



Interaction Concentrator 8.0

User's Guide

The information contained herein is proprietary and confidential and cannot be disclosed or duplicated without the prior written consent of Genesys Telecommunications Laboratories, Inc.

Copyright © 2006–2011 Genesys Telecommunications Laboratories, Inc. All rights reserved.

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 13](#). 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: 80icon_us_05-2011_v8.0.002.00



Table of Contents

List of Procedures	9	
Preface	11	
About Interaction Concentrator.....	11	
Intended Audience.....	12	
Making Comments on This Document	12	
Contacting Genesys Technical Support.....	13	
Documentation Change History.....	13	
New and Updated in Document Version 8.0.002.00	13	
Part 1	About Interaction Concentrator..... 15	
Chapter 1	Product Overview..... 17	
	Basic Architecture..... 17	
	Components and Functions..... 18	
	ICON Server	18
	Interaction Database.....	20
	Supported Features and Functionality.....	21
	New in This Release.....	23
Chapter 2	Introducing IDB Schema..... 25	
	Operational Tables.....	25
	Interaction-Related and Party-Related Tables	26
	Agent State–Related and Login Session–Related Tables	30
	Custom State–Related Tables	31
	Attached Data–Related Tables	31
	Outbound-Related Tables	33
	Virtual Queue Table	33
	Configuration Tables.....	33
	Object Tables	33
	Object Links Tables.....	33
	Data Source Session Control Tables.....	34

	G_DSS_*_PROVIDER Tables	34
	Service and Dictionary Tables	37
	Log Tables	37
Chapter 3	How ICON Works	39
	ICON Processing	39
	Populating Configuration Data	40
	Persistent Cache for Configuration Data	41
	Reading the Configuration Database on Startup	41
	Persistent Cache Is Not Available	41
	ICON Receives Dynamic Notifications	42
	ICON Reads the Configuration History Log	42
	User Request For Resynchronization	43
	Populating Interaction Data	43
	T-Server TEvents	43
	Extracting Interaction Data	44
	Identifying Who Released the Call	45
	Determining Data Availability and Reliability	49
	Determining Data Availability	50
	Determining IDB Availability	51
	Determining Data Reliability	53
Chapter 4	Integrating with Multimedia	55
	Multimedia Objects	56
	Endpoints	56
	DNs	56
	Populating Multimedia Interaction Data	57
	Multimedia Reporting Protocol Events	57
	Multimedia Interactions	57
	Agent Activity	59
	Supported Scenarios	59
	Handling Active Multimedia Interactions	59
	Removing Interactions from Memory	60
	Reconstructing Interaction Data	60
	isOnline Chat Attribute	61
	Processing the isOnline Attribute	61
Chapter 5	Processing Attached Data	63
	Attached Data Specification File	63
	Attached Data Processing for Voice Calls	64
	T-Server Interactions	64

	Call-Specific Data	64
	Historical Data	64
	Post-Routing User Data Processing	65
	Database Schema Extensions for Voice Attached Data	66
	Attached Data Security	67
	Multi-Site and Distributed Environments.....	67
	Customized Attached Data Processing	67
	Custom Dispatchers	68
	Attached Data Processing for Multimedia	68
	Interaction Server Interactions.....	68
	Multimedia Interaction–Specific Data	69
	Database Schema Extensions for Multimedia Attached Data	69
	Tracking User Data Changes.....	69
Chapter 6	Monitoring Virtual Queues and Routing Points.....	71
	Monitoring Route Results on Virtual Queues	71
	Storage Modes	72
	Data Processing Steps	72
	Monitoring Route Results on Routing Points.....	75
Chapter 7	Agent States and Login Sessions.....	79
	Agent and Login Session Models	79
	Agent and Login Session Objects.....	80
	Agent State Model—Voice and Multimedia	81
	Agent State Model—SIP Chat	84
	Login Sessions Model.....	84
	Available Agent State and Login Session Data	85
	Agent State and Login Session Data.....	86
	Precalculated Agent State Metrics.....	86
	After-Call Work and Not-Ready Agent States	87
	Uninterrupted ACW or Not-Ready Duration.....	87
	Interrupted ACW or Not-Ready Duration	87
	Associating ACW with an Interaction	88
	Populating Agent Login Session Data	88
Chapter 8	Integrating with Outbound Contact	89
	Outbound Objects and Models	89
	Outbound Contact Objects	89
	Campaign Model.....	90
	Chain Model.....	91
	Available Outbound Data.....	91

	ICON and OCS Communications	92
	Outbound Objects Data	92
	Precalculated OCS Metrics.....	92
	Outbound Contact Deployment Scenarios	95
	Extracting Outbound Contact Data.....	97
Chapter 9	Processing User Events and Custom-Defined States	99
	Custom States in Interaction Concentrator.....	99
	Storing Data from EventUserEvent	100
Part 2	Implementing High Availability	101
Chapter 10	The Interaction Concentrator HA Model.....	103
	Overview.....	103
	ICON HA Model.....	104
	Sources of Data Interruption	105
	Consequences of Failures.....	106
Chapter 11	Extracting Data in an HA Deployment.....	109
	Extracting HA Data	109
	Scenario: Interrupted Data Flow with Calls.....	110
	Extracting HA Data: Approach 1	114
	Extracting HA Data: Approach 2	114
	Extracting Multimedia Data.....	115
	Extracting Virtual Queue-Specific Data	116
	Extracting Configuration Data.....	116
	Extracting Agent-Specific Data.....	116
Part 3	Administering Interaction Concentrator	119
Chapter 12	Starting and Stopping Interaction Concentrator	121
	Overview.....	121
	Before You Begin.....	122
	Starting the cfg Role for an Oracle IDB	122
	Verifying ICON Connections and Configuration.....	122
	Command-Line Parameters	123
	Starting ICON	123
	Starting ICON with Solution Control Interface	124
	Starting ICON Manually	125
	Starting ICON on Windows	126

	Starting ICON as a Windows Service	127
	Stopping ICON	127
	Stopping ICON with Solution Control Interface.....	128
	Stopping ICON on UNIX	128
	Stopping ICON on Windows	129
	Stopping ICON as a Windows Service	130
Chapter 13	Monitoring Interaction Concentrator	131
	Accessing the Performance Counter.....	131
	Performance Counter Pages	132
Chapter 14	Filtering IDB Data	133
	Overview.....	133
	What Data Can Be Filtered?.....	134
	Party History Data.....	134
	Party Metrics	135
	External Parties	135
	User Data History	135
	Call Metrics	136
	Call History	136
	Interaction Record History	136
	Agent Activity Data	136
	Service Observer Data	138
	Strategy Activity Data	139
Chapter 15	Resynchronizing Configuration Changes.....	141
	How Resynchronization Works.....	142
	How Long Does Resynchronization Take?.....	144
	When to Resynchronize IDB	144
	Setting an Alarm Condition	145
	How to Resynchronize Configuration Data.....	146
	Recommendations for Resynchronization.....	147
	Restoring the Configuration Database from Backup	148
	Modifying the Configuration Database.....	148
Chapter 16	Using Special Stored Procedures.....	151
	Merge Stored Procedure	152
	gsysIRMerge and gsysIRMerge2	152
	Setting Up the Merge Procedure	156
	gsysIRMerge2 Parameters	158
	Executing the Merge Procedure	160

	Purge Stored Procedures	162
	Purge Procedures	163
	Using the gsysPurge80 Stored Procedure	164
	Using Separate Purge Procedures	170
	Stored Procedure gsysInitTimeCode	178
	Setting Up the Time-Setting Procedure	179
	Executing the Time-Setting Procedure	179
	Custom Dispatchers	179
Chapter 17	Troubleshooting ICON Installation and Deployments	181
	Restrictions	181
	Startup Problems	181
	No Connection to the Configuration Server	182
	ICON Exits at Startup	182
	Runtime Problems	183
	No Connection to T-Server or Interaction Server	184
	ICON Does Not Receive Call-Related Events from T-Server	185
	ICON Does Not Write Information to the Database	185
	ICON Has Lost Synchronization with the Configuration Database ...	187
	ICON Does Not Function Correctly	187
	Merge Procedure Problems	187
	Merge Procedure Does Not Complete Successfully	188
	Merge Procedure Does Not Execute	188
	Merge Procedure Performance Is Slow or Unstable	189
Supplements	Related Documentation Resources	191
	Document Conventions	193
Index	195



List of Procedures

Starting ICON with SCI.	124
Starting ICON manually on UNIX.	125
Starting ICON on Windows.	126
Starting ICON as a Windows service	127
Stopping ICON using SCI.	128
Stopping ICON on UNIX from the command line	128
Stopping ICON on UNIX from the console window	129
Stopping ICON on Windows from the console window	129
Stopping ICON running as a Windows service.	130



Preface

Welcome to the *Interaction Concentrator 8.0 User's Guide*. This document provides an overview of the basic Interaction Concentrator architecture, and detailed information about Interaction Concentrator features and functionality. It also provides users with the information they need to know to perform ongoing maintenance and administration of Interaction Concentrator 8.0. This document is valid only for the 8.0 release(s) of this product.

