Party Object connected to `Agent 1` remains in the `Handling` state.

`Note:` An interaction transfer might be rejected automatically if a configured timeout expires before `Agent 2` accepts the transfer.

Conferencing an Interaction

Some interactions, such as chat, might reasonably be handled by several agents simultaneously rather than by one. This sort of interaction processing is called conferencing (see Figure 42).

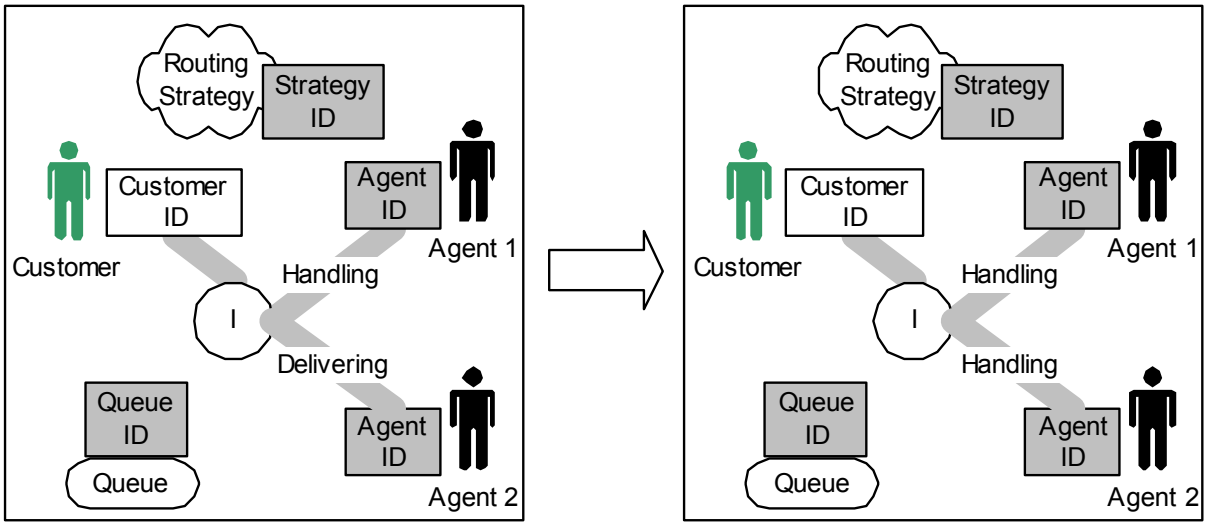

Figure 42: Conferenced Interaction

During a conference, a new Party Object is created and changes from the initial `Idle` state (not shown) to the `Delivering` state. The original Party Object remains in the `Handling` state and emits an event that indicates the creation of the conference. When the additional agent or agents accepts the interaction, the second Party Object enters the `Handling` state, as shown in the right-hand pane of Figure 42. Unlike a transfer, a conference results in multiple Party Objects in the `Handling` state. The original Party Object is not destroyed.

Note: A conference and the resulting creation of the new Party Object might be initiated either by someone already handling the interaction or by the agent joining the conference. In either case, the conferencing procedure is the same.

Stopping the Processing of an Interaction

You can stop an interaction from being processed further in several ways. One way is for an agent to stop the processing. This scenario is shown in Figure 43.

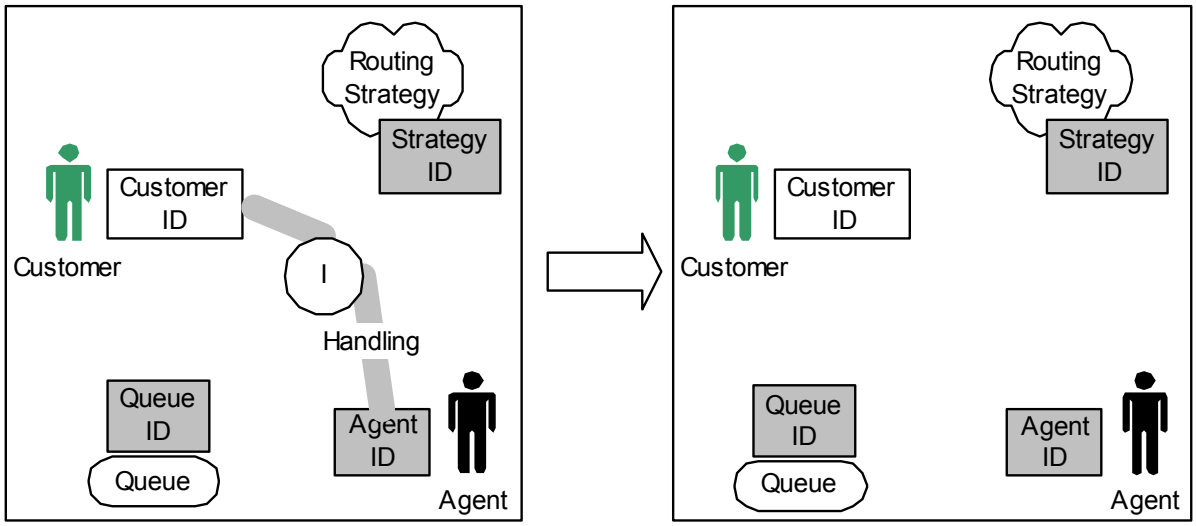

Figure 43: Stopping the Processing of an Interaction

Before the agent decided to stop the interaction, the Party Object was in the `Handling` state, as shown in the left-hand pane of Figure 43. Afterwards, the Party Object changes to the `Stopped` state and then to the `Deleted` state, which corresponds to the destruction of the Party Object, as shown in the right-hand pane of Figure 43.

Interaction processing might also be stopped by:

- A routing strategy, if it determines that an interaction should be sent to a `Stop` routing object.
- Chat Server, if the customer disconnects before the chat session has been distributed to an agent. If the chat session has been distributed to an agent, only a routing strategy or an agent can stop the interaction.

E-Mail Processing Example

This section puts together the various segments of interaction processing detailed in the preceding section to present an overview of a full sequence of steps that might be involved in processing an inbound e-mail.

Note: This is an example and might not reflect actual e-mail processing stages in your environment.

This example shows an e-mail arriving, sending of an acknowledgement, processing by an agent, and the agent sending a response back to the customer.

The processing environment appears in Figure 44. The interaction waits in `Queue 1` until `Strategy 1` processes it. Strategy 1 sends an acknowledgement to the customer and determines the appropriate agent. `Queue 2` and `Strategy 2` are used to send the outbound e-mail interactions.

Figure 44 shows the initial situation, in which no interaction exists.

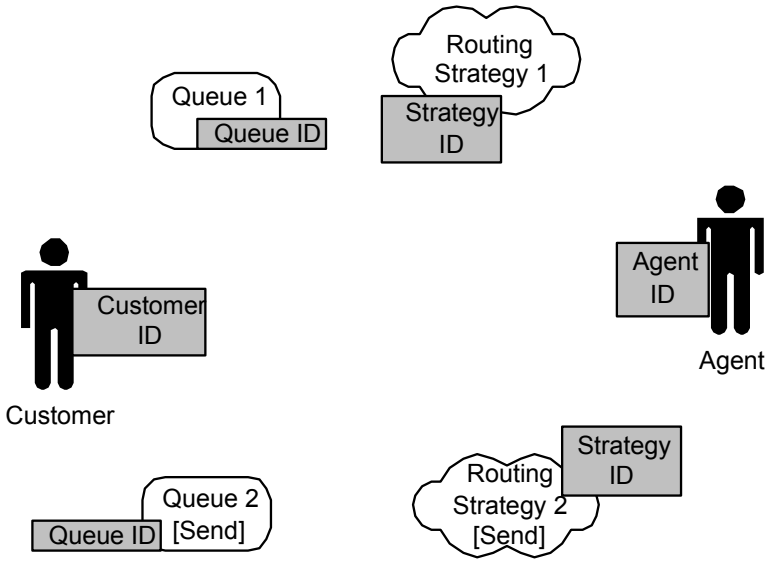

Figure 44: Initial Stage: No Interaction Exists

The customer then sends an e-mail to the contact center. An interaction of the e-mail type is created, as shown in Figure 45. It is currently composed of two Party objects, one of which associates the interaction with the customer. The other one associates the interaction with `Queue 1`, where it waits for `Strategy 1` to process it. This party is in the `Queued` state.

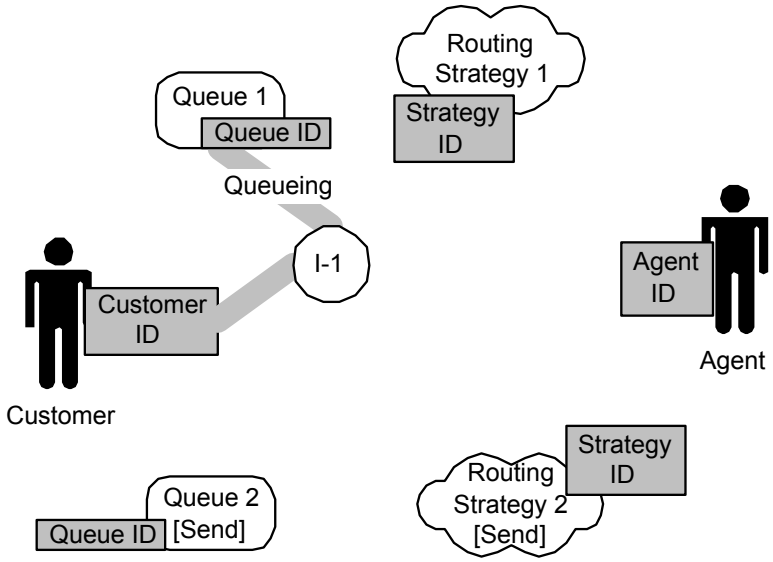

Figure 45: The New Interaction in Queue

Next, the Queued Party is transferred to Strategy 1, at which point it changes to the `Routing` state, as shown in Figure 46. The Strategy processes the interaction, first by sending an acknowledgement that the e-mail was received and then by determining the most appropriate agent to handle the interaction.

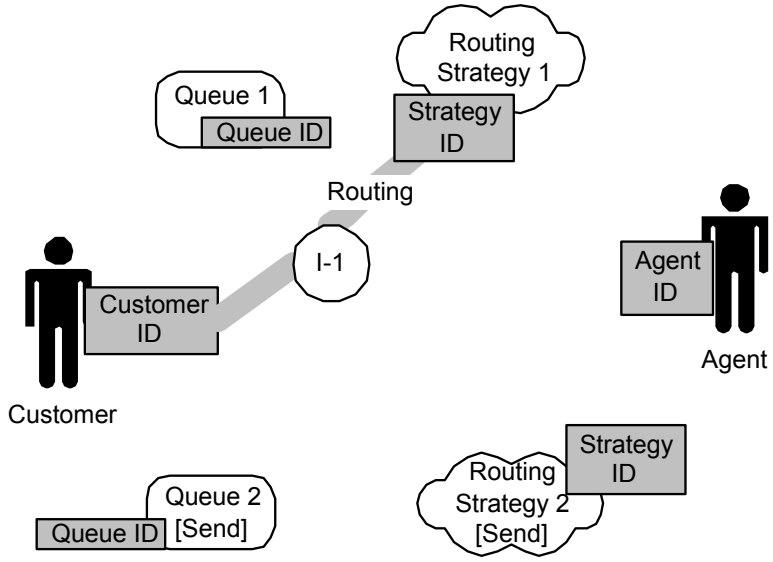

Figure 46: Routing the E-Mail

`Strategy 1` initiates a new outbound interaction, `I-2`, for the acknowledgement. The new interaction has a Party object associated with the customer and a Party object that is associated with `Queue 2` and which is in the `Queued` state. At this point, the interaction is waiting for `Strategy 2` to process it (see Figure 47).

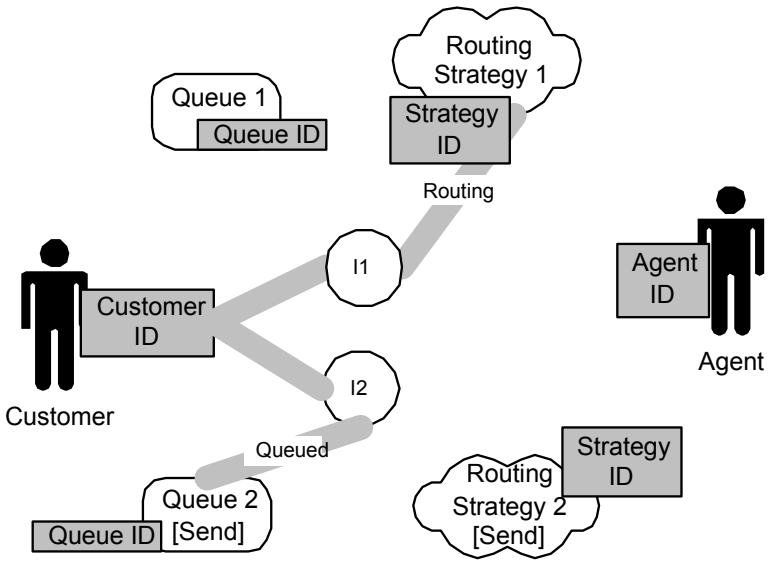

Figure 47: The Auto Response in Queue

The outbound acknowledgement interaction is processed by `Strategy 2` and sent to the customer (see Figure 48). After the auto response is sent, `I-2` and the associated Party Objects, disappears.

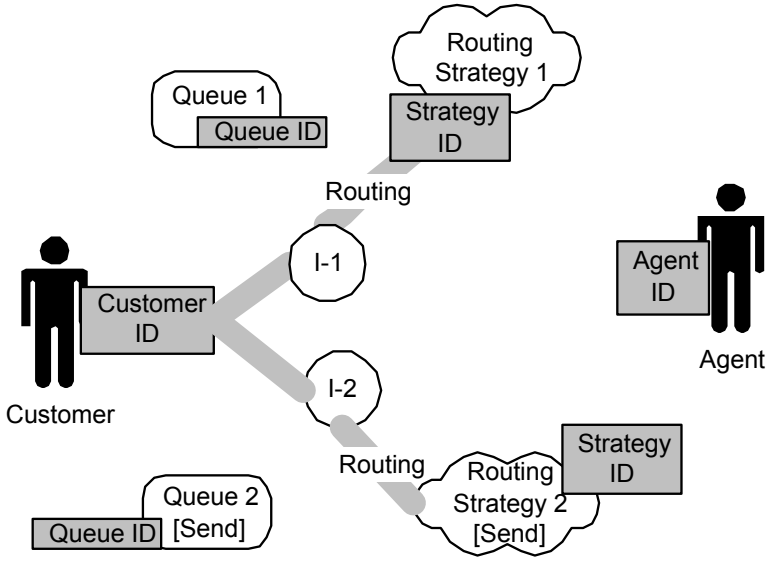

Figure 48: Strategy 2 Sends the Auto Response to the Customer

After creating the acknowledgement interaction, `Routing Strategy 1` checks for an agent who can process the interaction. The interaction Party Object moves to the agent and enters the `Delivering` state. When the agent accepts the interaction the Party changes to the `Handling` state, as shown in Figure 49.

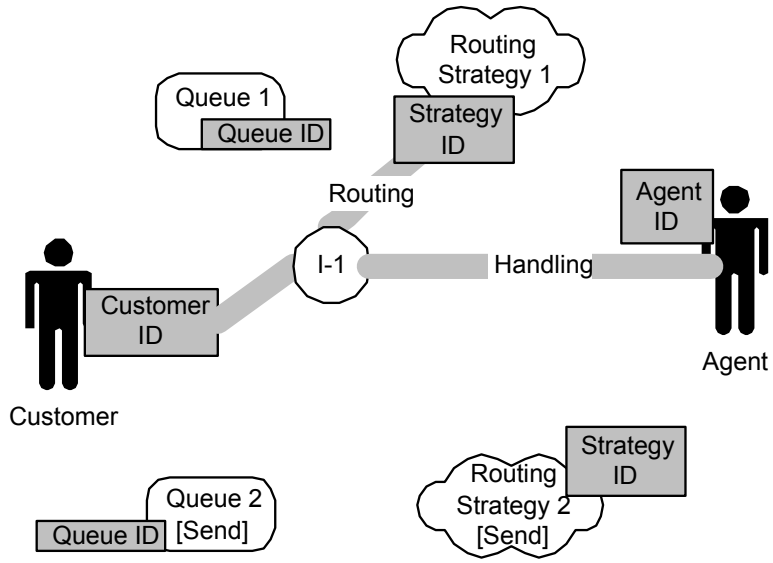

Figure 49: The Agent Accepts the Interaction

The agent handles the interaction, which includes preparation of a reply e-mail that the agent sends to the customer. The reply e-mail creates a new outbound interaction, `I-3`, with two Party Objects, one associated with `Queue 2` and the other with the customer, as shown in Figure 50.

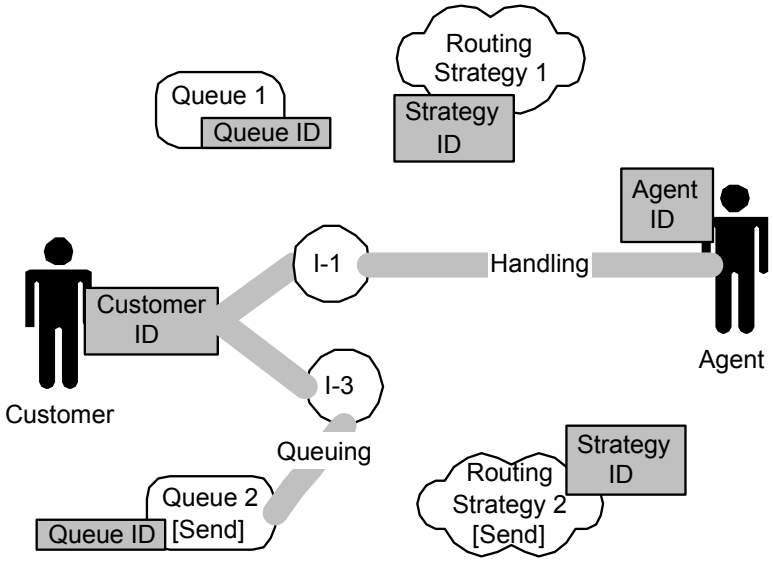

Figure 50: Interaction 3: The Reply E-Mail

In order for the reply to reach the customer, it is queued, sent to a routing strategy, and then to the customer (see Figure 51). After the interaction is sent to the customer's e-mail server, Interaction 3 disappears.

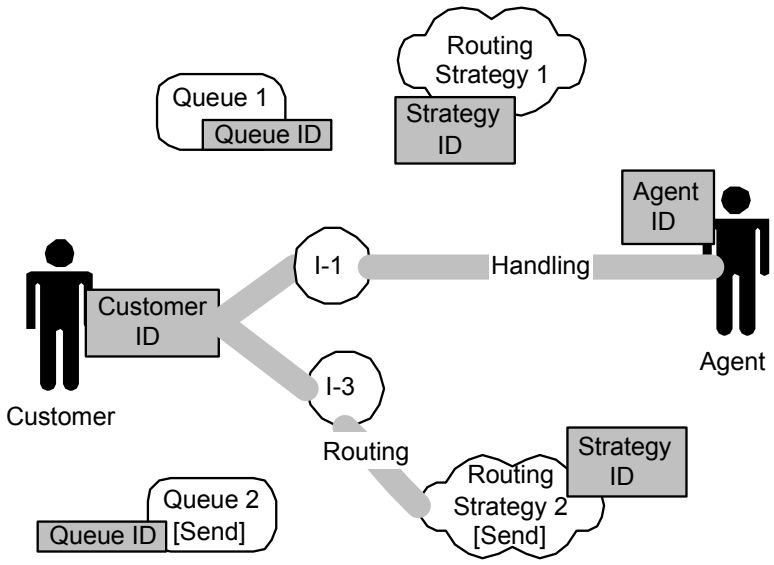

Figure 51: Routing the Reply E-Mail to the Customer

After the reply is sent, the agent stops processing Interaction 1, which then disappears as well. The situation returns to the starting point, with no interactions present, as shown in Figure 44 on page 64.

This example shows a typical scenario for processing inbound e-mail in terms of the Multimedia Interaction Model. Note that, in this scenario, e-mail processing involves three different interactions.

The Statistical Model

The Genesys Statistical Model embodies the main principles of Stat Server operation. This document discusses only the part of the Statistical Model relevant to Solution Reporting.

Stat Server can work with several T-Servers and/or Interaction Servers; that is, it can be the client of several T-Servers and Interaction Servers simultaneously. Stat Server is also a client of Configuration Server, from which it retrieves information about objects in a contact center. Stat Server uses Configuration Server information to maintain its own system of objects, which is derived from the object model in the Configuration Server. Stat Server processes information received from T-Server and Interaction Server to provide its clients with more elaborated and statistically useful information. Figure 52 shows a typical Stat Server environment.