Note: For versions of this document created for other releases of this product, 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 contains the following sections:

- [About Interaction Concentrator, page 11](#)
- [Intended Audience, page 12](#)
- [Making Comments on This Document, page 12](#)
- [Contacting Genesys Technical Support, page 13](#)
- [Documentation Change History, page 13](#)

For information about related resources and about the conventions that are used in this document, see the supplementary material starting on [page 191](#).

About Interaction Concentrator

Interaction Concentrator collects and stores detailed data about the interactions and resources in customer interaction networks that use Genesys Framework (contact center, enterprise-wide, or multi-enterprise telephony and computer networks). Downstream reporting systems can access Interaction Concentrator data in near real time.

Intended Audience

This document is primarily intended for system integrators. It has been written with the assumption that you have a basic understanding of:

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

You should also be familiar with:

- Genesys Framework architecture and functions
- Genesys solutions deployed in your contact center
- Your real-time and historical reporting objectives

Making Comments on This Document

If you especially like or dislike anything about this document, feel free to e-mail your comments to Techpubs.webadmin@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 scope of this document only and to the way in which the information is presented. Contact your Genesys Account Representative or 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) +61-7-3368-6868 (International)	support@genesyslab.com.au
Japan	+81-3-6361-8950	support@genesyslab.co.jp
Before contacting technical support, refer to the <i>Genesys Technical Support Guide</i> for complete contact information and procedures.		

Documentation Change History

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

New and Updated in Document Version 8.0.002.00

- The references to GSYPurge76 have been updated to GSYPurge80. Also, the description of this procedure includes the fact that it now purges Outbound Contact data as well as voice and multimedia interactions, attached data, and agent login session data. For details, see “Purge Stored Procedures” on [page 162](#).
- The explanation of how ICON handles ‘no data’ situations has been expanded and updated. For details, see “Extracting Interaction Data” on [page 44](#).
- An note has been added to the effect that Genesys Info Mart 8.0 does not use the Job_MergeGIM stored procedure. Instead, it merges interactions automatically as a part of the scheduled ETL process (see [page 152](#)).

- Updated the chapter on extracting high-availability (HA) data to indicate that ICON now supports HA for multimedia attached data (see [page 115](#)).



Part

1

About Interaction Concentrator

This part of the document provides detailed information about Interaction Concentrator data, features, and functionality. It also provides a high-level overview of the basic architecture and components of Interaction Concentrator.

This information appears in the following chapters:

- Chapter 1, “Product Overview,” on [page 17](#)
- Chapter 2, “Introducing IDB Schema,” on [page 25](#)
- Chapter 3, “How ICON Works,” on [page 39](#)
- Chapter 4, “Integrating with Multimedia,” on [page 55](#)
- Chapter 5, “Processing Attached Data,” on [page 63](#)
- Chapter 6, “Monitoring Virtual Queues and Routing Points,” on [page 71](#)
- Chapter 7, “Agent States and Login Sessions,” on [page 79](#)
- Chapter 8, “Integrating with Outbound Contact,” on [page 89](#)
- Chapter 9, “Processing User Events and Custom-Defined States,” on [page 99](#)



Chapter

1

Product Overview

This chapter describes the basic architecture and components of Interaction Concentrator. It also provides a high-level overview of Interaction Concentrator functionality. It contains the following sections:

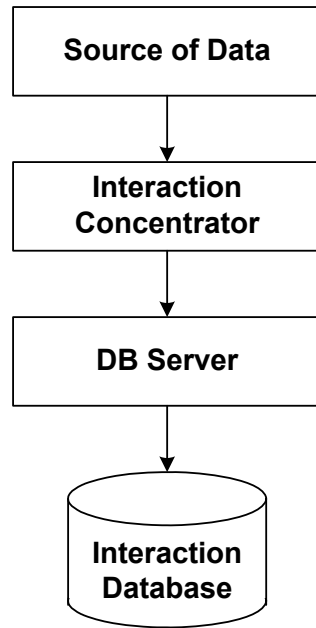
- [Basic Architecture, page 17](#)
- [Components and Functions, page 18](#)
- [Supported Features and Functionality, page 21](#)
- [New in This Release, page 23](#)

Basic Architecture

Interaction Concentrator is a Genesys product that collects and stores detailed data from various sources in a contact center that utilizes Genesys software. Downstream reporting systems can access Interaction Concentrator data in near real time.

Operating on top of Genesys Framework, the Interaction Concentrator product consists of a server application called Interaction Concentrator (ICON) and a database called Interaction Database (IDB). The server receives data from the data sources such as Configuration Server, T-Server, or particular Genesys solutions; it then stores this data into IDB through Genesys DB Server.

Figure 1 on [page 18](#) depicts the basic ICON architecture, omitting most of the Framework components for the sake of simplicity.



Sources of data for Interaction Concentrator 8.0 include:

- Configuration Server
- T-Server or SIP Server
- Interaction Server
- Outbound Contact Server (OCS)

Figure 1: Basic Interaction Concentrator Architecture

Components and Functions

Interaction Concentrator consists of the following elements:

- ICON server
- Interaction Database

The following subsections provide brief descriptions of the Interaction Concentrator components. For more details, see the overview chapter in the *Interaction Concentrator 8.0 Deployment Guide*.

ICON Server

The ICON server:

- Performs preprocessing of events received from Configuration Server, T-Server, Interaction Server, and Outbound Contact Server (OCS), according to the role configured for the ICON instance.
- Prepares the data that will be stored in IDB.
- Writes the prepared data from the in-memory queue to the persistent queue and persistent caches. For more information, see “Persistent Queue and Persistent Caches” on [page 19](#).
- Manages the data in the persistent queue and persistent caches.
- Writes data from the persistent queue into IDB.
- Writes data from the persistent cache for configuration data (`cfg-sync.db`) into IDB.

For detailed information about the configuration options that determine ICON functionality and performance, see the *Interaction Concentrator 8.0 Deployment Guide*.

Persistent Queue and Persistent Caches

The persistent queue is a file that ICON creates and uses to store data before writing it to IDB. The persistent queue also stores information about database-writing requests that are made to IDB. Each ICON instance creates its own persistent queue file (default name `icon_<dbid>.pq`), which stores data for all the roles that are configured for that ICON. Data in the persistent queue survives a shutdown and restart of ICON. Alarm thresholds can be used to monitor ICON performance.

For information about the configuration options that control persistent queue functionality, see the *Interaction Concentrator 8.0 Deployment Guide*.

Persistent Cache for Configuration Data

There is an additional persistent cache (`cfg-sync.db`) for the ICON instance that performs the `cfg` role. This cache plays an important role in maintaining IDB synchronization with the Configuration Database. For more information about the role of the persistent cache in maintaining synchronization, see “Populating Configuration Data” on [page 40](#).

Persistent Cache for Agent Login Session Data

For the ICON instances that perform the `gcc`, `gls`, and `gud` roles, there is an additional persistent cache (default name `apstorage.db`) for agent login session data. ICON uses this cache to prevent stuck login sessions and to prevent duplicate storage of agent login sessions in IDB. For more information, see “Populating Agent Login Session Data” on [page 88](#) and “Extracting Agent-Specific Data” on [page 116](#).

ICON Server Interfaces

The ICON server interfaces with:

- Solution Control Server (SCS) through Local Control Agent (LCA), to control when the ICON server starts and stops.
- Configuration Server, to read Interaction Concentrator application configuration options and other configuration objects and options that affect Interaction Concentrator functionality. (This interface is logically separate from ICON’s connection to Configuration Server as a source of data about contact center resources—see “Supported Features and Functionality” on [page 21](#).)
- Message Server, to log messages to the Central Logger.

Note: Interaction Concentrator does not support the use of the Transport Layer Security (TLS) protocol to secure data exchange between the components with which the ICON server interfaces.

Interaction Database

The Interaction Database stores data about contact center interactions and resources at a granular level of detail. IDB is a database optimized for storage (in other words, primarily for inserting data). Interaction Concentrator itself does not provide a reporting facility. You can use IDB as a consistent and reliable data source for downstream reporting applications.

For a high-level description of the IDB architecture, see “Introducing IDB Schema” on [page 25](#). For a complete table structure and descriptions of all IDB tables and fields, see the *Interaction Concentrator 8.0 Physical Data Model* document for your particular RDBMS.

Stored Procedures

Interaction Concentrator uses a number of stored procedures. Most of these are completely internal to Interaction Concentrator functioning. Therefore, detailed information about them is not relevant for end users.

The following stored procedures are exposed for end-user use and require user input or action:

- Merge**
 - `gsysIRMerge` and `gsysIRMerge2`—The merge procedures that finalize data processing of closed single-site and multi-site interactions.
 - `gsysIRMergeReset`—The procedure that resets the merge procedure to recover from a failed state.
- Purge**
 - `gsysPurgeIR`, `gsysPurgeUDH`, `gsysPurgeLS`, and `gsysPurgeOS`—The procedures that safely purge voice interactions, user data history, agent login session, and Outbound Contact data, respectively, from IDB.
 - `gsysPurge80`—An alternative procedure that safely purges voice, multimedia, and outbound interactions; attached data; and agent login session data from IDB.

Note: Interaction Concentrator release 8.0.000.32 and earlier includes the prior version of the purge procedure, which does not purge Outbound Contact data.

- Time-Setting**
 - `gsysInitTimeCode`—The stored procedure that populates the `G_TIMECODE` table, to enable time-interval reporting.
- Custom Dispatchers**
 - `gudCustDISP1` and `gudCustDISP2`—The stored procedures that customize attached data processing.

For more information about these stored procedures, see “Using Special Stored Procedures” on [page 151](#).

Supported Features and Functionality

In addition to new functionality described in “New in This Release” on [page 23](#), and depending on the role configured for the Interaction Concentrator instance, Interaction Concentrator provides the following features and functionality that support reporting about contact center activities:

- Captures and stores information about the current contact center configuration (objects and associations), and preserves information about deleted configuration objects and terminated associations.
- Captures and stores detailed information about active and completed voice interactions, including switch, DN, time, and routing information about calls and parties. Interaction Concentrator uses a globally unique call identifier. For more information, see Chapter 3 on [page 39](#).
- Captures and stores detailed information about multimedia interactions (e-mail, non-SIP chat, and custom-designed media such as fax and web forms). For more information, see Chapter 4 on [page 55](#).
- Captures and stores detailed information about agent states and login sessions, for agents handling voice as well as multimedia (e-mail, non-SIP chat, other third-party media) and SIP chat interactions. For more information, see Chapter 7 on [page 79](#).

For voice interactions, functionality includes the option to specify which interaction—the first one or the last one—ICON associates with after-call work (ACW), as well as the option to continue or to interrupt the ACW and NotReady agent states while an agent is handling another call. For more information, see “After-Call Work and Not-Ready Agent States” on [page 87](#). See also the information about configuring agent login session metrics in the *Interaction Concentrator 8.0 Deployment Guide*.

- Supports custom agent states, for agents handling voice interactions. For more information, see Chapter 7 on [page 79](#).
- For all types of interactions, captures and stores detailed information about virtual queue usage in interaction processing. For more information, see Chapter 6 on [page 71](#).
- For voice interactions, captures and stores detailed information about Routing Point (RP) usage in call processing.
- Captures and stores detailed information about interactions that are generated in a network-based contact solution environment.
- Captures and stores detailed information about interactions that are generated in a network call parking environment.
- Stores attached data and captures the history of attached data changes for voice interactions as well as eServices/Multimedia (e-mail and chat) and 3rd Party Media interactions. For more information, see Chapter 5 on [page 63](#).

- Supports customized attached data processing for voice calls. For more information, see “Customized Attached Data Processing” on [page 67](#).
- Captures and stores detailed information about outbound campaigns, including:
 - History of campaign processing
 - History of chain processing
 - Precalculated metrics provided by OCS

For more information, see Chapter 8 on [page 89](#).

- Provides a configurable filtering mechanism for certain types of data, to enable the optimization of database size and performance. For more information, see Chapter 14 on [page 133](#).
- Provides the ability to resynchronize the configuration data in IDB with Configuration Database on demand. For more information, see Chapter 15 on [page 141](#).
- Supports high availability (HA) of all types of data through the use of parallel ICON instances, each with its own instance of IDB, in combination with supplementary data that provides information about the availability and reliability of the data stored in IDB. For more information, see Chapter 10 on [page 103](#).
- Supports near–real-time, intraday reporting by writing data to IDB as soon as the data is available (as opposed to after the interaction is completed).
- Provides a sophisticated recognition mechanism, utilizing Inter-Site Call Linkage (IS-Links), to process multi-site interactions. ICON receives information from T-Server regarding the relationship between a given interaction and an interaction at a different site. As a result, complete data is available for reporting across sites. Interaction Concentrator provides a stored procedure to merge the interaction records for multi-site interactions. For more information about the merge procedure, see Chapter 16, “Using Special Stored Procedures,” on [page 151](#).
- Supports multibyte character encoding.
- Stores time information in two formats:
 - Greenwich Mean Time (GMT)—As a `datetime` data type.
 - Coordinated Universal Time (UTC) seconds—As an `integer` data type.

ICON obtains the time information from the timestamps of the data provider events (for example, T-Server TEvents), in the form of UTC seconds.

- Provides mechanisms to purge voice and multimedia interactions, agent login session data, attached data, and OCS data that is stored in IDB. For more information about the purge procedures, see “Purge Stored Procedures” on [page 162](#).

New in This Release

The 8.0 release of Interaction Concentrator includes the following new features or changed functionality that are relevant to the topics discussed in this document:

- Reliably indicates whether the endpoint associated with a party is an IVR device. ICON uses a formerly reserved field in the `G_PARTY_HISTORY` table, `GSYS_EXT_INT1`, to store information that identifies when the party is associated with an IVR device—a value of 1 in this field, in the first historical record related to the party, indicates that the associated endpoint is an IVR.

ICON determines that the party is associated with an IVR device based on the configuration of the DN or IVR port. For more information, see the description of the `ivr` option, in the `gts` section of the DN configuration object, in the *Interaction Concentrator 8.0 Deployment Guide*.

- Properly handles user data that is updated by a routing strategy or agent after the party's association with the interaction has been terminated (for example, the call was transferred). For more information, see “Post-Routing User Data Processing” on [page 65](#).
- Identifies whether the UserEvent that caused a record to be written to a custom states table came from a device at a time when that device was participating in an active call. ICON uses a formerly reserved field, `GSYS_EXT_INT1`, as a flag—a value of 1 in this field in the `G_CUSTOM_DATA_P` or `G_CUSTOM_DATA_S` tables indicates that the UserEvent was generated while the device was participating in an active call.
- If a virtual queue is involved in routing an interaction, stores the database identifier that is assigned to the virtual queue by Configuration Server (the DBID of the virtual queue) in the `G_ROUTE_RESULT` table in IDB, if this data is provided by URS. For more information, see “Virtual Queue DBID” on [page 76](#).
- Enables downstream reporting applications to identify when data was not available and to evaluate the reliability of available data provided by T-Server, Interaction Server, Outbound Contact Server, and Configuration Server.

This functionality:

- Provides a mechanism for downstream reporting applications, such as Genesys Info Mart, to support HA for all types of data, including multimedia and Outbound Contact data, when one of a pair of ICON instances fails.
- Enhances support for HA by enabling downstream reporting applications to detect in advance when to switch over to another IDB for data extraction.

- Improves data integrity and ICON functioning by enabling downstream users to identify the periods when configuration data was not captured or stored and, therefore, to take the necessary action to resynchronize configuration data in a timely manner.

To enable this functionality, new tables in IDB, one for each type of data provider (as specified by the ICON role), store detailed data about connections, timestamps, and events from the specified data sources. For more information about the new tables, see the *Interaction Concentrator 8.0 Physical Data Model* document for your particular RDBMS.

For more information about using this new functionality, see “Determining Data Availability and Reliability” on [page 49](#) and “Extracting HA Data” on [page 109](#).

- Provides sufficient information at the call and party levels for downstream reporting applications to determine which party released the call.

To support this functionality, ICON uses the GSYS_EXT_VCH1 and GSYS_EXT_VCH2 fields in the G_CALL_STAT table and the GSYS_EXT_INT1 field in the G_PARTY_STAT table to store information, if it is reported by T-Server. A new configuration option, store-releasing-party, enables this functionality.

This feature requires T-Server release 8.0 or higher. Interaction Concentrator 8.0 supports this feature for the Alcatel A4400/OXE switch. Interaction Concentrator 8.0.000.35 and higher also supports this feature for Avaya switches; for Avaya, this functionality requires T-Server for Avaya Communication Manager release 8.0.101.05 or higher.