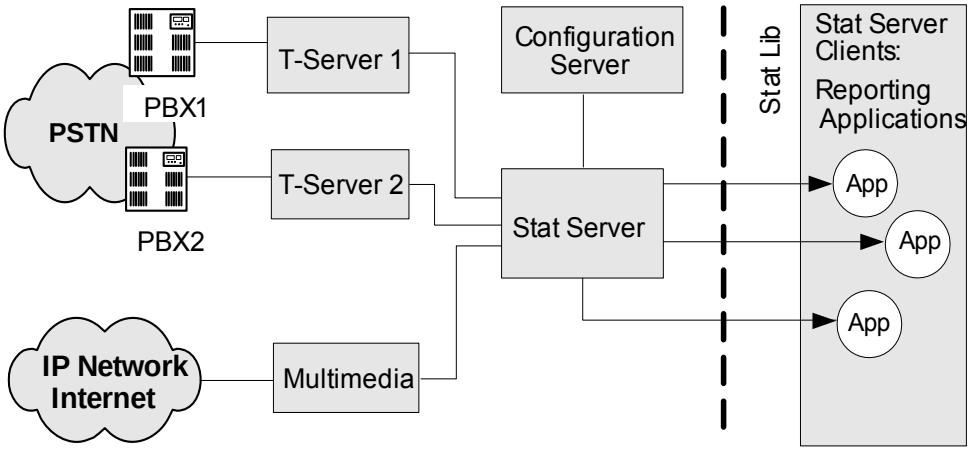

Figure 52: Stat Server Environment

To provide its clients with statistical information, Stat Server uses its internal API, the Stat Lib. This API principally supplies metrics and statistics that tell Stat Server what information clients require. Roughly speaking, the metrics specify information about what object type is needed, what data is of interest about these objects, and how to calculate information. The statistic is the application of the metric to a specific object.

Structure of a Statistic

To receive statistical information from Stat Server, its clients must specify what kind of information they need. This specification consists of a request for statistics retrieval from the Stat Server API. Stat Server collects the statistical values and sends them to the client only when requested.

The structure of the components of a statistic is shown in Figure 53.

Within the Genesys Statistical Model, a *statistic* is defined as a metric applied to a specific object in a contact center. A *metric* is comprised of a statistical type, time profile, time range, and filter. (The latter two are optional.) A *statistical type* is a collection of masks, object types, category, and subject.

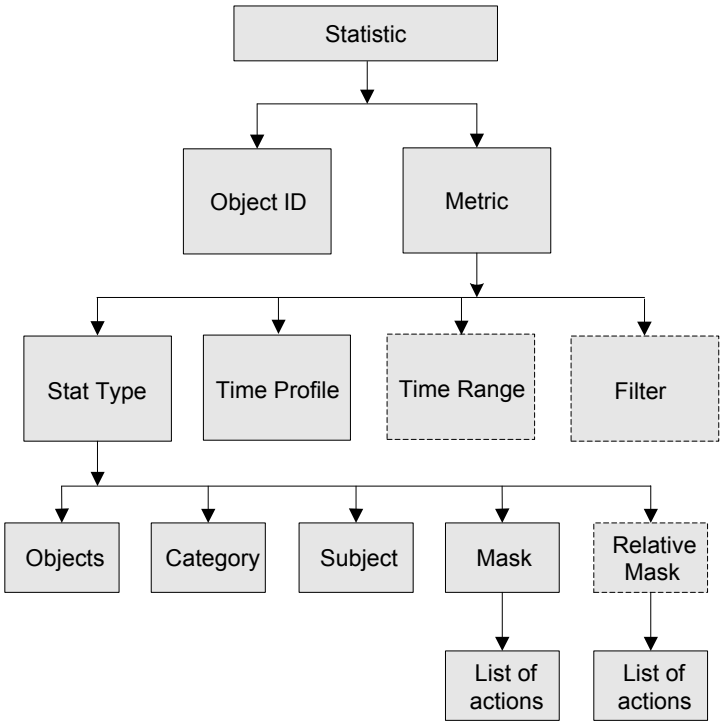

Figure 53: Structure of Request for Statistical Information

Statistical Objects

Stat Server provides applications with information about the following contact center objects:

- **Regular Directory Numbers (RegDN).** Represents regular devices in the contact center (T-Server) identified by a directory number (DN).
- **Agent.** Represents a living person (operator) registered (logged in) in a contact center as an agent.
- **Place.** Represents the agent's working place and is usually composed as a set of regular DNs.
- **Queue.** Represents an ACD queue, interaction queue, or workbin of interactions waiting for distribution. The functionality of a queue is usually implemented within PBX hardware that T-Server can access.
- **Routing Point (RoutePoint).** Represents a Routing Point device, which is a DN that can route calls to other DNs. Routing Point functionality is implemented within T-Server software.
- **Group of Agents (GroupAgents).** Represents a group of agents, usually grouped on the basis of established business rules, for example, the Sales and Help Desk groups.
- **Group of Places (GroupPlaces).** Represents a group of places, usually formed for administrative or geographical purposes.

- **Group of Queues (GroupQueues).** Represents a group composed of Queue and/or Routing Point objects.
- **Staging Area.** Represents an interaction queue in which multimedia interactions exist while they are being processed.
- **Tenant.** Represents an entire contact center. Used in the Genesys eServices e-mail and chat statistics.

Stat Server gets all its information about these objects from the Configuration Server. Refer to the *Framework 8.0 Deployment Guide* and *Framework 8.0 Configuration Manager Help* for further information.

Statistical Actions and Statuses

The notion of an action is fundamental to the Genesys Statistical Model. Actions are the building blocks for constructing metrics and ultimately for calculating statistical data. Stat Server actions function much like T-Server events (TEvents). Despite their elementary nature, actions are built on and could be strictly defined in terms of the Genesys Call Model. However, for the Statistical Model, actions are elementary notions.

Actions are associated with objects of device type: regular DN, Routing Point DN, and queue DN. An action on such an object tells what is going on with the object or activity on the object. More exactly, actions are related to parties associated with a particular object. For example, the transition of a party on a regular DN from a `Ringing` state to an `Established` state originates the `Answered` action for the agent logged in at the corresponding DN.

A set of rules prescribes how actions on objects of a lower level of hierarchy are propagated and aggregated to a higher level.

Actions are either *instantaneous* or *durable*. (For a more detailed classification, see “Action Classification” on page 72.) An action might last on an object (if it is durable) or occur (if it is instantaneous). Several actions might affect an object simultaneously. For example, the `CallOnHold` durable action on a regular DN indicates that this DN is on hold. At the same time though, the `Monitored` durable action reveals that this DN is monitored.

Making sense of any Statistical Model metric requires an understanding of how all actions used in that statistic’s computation are generated. All actions are generated on the basis of TEvents or Interaction Server events. Each instantaneous action is triggered by one event while a durable action is triggered by two.

The current version of Stat Server predefines the set of all actions, which cannot be extended.

Note: The following discussion of actions, although it is based on the telephony call model, applies to multimedia actions as well unless otherwise specified. The Multimedia Interaction Model of Genesys eServices uses a separate set of actions, some of which are parallel to telephony actions, other of which are unique to multimedia. For a discussion of multimedia actions, refer to the *Framework 8.0 Stat Server User's Guide*.

Action Classification

All Stat Server actions can be divided into two groups: durable actions and instantaneous actions as depicted in Figure 54. It is convenient to associate durable actions with the state of a state machine having a starting and ending moment and lasting some duration. Therefore, a durable action can be associated with its duration time.

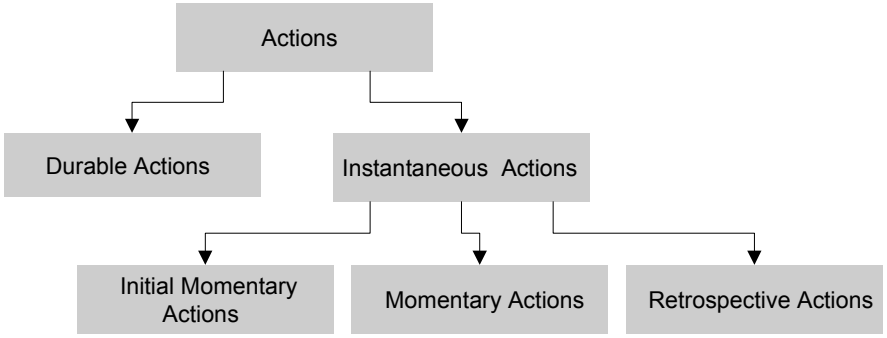

Figure 54: Types of Actions

Instantaneous actions can be naturally associated with the transition of a state machine corresponding to transition from one state to another. Therefore, these actions are indivisible and occur at a single moment of time. Instantaneous actions, in turn, can be further divided into initial momentary actions, momentary actions, and retrospective actions. Figure 55 illustrates the interrelationship between action types.

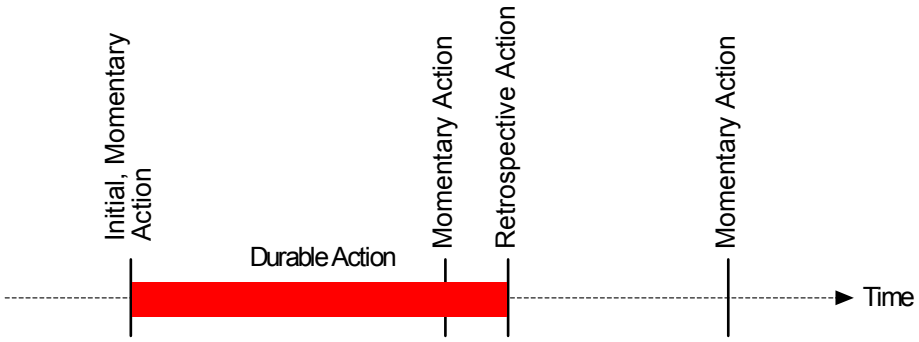

Figure 55: Interrelationship Between Actions

The durable action is depicted as the bold horizontal bar along the timeline clearly showing a duration in time. The onset of a durable action is an initial, momentary action, which is instantaneous and has a duration equal to zero. A retrospective action terminates the durable action. Despite the fact that retrospective actions, too, are instantaneous in nature, Stat Server assigns to them the duration of their corresponding durable action. Momentary actions have no relation to durable actions and endure only for the moment (duration equals zero).

Figure 56 illustrates the relationship between several Stat Server actions. This simplified example shows a sequence of actions related to a regular DN. In layperson's terms, a phone rings; an agent answers the phone and places the call on hold; the agent then releases the call from hold.

The initial momentary action, `CallRingingStarted`, causes the start of the `CallRinging` durable action. The `CallRinging` action ends normally (for example, the phone is answered) as signaled by the `CallAnswered` retrospective action. At the same time, the `CallInboundStarted` momentary action occurs marking the beginning of the `CallInbound` durable action. During this action, the call might be placed on hold, triggered by the `CallHeld` initial, momentary action that marks the start of the `CallOnHold` durable action. Now, two durable actions, `CallInbound` and `CallOnHold`, coexist. When the `CallOnHold` action terminates, the `CallRetrievedFromHeld` retrospective action occurs. At any time, a `UserEvent` momentary action can occur without influencing any of the durable actions.

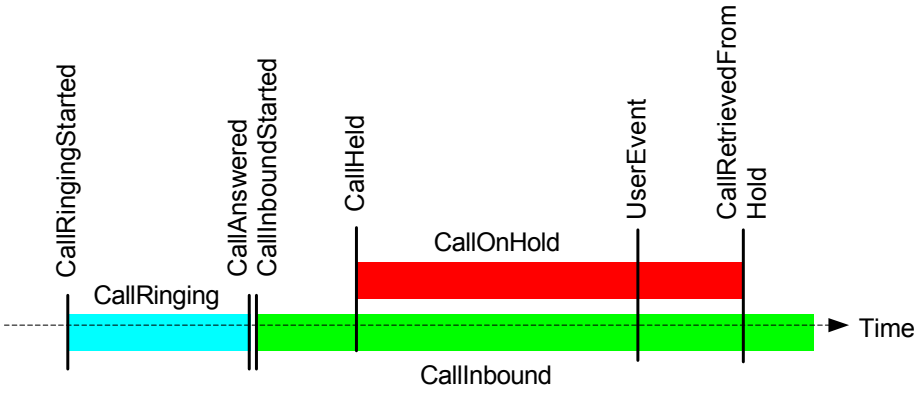

Figure 56: Example of Actions

Note that not every durable action is triggered by momentary initial and retrospective actions. For example, the two complementary actions, `Monitored` and `NotMonitored`, fall into this category.

Interrelationships Among Multimedia Actions

Multimedia actions function similarly to telephony actions. For example, a customer might initiate a chat session that is routed to an agent. The agent requests coaching and then, when the customer issue has been resolved, the agent stops the session. This interaction produces this sequence of actions:

The initial momentary action, InteractionDeliveringStarted, coincides with the start of the Interaction Delivering durable action. The Interaction Delivering action ends normally (for example, the agent accepts the interaction) as signaled by the InteractionAccepted retrospective action. At the same time, the InteractionHandlingStarted momentary action occurs marking the beginning of the InteractionHandling durable action. During this action, the agent might decide to conference in another agent. This triggers the InteractionConferenceMade instantaneous momentary action, followed by the InteractionDeliveringStarted initial, momentary action, which marks the start of the InteractionDelivering durable action. This Interaction Delivering action applies to a Party Object associating the current Interaction Object with a new agent. The Party Object associating the Interaction Object with the first agent continues to exist in the Handling state. When the second agent accepts the conference, it is marked first by the InteractionAccepted retrospective action, which ends the Delivering state, then by the InteractionConference Joined instantaneous momentary action, followed by the Interaction Handling Started initial, momentary action, which marks the start of the Interaction Handling durable action for the new Party Object. Now the interaction has two coexisting instances of the InteractionHandling durable actions. When the conference terminates, the InteractionStopped retrospective action occurs for the agent that has left the conference. When the chat session is complete, the remaining agent triggers the InteractionStopped retrospective action, ending the chat session.

Relation of Actions to Events

Occurrences of instantaneous and durable actions in the Statistical Model are triggered by events T-Server or Interaction Server sends. To understand what the action really means, it is useful to see a correlation between receiving events and the occurrence of actions.

Note: Multimedia interaction actions and events can be analyzed in an analogous way to the telephony event/action relationship described in the following sections.

Figure 57 shows an example of a CallRinging durable action and related instantaneous actions influenced by TEvents. As before, this is diagrammed using a state machine.

This state machine describes all actions related to ringing. The CallRinging durable action is represented by state **s1.** This action follows the CallRingingStarted initial momentary action to the left, which, in turn, is triggered upon receiving the EventRinging TEvent (or EventPartyChanged TEvent for a consulting call).

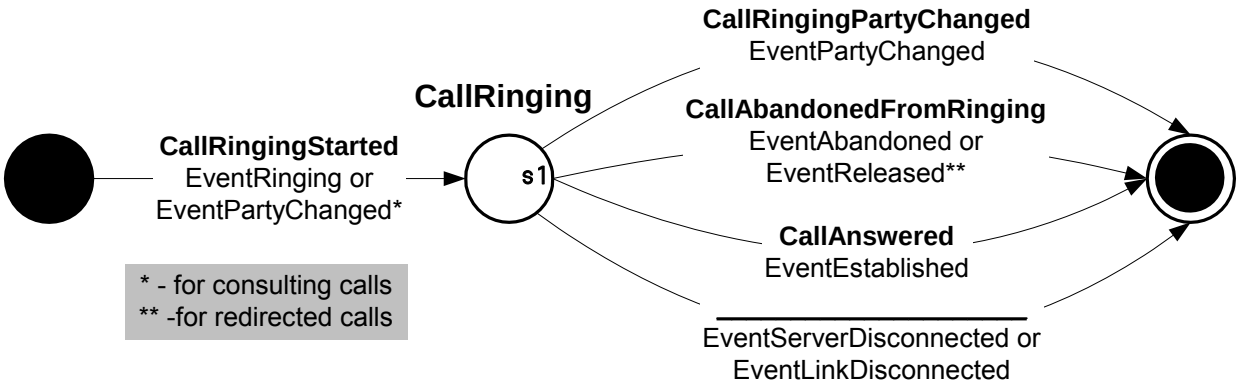

Figure 57: CallRinging Action

The `CallRinging` durable action endures until one of three instantaneous actions occurs: `CallAnswered`, `CallAbandonedFromRinging`, or `CallRinging PartyChanged`, which correspond to the three of the four possible state transitions to the right of **s1.** The fourth state transition indicates some kind system failure that forced a disconnect. These actions report about the possible end of the `CallRinging` action.

The `CallAnswered` action relates to answering the call and is triggered by the `EventEstablished` TEvent. The `CallAbandonedFromRinging` action indicates an abnormal end where the call was released before being answered (triggered by `EventAbandoned`) or as a result of redirection (triggered by `EventReleased`). Another exit from the `CallRinging` action, `CallRingingPartyChanged`, might be triggered by the `EventPartyChanged` TEvent and tells about completing a transfer of the call.

Note: Upon detecting a stuck call, T-Server distributes an `EventAbandoned` or `EventReleased` event coupled with a `AttributeReliability` attribute other than `TReliabilityOk`.

Stat Server distinguishes stuck calls, which are caused by a missynchronization between two or more interdependent contact center components (such as T-Server and the switch, Stat Server and T-Server, the Genesys Router and Stat Server), from those calls that are abandoned for other reasons (the customer hanging up, for example).

All of these instantaneous actions are retrospective actions.

The `CallRinging` durable action might also terminate when the `NotMonitored` action starts triggered either by the `EventServerDisconnected` or `EventLinkDisconnected` T-Events. These events indicate disconnection with T-Server or the breaking of the CTI link, respectively. Note that in this case there are no corresponding retrospective actions.

Comparison of Call Model and Statistical Model

Now compare the state machine in Figure 57 with the state machine for a Regular Party in Figure 20 on page 40. In Figure 58, the fragment of the state machine relating to the `Ringing` state is reproduced.

Note that from the Call Model point of view, there is only one way to enter the `Ringing` state, that is via the `EventRinging` TEvent, and but three ways to exit (via `EventEstablished`, `EventReleased`, and `EventAbandoned` transitions). The `EventPartyChanged` TEvent does not change the `Ringing` state but can change the `Call` object of the party. The Statistical Model interprets this situation slightly differently.

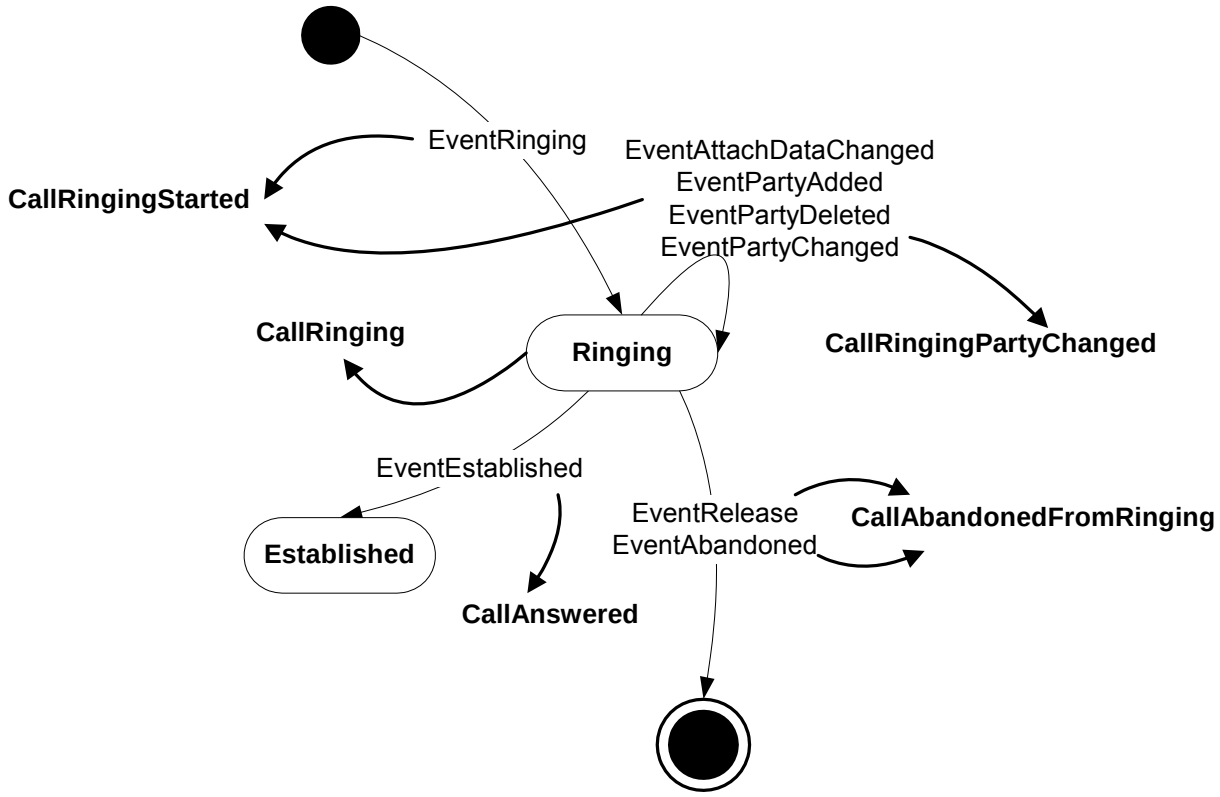

Figure 58: Fragment of Call Model State Machine

First, the `EventPartyChanged` T-Event with changed `ConnID` is treated as the end of one `CallRinging` durable action and the start of a new one. Second, the `EventPartyChanged` and `EventRinging` T-Events cause one initial momentary action: `CallRingingStarted`. Third, there might be three possible ends of the `CallRinging` action: retrospective action `CallAnswered` triggered by `EventEstablished`, `CallAbandonedFromRinging` triggered by `EventRelease` or `EventAbandoned`, and `CallRingingPartyChanged` triggered by `EventPartyChanged`.