For more information about using the new functionality, see “Identifying Who Released the Call” on [page 45](#). See also the information about the G_CALL_STAT table and the G_PARTY_STAT table in the *Interaction Concentrator 8.0 Physical Data Model* for your particular RDBMS.

- Improves merge functionality.
 - Implements database locking in a way that reduces the potential for problems to occur during execution of the merge procedure.
 - Introduces a new stored procedure, gsysIRMergeReset, that simplifies the steps to reset the merge procedure if required. For more information about using the gsysIRMergeReset procedure, see “Resetting the Merge Procedure” on [page 162](#).
- Streamlines future migration by packaging the stored procedures that support each 8.x release in a schema-specific set. A particular ICON release works only with the corresponding stored procedures package. Multiple sets of packages can exist in the same IDB, and an earlier set of stored procedures can work with a later version of the IDB schema. For more information, see the Interaction Concentrator section of the *Genesys Migration Guide*.



Chapter

2

Introducing IDB Schema

The Interaction Database (IDB) stores in its tables all reporting data that the Interaction Concentrator server (ICON) provides. Logically, IDB tables can be divided into groups that correspond to the classes of information that ICON writes.

This chapter describes the logical groups of tables. This chapter also provides a summary of the precalculated call and party metrics that are available in the statistical tables. It contains the following sections:

- [Operational Tables, page 25](#)
- [Configuration Tables, page 33](#)
- [Data Source Session Control Tables, page 34](#)
- [Service and Dictionary Tables, page 37](#)
- [Log Tables, page 37](#)

Notes: For a complete table structure and description of all IDB tables and fields, see the *Interaction Concentrator 8.0 Physical Data Model* document for your particular RDBMS.

The data filtering feature limits the data stored to IDB. See “Filtering IDB Data” on [page 133](#) for more information.

Operational Tables

There are six subgroups of tables that contain operational data:

- Interaction–related and Party–related tables (see [page 26](#))
- Agent State–related and Login Session–related tables (see [page 30](#))
- Custom State–related tables (see [page 31](#))
- Attached Data–related tables (see [page 31](#))

- Outbound-related tables (see [page 33](#))
- Virtual Queue-related table (see [page 33](#))

These groups of tables are described in detail in the following subsections.

Interaction-Related and Party-Related Tables

There are three subgroups of tables that contain data about interactions and parties:

- Core tables
- History tables
- Statistical tables

The following subsections describe each of these subgroups in turn.

Core Tables

The core tables contain current (the most up-to-date) information about interactions and parties, as well as all related information.

The five tables in this subgroup are:

- G_CALL—Contains information about telephone calls (both completed and active), SIP chat, eServices/Multimedia, and 3rd Party Media interactions.
- G_PARTY—Contains information about parties to the interaction.
- G_IR—Contains information about the data that is common to all of the interactions in a particular scenario.
- G_IS_LINK—Contains information about inter-site interaction links in a multi-site scenario.
- G_ROUTE_RESULT—Contains information about the results of routing for the interaction.

History Tables

The history tables contain intermediary (history) states of the data that was previously stored in the core tables. The history tables are named G_CALL_HISTORY, G_PARTY_HISTORY, G_IR_HISTORY, and G_IS_LINK_HISTORY. They correspond to the G_CALL, G_PARTY, G_IR, and G_IS_LINK core tables, respectively.

Statistical Tables

The two statistical tables, G_CALL_STAT and G_PARTY_STAT, contain statistical information about calls and parties, respectively.

Interaction Metrics Table 1 lists the Interaction Concentrator metrics that are stored in the G_CALL_STAT table, in the order of their appearance in that table. Table 1 also indicates the media type for which the metric is calculated.

Table 1: Available Interaction Metrics, by Media Type

Metric Name	Description	Media Type Supported			
		Voice	eServices/Multimedia		
			E-Mail	Chat	3rd Party Media
F_CONN	Flag Connected.	Yes	Yes	Yes	Yes
F_CONN_EXTN	Flag Connected on Extension.	Yes	Yes	Yes	Yes
F_TE_ABND	Flag Event Abandoned.	Yes	No	No	Depends on the media type and media server
CNT_HOLD	Number of times that the call was placed on hold.	Yes	No	No	No
CNT_DIVERT	Number of times that the interaction was diverted (for example, from a Queue or Routing Point).	Yes	Yes	Yes	Yes
CNT_TRANSFER	Number of times that the interaction was transferred, given that the transfer was completed.	Yes	Yes	Yes	Yes
CNT_TRANSFER_LGIN	Number of times that the interaction was transferred by a party associated with a device that had a logged-in agent.	Yes	Yes	Yes	Yes
CNT_CONFERENCE	Number of times that the interaction was conferenced.	Yes	No	Yes	No
T_DURATION	Duration of the interaction.	Yes	Yes	Yes	Yes
T_CONN	Time until Connected with the first party in the interaction. This metric's value can be considered the time to answer on any device (either by an agent or IVR).	Yes	Yes	Yes	Yes

Table 1: Available Interaction Metrics, by Media Type (Continued)

Metric Name	Description	Media Type Supported			
		Voice	eServices/Multimedia		
			E-Mail	Chat	3rd Party Media
T_CONN_EXTN	Time until Connected on Extension.	Yes	Yes	Yes	Yes
T_TE_ABND	Time until Abandoned.	Yes	No	Yes	Yes
TT_ALERTING	The sum of all the time interval durations if there was at least one internal party in a call in the ALERTING state.	Yes	Yes	Yes	Yes
TT_CONNECTED	The sum of all the time interval durations when all parties in a call were in a CONNECTED state.	Yes	Yes	Yes	Yes
TT_HOLD	The sum of all the time interval durations when there was at least one internal party in a call in the HOLD state.	Yes	No	No	No
TT_QUEUED	The sum of all the time interval durations when there was at least one internal party in a call in the QUEUED state.	Yes	Yes	Yes	Yes
CM_EXT_1– CM_EXT_10	Reserved for custom metrics calculation. You can perform calculations of custom metrics by using custom stored procedures.	Yes	Yes	Yes	Yes

Party Metrics Table 2 lists the Interaction Concentrator metrics that are stored in the G_PARTY_STAT table, in the order of their appearance in that table. Table 2 also indicates the media type for which the metric is calculated.

Table 2: Available Party Metrics, by Media Type

Metric Name	Description	Media Type Supported			
		Voice	eServices/Multimedia		
			E-Mail	Chat	3rd Party Media
TT_ALERTING	Total time that a party spent in the ALERTING state.	Yes	Yes	Yes	Yes
TT_CONNECTED	Total time that a party spent in the CONNECTED state. The states of other parties in the call do not affect this metric.	Yes	Yes	Yes	Yes
TT_HOLD	Total time that a party spent in the HOLD state.	Yes	No	No	No
TT_QUEUED	Total time that a party spent in the QUEUED state.	Yes	Yes	Yes	Yes
TT_ACW	Total time that a party spent in after-call work (ACW).	Yes	No	No	No
CNT_ALERTING	Number of times that a party changed its state to Alerting.	Yes	Yes	Yes	Yes
CNT_CONNECTED	Number of times that a party changed its state to Connected (for example, from the Alerting or Hold state).	Yes	Yes	Yes	Yes
CNT_HOLD	Number of times that a party changed its state to Hold.	Yes	No	No	No
CNT_QUEUED	Number of times that a party changed its state to Queued.	Yes	Yes	Yes	Yes
CNT_ACW	Flag of ACW presence for this party.	Yes	No	No	No
TT_ON_ALERT	Total time that another party in the interaction spent in the ALERTING state.	Yes	Yes	Yes	Yes
TT_ON_HOLD	Total time that another party in the call spent in the HOLD state.	Yes	No	No	No

Table 2: Available Party Metrics, by Media Type (Continued)

Metric Name	Description	Media Type Supported			
		Voice	eServices/Multimedia		
			E-Mail	Chat	3rd Party Media
TT_ON_QUEUE	Total time that another party in the interaction spent in the QUEUED state.	Yes	Yes	Yes	Yes
TT_ON_CONNECTED	Total time that all parties in the call were simultaneously in the CONNECTED state.	Yes	Yes	Yes	Yes
T_DURATION	Duration of a party's existence.	Yes	Yes	Yes	Yes
PM_EXT_1– PM_EXT_10	Reserved for custom metrics calculation. You can perform calculations of custom metrics by using custom stored procedures.	Yes	Yes	Yes	Yes

Agent State–Related and Login Session–Related Tables

There are three subgroups of tables that contain historical data about contact center agents.

- Core tables
- History tables
- Statistical tables

The following subsections describe each of these subgroups in turn.

Core Tables

The core tables contain information about agent states, login sessions, and the association of sessions with endpoints (DNs).

The tables in this subgroup are:

- G_LOGIN_SESSION—Contains information about agent login sessions.
- GX_SESSION_ENDPOINT—Contains information about the association between a login session and an endpoint (DN).
- G_AGENT_STATE_RC—Contains information about changed or terminated reason codes for agent states.
- G_AGENT_STATE_RC_A—Contains information about active reason codes for agent states.

History Tables

The history tables contain historical data relating to agent states and login sessions at a DN.

The two tables in this subgroup are:

- **G_AGENT_STATE_HISTORY**—Stores the history of agent states within a given login session.
- **G_DND_HISTORY**—Stores the history of DND (Do Not Disturb) feature activation and deactivation on a device (DN).

Statistical Tables

The two statistical tables contain several statistics related to agent states and login sessions.

The two tables in this subgroup are:

- **GS_AGENT_STAT**—Stores durations of agent states.
- **GS_AGENT_STAT_WM**—Stores durations of work modes for agent states.

Custom State–Related Tables

Interaction Concentrator supports customer-defined states and attached data in UserEvents for compatibility with Call Concentrator.

The three tables related to custom states are:

- **G_CUSTOM_DATA_P**—This is a flat table with key values delivered in UserEvents associated with voice calls.
- **G_CUSTOM_DATA_S**—This is a generic table. Information about key values are delivered in UserEvents associated with voice calls.
- **G_CUSTOM_STATES**—Stores detailed information about an agent's state changes during the agent login session.

Attached Data–Related Tables

The tables related to attached data (also sometimes referred to as *UserData*) contain data that T-Server and, if applicable, Interaction Server attach to an interaction. ICON selects the data from TEvents or Multimedia reporting events. The exact data that ICON selects depends on the way in which ICON has been configured. The data in these tables can include the current state of attached data or the history of attached data changes, if so configured.

There are three subgroups of attached data–related tables:

- Attached Data State tables for voice (*flat* tables)
- Attached Data History tables for voice (*generic* tables)
- Multimedia Attached Data tables

The following subsections describe each of these subgroups in turn. For more information, see Chapter 5, “Processing Attached Data,” on [page 63](#).

Attached Data State Tables

The Attached Data State (flat) tables store the current (latest received) state of the attached data attributes that are associated with calls.

The four tables in this subgroup are:

- G_CALL_USERDATA—Stores predefined attached data attributes.
- G_CALL_USERDATA_CUST—Stores custom attached data attributes.
- G_CALL_USERDATA_CUST1—Stores custom attached data attributes.
- G_CALL_USERDATA_CUST2—Stores custom attached data attributes.

For information about multimedia-specific attached data, see “[Multimedia Attached Data Tables](#)” below.

Attached Data History Tables

The Attached Data History (generic) tables store historical information about attached data for voice and multimedia interactions.

The two tables in this subgroup are:

- G_USERDATA_HISTORY—Stores information about the attached data fields that require no security protection.
- G_SECURE_USERDATA_HISTORY—Stores information about the sensitive attached data fields that do require security protection (for example, a customer’s Social Security number).

Multimedia Attached Data Tables

The Multimedia Attached Data tables store information about multimedia-specific attached data.

The two tables in this subgroup are:

- GM_L_USERDATA—Stores the values of attached data keys for suggested and auto responses and acknowledgements, customer IDs, and reasons for stopping processing.
- GM_F_USERDATA—Stores metadata information about e-mail and chat interactions (for example, the sender’s name and e-mail address, the subject, and the type).

For more information about how ICON populates these tables, see “Database Schema Extensions for Multimedia Attached Data” on [page 69](#).

Outbound-Related Tables

The outbound-related tables include either the `GO_` or `GOX_` prefix in their names. These tables store information about the entities and attributes that are related to the processing of outbound calls and campaigns as reported by Outbound Contact Server (OCS).

For more detailed information about stored outbound data, see “Available Outbound Data” on [page 91](#).

Virtual Queue Table

The table that stores information related to interaction processing at virtual queues is called `G_VIRTUAL_QUEUE`. It stores the history of associations between interactions and virtual queues, as reported by T-Server, provided that Universal Routing Server (URS) provides this information to T-Server.

For more detailed information about stored virtual queue data, see “Monitoring Virtual Queues and Routing Points” on [page 71](#).

Configuration Tables

There are two subgroups of tables that contain configuration data:

- Object tables
- Object links tables

The following subsections describe each of these subgroups in turn.

Object Tables

The object tables include the `GC_` prefix in their names. These tables contain current (the most up-to-date) information about the configuration objects that ICON tracks. These tables also preserve information about the configuration objects that have been deleted from the Configuration Database.

Object Links Tables

The object links tables include the `GCX_` prefix in their names. These tables contain information about the *associations* (links) between configuration objects in the Configuration Database. Examples of associations between configuration objects include assignments of `Skills` or `Logins` to `Agents` and assignments of `Agents` to `Agent Groups`. These tables also preserve information about terminated associations—for example, information about the fact that an `Agent` was removed from an `Agent Group`.

Data Source Session Control Tables

Provider The Interaction Concentrator 8.0 IDB schema introduces five control tables, one for each ICON provider. *Provider* refers to the ICON functionality that provides data for a particular ICON database schema. The provider functionality corresponds to the ICON role. Each provider can be considered as an independent ICON process, with its own set of IDB tables.

G_DSS_*_PROVIDER Tables

The provider control tables are:

- **G_DSS_CFG_PROVIDER**—The control table for the `cfg` role, which stores configuration-related information. The data source is Configuration Server.
- **G_DSS_GCC_PROVIDER**—The control table for the `gcc` role, which stores interaction-related and party-related information. The data sources are T-Server and Interaction Server.
- **G_DSS_GLS_PROVIDER**—The control table for the `gls` role, which stores data that pertains to agent states and agent login sessions. The data sources are T-Server and Interaction Server.
- **G_DSS_GOS_PROVIDER**—The control table for the `gos` role, which stores data that pertains to outbound calls and campaigns. The data source is Outbound Contact Server (OCS).
- **G_DSS_GUD_PROVIDER**—The control table for the `gud` role, which stores data that pertains to attached data associated with interactions. The data sources are T-Server and Interaction Server.

ICON populates a particular provider table only if the corresponding role option has been defined for the `ICON Application`. For more information about the role option, see the chapter about configuration options in the *Interaction Concentrator 8.0 Deployment Guide*.

For each `ICON Application` instance that performs a particular provider role, the provider table stores information about:

- | | |
|------------------------|--|
| Data Sources | <ul style="list-style-type: none"> • The identity of the applications that are data sources for that particular provider—for example, the identifier that Configuration Server assigns to the primary and backup data source applications (such as <code>DS_DBID</code> and <code>DS_DBID_PRIM</code>). |
| Connection | <ul style="list-style-type: none"> • The connection between the data source and ICON—for example, the timestamp when the connection was established, and the timestamp when a disconnection was detected. |
| Connection Type | <p>An important connection-related field, <code>DSCONN_TYPE</code>, describes the type of connection. This indicates the reason that a new record was created in the provider table:</p> |

- 1—Indicates that this is the first connection following a restart of the ICON application.
 - 2—Indicates that this is a reconnection to the data source.
 - 3—Indicates that this is a reconnection to the data source following a detected restart of the data source application (information about the restart was received).
 - 4—Indicates that a reconnection of the data source to the switch was detected.
 - 5—Indicates that a switchover between primary and backup data sources was detected.
- ICON Application**
- The identity and status of the ICON application—for example, the DBID of the ICON application (ICON_DBID), and the application startup or shutdown time.
- Data Source Events**
- Events from the data source—for example, the timestamp of the first and last saved events.

For full details about the fields in the G_DSS_*_PROVIDER tables, see the *Interaction Concentrator 8.0 Physical Data Model* document for your particular RDBMS.

Populating the Provider Control Tables

On startup (start or restart), each ICON reads its configuration (connection list) from Configuration Server, and determines the DBIDs of all the applications connected with that ICON. However, simply establishing a connection between ICON and a data source does not trigger the creation of a record in any provider table.

- The first record in a provider table is created the first time an ICON instance processes an event from an identified connection, and prepares data for storage in IDB for that provider.
- Thereafter, every time a transaction for that provider is being written, the record for the connection is updated to include data pertaining to the last stored event on the connection. The provider table record is updated before the new transaction is committed.

In some cases, ICON updates particular records in active provider control tables when it detects disconnection or shutdown of a data source application that is connected with that ICON.

- Data Source Session ID**
- A data source session ID (DSS_ID) uniquely identifies the connection between the ICON application, the data source application (for example, T-Server), and the switch, as well as the timeframe during which the connection was active. The DSS_ID is obtained from the value of a system field (GSYS_DOMAIN) in all operational tables, which identifies the session that was active when the data was processed by ICON.