The momentary action is illustrated by the state machine depicted in Figure 59. This machine has one state and one transition. The transition corresponds to a `UserEvent` momentary action and is triggered by the `EventUserEvent` TEvent. The transition can trigger at any time and does not change.

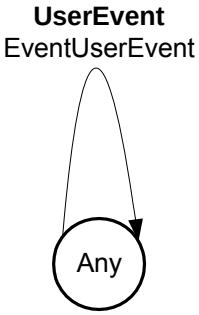

Figure 59: Momentary Action UserEvent

Action Generation and Propagation

The generation of actions starts when Stat Server receives events from T-Server and Interaction Server. Figure 60 illustrates the schema of action generation. Each T-Server event is identified by its DN (event parameter `ThisDN`). Therefore, it initially causes occurrence of DN action(s) according to action definition. Simultaneously, the status of each DN object is determined.

Interaction Server events are identified by the `InteractionID` parameter. The actions generated, such as `InteractionDelivering`, are related to the interaction.

Actions might propagate to other objects associated with this DN where they contribute to determining statuses of these objects. (The determining procedure will be explained later.)

For example, receiving an `EventRinging` TEvent for a regular DN with an inbound call type invokes:

- Initial, momentary actions—`CallRingingStarted` and `CallRingingStarted Inbound`.
- Durable actions—`CallRinging` and `CallRingingInbound`.

DN status changes to `CallRinging`, and at the same time, these actions propagate to the corresponding Place, Agent, and Group objects where the `CallRinging` status is assigned to each.

This propagation also true of multimedia interactions. In the multimedia equivalent to the preceding example, the Place, Agent, and/or Group objects would take on the `InteractionDelivering` status.

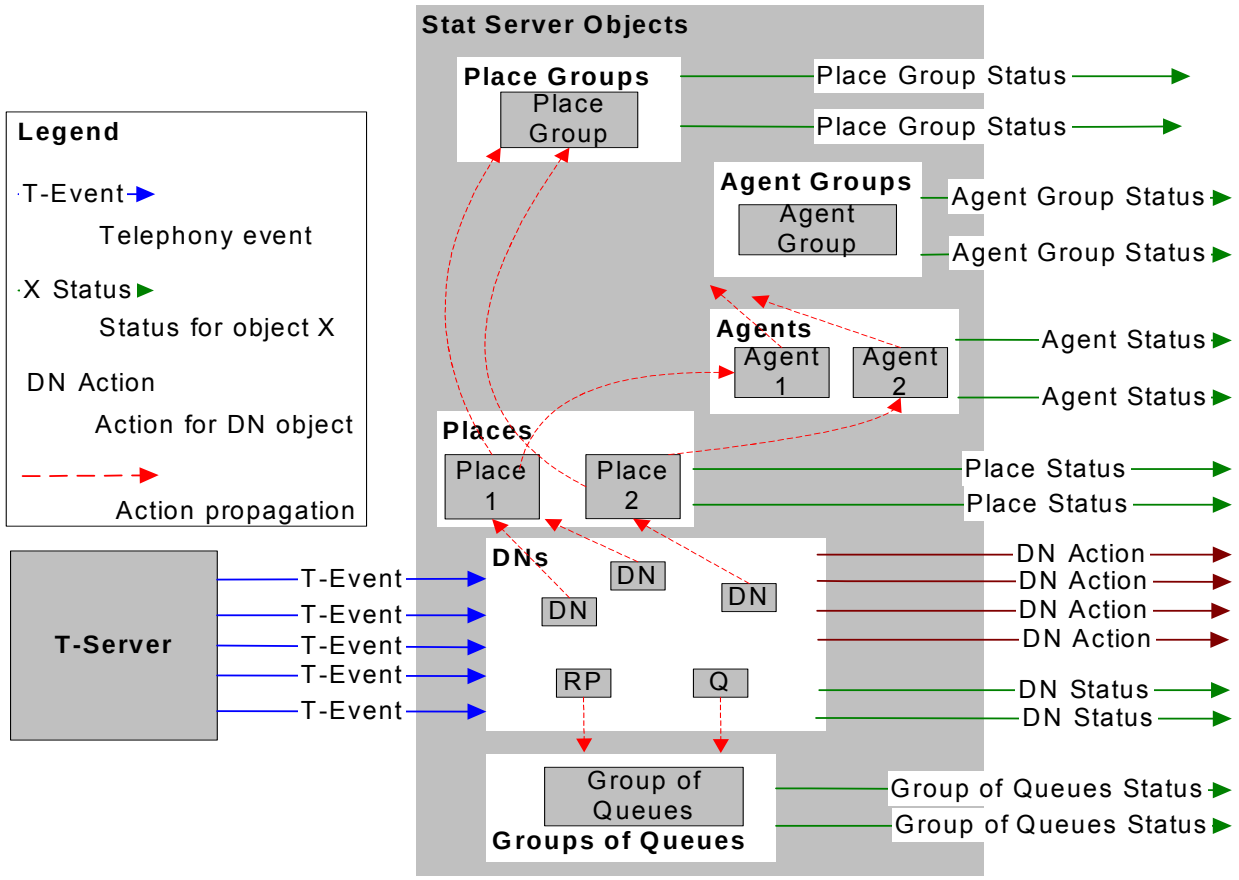

Figure 60: Action Propagation Among Objects

Telephony Actions and Events

The Statistical Model telephony actions are comprised of attributes generated on the basis of underlying TEvents. These attributes are described in Table 2.

Table 2: Stat Server Action Attributes

Attributes	Description
ANI	The Automatic Number Identification indicating all or part of the caller's telephone number. For durable actions, this attribute is inherited from the first TEvent and does not change during the action's life. For instantaneous actions, ANI directly transfers from the corresponding TEvent.
DNIS	The Dialed Number Identification Service indicating all or part of the telephone number that was dialed to make a call. For durable actions, this attribute is inherited from the first TEvent and does not change during the action's life. For instantaneous actions, DNIS directly transfers from the corresponding TEvent.
CustomerID	The attribute indicating the tenant ID in a multitenant environment.

Table 2: Stat Server Action Attributes (Continued)

Attributes	Description
MediaType	The attribute indicating the kind of environment through which the interaction is distributed. The predefined media types include voice, e-mail, and chat.
ThisQueue	The number of the queue.
TreatmentType	The type of the treatment applied to a call, such as silence, music, busy, and so forth.
UserData	Calculated on the basis of attached data to TEvents. For instantaneous actions, `UserData` is directly inherited from its corresponding TEvent. For durable actions, this attribute might vary. This attribute is initially set based on the first triggering TEvent. If triggered during an action, other TEvents with attached data can arrive to update this attribute. For instance, receiving an `EventAttachedDataChanged` TEvent updates this attribute. A final update is made following the ending TEvent. `UserData` uses TKV-List format with a list of TKV pairs.
Extensions	Enables Stat Server to filter switch-specific and other features on any specified key-value pair recorded in the `AttributeExtensions` attribute of select TEvents.
Reasons	Refers to additional data that is included in the TEvent to provide reasons for and results of actions taken by an agent.

Statistical Model Telephony Actions

The current version of Stat Server offers a large set of actions, which are described in the *Framework 8.0 Stat Server User's Guide*.

Actions can be separated into two groups:

- Interaction-related actions, reflecting events arising from particular interactions, such as TEvents that carry a `ConnID` or multimedia events that carry an `IxnID`.
- Non-interaction-related actions, caused by events not stemming from any particular interaction.

Note that Stat Server remembers the connection or interaction ID of an interaction because this ID provides the criterion for distinguishing between actions. More than one interaction-related action of the same kind can occur simultaneously at the same DN.

Telephony Object Statuses

Stat Server defines a special sort of durable action to characterize an object's status. Contrary to actions, objects can hold only one status at any point in time. Object statuses cannot overlap.

At any moment, object status is determined based on its ongoing durable actions and ranking in the `Status Priority` tables. The `Status Priority` tables list durable actions in order of priority. For example, the `Regular DN Status Priority` table (for instance, a priority table for Regular DN object) looks as follows:

```
NotMonitored<Monitored<OnHook<WaitForNextCall<OffHook
<CallDialing<CallRinging<NotReadyForNextCall<AfterCallWork
<CallOnHold<CallUnknown<CallConsult<CallInternal
<CallOutbound<CallInbound<ASM_Engaged<ASM_Outbound
```

Note: The `Regular DN Status Priority` table is the default priority table. Your system administrator might modify the table; see the *Framework 8.0 Stat Server User's Guide* for details.

Here, actions are listed in order of increasing priority. For example, the `CallInbound` action has higher priority than the `OffHook` action.

The status of a Regular DN object is determined by its highest-priority, ongoing durable action (illustrated in Figure 61). It shows an action flow for a Regular DN object. The object participates in several actions simultaneously, which are graphically depicted in the top four bars. The fifth bar shows the object's status and how it changes over time. Clearly, `Status Priority` tables are necessary tools.

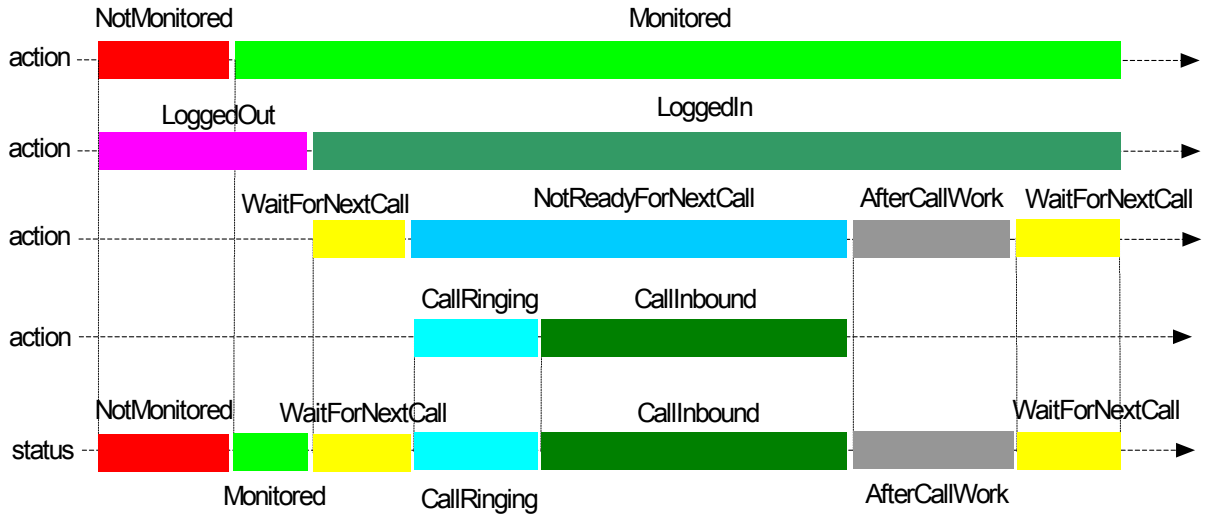

Figure 61: Actions and Status for Regular DN Object

Durable actions that do not appear in the `Status Priority` table have no effect on an object's status. For example, the `LoggedIn` and `LoggedOut` actions in Figure 61 do not affect the status of the Regular DN object.

Notice also that the duration of an action and its status need not coincide. In the figure above, the `Monitored` action and the `Monitored` status have different durations.

The `Status Priority` table for a Mediation DN object is simpler and could be written as follows:

`NotMonitored < Monitored < CallWait`

Figure 62 illustrates how the status of a Mediation DN object is determined.

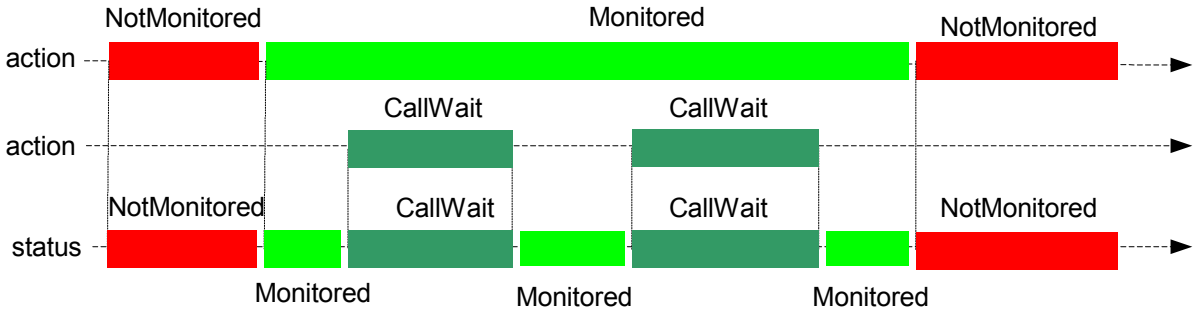

Figure 62: Actions and Status for a Mediation DN Object

The statuses of a `Place` object and related `Agent` objects are determined from the status of the DN associated with the place. If a `Place` object has only one DN, then its status assumes that DN's status. If the `Place` object is associated with several DNs, then status is determined by the `Regular DN Status Priority` table. For example, if the `Place` object has two DNs, and one is in `WaitForNextCall` status, the other in `CallInbound` status, then the place assumes the `CallInbound` status, which ranks higher in the `Status Priority` table than `WaitForNextCall`. The statuses of agents registered to that place are the same.

The statuses of place groups and agent groups are determined in a simpler and more intuitive manner. A place group can have but one of the following statuses:

- `Monitored`
- `NotMonitored`
- `WaitForNextCall`
- `NotReadyForNextCall`

A group of places is assigned the `NotMonitored` status if all its places have `NotMonitored` statuses. But if even one place reaches `Monitored` status, then the group is assigned the `Monitored` status. If at least one place reaches the `WaitForNextCall` status, then the group has `WaitForNextCall` status. If all places in the group reach `NotReadyForNextCall` status, then the group has `NotReadyForNextCall` status.

Therefore, the group is in `WaitForNextCall` status if it can receive a new call and it has a `NotReadyForNextCall` status if all of its places are busy (that is, all have `NotReadyForNextCall` or `NotMonitored` statuses).

The status of a group of agents is determined in a similar fashion based on the places where agents log in. A group of agents might have the following statuses:

- `Monitored`
- `NotMonitored`
- `WaitForNextCall`
- `NotReadyForNextCall`
- `LoggedOut`

If not one agent of a group is logged in to a place, then the group is said to have `LoggedOut` status.

Tracking Agent Status in a Multimedia Environment

Besides interaction processing, the multimedia interaction model of Genesys eServices tracks agent status. Figure 63 shows all the conditions that affect agent status.

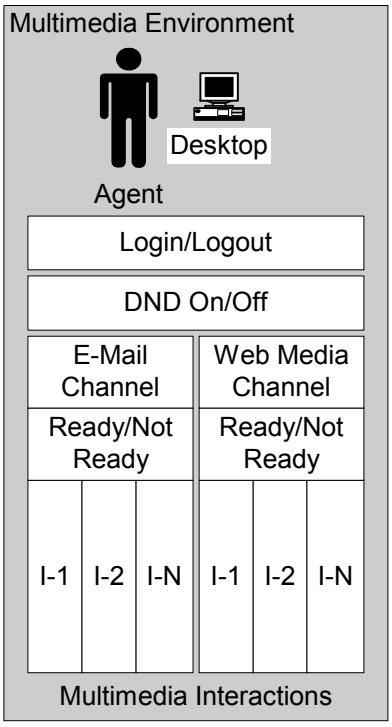

Figure 63: Agent Status Conditions

An agent might access to several media channels. Figure 63 shows two channels, e-mail and web media (chat). Each channel might have several active interactions. Moreover each channel has a `Ready/NotReady` status indicating whether the agent can accept more interactions for this channel.

Besides the media channels and their conditions, agent status is characterized by `Login/Logout` and `Do Not Disturb On/Off` states.

Agent Status State Machine

The state machine representing agent status is presented in Figure 64. It includes four controlling factors, `Login/Logout`, `Add/Remove Media Channel`, `Channel Ready/Not Ready`, `Do Not Disturb On/Off`, and depicts their interrelationships.

This state machine shows only two media types e-mail and web media (chat). However, you can construct a similar machine for other sets of media types.

This state machine represents the behavior of a monitored agent. This means that transitions from state to state are triggered by events received from Interaction Server.

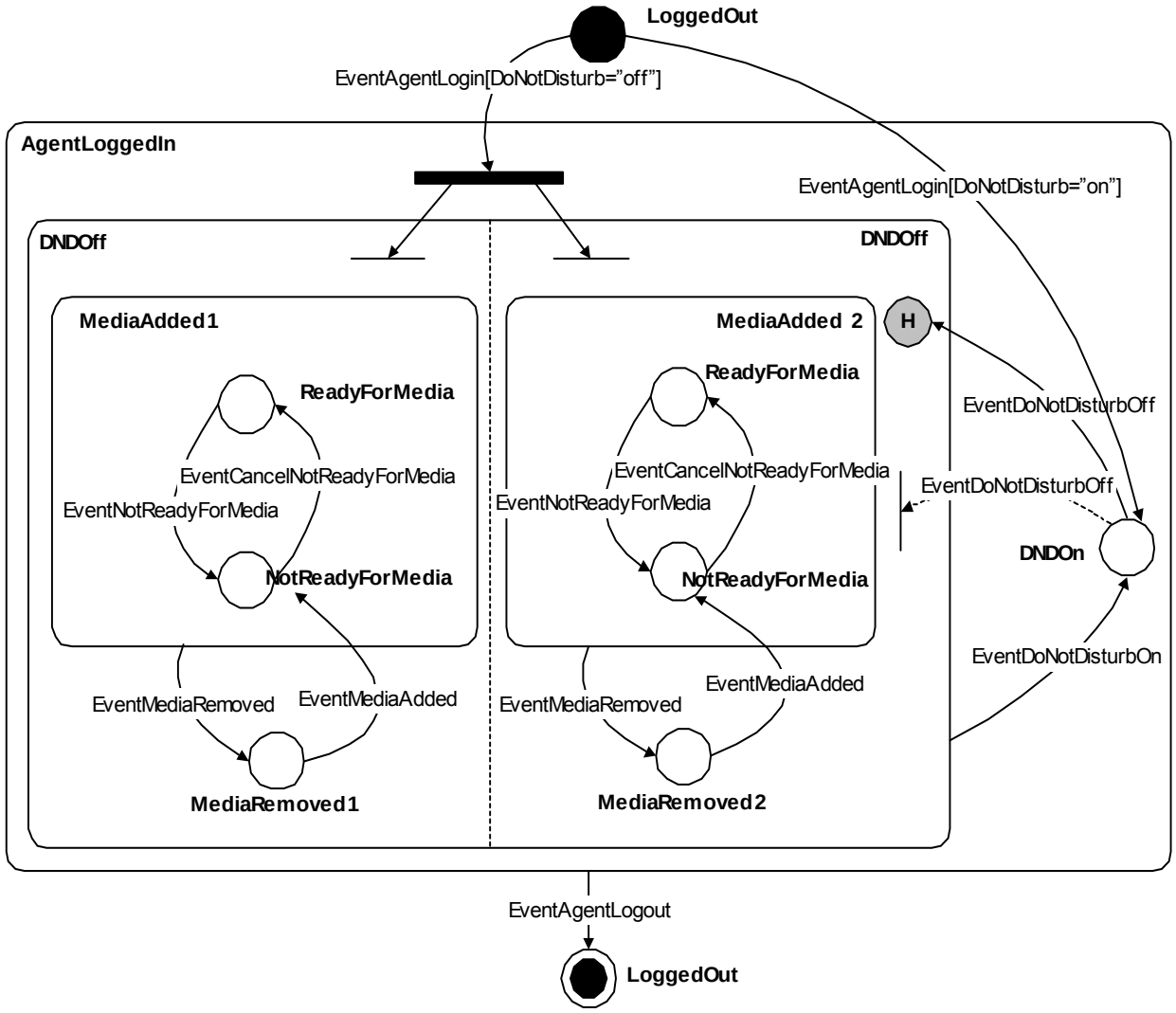

Figure 64: Multimedia Agent State Machine

Let us explain behavior represented in the diagram and interpret it in terms of agent status behavior.

Initial and Final States

The initial and final states, represented by a solid black dot and a black dot surrounded by circle respectively, correspond to the `Loggedout` agent state.

The AgentLoggedIn State

When Stat Server receives an `EventAgentLogin` event from Interaction Server, the machine proceeds to the `AgentLoggedIn` state. `AgentLoggedIn` is a super state that contains two states `DNDOff` and `DNDOn`. The state machine enters one of the

states depending on the setting for the DoNotDisturb parameter contained in the EventAgentLogin event.

The Media Channels

DNDOff is also a super state, which contains parallel sequences, one for each media channel. Each sequence consists of a state machine representing the status of the media channel. For example, the state MediaAdded_1 indicates that the specified media channel is in an active media list and can be used for receiving new interactions. The MediaRemoved_1 state indicates that the media channel does not exist in an active media list. The agent state machine can enter the MediaRemoved_1 state via an EventMediaRemove event or via an EventAgentLogin event that does not contain the particular media channel in its MediaTypeList parameter.

The MediaAdded and MediaRemoved States

The MediaAdded_*N* (*N*=1, 2) state is a super state which represents the availability of the specified channel or channels for receiving interactions. The state changes are triggered by the EventNotReadyForMedia and EventCancelNotReadyForMedia events.