If any of the applications that are part of the connection restarts, the value of the `DSS_ID` changes, and this triggers the creation of a new record in the provider table.

Limitations

- Interaction Concentrator release 8.0 does not support dynamic changes to HA configuration. In deployments in which the data source for the ICON instance is an HA pair, ICON might incorrectly identify the data source in the provider table record if the relationship between the primary and backup data sources is changed in the Configuration Layer while ICON is running.
- When ICON restarts after a shutdown, it does not store data about the previous data source session in the applicable provider control table under the following circumstances:
 - For configuration data (the `G_DSS_CFG_PROVIDER` table), if the `cfg-sync.db` file was deleted or the `cfg` role was disabled before ICON restarted.
 - For OCS data (the `G_DSS_GOS_PROVIDER` table), if the persistent queue (`.pq`) file was deleted or the `gos` role was disabled before ICON restarted.
 - For interaction-related, agent-related, and attached data-related data (the `G_DSS_GCC_PROVIDER`, `G_DSS_GLS_PROVIDER` and `G_DSS_GUD_PROVIDER` tables), if the persistent queue (`.pq`) file was deleted before ICON restarted.

Note: When ICON starts up, the `gcc`, `gls`, and `gud` providers start automatically, regardless of the role that has been configured for the ICON Application, in order to perform certain internal functions. The respective provider control tables might therefore contain partial records (for example, containing information about the start time or end time of the ICON application), even if that ICON application does not perform the particular provider role. However, the ICON provider stores data in IDB only if the ICON Application has been configured to perform the applicable role. Full records in the corresponding provider table are created and updated only in connection with data storage activities.

Purpose of the Provider Control Tables

The provider control tables support data integrity analysis. They provide a mechanism to determine the availability and reliability of various types of data in IDB. In ICON HA deployments, the downstream reporting application can use this information to determine which IDB is the better source from which to extract data for a particular time period.

For more information about using the provider control tables, see “Determining Data Availability and Reliability” on [page 49](#) and “Extracting HA Data” on [page 109](#).

Service and Dictionary Tables

The service and dictionary tables are used for ICON internal purposes or describe fields in other tables.

The six tables in this group are:

- `G_DICTIONARY` and `G_DICT_TYPE`—Contain dictionary information for certain enumerator fields in other tables (for example, the `STATE`, `STATUS`, and `CAUSE` fields).
- `G_DB_PARAMETERS`—Contains general information about the database schema (for example, the schema version).
- `G_SYNC_CONTROL` and `G_PROV_CONTROL`—Are used by ICON to implement internal transaction control.
- `G_TIMECODE`—Expands the timecode values that are referenced in other tables (for example, `CREATED_TCODE` and `DELETED_TCODE`) into specific time value entities such as month, day of the week, day of the month, and so on.

Log Tables

The log tables are internal tables used by ICON’s system procedures.

The five tables in this group are:

- `G_LOG_ATTRS`—Stores attributes about the messages stored in the `G_LOG_MESSAGES` table.
- `G_LOG_MESSAGES`—Stores messages from the stored procedures about merge operations, purge operations, and stuck calls.
- `G_LOG_GETIDRANGEREQ`—Stores information that Solution Control Interface (SCI) uses internally for selecting log records.
- `GSYS_DNPREMOTELOCATION`—Stores information about the remote locations involved in an interaction.
- `GSYS_SYSPROCINFO`—Stores information that ICON uses internally for processing.



Chapter

3

How ICON Works

This chapter describes basic Interaction Concentrator functioning—how the Interaction Concentrator server (ICON) collects, processes, and stores configuration and interaction data in Interaction Database (IDB).

This chapter contains the following sections:

- [ICON Processing, page 39](#)
- [Populating Configuration Data, page 40](#)
- [Populating Interaction Data, page 43](#)
- [Identifying Who Released the Call, page 45](#)
- [Determining Data Availability and Reliability, page 49](#)

ICON Processing

ICON is a client of the data sources that are specified on the **Connections** tab of the **ICON Application—T-Server, Interaction Server, Configuration Server, or Outbound Contact Server (OCS)**. ICON monitors events from these data sources, and it processes events for the types of data that the ICON instance has been configured to collect, as specified by the **ICON role** option.

Processing occurs in the in-memory queue (*accumulator*), as ICON prepares the data for storage in IDB. For all types of data except configuration data, ICON then writes the prepared data to the persistent queue, and from the persistent queue into IDB. For information about how ICON completes the processing of configuration data, see “[Populating Configuration Data](#)” on [page 40](#).

You can configure the size of the in-memory queue or the interval at which data is written from it to the persistent queue.

You can also configure the maximum number of keep-in-memory interactions that concurrently reside in an interaction queue or interaction workbin.

Memory optimization configuration options on the **ICON Application** enable this functionality, which requires Interaction Server release 7.6.1 or higher.

For more information about the persistent queue, see “Persistent Queue and Persistent Caches” on [page 19](#). See also the equivalent section in the overview chapter in the *Interaction Concentrator 8.0 Deployment Guide*, which provides additional information.

For more information about the ICON configuration options, see the *Interaction Concentrator 8.0 Deployment Guide*.

Note: ICON monitors the activity of its data sources and other configuration objects, in accordance with the role and option settings that are configured for the ICON instance. This means that options or features that are configured on other Genesys components might not be reflected in IDB data. For example, if an endpoint has been disabled in the Configuration Layer by an Administrator, configuration data in IDB will correctly show the disabled state (the STATE field in the GC_ENDPOINT table), but any activity on that DN will continue to generate data in the interaction-related tables (for example, G_PARTY).

Populating Configuration Data

If configured to do so, ICON gathers information about contact center configuration objects from Configuration Server on initial startup, and keeps track of changes made to these objects throughout its operation by monitoring dynamic real-time notifications from Configuration Server.

ICON collects and stores configuration-related data if the value of the role configuration option (in the callconcentrator section of the ICON Application) is set to `cfg` or `all`.

An ICON performing the `cfg` role collects configuration-related data:

- On startup—This is the first deployment of ICON, when the IDB is empty. For more information, see “Reading the Configuration Database on Startup” on [page 41](#).
- When the persistent cache is unavailable. For more information, see “Persistent Cache Is Not Available” on [page 41](#).
- Through real-time object change notifications. For more information, see “ICON Receives Dynamic Notifications” on [page 42](#).
- Upon receipt of the Configuration Server’s history log file. For more information, see “ICON Reads the Configuration History Log” on [page 42](#).
- Upon user request for resynchronization. For more information, see “User Request For Resynchronization” on [page 43](#).

ICON stores configuration-related data in the following tables in IDB:

- Tables prefixed with `GC_`—Information about the addition of new objects and the deletion or update of existing objects
- Tables prefixed with `GCX_`—Information about object relationships

Note: ICON stores information in GC_APPLICATION about the following application types only—Outbound Contact Server, CPD Server, T-Server, and Agent Desktop.

For more information about the configuration tables, see the *Interaction Concentrator 8.0 Physical Data Model* for your RDBMS.

Persistent Cache for Configuration Data

The ICON instance that performs the `cfg` role maintains a persistent cache for configuration data. The name of this local file is `cfg-sync.db`, and it cannot be changed. Data in the persistent cache survives a shutdown and restart of ICON.

When it receives data from Configuration Server, ICON writes the data from its in-memory queue to the persistent cache, and then from the persistent cache into the configuration tables in IDB.

The persistent cache enables ICON to synchronize configuration data in IDB with current Configuration Server information. The persistent cache contains a timestamp for configuration data changes. On startup, ICON requests from Configuration Server all configuration changes that occurred after that timestamp. ICON then updates the persistent cache, transfers the configuration data to IDB, and continues to monitor real-time notifications from Configuration Server.

Reading the Configuration Database on Startup

Upon initial startup, or if the local cache file is not available (see “[Persistent Cache Is Not Available](#)” below), ICON queries Configuration Server for all active configuration objects and active relationships. ICON loads the persistent cache with the information it gathers about these objects and relationships. It then submits the updated transactions to its persistent queue, from which it updates the configuration-related tables in IDB.

Persistent Cache Is Not Available

If the persistent cache is not available during a *routine* startup (that is to say, not the initial startup when the IDB is empty), ICON performs a resynchronization of IDB automatically. When it restarts, ICON verifies the content of the IDB using the last processed real-time notification from Configuration Server. If there is no information about the last notification because the persistent cache is not available, ICON requests all configuration-related information from Configuration Server and recovers the persistent cache.