Note: Note that the initial states of super states, represented by arcs that end at a thick bar, are not specified. The actual states depend upon parameters in EventAgentLogin event. This event contains MediaTypeList parameter that contains a list of media types that will be activated after login. If, for example, both media types are present, the initial states will be MediaAdded_1 and MediaAdded_2. If, however, the list will contain only second media then initial states will be MediaRemoved_1 and MediaAdded_2.

If some media channel, for example a channel called Media 1, is active (that is, the state machine indicates its condition with the MediaAdded_1 state) then at any time it might become inactive after receiving an EventMediaRemoved event. If, after receiving the EventMediaRemoved event, Stat Server receives an EventMediaAdded event, the state machine changes to the MediaAdded_1 super state with the NotReadyForMedia substate within it.

For each media type presented in MediaTypeList parameter the EventAgentLogin contains its initial state ReadyForMedia or NotReadyForMedia that point out corresponding initial states within super states MediaAdded_*N* (*N*=1, 2).

Note: The EventRemoveMedia event does not influence the state of interactions currently being processed.

Do Not Disturb Events

If agent has active channels and is in the `DNDOff` super state, a `EventDoNotDisturbOn` event effectively means that the agent cannot receive a new interaction by any channel. Note that a `EventDoNotDisturbOn` event freezes the activity that was happening when the agent was in the `DNDOff` state. These become active again after reception of an `EventDoNotDisturbOff` event. In this case, the internal `DNDOff` super state is restored, indicated in Figure 64 on page 83 by the H symbol. 

If the `EventAgentLogin` event has the `DoNotDisturb` parameter set to `on`, then the state machine enters the `DNDOn` state, where all media channels are inactive. They become active after receiving `EventDoNotDisturbOff` event. In this case, the initial states of the `DNDOff` super state are set according to the other parameters contained in the event. This case is indicated in Figure 64 by a dotted arc that ends in a vertical bar.

The EventAgentLogout Event

Receiving `EventAgentLogout` event at any time results in termination of all activities, which corresponds to the clearing all states within the `AgentLoggedIn` super state.

Metrics: Their Composition and Definition

A *metric* defines what and how Stat Server is to measure certain interactions within a contact center. A metric is defined by four elements:

- Statistical type
- Time range
- Time profile
- Filter

Each is described in the following subsections.

Statistical Type

A *statistical type (stat type)* is comprised of one or more of the following statistical parameters:

- Objects
- Category
- Subject
- MainMask
- RelMask (optional)
- Description (optional)
- MediaType (optional)
- Formula (optional)
- UseSourceTimeStamps

The “Stat Server Stat Type Definitions” chapter in the *Solution Reporting Templates* book of the *Reporting Technical Reference* series offers all of the predefined statistical types used in the Genesys-provided reports to define

metrics. In addition, you can create you own stat types. The “Creating Custom Stat Types” chapter in the *Customization* book of this series shows you how.

Note: When loaded, Java Stat Server Extensions (SSJE) pass their own stat type definitions for all inherent statistical types to Stat Server, making them available to Stat Server clients. These stat types can be real-time or historical and, unlike regular stat types, are dynamic in nature. This means that they are enabled only if the corresponding SSJE is loaded.

These are the parameters for Java stat types:

- JavaSubCategory
- AggregationType
- Description
- Category
- Objects
- <business attribute> (MediaType)

Objects

The *object types* assigned to a stat type are formed from the list of object types Stat Server supports, namely:

- Agent
- Agent Group
- Calling List
- Campaign
- Campaign Calling List
- Campaign Group
- Place
- Place Group
- Queue
- Queue Group
- Regular DN
- Routing Point
- Routing
- Staging Area
- Switch
- Tenant
- Virtual Queue
- Virtual Routing Strategy
- Workbin

A stat type is typically defined for several compatible object types, rather than just one, which allows one stat type to serve several metrics. However, a stat type can only be applied to object types within the same *compatibility group*. The group of object types is said to be compatible if objects within the group are reachable during the propagation of actions (see “Action Generation and Propagation” on page 77). For example, the RegDN, Place, Agent, GroupAgents, and GroupPlaces object types all belong to the same compatibility group because all of them are reachable during propagation of an action started at the RegDN object.

Figure 65 shows the partitioning of object types into the Agent, Queue, and Campaign compatibility groups. Staging Area (an e-mail “queue”) and Tenant are each the sole member of their group.

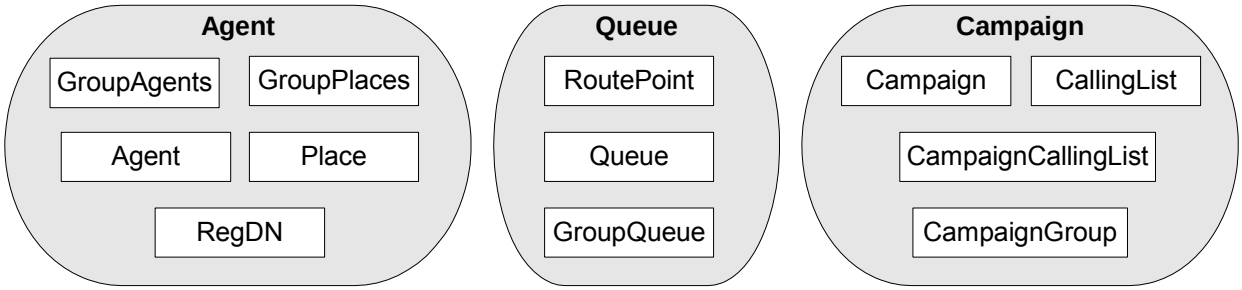

Figure 65: Partitioning Objects into Compatibility Groups

For example, assigning the RegDN, Place, and Agent object types to one stat type is valid, but assigning the RegDN, Place, and GroupQueue object types to one stat type is not. The GroupQueue object falls in a different compatibility group than RegDN and Place objects.

MainMask

A *main mask* specifies the set of actions or statuses Stat Server considers when calculating a statistic (see also "RelMask" on page 88). In the simplest case, the main mask might specify only one action or status. For example, a main mask specifying the `CallInbound` action provides a statistic related to this one action. If you need a total number of inbound calls on some DN, then the statistic calculates the number of occurrences of the `CallInbound` action within a specified time interval. If you need total duration of inbound calls on some DN, the metric sums the duration of all `CallInbound` actions within the specified time interval.

The main mask might be constructed from several actions (or statuses). If so, the statistic considers all specified actions without differentiating between them. For example, suppose the main mask is constituted of two actions: `CallInbound` and `CallOutbound`. To calculate total time of external (inbound and outbound) calls, use this mask. The metric sums the time duration of all these actions.

Genesys does not recommend assigning overlapping actions to one mask. For example, an assignment of the `Monitored` and `LoggedIn` actions results in the return of senseless data because, by definition, statuses cannot overlap.

Specify a main mask by a list of comma-separated action (or status) names. Use the asterisk (*) symbol to specify all actions (or statuses); use the tilde (~) symbol to exclude a particular action (or status). The following examples are valid main-mask specifications:

```
MainMask = CallInbound
MainMask = CallInbound,CallOutbound
MainMask = *, ~LoggedOut
```

In the last line, the main mask is specified as all statuses except `LoggedOut`.

It is very important to specify masks correctly. For example, these two main masks are not equivalent:

```
MainMask = LoggedOut
MainMask = *, ~LoggedIn
```

Indeed, despite the fact that `LoggedIn` and `LoggedOut` actions are complementary, calculations of duration time based on these two masks might report different results because the first mask indicates the duration when an agent was logged out and monitored, while the value of the second mask indicates the duration an agent was not logged in, which is equal to the time when agent was logged out and monitored plus the time s/he was not monitored. The latter value might be greater than the former.

RelMask

You specify a *relative mask* for calculating statistics reflecting relative values, such as percentages and averages. For example, a metric defining the percentage of inbound calls in all external calls (inbound and outbound) requires a relative mask. But assign the superset to the relative mask; otherwise the end result makes no sense:

```
MainMask=Inbound
RelativeMask=CallInbound,CallOutbound
```

The syntax is the same as for a main mask.

Subject

The *subject* of a stat type specifies the object type that will be considered as a source of statistical data. A subject might be one of the following:

- DNAction
- DNStatus
- Action
- AgentStatus
- PlaceStatus
- GroupStatus
- CampaignAction

The first part of each compound name indicates the object type: `DN`, `Agent`, `Place`, `Group`, or `Campaign`. The second part indicates whether to consider actions or statuses. Hence, a *DNAction* subject assignment reveals that the source of statistics for all objects is the *actions* of a regular `DN`. The *AgentStatus* subject reveals that statistics will be gathered from the *statuses* of `Agent` objects.

The *Action* subject is used for multimedia statistics. It is analogous to the `DNAction` subject for telephony interactions.

To clarify the subject's role, consider the following example:

```
MainMask=CallOnHold
Object=GroupAgent
Subject=DNAction
```

This definition tell us that metrics should be calculated for the `GroupAgent` object. The data source is `Regular DN` objects, which is where the `CallOnHold` action is tracked. This action will be propagated from the `DN` object to the `GroupAgent` object (according to the action propagation process, as discussed

on page 77) and collected for this object. As a result, you receive a calculation of `CallOnHold` actions for the `GroupAgent` object.

If, however, you select `GroupStatus` as the subject, then you must redefine the main mask to track statuses pertaining to a group and you will receive a different statistical value.

Category

The *statistical category* element of a stat type determines how to calculate a statistical value. Think of a statistical category as the algorithm used to calculate a value. The calculation uses one or more masks as input parameters. Table 3 describes some of the statistical categories used for Historical Solution Reporting.

`Note:` Refer to the *Framework 8.0 Stat Server User's Guide* for complete information about statistical categories.

Table 3: Statistical Categories for Historical Solution Reporting

Statistical Category	Description
AverageTime	The quotient of the `TotalTime` aggregated value for the main mask and the `TotalNumber` aggregated value for the relative mask.
TotalNumber	For subject DN action, the total number of actions listed in the mask that ended (for durable actions) or occurred (for instantaneous actions) during the interval from which the statistic is calculated. For subject DN status (and, respectively, agent status and place status), this is the total number of statuses listed in the mask that either started or are in progress during the interval from which the statistic is calculated.
TotalTime	The sum of all durations of durable and retrospective, instantaneous actions or of statuses listed in the mask that: • Ended (for durable actions) • Occurred (for retrospective, instantaneous actions) • Either started or are in progress (for statuses) during the interval from which the statistic is calculated. Momentary actions listed in the mask are ignored since they do not have a duration. If a statistic is requested for statuses, Stat Server uses the status duration within the statistical interval for calculation; otherwise Stat Server uses the entire action duration.

Table 3: Statistical Categories for Historical Solution Reporting (Continued)

Statistical Category	Description
TotalTimeInTime Range	Represents the total duration of all durable and retrospective, instantaneous actions or of statuses listed in the mask that ended (for durable actions or for statuses) or occurred (for retrospective, instantaneous actions) during the interval from which the statistic is calculated and whose duration is within the specified time range. Unlike other historical aggregated values, these values depend not only on the mask and the interval from which the statistic is computed, but on the time range as well.
MaxTime	The maximum duration among all durations of durable and retrospective, instantaneous actions or statuses listed in the mask that: • Ended (for durable actions) • Occurred (for retrospective, instantaneous actions) • Either started or are in progress (for statuses) during the interval from which the statistic is calculated. Momentary actions listed in the mask are ignored since they do not have a duration. If a statistic is requested for statuses, Stat Server uses the status duration within the statistical interval for calculation; otherwise Stat Server uses the entire action duration.
MinTime	The minimum duration among all durations of durable and retrospective, instantaneous actions or of statuses listed in the mask that: • Ended (for durable actions) • Occurred (for retrospective, instantaneous actions) or • Either started or are in progress (for statuses) during the interval from which the statistic is calculated. Momentary actions listed in the mask are ignored since they do not have a duration. If a statistic is requested for statuses, Stat Server uses the status duration within the statistical interval for calculation; otherwise Stat Server uses the entire action duration.
TotalAdjusted Number	Sums the total number of occurrences of actions or statuses listed in the main mask that are ended during the interval from which the statistic is calculated. • The `TotalAdjustedNumber` category differs from `TotalNumber` only if reset-based notification is used for the statistic. For all other notification modes, `TotalAdjustedNumber` values are the same as `TotalNumber`.

Table 3: Statistical Categories for Historical Solution Reporting (Continued)

Statistical Category	Description
TotalAdjustedTime	If a statistic is requested with DN action specified, `TotalAdjustedTime` is the sum of all durations of durable and retrospective, instantaneous actions listed in the mask that: • Either ended or are in progress (for durable actions) • Occurred (for retrospective, instantaneous actions) during the interval from which the statistic is calculated. Momentary actions listed in the mask are ignored since they do not have a duration. Only the duration time that is within the interval is used in this calculation. For status-based statistics, `TotalAdjustedTime` is the sum of all durations of durable and retrospective, instantaneous actions listed in the mask that ended or occurred (for retrospective, instantaneous actions) during the interval from which the statistic is calculated. Stat Server uses the overall status duration in this calculation. A statistic of this category must be requested with the reset-based notification; that is, a statistic is reset to zero when a new interval starts. • The `TotalAdjustedTime` category differs from `TotalTime` only if reset-based notification is used for the statistic. For all other notification modes, `TotalAdjustedTime` values are the same as `TotalTime` values.

Using Statistical Categories: Examples

How does the choice of different statistical categories and subjects influence the calculation method? Figure 66 presents a simple example of the `CallInbound` action and status on a single regular DN. Let us consider two stat types formulated for the same statistical category, one with Subject=`DNAction`, the other with Subject=`DNStatus`.

For simplicity, there is no more than one action at a time and each occurrence of the action changes the status to the same type, `CallInbound`, as the action.

Suppose you observe these actions and statuses during two time intervals: 10:00-10:15 and 10:15-10:30. During these periods, a `CallInbound` durable action occurs on the DN several times. Each occurrence is preceded by a `CallInboundStarted` initial momentary action (not shown).

> Note: For graphical simplicity, Figure 66 shows calls in minutes. However, because Stat Server tracks seconds, the results for each statistical category in the following discussion are given in seconds (where appropriate) rather than minutes.

These stat types have different meanings that are reflected in the statistical calculations. In the examples that follow, the choice of subject for the stat type is shown in bold typeface.

- **Subject is Action Based**—The number and duration of actions during a time interval are not calculated until the action has been completed. Therefore, an action that starts in one interval and ends in the next is only counted in the second interval. However, the duration time given in the second interval includes the entire duration of the action, even the part that occurred during the first interval.
- **Subject is Status Based**—The number and duration of each occurrence of a status during an interval is calculated for that interval, even if the status continues past the end of the interval. Only the time that falls within the interval is calculated, even for actions that start or end outside of the interval.

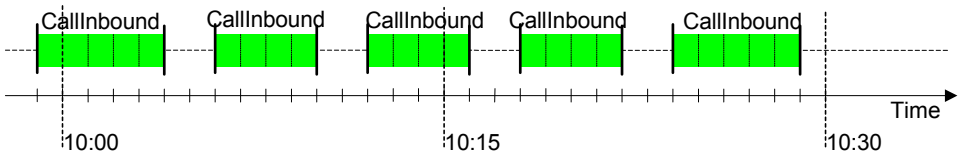

Figure 66: Example of CallInbound Action and Status

TotalNumber Statistical Category

First consider the `TotalNumber` statistical category and the values it yields in the example when applied to the `CallInbound`:

Action—For the first interval 10:00-10:15, `TotalNumber` returns a value of 2 because only two actions ended within this interval. For the second interval, 10:15–10:30, `TotalNumber` yields 3 since three `CallInbound` actions ended during this interval.

Status—Both intervals yield 3 because there were 3 occurrences of the status during each interval.

Initial Momentary Action—If you apply the `TotalNumber` category to the `CallInboundStarted` initial momentary action, then you obtain a value of 2 for each interval, because the action occurred two times within each interval.

TotalTime Statistical Category

Now consider the `TotalTime` statistical category and the values it yields in the same example when applied to the `CallInbound`:

Action—The first interval yields 540 seconds, representing the total duration of the two actions that ended within the interval. The third action that occurs at 10:12 does not contribute to the final result as it does not end within the interval. For the second interval, you obtain 780 seconds since all three actions end within the interval.

Status—The first interval yields 660 seconds; the second interval yields 600 seconds. These values include the duration of each occurrence of the `CallInbound` status during the interval.

Initial Momentary Actions—Applying the `TotalTime` category to the `CallInboundStarted` momentary action results in a value of `0` because initial momentary actions have no duration.

MaxTime Statistical Category

Now apply the `MaxTime` statistical category to the `CallInbound`:

Action—You obtain a value of 300 seconds for the first interval. Two actions (the first and second) ended during the interval. The total duration of each action that ended in the interval is counted, so the longer first action provides the value. For the second interval, you obtain a value of 300 because the longest action in the interval lasts 300 seconds.

Status—You receive values of 240 and 300 for first and second intervals respectively.

MinTime Statistical Category

The `MinTime` statistical category returns:

Action—A value of 240 for both intervals when applied to the `CallInbound` action.

Status—A value of 240 for both intervals when applied to the `CallInbound` status.

TotalAdjusted Time Statistical Category

Action—Returns values of 660 seconds and 600 seconds, respectively, for the `CallInbound` action.

Status—Returns values of 540 and 780 for the first and second interval, respectively, for the `CallInbound` status.

TotalAdjusted Number Statistical Category

The `TotalAdjustedNumber` category returns the values `2` and `3` for the first and second interval, respectively, for both `CallInbound` action and status.

Calculation Rules for Statistical Categories

The standard Genesys Historical Reporting reports statistical data for a particular time interval; each reported number is calculated only for that time interval (whether it be 15 minutes, 1 hour, 1 day, or other). Because the specified time interval and the corresponding interactions might overlap in different ways, you must understand how each metric is calculated.

There are four possible scenarios for how interactions might overlap during a reporting time interval, namely:

- Scenario 1—The interaction starts and ends within the time interval.
- Scenario 2—The interaction starts before the time interval and ends within the time interval.
- Scenario 3—The interaction starts during the time interval and ends after the time interval.
- Scenario 4—The interaction starts before the time interval and ends after the time interval.

For each scenario, the following calculation rules clarify how the statistical categories function:

- `TotalNumber` calculates the total number of interactions finished during the time interval.
- `TotalTime` calculates the total number of interactions existing during the time interval.

- `AverageTime` calculates the average duration of all completed interactions within the time interval.

Scenario 1 The first and simplest scenario occurs when a particular interaction both starts and ends within a time interval. Figure 67 depicts this scenario.

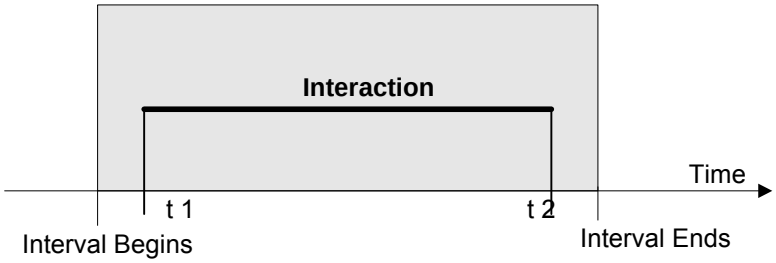

Figure 67: Entire Interaction Occurs Within Interval

The `TotalNumber` statistical category yields a value of 1 because the interaction ends within the time interval. `TotalTime` yields the time indicated by t_2–t_1. `AverageTime` yields the result of (t_2–t_1)/1.

Scenario 2 Scenario 2 occurs when the interaction starts before the time interval and ends within the time interval (see Figure 68).

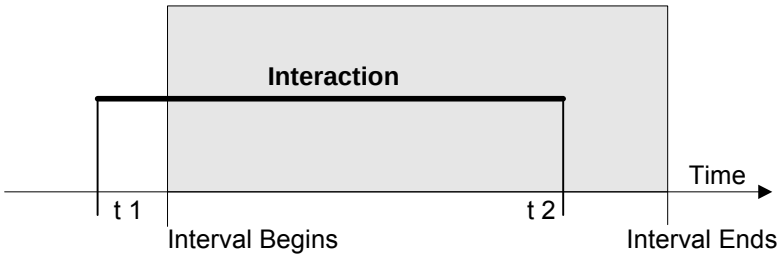

Figure 68: Interaction Begins Before but Ends Within Interval

The `TotalNumber` statistical category yields a value of 1 because the interaction ends within the time interval. `TotalTime` yields the time indicated by t_2–[Interval Begins]. `AverageTime` yields the result of (t_2–t_1)/1.

Two `TotalTime` metrics are calculated for this scenario:

- The total duration of all interactions occurring within the time interval, counting only the time that falls within the specified time interval, that is, t_2–[Interval Begins].
- The total duration of all interactions ending within this time interval, counting the entire duration of interactions including even duration that falls outside of the time interval. In this case it is the result of (t_2–t_1).

  A second stat type is necessary to calculate the average interaction time for the aggregated time intervals.

Scenario 3 Scenario 3 (see Figure 69) occurs when the interaction starts within the time interval, but ends following the time interval.