ICON Receives Dynamic Notifications

ICON is a client of Configuration Server. Whenever both applications are operating and changes are made to configuration objects or their relationships to other objects within Configuration Manager, Configuration Server immediately notifies its clients of the change. (Genesys does not support such notification if objects are changed directly within the Configuration Database.) The persistent cache is designed to always be synchronized with Configuration Database. When ICON receives the notification, it immediately sends the information to the persistent cache, and records it in the appropriate IDB table using the actual timestamps when the object was changed in Configuration Server.

ICON Reads the Configuration History Log

Configuration Server maintains a history log for the purpose of enabling clients to restore a session that was terminated by a service interruption and to request any changes to configuration objects that occurred during that the interruption. Dynamic changes made to configuration objects are reported directly by Configuration Server. ICON requests this information from Configuration Server every time it connects to it.

With earlier releases of Configuration Server (prior to 7.6), you must ensure that the Configuration Server's `history-log-active` option is set to `true`. If set to `false`, configuration changes are not recorded in the history log file. Therefore, if ICON lose connection to Configuration Server during this time, it cannot later retrieve information about configuration changes during the time of the disconnect. Setting this option to `false`, however, does not prevent resynchronization.

In Configuration Server release 7.6 or later, you can set the following Configuration Server `[history-log]` configuration options to control the history log functionality:

- `all`
- `expiration`
- `client-expiration`
- `max-records`
- `active`
- `failsafe-store-processing`

Note: If `failsafe-store-processing` is set to `false`, the history log database may not be wholly preserved.

Refer to the configuration history log section in the *Framework 8.0 Deployment Guide* and the history log section in the *Framework 8.0*

Configuration Options Reference Manual for more information about these options.

User Request For Resynchronization

On demand resynchronization occurs when a customer manually runs the resynchronization procedure (see “How to Resynchronize Configuration Data” on [page 146](#) for complete step-by-step instructions). When instructed to start resynchronization, ICON requests all configuration data from Configuration Server and stores it in its persistent cache. At the same time, all other activity—such as dynamic notifications—between ICON and Configuration Server is disabled. ICON then transfers configuration data in the persistent cache to the IDB and begins to monitor real-time notifications from Configuration Server again.

Populating Interaction Data

This section describes aspects of basic ICON functioning to capture information about voice calls.

For information about how to capture information about multimedia interactions, see Chapter 4 on [page 55](#).

For information about the way in which ICON handles attached data for voice calls, see “Attached Data Processing for Voice Calls” on [page 64](#).

T-Server TEvents

ICON connects to T-Server, and it receives notifications, in the form of TEvents, about voice call processing.

ICON provides two tracks for operational reporting: call-based and party-based.

Real-time interaction data, such as voice-specific interactions, is stored in the following IDB tables:

G_IR	G_CALL	G_CALL_USERDATA
G_IR_HISTORY	G_CALL_HISTORY	G_CALL_USERDATA_CUST
G_IS_LINK	G_CALL_STAT	G_CALL_USERDATA_CUST1
G_IS_LINK_HISTORY	G_PARTY	G_CALL_USERDATA_CUST2
G_ROUTE_RESULT	G_PARTY_HISTORY	G_USERDATA_HISTORY
	G_PARTY_STAT	G_SECURE_USERDATA_HISTORY

For detailed information about the tables in IDB in which ICON stores interaction data, see the *Interaction Concentrator 8.0 Physical Data Model* document for your particular RDBMS.

Extracting Interaction Data

Genesys recommends the following approach to extracting interaction data from IDB:

1. Use records in the G_IR table as the base records for data extraction. Use the value of the IRID field as an identifier to determine which records to extract from the G_IR_HISTORY and G_CALL tables.
2. After data extraction from the G_CALL table, use the value of the CALLID field as an identifier to determine which records to extract from the other interaction-related tables.
3. After data extraction from G_PARTY, use the value of the PARTYID field as an identifier to determine which records to extract from the G_PARTY_HISTORY and G_PARTY_STAT tables.

Stuck Records

Stuck records can result when ICON restarts if, during the time that ICON was shut down:

- A change occurred in agent session data (for example, an agent logged in or out).
- Outbound campaign processing completed.
- A call was distributed from a virtual queue.
- A call or party was deleted.

Stuck calls or parties can also result from stuck calls or parties on the T-Server or Interaction Server side.

Note: The disconnection of ICON from T-Server or Interaction Server does not in itself result in stuck calls. ICON can retrieve a snapshot of the active calls and compare this to the calls in memory.

Stuck Call Resolution Procedure

When data is incomplete, ICON uses an internal stored procedure to resolve stuck calls and to determine whether to process the available data. This procedure is based on a timeout mechanism. It allows for continued data processing in the event of partial data loss. Within IDB, ICON marks the records it detects to be incomplete as having low reliability. For more information, see the G_IR.GSYS_EXT_VCH2 field description in the *Interaction Concentrator 8.0 Physical Data Model* document for your particular RDBMS.

Identifying Who Released the Call

Provided that T-Server includes the required attributes in TEvent messaging, ICON stores information in IDB that enables downstream reporting applications to identify the party that released the call.

ICON provides two kinds of call-release reporting. The two kinds of call-release reporting are independent from each other.

- **Call-based**—In the `G_CALL_STAT` table, ICON stores information about the device (endpoint), and possibly also the associated party, that initiated termination of the call. For more information, see [“Call-Based Reporting”](#) below.
- **DN-based**—In the `G_PARTY_STAT` table, ICON stores information that indicates whether termination was initiated locally (on this party’s endpoint) or remotely (on the other party’s endpoint). ICON stores this information only for parties that are associated with endpoints that are monitored by that ICON instance. For more information, see [“DN-Based Reporting”](#) on [page 46](#).

For information about the implications for multi-site deployments, see [“Multi-Site Deployments”](#) on [page 48](#).

Call-Based Reporting

When a call is terminated and ICON receives `EventCallDeleted` from T-Server, ICON processes the event to obtain the following information:

- From the `AttributeCallUUID` attribute, ICON gets the `CallID` that identifies the call. ICON includes this value in the `G_CALL_STAT` record.
- From the `AttributeCtrlParty` attribute, ICON gets a string that specifies the device (endpoint) that initiated release of the call. ICON stores this string, exactly as T-Server delivered it, in the `GSYS_EXT_VCH1` field in the `G_CALL_STAT` table.

If T-Server does not report the `AttributeCtrlParty` attribute in `EventCallDeleted`, ICON stores an empty string in the `GSYS_EXT_VCH1` field in the `G_CALL_STAT` table.

- Based on the value of the `AttributeCtrlParty` attribute, ICON might be able to identify the party associated with the monitored DN.
 - If it can identify the associated party, ICON stores the `PartyID` in the `GSYS_EXT_VCH2` field in the `G_CALL_STAT` table.

- If it cannot identify the associated party, ICON stores an empty string in the `GSYS_EXT_VCH2` field in the `G_CALL_STAT` table.

Note: T-Server does not report the `AttributeCtrlParty` attribute when a consultation call is terminated as the result of a completed two-step transfer or two-step conference. Therefore, in these scenarios, ICON stores an empty string in the `GSYS_EXT_VCH1` and `GSYS_EXT_VCH2` fields in the `G_CALL_STAT` table in records that are related to the consultation call.

DN-Based Reporting

When a call is terminated and ICON receives `EventReleased` or `EventAbandoned` from T-Server, ICON writes a record in the `G_PARTY_STAT` table. ICON uses the value of the `ReleasingParty` extension key, in the `AttributeExtensions` of the event, to populate the `GSYS_EXT_INT1` field of the `G_PARTY_STAT` record with one of the following values:

G_PARTY_STAT Values

- 0—T-Server did not provide `ReleasingParty` data in `EventReleased` or `EventAbandoned`.
- 1—Local. Call termination was initiated by this party (`ThisDN` in `EventReleased` or `EventAbandoned`), and T-Server sends `ReleasingParty` = 1 Local in the event.
- 2—Remote. Call termination was initiated by another party (not `ThisDN` in `EventReleased` or `EventAbandoned`), and T-Server sends `ReleasingParty` = 2 Remote in the event.
- 3—Unknown. T-Server was unable to determine the initiator of call termination, and sends `ReleasingParty` = 3 Unknown in `EventReleased` or `EventAbandoned`.

If ICON is monitoring the DNs for more than one of the parties involved in a call, there will be multiple `G_PARTY_STAT` records—one for each party.

Notes:

- The T-Server configuration option `releasing-party-report` must be enabled on the T-Server Application in order to report this extension.
 - T-Server does not report `ReleasingParty` information when a consultation call is terminated as the result of a completed two-step transfer or two-step conference. Therefore, in this scenario, ICON stores the value 0 (no data provided by T-Server) in the `GSYS_EXT_INT1` field in the `G_PARTY_STAT` table.
-

Reporting Summary

Table 3 summarizes the values that ICON might report in the G_CALL_STAT table for the party that released various types of calls. For the values that ICON might report in the GSYS_EXT_INT1 field in the G_PARTY_STAT table, see “G_PARTY_STAT Values” on page 46.