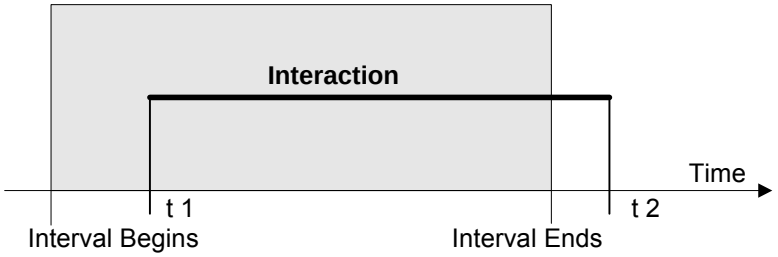

Figure 69: Interaction Begins During but Ends Outside Interval

The `TotalNumber` statistical category yields a value of 0 because the interaction does not end within the time interval. `TotalTime` yields the time indicated by [Interval Ends]–t_1. `AverageTime` yields 0 because no interactions ended within the time interval.

Two `TotalTime` metrics are calculated for this scenario:

- The total duration of all interactions occurring within the time interval, counting only the time falling within the specified time interval (<Interval Ends>–t_1).
- The total duration of all interactions ending within the time interval including the entire interaction time, even the duration falling outside the time interval. In this case, the result is 0 because no interaction ends within the interval.

Scenario 4

Scenario 4 (see Figure 70) occurs when the interaction starts before the time interval and ends following it.

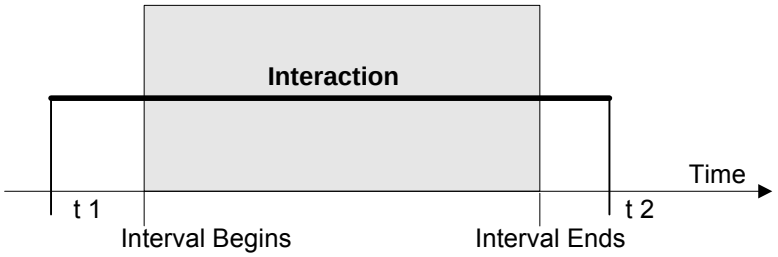

Figure 70: Interaction Begins Before but Ends After Interval

The `TotalNumber` statistical category yields a value of 0 because the interaction does not end within the time interval. `TotalTime` yields the time indicated by [Interval Ends]–[Interval Begins]. `AverageTime` yields 0 because no interactions ended within the interval.

Two `TotalTime` statistical categories are calculated for this scenario:

- The total adjusted duration of all interactions happening within the time interval, counting only the duration within the specified time interval, namely: [Interval Ends]–[Interval Begins], the length of the interval.
- The total duration of all interactions ending within time interval. This does not apply to this scenario, so the result is 0.

Statistical Category Summary Example

The `TotalAdjustedTime` statistical category is applicable to reset-based time profiles and behaves as follows:

- For a `DNAction` or `Action` subject, `TotalAdjustedTime` reports finished and unfinished actions for the current interval. This is necessary for reports where you want to see that an agent has performed work, even if it is not yet finished.
- For the `DNStatus` and `AgentStatus` subjects, however, `TotalAdjustedTime` causes Stat Server to report the entire time a DN or Agent status occurs only if it ends within the time interval. This is necessary to correctly calculate averages after daily, weekly, monthly, quarterly, and yearly aggregations.

For example, an agent performs an action that spans two 15-minute intervals. The action endures 14 minutes, consuming 7 minutes in each interval (see Figure 71).

Figure 71: Statistical Category Summary Example

Historical statistical categories would report the following:

	TotalTime		TotalAdjustedTime		TotalNumber		TotalAdjustedNumber	
	1st Interval	2nd Interval	1st Interval	2nd Interval	1st Interval	2nd Interval	1st Interval	2nd Interval
Action	0	14	7	7	0	1	0	1
Status	7	7	0	14	1	1	0	1

Formula

Custom formula is an element of a statistical type. This formula might contain a component called `DistByConnID` (or DCID for short) to distinguish actions related to the same call (that is, having the same `ConnID`). When used, this qualifier brings the call identifier into the equation when determining statistical values.

Suppose, for example, the six durable `CallOnHold` actions illustrated in Figure 72 are observed on a regular DN within a 15-minute period. If you calculate the total number of occurrences of this action without using the `DistByConnID` qualifier, you get a value of 6, which could be interpreted as the number of times the *DN* was on hold. If, however, you want to know how many *calls* were on hold during the interval, you must apply the `DistByConnID`

qualifier. In this case, the metric calculates only first, third, and fourth actions. The second action is not considered because it has the same connection ID as the first action. The fifth and sixth actions are also dropped from the calculation for the same reason. As a result, you get a value of 3—exactly the number of calls held during the interval.

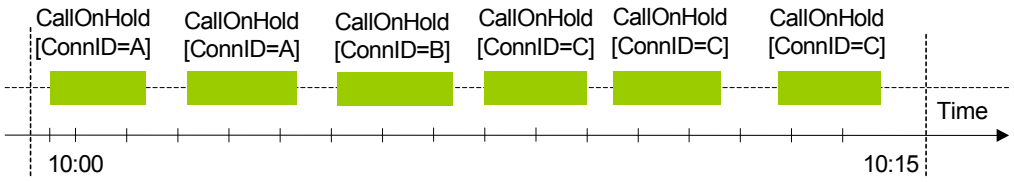

Figure 72: Using DistByConnID Qualifier

The `DistByConnID` component affects only number-related, percentage, and average metrics. Time-related metrics are not affected and, in fact, ignore this qualification. This means that the total time of `CallOnHold` in the previous example will be summed as normal.

Description

An optional description of the stat type. Stat Server does not use this parameter to calculate values or to generate actions.

MediaType

MediaType is the business attribute that you can use to distinguish the type of media for which Stat Server is collecting data using the associated stat type.

You can configure a single core stat type and use it to collect statistics for multiple media types, using filters to distinguish the media types. However, statistics that derive from the Stat Server Java extension are distinguished by the value of the `MediaType` parameter. You cannot use filters to distinguish media for these statistics. Therefore, you must configure separate stat types for each media type using extension statistics.

`Note:` You can use only one business attribute when defining a stat type.

UseSourceTimeStamps

For those metrics that qualify, this parameter specifies whether Stat Server uses the actual time that events were transmitted to Stat Server (source timestamp) or the time that Stat Server acknowledges receipt of the events (the default behavior) when calculating metric duration.

JavaSubCategory

This parameter is used only in Java stat types; that is statistics drawn from the Stat Server Java Extension. It is the name of the Java subclass that implements statistic calculation.

AggregationType

This parameter is only used in Java stat types. It indicates the kind of aggregation that the client application is to perform on the data sent by Stat Server.

Stat Type Examples

Now apply these concepts to a couple of stat types.

Stat Type: *TotalHandleStatusTime*

```
Objects = Agent, Place, GroupAgents, GroupPlaces
Category = TotalTime
MainMask = CallInbound, CallOutbound, AfterCallWork
Subject = AgentStatus
```

Use the `TotalHandleStatusTime` stat type to calculate the total call-handling time by agents. You can apply to this stat type four objects:

- Agent
- Place
- GroupAgents
- GroupPlaces

If the object is Agent, then the calculation sums the duration (category `TotalTime`) of times that the agent spends processing inbound and outbound calls along with after-call-work times. This calculation is defined by a list of actions in the main mask of the type. The calculation yields an identical result for the Place object.

If the object type is group of agents (or group of places), then the value is calculated as a sum of all total times for all agents (places) of the group.

Stat Type: *TotalNumberInboundCalls*.

```
Objects = RegDN, Agent, Place, GroupAgents, GroupPlaces
Category = TotalNumber
MainMask = CallInbound
Subject = DNAction
```

The `TotalNumberInboundCalls` stat type defines the parameters for calculation of the total number of inbound calls for objects like Regular DN, Agent, Place, Group of Agents, and Group of Places. If the object is a Regular DN, then Stat Server sums all inbound calls occurring on this DN. If the object is an agent who has only one DN, then the calculated value is identical to its DN. If, however, the agent has two or more DNs, then the value is a sum of the values of all its DNs. A similar summation is performed for the Place and Group object types.

Time Profile

The *time profile* parameter defines the time intervals used for calculating historical aggregate values for statistics. Historical Reporting uses the `CollectorDefault` time profile which has a `Growing` interval type. This time profile defines moments of time when Stat Server returns statistical values to a client and resets statistics to zero to start collecting data for the next time period.

Note: You can find other time profile types described in the *Framework 8.0 Stat Servers User's Guide*.

Time profile is defined in the form `<time> +<increment>` where `<time>` is the time for the initial resetting of the statistics and `<increment>` defines a series of times for resetting statistics. For example, the time profile expressed as `08:00 +00:15` says that the initial reset procedure is performed daily at 8 AM and is reset every 15 minutes, at which point the statistical data is sent to Stat Server clients.

Thus, the time profile simultaneously defines intervals for collecting statistical values and a schedule for delivering these values to clients.

The time profile might have a more complicated form with several expressions separated with commas. For example, a time profile defined as "`08:00 +00:15; 17:00 +00.30`" says that the first reset will be at 8 AM. Resets will occur every 15 minutes until 5 PM. At 5 PM, resets will occur every 30 minutes until 8 AM of the next day, and so on.

Time Range

This statistical parameter defines the time range for collecting data for several stat types that calculate values occurring within the time range interval.

The time range is specified as two digits separated by a hyphen. The first digit corresponds to the starting point and the second to the end point, in seconds. Thus, the 0–30 time range defines a range between 0 and 30 seconds.

Figure 73 illustrates a time range applied to five calls within a queue. Suppose you wish to calculate the total number of calls distributed from a queue within 2 minutes (120 seconds). The time range is 0–120. Further, suppose that five `CallWait` durable actions occur within the 15-minute interval illustrated in Figure 73. Notice that four actions end with `CallDistributed` retrospective actions and one of them ends with `CallAbandoned`.

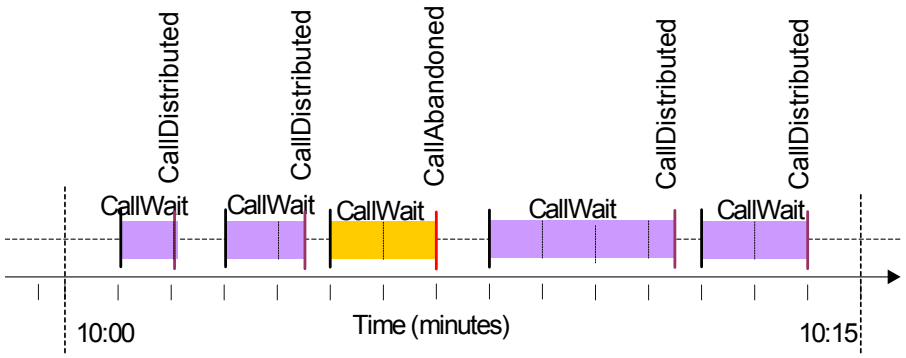

Figure 73: Using the Time Range Parameter

Now assign the `CallDistributed` retrospective action to the main mask of your stat type. Select the `TotalNumberInTimeRange` statistical category. A metric defined as such yields a value of 3. Indeed, the first, second, and fifth actions play into the calculation because these actions were distributed from the queue within 120 seconds. The third action does not because it does not meet the main mask specification; the call was abandoned, not distributed. Nor does the fourth action play into the calculation because its duration, exceeding 120 seconds, falls outside the time range.

Filters

A *filter* is the part of a metric that refines calculations of aggregated values. More specifically, filters exclude certain actions based on criteria specified in a logical condition. Filters are based on action attributes, such as `DNIS`, `ANI`, `CustomerID`, `MediaType`, `ThisQueue`, and `Treatment`.

Note: You cannot apply filters to Java-category statistical types.

You specify a filter by a text string containing a logical condition, which must be evaluated for each action. If the result is true, then the action is included in the calculation. Otherwise, the action is excluded.

These are examples of valid filters:

```
PairExist("CS","Gold")
ANI="1347"
(MediaType!=EMail)&(PairExist("CS","Platinum"))
```

The first filter determines if the action has user data with a TKV pair `("CS","Gold")`. The second filter checks to see if the action's `ANI` attribute is equal to `1347`. The third filter exacts a complex logical condition. The condition is true only if the action contains a `MediaType` attribute not equal to `e-mail` and if the TKV pair `("CS","Platinum")` exists in the user's data.

Chapter

3 Historical Reporting

This chapter describes the Genesys approach to Historical Reporting and includes these sections:

Introduction

Historical Reporting collects and presents information about contact center activities over long periods of time—weeks, months, and years.

The Historical Reporting architecture is presented in Figure 74. The primary sources of historical information are Stat Server and Configuration Server.

- Stat Server tracks statistics for contact center objects such as Agents, Agent Groups, Places, Group of Places, and so on. It interprets events generated by T-Server, SIP Server, and Interaction Server (for Genesys eServices).
- Configuration Server tracks contact center configuration information, dynamically updating data such as agents, their groups, and their skills.

Data from these sources goes through three stages: collection, transformation and aggregation, and delivery. Three subsystems support these activities:

- The Data Collection Services
- The Data Mart Services
- The Information Delivery Services

Data Collection Services

The three components of Data Collection Services gather raw information about objects and data generated by Stat Server:

- **Data Sourcer**—Obtains object statistics from Stat Server at each collection time interval and writes them to ODS.
- **Data Modeling Assistant (DMA)**—A graphical user interface (GUI) application with which contact center administrators define report layouts and set time intervals for data retrieval from Stat Server.
- **Operational Data Storage (ODS)**—Temporarily holds raw statistical data about objects sourced from Stat Server.

Data Mart Services

After collection in ODS, data enters the Data Mart Services, which organizes, summarizes, and optimizes data for Solution Reporting using these components:

- **ETL Runtime**—Maps ODS tables to Data Mart tables, which are used for Solution Reporting and analysis. At each time interval, ETL Runtime extracts raw statistics from ODS, applies transformation and summarization rules, and loads the data into the Data Mart.
- **Data Mart**—The target database for static and ad-hoc reporting.
- **ETL Assistant**—A GUI tool enabling contact center managers to specify the Data Mart loading interval, ODS, and purging rules—the ETL time profiles—and browse available aggregation levels.

Information Delivery Services

Finally, Information Delivery Services provide report development and distribution. Both CCPulse+ and CC Analyzer draw on the Data Collection Services and Data Mart Services components to present historical information.

- **CCPulse+**—Generates real-time views as well as historical and query based views.
- **CC Analyzer**—Uses Hyperion Interactive Reporting, which is redistributed by Genesys, to generate historical reports.

The following sections describe these data-processing stages in more detail.

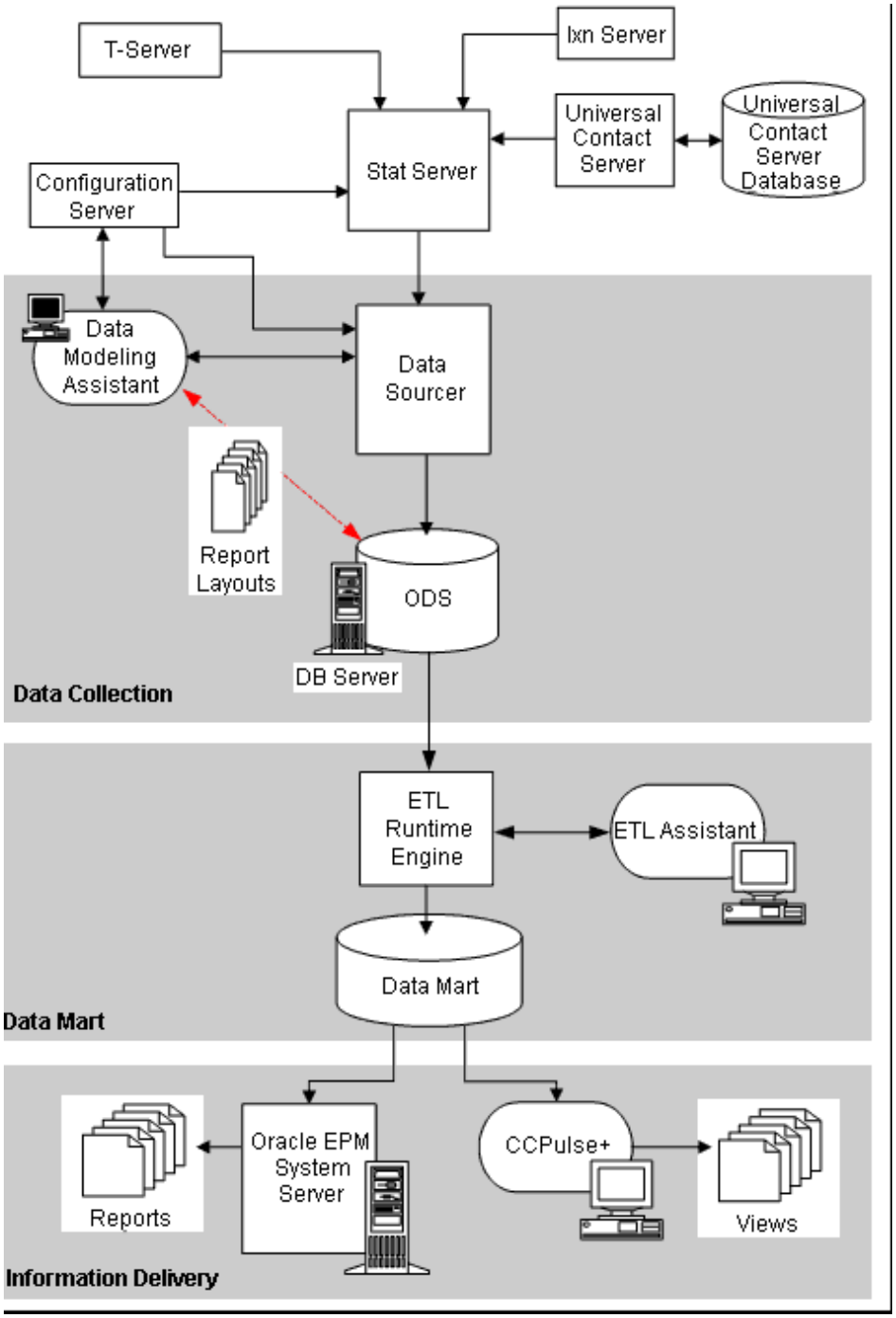

Figure 74: Historical Reporting Architecture

Data Collection Services

The Data Collection Services capture statistics from Stat Server and loads them into ODS for further processing.

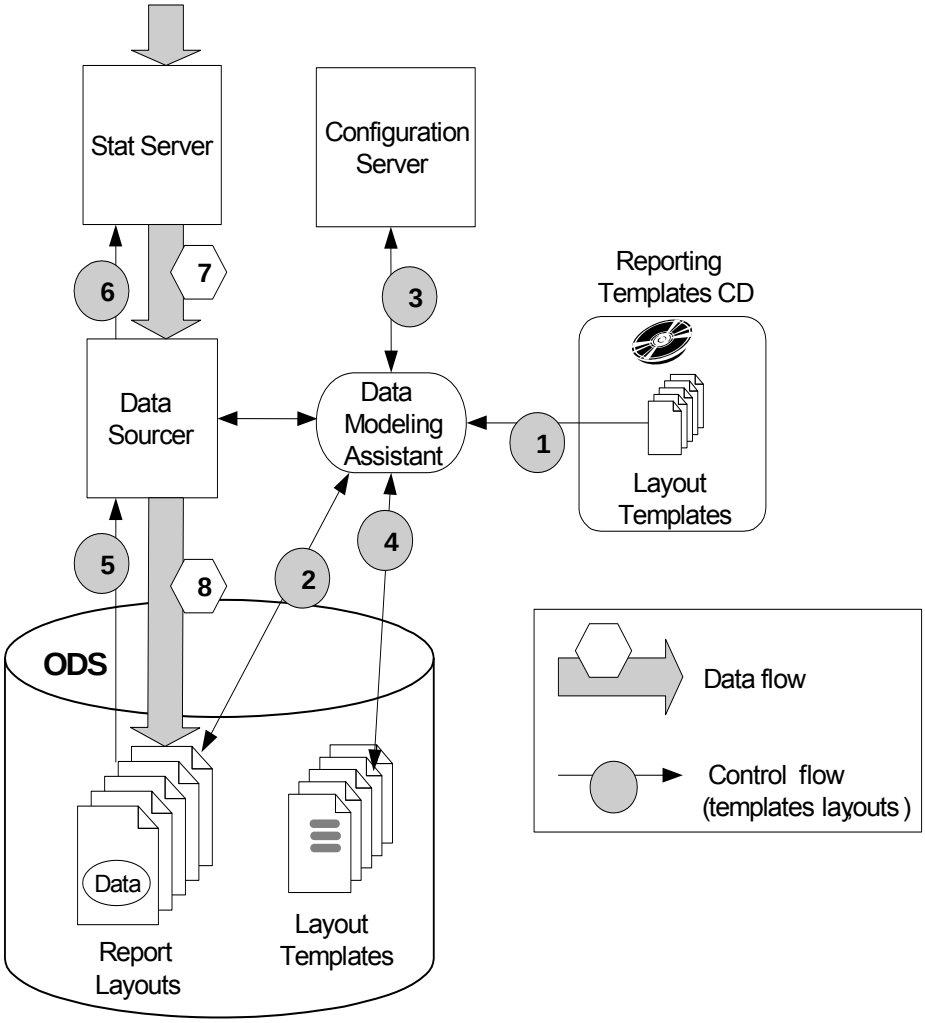

Figure 75: Data Collection Data Flow

The detailed data flow of Data Collection Services shown in Figure 75 depicts a typical scenario using layout templates.

Typically a contact center administrator configures data collection using DMA. Here is the process.

1 The administrator identifies the appropriate predefined solution layout template(s) to use. These layout templates are part of the canned (predefined) templates provided for some Genesys solutions. (“Layout Template Structure” on page 106 discusses layout template structure in detail.)

Note: You can create new layout templates from scratch or based on the provided templates. In addition, you can design custom statistical parameters to use in your templates. “Creating a Layout Template” in the *Customization* book of the *Reporting Technical Reference* series describes how.

2 The administrator imports the chosen templates into ODS using DMA.

3 From the Configuration Server, DMA gathers information about the statistical parameters specified in the imported layout templates and writes this information to ODS.

4 The administrator then creates report layouts based on these layout templates, creating an association between the layout template and the actual object in the contact center. The administrator next activates the report layout.

Note: You also have the option of creating report layouts from scratch, not based on any particular template.

5 Data Sourcer loads the active report layout from ODS to begin data collection.

6 Data Sourcer requests the statistics specified in the report layout from Stat Server.

7 Stat Server sends the requested information to Data Sourcer.

8 Data Sourcer writes this information to ODS into data fields of the corresponding report layout for further processing.

9 Configuration Server dynamically updates stat types, filters, and time ranges as they are changed in Configuration Server.

Layout Template and Report Layout

To generate a report about contact center behavior using the Genesys Historical Reporting tools, an administrator must precisely specify the information needed. As with other documents, you can distinguish the content of a report from its presentation. The content is the information gathered. The report presentation indicates how that content looks on the screen or page.

Administrators must specify both the report’s content and its presentation. They create the presentation either in CCPulse+ views or, in CC Analyzer, with the aid of third-party tools such as Hyperion Interactive Reporting Studio. Presentation options are discussed in the “Information Delivery Services” section on Page 118. To specify report content, the administrator either creates a report layout containing all desired information or uses an existing one. More specifically, the report layout defines which contact center objects and what data about those objects are of interest.

Report layout content includes information about contact center objects, the objects' statistics, time frames in which the statistics should be gathered, and so forth. To simplify report layout creation, Genesys Solution Reporting uses layout templates. Layout templates define the content of a report layout but do not relate to any actual contact center object.

The relationship between a layout template and a report layout is similar the relation between a document template and a document in word-processing applications.

To understand how Genesys Historical Reporting functions, you must be able to distinguish between report layouts and layout templates. These topics are covered in the next subsections.

Note: The discussion in the following sections applies to layout templates for all media types, including custom media types you create using Genesys Open Media. However, you must perform some preliminary configuration before you can create custom media layout templates. Refer to the *Customization* book of the *Reporting Technical Reference* series for more information.

Layout Template Structure

Figure 76 shows the structure of a layout template.

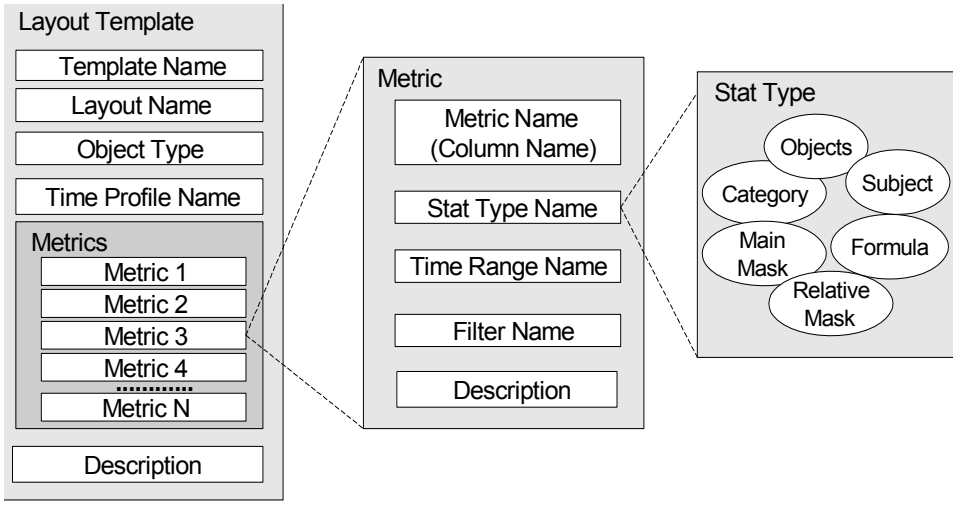

Figure 76: Structure of a Layout Template

Template Name

Each layout template has a unique name (`Template Name` field). The layout template might specify a report layout name (`Layout Name` field) to be used when a report layout is created from the template and activated automatically (see "Report Layout Data" on page 111).

Template Names and Data Mart View Names

Your template name is the source for the Data Mart view name. The view names are created in the format `dimension_tname_agglevel`.

- *Dimension* is one of [`S` | `O` | `T` | `V`] (stat, object, time, or value dimension).
- *Tname* is the name of the layout template as defined in DMA
- *Agglevel* is one of [`DAY` | `HOUR` | `MONTH` | `WEEK` | `QUARTER` |`YEAR` | `NO_AGG`].

The advantages of this naming system are these:

- You can use the same queries and Hyperion report layouts in a multi-tenant environment. Each account uses aliases—or views depending on the database engine—that point to the aliased tables containing report data. As a tenant, you see and use the same table names as other tenants, but the content of generated reports describes your own tenant-specific activity. Tenants can view the report layouts and folders of other tenants but cannot access the tables or the data on which these report layouts and report folders are based.
- You know the table names without having to run ETL Assistant.

Template Object Type

The `Object Type` field indicates the type of contact center object to which you can apply this layout template. For example, you can apply a layout template specifying the Agent object type to an actual Agent object in a contact center.

Template Time Profile

The `Time Profile Name` field defines the time profile to be used for collecting all statistical data.

Layout Template Metric Definitions

The layout template defines a set of metrics to be collected for a specified object. Each metric is described by its own structure.

`Note`: Strictly speaking, in terms of Stat Server, the layout template's metrics are not *metrics*, but rather *statistics*. They become metrics when they are applied to capture behavior of an object within a configured time profile. By design, one, and only one, time profile can be applied to all of the statistics in a layout template.

Each metric is identified by its unique name or its column name. A metric is comprised of a stat type, a time range (if appropriate), a filter (if appropriate), and a description.

Custom media metrics require specially configured stat types, as explained in "Creating Custom Stat Types" in the *Customization* book of the *Reporting Technical Reference* series.

The stat type name indicates the statistical type employed during statistic calculation. Stat types here hold exactly the same principle as they do in Stat Server. Each stat type contains a list of one or more objects, statistical category, subject, one or more main masks, and, optionally, a custom formula.

Figure 77 shows the layout template structure for an Agent object type within DMA. Note the template name—`AGENT`—and its layout name—`Agent Layout`. The time profile associated with the layout template is `CollectorDefault`. The template contains 28 statistics that measure agent activity. For example, the `N_INBOUND` statistic is used to calculate the total number of inbound calls.

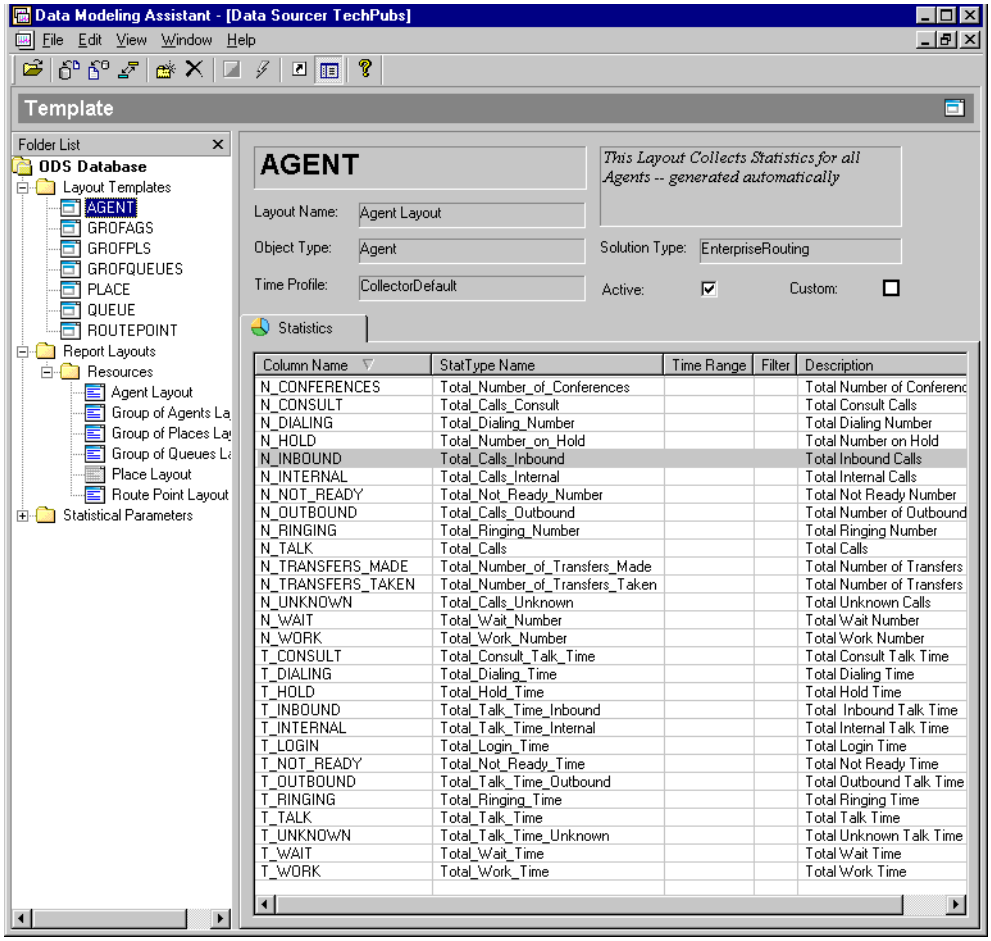

Figure 77: Layout Template Structure, Shown in DMA

If the `Active` check box is selected, the layout template is active. Active layout templates cause Data Sourcer to automatically create and activate report layouts when the new tenant appears in the system.

Under the `Statistics` tab, the layout template displays the definitions of all of its statistics. Each statistic is identified by its column name. The tab also displays the stat type definition each statistic follows and the time range and filter if appropriate. These statistical parameters are retrieved from ODS's statistic parameters section.

Figure 78 shows the stat type definition for the N_INBOUND metric within DMA. Here, when the Total_Calls_Inbound stat type is highlighted in the left pane, the stat type's properties are displayed to the right.

Report Layout Structure

A report layout is a structure that defines report content. It has much in common with layout template structure. The main difference is that the report layout identifies the specific object(s) of a contact center for which Solution Reporting information should be collected and processed.

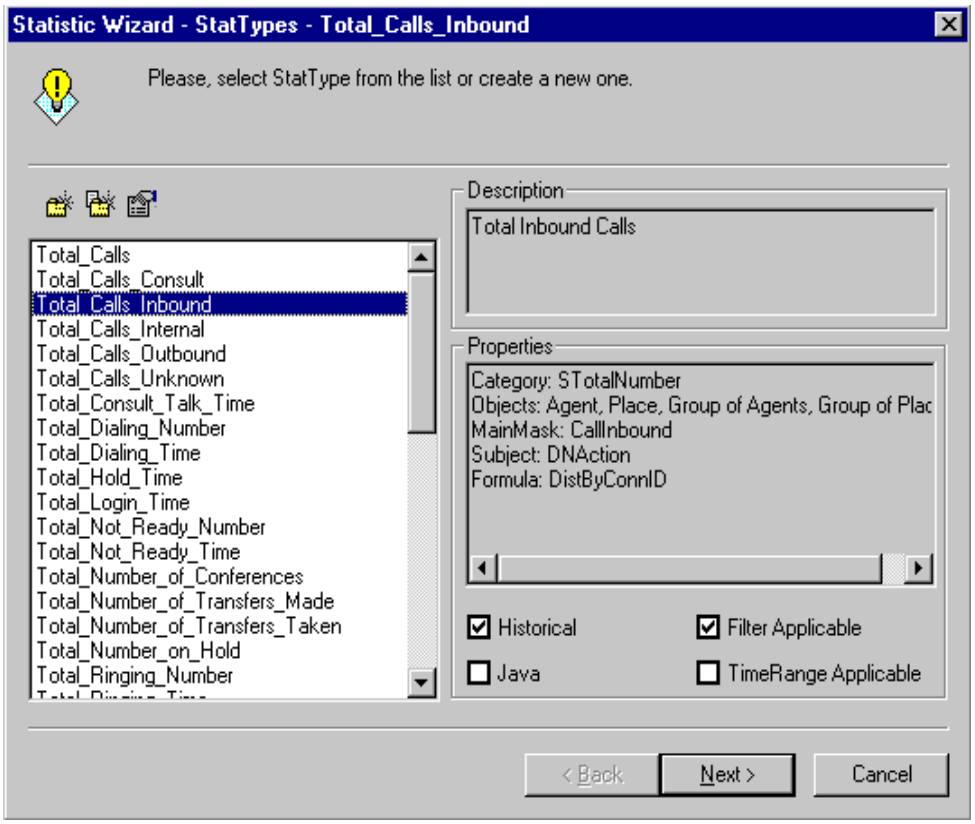

Figure 78: Stat Type Structure (DMA Snapshot)

Report layout structure is illustrated in Figure 79 on page 111. If a report layout is built upon an existing layout template, the report layout inherits the layout template's object type and statistical set. If you build the report layout from scratch, you must specify this information within DMA and associate it with the report layout:

- Layout Name
- Tenant Name
- Metagroup Name
- Template
- Description

In a multitenant environment, the report layout has reference to a specific `Tenant` object in the contact center.

The reference to the object itself is organized with the aid of metagroups. The report layout cannot refer to a specific object itself but rather to its metagroup. For some objects, the metagroup is defined as the group to which they belong. For example, if you are interested in the activities of a particular agent, then you should select one of the agent groups to which the agent belongs. (Remember, the agent might simultaneously belong to several agent groups.) You can find an example of assigning a metagroup to a report layout in the "Creating Custom Report Layouts" chapter of the *Customization* book of the *Reporting Technical Reference* series.

For each object, the default metagroup, `All Objects`, defines all objects of the same type. Therefore, for a particular agent you can to select one of the agent groups to which the agent belongs or the `All Agents` default group.

For those objects having no outer group object, use the all objects group default. For example, the `AgentGroup` object has no outer group object and therefore `All Agent Groups` is used.

The default metagroups are:

- All Agents
- All Agent Groups
- All Calling Lists
- All Campaigns
- All Tenants
- All Place Groups
- All Queue Groups
- All Campaign Groups
- All Campaign Calling Lists
- All Staging Areas
- All Places
- All Queues
- All Regular DNs
- All Routing Points

`Note:` You cannot create a report layout that monitors the performance of any specific agent. To track the agent's performance, select one of the groups to which the agent belongs or select the `All Agents` metagroup. In both cases, you will receive information about all agents of the metagroup. Then at the next stage (Information Delivery Services) you can extract Solution Reporting information about that specific agent.

Use DMA to observe report layout structure. Figure 79, for example, shows the structure of the Group of Agents report layout based on the `AgentGroup` object type.

A report layout contains all template-like information, including its metrics. The layout also has some additional information such as activation and deactivation time.

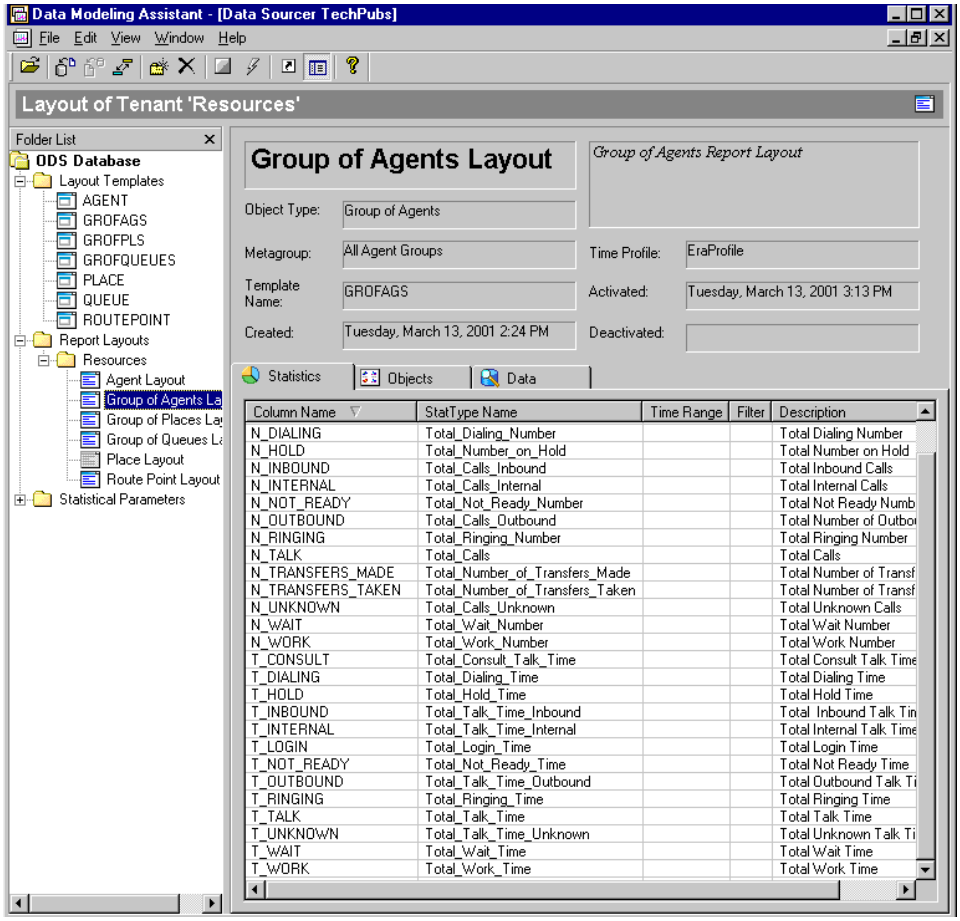

Figure 79: Report Layout Structure, Shown in DMA

Figure 80 shows the relationship between layout templates and report layouts. As you can see, a report layout can refer to only one layout template. At the same time, a layout template might be used by several report layouts. Report layouts need not refer to any layout template (Agent Group Layout 3 in the figure) as they can be created from scratch.

Report Layout Data

When a report layout is activated, Data Sourcer begins collecting Solution Reporting information from Stat Server as specified by this layout. More specifically, Data Sourcer, based on report-layout data, forms requests for needed metrics and then sends the requests to Stat Server. Stat Server calculates the requested data and returns it to Data Sourcer according to the time profile specified by the report layout. Data Sourcer gathers the information into special tables.

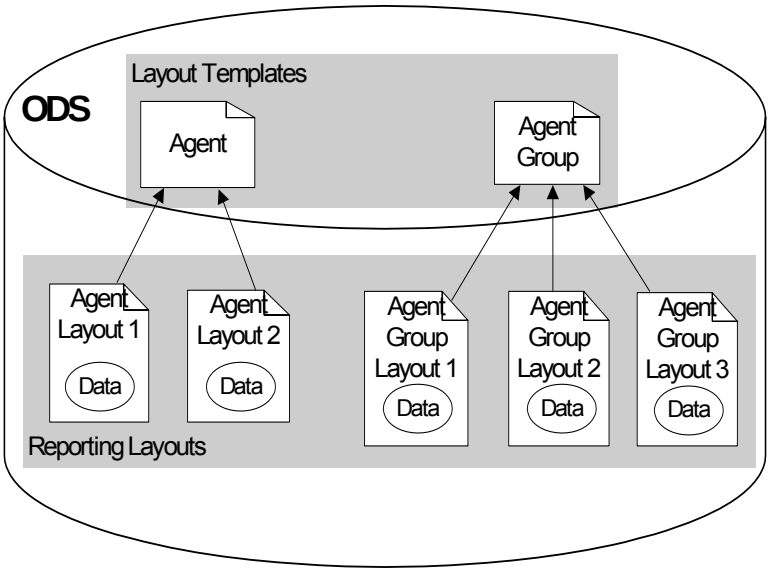

Figure 80: Layouts and Templates

Logically, this data is organized as a three-dimensional array with objects, metrics, and time each claiming a dimension. (Remember, data is collected only from metagroups of objects.) Figure 81 illustrates this concept.

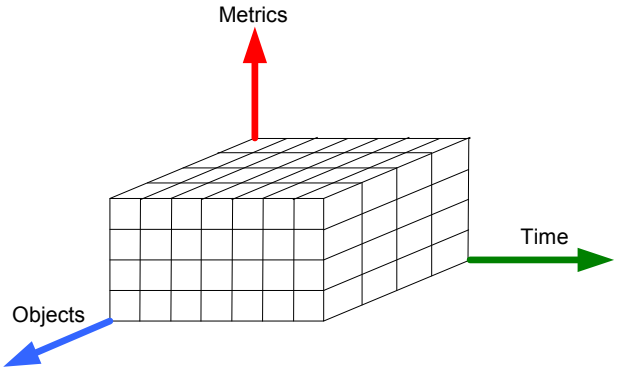

Figure 81: Layout Data Structure

Statistical data is dropped periodically according to layout time profile. For example, the layout might request statistical information every 15 minutes. Statistical data collected and stored for such a period is called a *data chunk.* The new data chunk fills a new layer of the array.

Notice that the data cube might vary in object dimension because objects might appear or leave the contact center at any time during the reporting period. For example, an agent might log in (or log out). This action has a corresponding data stream that is either turned on or off.

The data cube cannot change in the metrics dimension because the set of statistics cannot change (except in custom report layouts).

Using Data Modeling Assistant

Data Modeling Assistant (DMA) is a GUI application that contact center administrators use to define report layouts. These report layouts define the relationships between objects and statistics. This tool also enables administrators to define a *collection time profile*, the interval at which data is drawn from Stat Server.

More specifically, DMA offers the following functionality:

- Import, export, and creation of layout templates
- Import and creation of metrics
- Creation of statistical types
- Creation of custom formulas for statistical categories
- Creation of time profile and time range parameters
- Creation of filters for metrics
- Creation and activation of report layouts based on existing templates or from scratch
- Monitoring of a collection of data in real time

Figure 76 on page 106 shows a sample DMA window.

Collecting Data from Multimedia

Historical Reporting for eServices draws statistical data from Stat Server. Stat Server receives the data, which is stored in the Universal Contact Server database, from Universal Contact Server and Interaction Server. In addition to the objects about which you can also collect telephony data, multimedia data also includes information about Tenant and Staging Area objects.

The Tenant Object

The Tenant object represents the whole contact center and comprises both the e-mail and web media (chat) channels. The information reported about this object includes interactions for each media type; for example, the number of e-mail messages, response times, and so forth.

The Staging Area Object

The Staging Area object represents an interaction queue in which interactions reside during processing. Typical information about the Staging Area object might include e-mail interactions such as the total number of e-mail messages, SMS messages waiting for handling, and so forth.

Data Mart Services

The general configuration of the Data Mart Services and data flow are depicted in Figure 82.

The core component of the Data Mart Services is ETL Runtime—the data-organizing component of Genesys Historical Reporting. ETL Runtime comprises several processes including transformation and loading, aggregation, purging, object tracking, and tenant tracking. ETL Runtime can work with several ODSs simultaneously transferring data from each of them into one Data Mart.

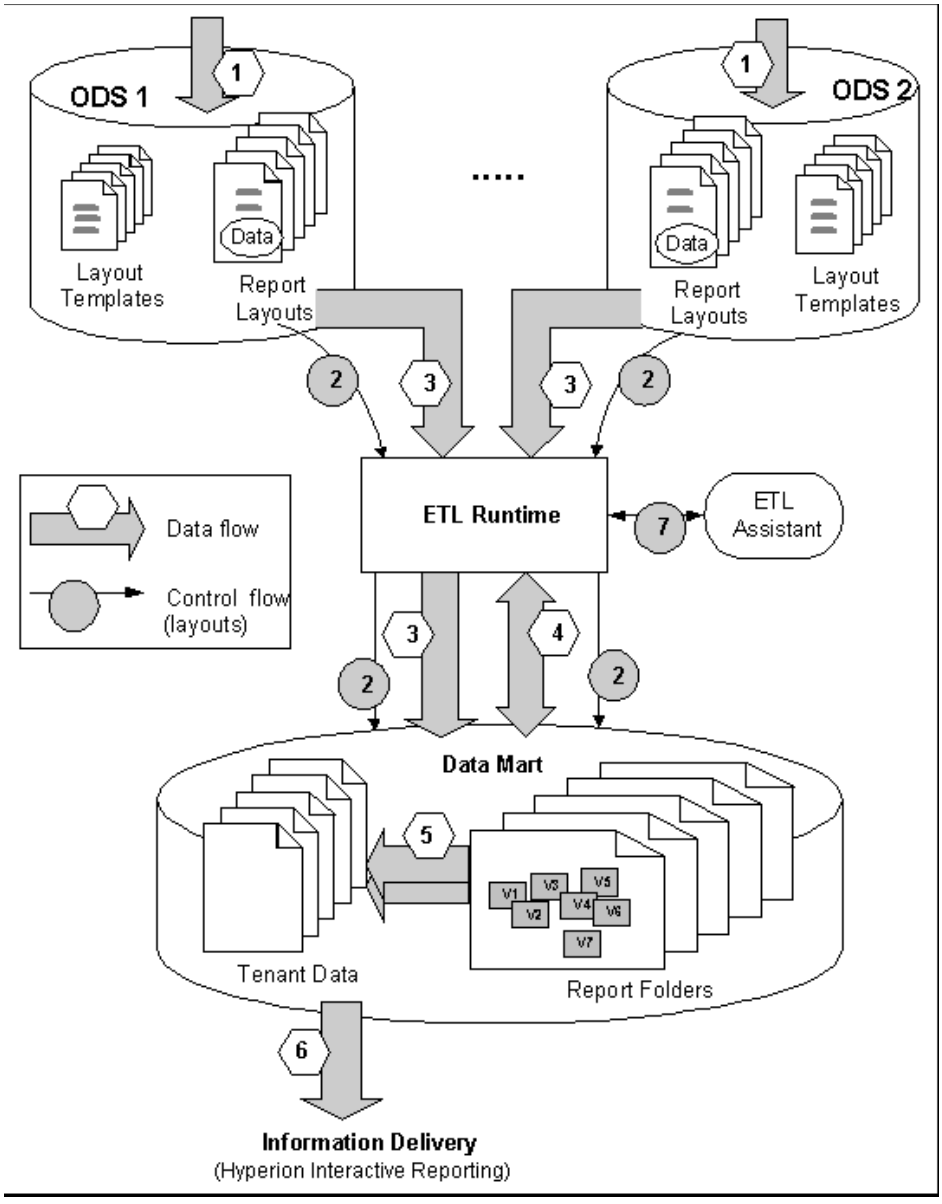

Figure 82: Data Aggregation

The typical data flow scenario illustrated in Figure 82 comprises the following steps:

1. Data Sourcer collects raw statistical data and writes it to ODS.
2. Based on active report layouts that ETL Runtime reads from ODSs, ETL Runtime creates report folders in Data Mart. For each report layout, ETL Assistant creates one report folder. Sometimes this process is referred to as loading metadata.
3. ETL Runtime's data-loading process detects data generated after it performed its last load and writes the new data into the appropriate report folders. Raw statistical data is transformed from its initial format into a format suitable for further processing within the Data Mart without manipulating data content. Data is written to a low-level, 15-minute (by default) view. Once data has transferred successfully, ETL's transformation process deletes the data from ODS (if the `dropTransferredTables` parameter is set).
4. ETL Runtime's aggregation process aggregates the data from 15-minute views to higher aggregation levels. Each new level is written as a view to the report folder. By default, the 15-minute level is aggregated to the hour level. Hour-level data is aggregated to the day level and so forth. The aggregation process might run in parallel with the loading and transformation processes.

   Note: The default time profiles for multimedia data are `hourly` and `daily`.

5. ETL Runtime's tenant-tracking process searches report folders related to the same tenant and consolidates data from them into one place—Tenant data views. Usually according to predefined purging rules, ETL Runtime purges Data Mart of unneeded data and frees up space for further processing.
6. Information Delivery Services can then retrieve the aggregated and tenant-tracked data.
7. The contact center administrator can monitor Data Mart using ETL Assistant.

Data Loading and Transformation

ETL Runtime uses its loading and transformation process to copy data from ODS sources defined using ETL Assistant to the Data Mart.

The ETL Runtime data-loading process detects new data generated after it performed its last load, creates tables associated with the corresponding report folder, and writes the data into this report folder.

Figure 83 illustrates ETL's process of reading and deleting data from ODS. Here, you see portions of data before (left side) and after (right side) the loading and deleting procedure. On the left side you can see three portions (implemented as tables) of statistical data for one object and one metric. The first two portions are completed; the third portion is incomplete. The ETL

Runtime loading process reads the data and deletes only the two first completed portions, leaving the incomplete one in ODS. If the loading and transformation processes are successful, the completed portions are deleted (when the `dropTransferredTables` parameter has been set).

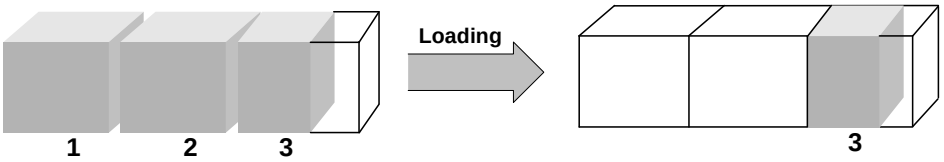

Figure 83: Reading and Deleting Data from ODS

The transformation procedure converts the format of loaded data to a format more suitable for aggregation and retrieval.

Data Aggregation

ETL Runtime's aggregation process derives the data for different aggregation levels. The aggregation levels are presummarized data tables containing data of different aggregation levels. There are seven default aggregation levels:

- no aggregation
- 1 hour
- 1 day
- 1 week
- 1 month
- 1 quarter
- 1 year

Note: Historical Reporting allows partial period aggregations. For example, you can build reports on a partial day. For more detail, refer to the *Reporting 8.0 Reference Manual*.

Figure 84 shows the order in which ETL Runtime aggregates data. Note that day data is used for calculating both month-level and week-level data. Similarly, month-level data is used for building year-level and quarter-level data.

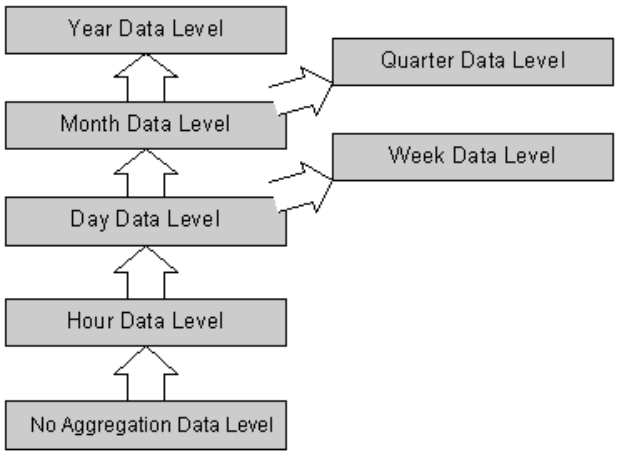

Figure 84: Data Aggregation Levels

Data for each aggregation level is stored in report views within the corresponding report folder. Therefore, by default, a report folder contains seven views corresponding to seven levels of aggregation.

The aggregation procedure uses the corresponding report layout to define the method of calculating aggregated data. Figure 85 illustrates the aggregation of hour-level data from the no-aggregation level to the hour level. The calculation is performed for one object and one metric. The four 15-minute data items shown at the left are `18`, `5`, `10`, and `6`. Suppose that a report layout uses a stat type with the `TotalTime` statistical category. The aggregation procedure sums all four numbers to yield `39`. As a result, for hour-level aggregation, a new data item equal to `39` is created.

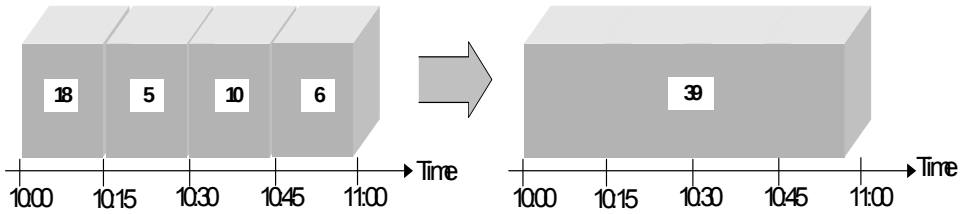

Figure 85: Example of Data Aggregation

If, for example, the stat type uses the `MaxTime` statistical category, then the result yields `18` since `18` is the maximum of the four numbers.

Working with ETL Assistant

ETL Assistant is a GUI tool that enables contact center managers to observe Data Mart configuration information and browse the available aggregation levels. More specifically, ETL Assistant displays the following configuration parameters:

- Information about configuration servers that have been set up for the Data Mart to collect information from. For instance, for each Configuration Server, you can view the server's ID, name, host name, and port number.
- Information about ODS sources such as their name, database information, user information (username and encrypted password), time zone in which data is to be aggregated, and so forth.

Using ETL Assistant, you can see information about all report folders. For each report folder, you can view the information about its Data Sourcer; corresponding report layout including the name of the report layout; tenant name; time profile; objects (metagroups); and its set of metrics.

Using ETL Assistant, you can browse the report views for all data levels. For each report view, you can view information about database tables, purging parameters, time records comprising the report view, and so forth.

Figure 86 shows an ETL Assistant report view of the hour-level aggregation.

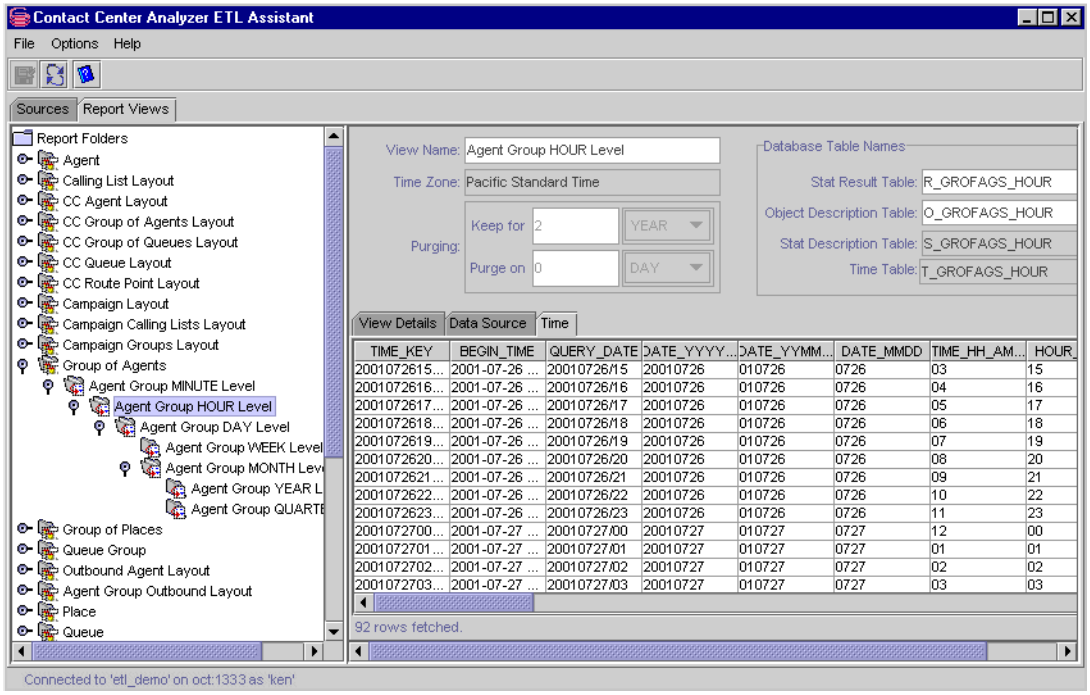

Figure 86: ETL Assistant Report View

Information Delivery Services

Genesys Historical Reporting uses two information delivery methods, CCPulse+ and CC Analyzer, both of which obtain historical data from the Data Collection and Data Mart services.

CCPulse+

With CCPulse+, you can associate real-time and historical statistics and of both create real-time, historical and query based views for most statistics (except `Current Status` statistics, which have no historical counterpart).

> `Note:` For information on generating and customizing historical and query based views in CCPulse+, refer to the *Reporting 8.0 CCPulse+ Help*.

CC Analyzer

CC Analyzer generates only historical reports. For these, Genesys provides Hyperion Intelligence reporting tools to provide report development and distribution functionality. In addition, you can access a set of out-of-box (canned) report templates through the Hyperion Intelligence client reporting tools. These templates are solution-specific for the Genesys Outbound Contact, Genesys eServices, SIP IM, and Universal Routing solutions. The *Solution Reporting Templates* book in the *Reporting Technical Reference* series covers just about every aspect of the solution-provided templates.

Hyperion Interactive Reporting–CC Analyzer Report Generation

The Oracle EPM System (including Hyperion Interactive Reporting) integrates query, analysis, and reporting client tools and an enterprise server solution in a secure, flexible, and scalable environment. The product suite consists of several components, which enable design and delivery of reports via client/server and the Web.

You can find a fully detailed description of Hyperion Interactive Reporting in Oracle EPM documentation. The sections below briefly introduce Hyperion Interactive Reporting and its functionality within Genesys Historical Reporting.

Report Creation and Administration

Hyperion Interactive Reporting (HIR) delivers query, analysis, and reporting capabilities. Users can use predefined data models or create new data models from database tables for their own use and/or further distribution. They can also create custom, Web-enabled dashboards using Web Analysis component. HIR relies on the other parts of Oracle EPM System, such as Oracle EPM System Workspace, to provide centralized solution administration and centralized repository for reports.

Hyperion Interactive Reporting Studio (HIRS) is client-side part of HIR which delivers query, analysis, and reporting capabilities for expert users who need to access data sources directly-or explore the information organized in prebuilt data models taken from the centralized repository (see Figure 87).

Oracle EPM System Server

The Oracle EPM System server part includes Workspace and Interactive Reporting Framework applications that together deliver centralized storage of reports, automated query processing and report distribution via e-mail, the Web, printers, and networks. You can build queries and reports and schedule them for processing based on date, time, or event, such as a database update. You can publish reports-with or without saved result sets-to specified users or groups of users who can then perform additional analysis without ever accessing the database. With Oracle EPM System 11 you need only web-browser like Microsoft Internet Explorer or Mozilla Firefox to discover this content.

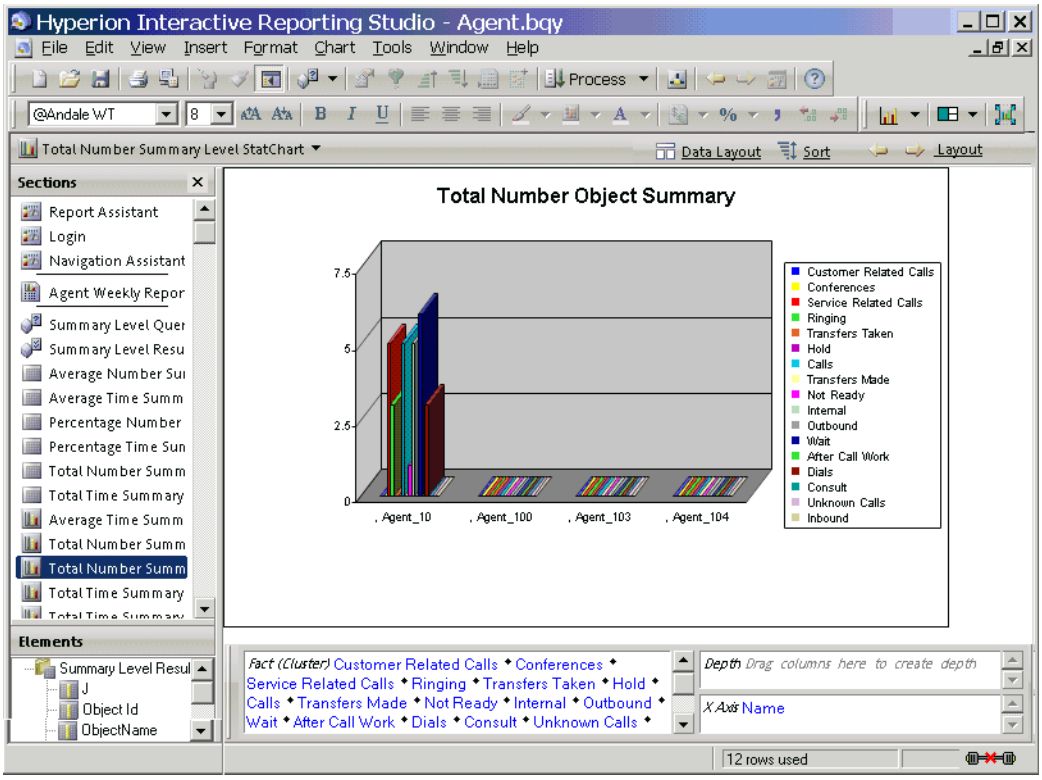

Figure 87: Hyperion Interactive Reporting Studio Interface

Standard Report Repository Reports

The Standard Report Repository includes out-of-box reports for analyzing agent, place, group, queue, and Routing Point status and performance, aggregated over 15-minute, hourly, daily, weekly, monthly, quarterly, and yearly periods.

Agents Status and Performance—Displays the number and type of interactions handled by agents, agent efficiency over time, average number of interactions handled for each communication channel, interaction response time, hold time, after interaction work time, and ratio of customer interactions handled to number of consultations or conference calls.

Queue Management—Indicates the reason interactions failed to connect, network flow for capacity planning, network efficiency over time, downtime trends, and time waiting in queue versus definable thresholds and metrics.

Routing and IVR Performance—Displays calls handled by an IVR versus calls directed to agents via Interaction Router.

Sizing and Scalability

The Data Collection and Data Mart Services make CC Analyzer and CCPulse+ scalable applications for large-volume, multi-site, multi-tenant contact center environments. The Services achieve scalability by relying on industry-standard database management systems and by enabling the distribution of multiple Collection Services. A large contact center might require installation of multiple data collection units. This section discusses how to calculate the size database you require for ODS and Data Mart. It also provides guidelines for distributing data collection processing.

Estimating ODS Size

Genesys recommends reserving enough space in the ODS database for at least two additional days of data collection in the event data is not removed from the ODS as anticipated. An appropriate ODS size depends on the number of requests, time profiles, request record size, and how often the database is cleared.

Use the following formula to estimate minimum ODS size:

ODSSize = *NRequests* x *NTimesPerDay* x *RequestsRecordSize* x *(NDays* + 2)

where:

ODSSize is the size of the Operational Data Storage in bytes.
NRequests is the number of requests made to Stat Server.
NTimesPerDay is the number of Collection Time Profiles per day.
RequestsRecordSize is the request record length in bytes.
NDays is the number of days data is stored in the ODS.

Data Sourcer stores the data it requests from Stat Server in `OL_DATA`*n* tables in ODS. This table's record length depends on your relational database management system and its storage parameters. Table 4 provides record length estimates for the supported RDBMSs. The actual record length in your `OL_DATA`*n* tables might differ.

Table 4: Estimated Length of Request Records by RDBMS

	Microsoft SQL	Oracle	DB2	Sybase
Record length	66 bytes	42 bytes	58 bytes	83 bytes

You can calculate the number of requests made to Stat Server as follows:

$$NRequests = \sum_{i=1}^{NLayouts} NObjects_i \times NStatistics_i$$

where:

NObjects is the number of objects in your report layout.
NStatistics is the number of statistics collected by each report layout.
NLayouts is the number of active report layout in Data Sourcer.

Note: DMA shows the number of requests for all active report layouts in the status bar when the topmost report folder (the default name is `Report Layouts`) is selected in the folder pane. DMA displays the total number of statistics for a particular report layout when that report layout is selected in the folder pane.

Example

Assume the following: 100,000 requests, a 15-minute time profile, an Oracle RDBMS, and ODS is cleared once daily.

- *NRequests* = 100,000
- *NDays* = 1
- *NTimesPerDay* = 4 collections/1 hr x 24 hrs/day = 96 collections/day

An appropriate database size for this scenario is ˜1.2 GB (100,000 x 96 x 42 x [1+2]). And it would be a good idea to factor in some extra space.

Calculating the Number of Requests

Genesys Outbound Contact, Enterprise Routing (Inbound), and Multimedia use the solution-specific layout templates listed in Table 5. You can use these templates as starting points for creating report layouts that measure the status and performance of certain object types. The table also shows the number of statistics collected.

Refer to the “ODS Layout Templates” chapter of the *Reporting Technical Reference Solution Reporting Templates* book in the RTRG series for more information about the statistics gathered.

Table 5: Solution Layout Templates

Outbound Contact Layout Templates		Enterprise Routing Layout Templates		eServices Layout Templates	
Template Name	**Number of Statistics Collected**	**Template Name**	**Number of Statistics Collected**	**Template Name**	**Number of Statistics Collected**
CALL_LS	24	AGENT	28	**ERMS**	
CMP	25	GROFAGS	28	EMAIL_AGENT	11
CMP_CALL_L	24	GROFPLS	28	EMAIL_GAGENT	11
CMP_GR	7	GROQUEUES	11	EMAIL_GPLACE	11
GROFPLS	28	PLACE	28	EMAIL_IQUEUE*	5
GROFQUEUES	11	QUEUE	11	EMAIL_PLACE	11
O_AGENT	32	ROUTEPOINT	11	EMAIL_TENANT*	11
O_AGENT_GR	32			**Web Media**	
PLACE	28			CHAT_A	6**
QUEUE	11			CHAT_GA	6**
ROUTEPOINT	11			CHAT_GH*	7
				CHAT_GP	6**
				CHAT_P	6**
				Voice	
				VOICE_A	22
				VOICE_AG	22
				VOICE_GQ	12
				VOICE_P	22
				VOICE_PG	22
				VOICE_Q	12
				VOICE_RP	12

Note: The layout template names followed by an asterisk (*) contain metrics which are sourced from a Stat Server Java Extension.

The number of statistics for some web media (chat) layout templates is followed by two asterisks (**). This indicates that the template includes additional metrics that are reserved for future use. The number listed in the table represents the number of active statistics.

Use the following formula to calculate the number of requests generated for an ODS containing all seven layout templates for Enterprise Routing:

NRequests = (*NAGENTs* x 28) + (*NGROFAGSs* x 28) + (*NPLACEs* x 28) + (*NGROFPLS* x 28) + (*NQUEUEs* x 11) + (*NROUTEPOINTs* x 11) + (*NGROFQUEUES* x 11)

Example

Consider the following sample environment:

Tenant 1		Tenant 2	
1,000 Agents	5 Queue Groups	2,000 Agents	5 Queue Groups
50 Agent Groups	15-min Time Profile (NTimesPerDay=96)	100 Agent Groups	15-min Time Profile (NTimesPerDay=96)
500 Places	Oracle RDBMS	500 Places	Oracle RDBMS
25 Place Groups	ODS cleared once daily	25 Place Groups	ODS cleared once daily
10 Queues		10 Queues	
20 Routing Points		100 Routing Points	

Using these figures in the equations above, you calculate *NRequests* and *ODSSize* as follows:

NRequests = [(1000 x 28) + (50 x 28) + (500 x 28) + (25 x 28) + (10 x 11) + (20 x 11) + (5 x 11)] +
[(2000 x 28) + (100 x 28) + (500 x 28) + (25 x 28) + (10 x11) + (100 x 11) + (5 x 11)]
= 44,485 + 74,765
= 119,250

ODSSize = 119,250 x 96 x 42 x (1 + 2)
= ~1.4 GB

Estimating Data Mart Size

The appropriate size for Data Mart depends on the number of objects stored, number of statistics gathered, and how long data is kept. This database is much larger than ODS because:

- It maintains a much longer history of contact center statistics; typically, it stores statistics for one year.

- Presummarized data is generated for several aggregation levels to improve reporting performance.

To calculate the Data Mart size, you must find the raw data size and then factor in whatever amount of overhead is appropriate for your enterprise. Steps for calculating the minimum size for the Data Mart appear in the next section.

Note: The size Data Mart you require depends on how much overhead—space required for such things as indexes and metadata—your enterprise chooses to add to the basic raw data size. The overhead size is a highly variable parameter.

As a guideline, note that in addition to storage requirements for raw data, you must also store three default indexes:

- One composite index for the `Fact` table on the `Object` and `Time` foreign keys.
- Two indexes, one each on the primary key indexes for the `Dimension` tables.

These three indexes and the two `Dimension` tables consume approximately one-third again as much space, so the total minimum space required for the Data Mart is calculated as follows:

$$DMSize = RawDataSize \times 1.33$$

Note: If you are using Genesys Info Mart, refer to the "Genesys Info Mart" section of the *Genesys Hardware Sizing Guide* for information on sizing estimates for your Info Mart.

Calculating Data Mart Size

Calculating the raw data size requires that you first calculate the number of aggregations you are using, and then use this figure in the equation for raw data size.

Calculating Number of Aggregations

Each report layout star schema contains two `Dimension` tables (`Object` and `Time`) and one `Fact` table for each aggregation level. `Fact` tables affect database size more than do `Dimension` tables. All `Fact` tables hold the number of aggregation periods maintained for each aggregation level.

For example, at the 15-minute level Data Mart maintains 35,040 aggregation periods for a one-year period (365 days x 24 hours/day x 4 aggregations/hour), while at the one-year level, it maintains just one aggregation period.

The total of the aggregation periods can be represented as follows:

$$TotalAggs = \sum_{i=1}^{NLevels} NAggregations_i$$

Calculate the total number of aggregations for the seven default aggregation levels as follows:

$$TotalAggs = 35040 + 8760 + 365 + 52 + 12 + 4 + 1 = 44234$$

Calculating Raw Data Size

For each report layout schema:

- The number of objects multiplied by the total number of aggregation periods translates into the number of rows.
- The number of statistics translates into the number of columns.

In addition, two keys in each row, the `Object` and `Time` foreign keys, point to the `Dimension` tables. Each statistic and the two keys take 4 bytes of space.

To calculate the total size of the raw data in the Data Mart, sum the star schema sizes for each report layout:

$$RawDataSize = TotalAggs \sum_{j=1}^{NLayouts} NObjects_j \times \langle \langle NStatistics_j \times 4 \rangle + \eta \rangle$$

where η is the size of the row key (size of the `TIME_KEY` and `OBJECT_ID` fields).

Example

To calculate Data Mart size, assume the following:

- The Data Mart is loaded daily.
- You are using the default aggregation levels.
- You are maintaining a one-year history in the Data Mart.

Tenant characteristics are as follows:

Tenant 1	Tenant 2
1,000 agents	2,000 agents
50 agent groups	100 agent groups
500 places	500 places
25 place groups	25 place groups
10 queues	10 queues
20 Routing Points	100 Routing Points
5 queue groups	5 queue groups
Oracle row-key size = 30	Oracle row-key size = 30

As shown above, the equation is as follows:

$$RawDataSize = TotalAggs \sum_{j=1}^{NLayouts} NObjects_j \times \langle \langle NStatistics_j \times 4 \rangle + \eta \rangle$$

You must perform the calculation separately for each layout, using the correct number of objects and number of statistics for each layout. Add these results together to obtain the raw data size.

Total Aggs	= 44234 (See the calculation in "Calculating Number of Aggregations" on page 125.)	
NLayouts	= 7	(Agent, Agent Group, Place, Place Group, Queue, Queue Group, Routing Point)
$NObjects_j$	3000 agents, 150 agent groups, 1000 places, 50 place groups 20 queues, 120 Routing Points, and 10 queue groups.	
$NStatistics_j$	The number of statistics for each layout as shown in Table 5 on page 123.)	
η	= 30	(Row key size)

Using these figures, the raw data size comes to 25.02664458 GB.

Minimum recommended Data Mart size is as follows:

Raw Data Size x 1.33 = 25.02664458 x 1.33 = 33.28543729 GB

Example—Alternative Calculation of Data Mart Size

You can also calculate the minimum Data Mart size as follows:

DMSize = (*NRequests* x *NTimesPerDay* x *NDays* x 8) + 20,000

where:

- *DMSize* is the size of the Data Mart in bytes.
- *NRequests* is the total number of requests from all Data Sourcers connected to the Data Mart.
- *NTimesPerDay* is the number of `Collection Time Profiles` per day.
- *NDays* is the number of days data is stored in the Data Mart.

Using the same number and types of objects as in the previous example, this is calculated as:

DMSize = (119,250 x 96 x 365 x 8) + 20,000 = 33,428,180,000 bytes

To convert the answer to GB, divide by 1073741824. This gives an appropriate database size for this scenario of ~32 GB. And it would be a good idea to factor in some extra space.

Note: *NRequests*= [(1,000 x 28) + (50 x 28) + (500 x 28) + (25 x 28) + (10 x 11) + (20 x 11) + (5 x 11)] +
[(2,000 x 28) + (100 x 28) + (500 x 28) + (25 x 28) + (10 x 11) + (100 x 11) + (5 x 11)]
= 44,485 + 74,765
= 119,250

Distributed Architecture

In estimating database- and hardware-sizing requirements, first determine the implementation architecture for the Data Collection Services. In most cases, a centralized configuration easily accommodates interaction volume. For large volumes—more than 30 interactions per second—Genesys recommends a distributed configuration.

Note: If you are using Genesys eServices/Multimedia, contact Genesys Technical Support for assistance in determining the correct number of multimedia collection units.

Calculating Number of Collection Units

Because Stat Server tracks and maintains statistics in memory, it can only handle a certain number of statistics. This limitation depends on interaction volume, DBMS throughput, CPU speed, and available memory. To scale beyond these limitations, distribute the monitoring and collection of statistics across multiple collection units.

Determining how many collection units to configure requires site-specific information on contact center volume, interaction complexity, and hardware and software environments. In general, configure one collection unit for every contact center or every tenant in a multi-contact center or multi-tenant environment. For a more precise determination of initial distribution, use the following procedure.

Note: The following procedure is only a guideline because accurate scaling of collection units requires ongoing monitoring and tuning.

1. Determine the number of calls per second each T-Server handles in the virtual contact center.
2. Organize the T-Servers into groups whose total volume adds up to approximately 30 contacts per second:

   $$\mathrm{GroupCV} \leq 30$$

3. For each group of T-Servers, calculate the number of requests for all report layouts associated with each T-Server:

   $$\mathrm{NRequests} = \sum_{i=1}^{\mathrm{NLayouts}} \mathrm{NObjects}_i \times \mathrm{NStatistics}_i$$

4. Calculate the number of collection units for each T-Server Group by multiplying its number of requests by its total call volume. Then divide the result by the product of the tested limits for call volume per second and requests for the hardware on which the collection unit will run:

$$NCollectionUnits = \frac{NRequests_T \times GroupCV}{CVMax \times NRequests_H}$$

5. Add the sum of collection units for each T-Server group to get the total number of collection units:

$$TotalCollectionUnits = \sum_{i=1}^{NGroups} NCollectionUnits_i$$

6. In case of fractional results, round up the number of collection units as a cushion for increasing volumes.

Note: The `MaxRequestsPerCollUnit` figure is based on both the performance of Stat Server and Data Sourcer, which, in turn, is based on a number of factors including disk space, and memory to mention a few. Refer to "Stat Server Performance" and "Data Sourcer Performance" in the "Performance Measurements" chapter of the *Reporting 8.0 Reference Manual* for more information.

Example

Adding to the previous example (see "Example" on page 122), Tenant 1 is serviced out of two equal-sized contact centers, each with a T-Server handling contact volume of approximately 10 contacts per second. Tenant 2 is also serviced out of two equal-sized sites, each with a T-Server handling 20 contacts per second. The total contact volume is 60 contacts per second, which would overload a single collection unit.

This example assumes Windows NT Servers with Pentium 400 processors and 256 MB RAM, which tests have shown to handle:

- Approximately 30 contacts per second.
- Approximately 50,000 requests.

These numbers depend heavily on call complexity, which can vary widely. The tests used contacts of average complexity (for example, few transfers, conferences, typical amounts of attached data, and so forth).

Tenant 1 (each Contact Center)

500 Agents	5 Queue Groups
25 Agent Groups	15-min Time Profile
250 Places	70-bit record size
12 Place Groups	2 T-Servers
5 Queues	10 contacts per second
10 Routing Points	

Tenant 2 (each Contact Center)

1,000 Agents	5 Queue Groups
50 Agent Groups	15-min Time Profile
250 Places	70-bit record size
12 Place Groups	2 T-Servers
25 Queues	20 contacts per second
50 Routing Points	

In making the collection unit calculation, you could distribute four collection units, one each to the four sites. However, you can optimize the distribution by following the process just described:

1. The T-Server contact volumes are:
   - T-Server1, 10 contacts per second.
   - T-Server2, 10 contacts per second.
   - T-Server3, 20 contacts per second.
   - T-Server4, 20 contacts per second.
2. You can pair each Tenant 1 site with a Tenant 2 site:
   - T-Server1 + T-Server3, 30 contacts per second
   - T-Server2 + T-Server4, 30 contacts per second
3. Since each of the paired sites has the same characteristics, the number of requests is the same for each:
   - NRequests

     = [(500 x 28) + (25 x 28) + (250 x 28) + (12 x 28) + (5 x 11) + (10 x 11) + (5 x 11)] + [(1000 x 28) + (50 x 28) + (250 x 28) + (12 x 28) + (25 x 11) + (50 x 11) + (5 x 11)]

     = 22,256 + 37,616 = 59,872
4. The number of collection units for each T-Server group is:

   $$\text{NCollectionUnits} = \frac{59872 \times 30}{1500000} = 1.2$$
5. The total number of collection units for the two T-Server groups is:

   $$\text{TotalCollectionUnits} = 1.2 + 1.2 = 2.4$$

If 2.4 is rounded up, you would distribute three collection units. In this case, instead of the two pairs of sites above, you could, alternatively, configure one collection unit for Tenant 1's two sites and one each for Tenant 2's two sites.

Appendix

Acronym List

This appendix provides the meaning of the acronyms used in this document.

Acronym	Meaning
ACD	Automatic Call Distribution
ACW	After-call Work
AHT	Average Handling Time
ANI	Automatic Number Identification
ASA	Average speed of answer
ASAP	As soon as possible
AWT	Actual Waiting Time
ANI	Automatic Number Identification
CCA	Genesys Contact Center Analyzer
CIM	Customer Interaction Management
CPD	Call Progress Detection
CTI	Computer-Telephony Integration
DCID	Distinguish by Connection ID
DMA	Data Modeling Assistant
DN	Directory Number
DNIS	Directory Number Information Service
ERS	Enterprise Routing Solution
ETL	Extraction, Transformation, and Loading

Acronym	Meaning
EWT	Estimated Waiting Time
GIM	Genesys Info Mart
GUI	Graphical User Interface
HIR	Hyperion Interactive Reporting
HIRS	Hyperion Interactive Reporting Studio
ICON	Interaction Concentrator
ICS	Internet Contact Solution
IDB	Interaction Database
IVR	Interactive Voice Response
Ixn	Interaction
OCS	Outbound Contact Solution
ODS	Operational Data Storage
PBX	Private Branch eXchange
PSTN	Public Switch Telephone Network
SDK	Software Developer Kit
SMS	Short Message Service
RGA	Report Generation Assistant
SIP	Session Initiation Protocol
SSJE	Stat Server Java Extension
UCS	Universal Contact Server
UML	Unified Modeling Language
URS	Universal Routing Server
VoIP	Voice over Internet Protocol
WCB	Web Callback

Related Documentation Resources

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

Reporting

- The *Reporting 8.0 Deployment Guide*, which provides step-by-step instructions for configuring and installing the Reporting components.
- The *Reporting 8.0 Reference Manual*, which provides general information about performance measurements, how Reporting behaves during time shifts, and how to set up custom reports for skills-based and partial-period reporting.
- The *Reporting 8.0 CCPulse+ Help*, which contains detailed instructions for using CCPulse+ features and functions.
- The *Reporting 8.0 CCPulse+ Administrator's Guide*, which presents information on customizing and troubleshooting your CCPulse+ application. It also includes tables showing which historical statistics link with which real-time statistics for all statistics included in the solution templates.
- The *Reporting 7.6 Data Sourcer User's Guide*, which describes the role Data Sourcer plays in your Reporting environment and includes the Configuration Server objects Data Sourcer tracks, how it organizes data, and how to fine-tune configuration and troubleshoot problems.
- The *Reporting 7.6 Data Modeling Assistant Help*, which explains how to import and export templates, create new statistical parameters, and create new layout templates and report layouts.
- The *Reporting 7.6 ETL Assistant Help*, which describes how ETL Assistant manages metadata in the Data Mart and allows you to view information about the results of data transformation and aggregation from different sources.

- The *Reporting 7.6 ETL Runtime User's Guide*, which describes the role that ETL Runtime plays in your Reporting environment. It includes a discussion of ETL Runtime's modules, the runtime parameters, options you can set to fine-tune configuration, and how to schedule ETL Runtime processes.

T-Server

- The *Genesys 7 Events and Models Reference Manual* and the *T-Library SDK 7.2 C Developer's Guide*, which provide detailed information on T-Server features and functions.

Framework

- The *Framework 8.0 Stat Server User's Guide*, which describes Stat Server architecture and functions, configuration steps and options, installation procedures, and statistical definitions and formulas.

Genesys

- The *Reporting Technical Reference Guide for the Genesys 7.2 Release* (the predecessor to this document).
- The *Genesys Technical Publications Glossary*, which ships on the Genesys Documentation Library DVD and which provides a comprehensive list of the Genesys and computer-telephony integration (CTI) terminology and acronyms used in this document.
- The *Genesys Migration Guide*, which ships on the Genesys Documentation Library DVD, and which provides documented migration strategies for Genesys product releases. Contact Genesys Technical Support for more information.
- The Release Notes and Product Advisories for this product, which are available on the Genesys Technical Support website at http://genesyslab.com/support.

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

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

Consult these additional resources as necessary:

- The *Genesys Hardware Sizing Guide*, which provides information about Genesys hardware sizing guidelines for the Genesys 8.x releases.

- The *Genesys Interoperability Guide*, which provides information on the compatibility of Genesys products with various Configuration Layer Environments; Interoperability of Reporting Templates and Solutions; and G*plus* Adapters Interoperability.
- The *Genesys Database Sizing Estimator 7.6 Worksheets*, which provides a range of expected database sizes for various Genesys products.

For additional system-wide planning tools and information, see the release-specific listings of System Level Documents on the Genesys Technical Support website, accessible from the system level documents by release tab in the Knowledge Base Browse Documents Section.

Genesys product documentation is available on the:

- Genesys Technical Support website at http://genesyslab.com/support.
- Genesys Documentation Library DVD, which you can order by e-mail from Genesys Order Management at orderman@genesyslab.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. Here is a sample version number:

80rtr_overview_10-2010_v8.0.001.00

You will need this number when you are talking with Genesys Technical Support about this product.

Screen Captures Used in This Document

Screen captures from the product graphical user interface (GUI), as used in this document, might sometimes contain minor spelling, capitalization, or grammatical errors. The text accompanying and explaining 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.

Universal Model Language

Statechart formalism, part of Universal Model Language, graphically represents the state machine in this figure. Statecharts are convenient for representing objects that can be created or destroyed. For instance, a black circle indicates that the object does not yet exist. A black circle enclosed within a hollow circle indicates that the object no longer exists. Rounded rectangles indicate object existence. For a full description of UML, see the current UML specification.

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 Also used to indicate placeholder text within code samples or commands, in the special case where angle brackets are a required part of the syntax (see the note about angle brackets on page 138).	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 . . .
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. Also used for any text that users must manually enter 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]`

Table 6: Type Styles (Continued)

Type Style	Used For	Examples
Angle brackets (< >)	A placeholder for a value that the user must specify. This might be a DN or a port number 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

L

M

N

O

P

Q

R

S

T

U

V

W