Table 3: Possible Values for Call-Release Reporting, by Call and Releasing Party Type

Releasing Party*	AttributeCtrlParty in EventCallDeleted	G_CALL_STAT Table	
		GSYS_EXT_VCH1	GSYS_EXT_VCH2
Call type = Inbound or Outbound			
External	No data	Empty string	Empty string
	“External DN”	“External DN”	<PartyID associated with "External DN" and associated with the call identified by CallID of this call>
Internal	No data	Empty string	Empty string
	“Internal DN”	“Internal DN”	<PartyID associated with EndpointDN = "Internal DN" and CallID of this call>
Call type = Internal			
Internal	No data	Empty string	Empty string
	“Internal DN”	“Internal DN”	<PartyID associated with EndpointDN = "Internal DN" and CallID of this call>
Call type = Consultation ^a			
External or Internal	No data	Empty string	Empty string
Call type = Conference (one internal party, all other parties external)			
Internal	“Internal DN”	“Internal DN”	<PartyID associated with EndpointDN = "Internal DN" and CallID of this call>
*Releasing party is in relation to the T-Server or switch monitored by the ICON instance.			

- a. The consultation call is terminated as the result of completion of a two-step transfer or two-step conference, after a CompleteTransfer or CompleteConference request to T-Server. T-Server does not report AttributeCtrlParty in EventCallDeleted in these cases.

Multi-Site Deployments

In multi-site configurations, regardless of whether each site is monitored by a separate ICON or by the same ICON instance, the record that reports the initiation of call termination on a site treats endpoints from the other site(s) as external resources (endpoints).

In most cases, this external resource is represented and reported by T-Server as an External Routing Point.

Example

Assume that there are two ICON instances, writing to separate IDBs, for two sites: ICON_1, which writes to IDB_1 and monitors Site 1; and ICON_2, which writes to IDB_2 and monitors Site 2. A call from DN_1 on Site 1 goes to DN_3 on Site 2, and then is released by DN 3.

ICON_1 creates two parties:

- One party is associated with DN_1.
- The second party is represented as an external party associated with external resource Ext_DN2. Ext_DN2 is usually represented and reported as an External Routing Point.

Table 4 summarizes the values that ICON will report for the party that released the call.

Table 4: Example of Call-Release Reporting in a Multi-Site Deployment

IDB	G_CALL_STAT Table			G_PARTY_STAT Table	
	Data Source	GSYS_EXT_VCH1	GSYS_EXT_VCH2	Data Source	GSYS_EXT_INT1
IDB_1	AttributeCtrlParty in EventCallDeleted received from the T-Server connected to ICON 1	"Ext_DN2" or "String submitted by T-Server"	<PartyID associated with EndpointDN = "Ext_DN2" as represented on this site>	ReleasingParty in EventReleased for the party associated with DN_1, received from the T-Server connected to ICON 1	2
IDB_2	AttributeCtrlParty in EventCallDeleted received from the T-Server connected to ICON 2	"DN_3"	<PartyID associated with EndpointDN = "DN_3" and CallID of this call>	ReleasingParty in EventReleased received from the T-Server connected to ICON 2	1

T-Server Support

To implement reporting on which party released the call, ICON relies on T-Server to provide the required data. This feature requires T-Server release 8.0 or higher. Interaction Concentrator 8.0 supports this feature for the Alcatel A4400/OXE switch. Interaction Concentrator 8.0.000.35 and higher also supports this feature for Avaya switches; for Avaya, this functionality requires T-Server for Avaya Communication Manager release 8.0.101.05 or higher.

For information about configuring T-Server to support this functionality and for information about the scenarios in which T-Server reports the required attributes, see the section about call release tracking in the *Framework 8.0 T-Server Deployment Guide* for the applicable T-Server.

Limitations

ICON is unable to determine the party that released the call in the following situations:

- If the call was terminated during a time that ICON was down. On restart, ICON reports the releasing party information as if no data was received from T-Server.
- When a consultation call is terminated after a request to complete the transfer or conference (see the Note on [page 46](#) for call-based reporting, and the Notes on [page 46](#) for DN-based reporting).

In these situations, ICON reports an empty string in the GSYS_EXT_VCH1 and GSYS_EXT_VCH2 fields in the G_CALL_STAT table, and the value 0 in the GSYS_EXT_INT1 field in the G_PARTY_STAT table.

Determining Data Availability and Reliability

Starting with release 8.0, ICON tracks detailed control data related to connections and events from ICON data sources, and stores the data in G_DSS_*_PROVIDER tables in IDB. For more information about the provider control tables, see “Data Source Session Control Tables” on [page 34](#).

Downstream reporting applications can analyze the control data to determine the availability and reliability of reporting data in a particular IDB. Based on the analysis, the downstream reporting applications can adjust the extraction, transformation, and loading (ETL) activities to optimize ETL processes. In high availability (HA) deployments, the downstream reporting application can use the results to identify which IDB is the better data source for a particular time interval. For more information, see “Extracting HA Data” on [page 109](#).

For information about the IDB inconsistencies that can result from unavailable data, see “Consequences of Failures” on [page 106](#).

Determining Data Availability

The following example of a typical scenario illustrates how the data in the control tables can be used to identify gaps in the reporting data.

The scenario tracks one ICON instance and one T-Server, and considers the connection information that is applicable to only the GCC provider.

[Table 5](#) shows scenario values for connection-related fields in the G_DSS_GCC_PROVIDER table for various startup, disconnection, reconnection, and shutdown events that occur at times t0 through t6.

Table 5: Scenario Example—G_DSS_GCC_Provider Table Field Values

Events	G_GCC_Provider Table Record	Selected G_DSS_GCC_PROVIDER Table Fields				
		DSCONN_TYPE ^a	ICON_STIME	DSCONN_STIME	ICON_ETIME	DSCONN_ETIME
ICON starts up at t0, connects to T-Server at t1, and writes some data to IDB.	New	1	t0	t1	Null	Null
ICON disconnects from T-Server at t2 (network failure case).	Updated	1	t0	t1	Null	t2
ICON reconnects to T-Server at t3 (network failure case).	New	2	t0	t3	Null	Null
T-Server disconnects from the switch at t4.	Updated	2	t0	t3	Null	t4
T-Server reconnects to the switch at t5.	New	4	t0	t5	Null	Null
ICON shuts down unexpectedly at t6. Or: ICON shuts down gracefully at t6.	No change when ICON restarts Updated when ICON restarts	4 4	t0 t0	t5 t5	Null t6	Null t6

a. For the meaning of the DSCONN_TYPE values, see “Connection Type” on [page 34](#).

Connection Analysis

The history of the connection to the data source indicates the points of interruption in the data flow. In the scenario illustrated in Table 5 on [page 50](#), the downstream reporting application can determine that, for a particular ICON instance, data from T-Server was not reliably available during the following time intervals:

- t2–t3
- t4–t5
- Starting with t6 (in the case of a planned ICON shutdown, or in the case of an unplanned ICON shutdown in an HA deployment)
- t5–the time that the next new record is created (in the case of an unplanned ICON shutdown)

The next new record is created after ICON restarts, reconnects to the data source, and receives the first event (at t7). From the existence of the new record, the ETL can infer that data was not available at some time between t5 and t7. The timestamp of the last processed event (LEVENT_DSTIME and LEVENT_ETIME) can help the downstream reporting application approximate the shutdown time (t6).

In an HA deployment, the absence of information after t6 in, say, ICON-1 and the presence of information after t6 in ICON-2 will enable the ETL to reliably determine that data was not available for ICON-1 from t6 to t7.

Determining IDB Availability

The G_DSS_*_PROVIDER tables for the GCC, GLS, GUD, GOS and CFG roles provide an indirect heartbeat mechanism that enables the downstream reporting application to distinguish between (a) the case in which there is no data for ICON to store and (b) the case in which ICON does not store data in IDB because of a problem between ICON and IDB.

Note: If you want the G_DSS_*_PROVIDER tables to be populated, you must set the value of the use-dss-monitor configuration option to true.

No Data to Store	For each provider, ICON stores in the persistent queue (.pq file) a timestamp of the last data that was written to the persistent queue. If no new data is written to the queue during a predefined interval, ICON creates a “no data” record for the applicable provider(s), and ICON sends this record to IDB in the usual way. The default value for this interval is 300 seconds; it can be changed if desired.
NODATA_IUTC Field	When the “no data” record is sent to IDB, the NODATA_IUTC field in the applicable G_DSS_*_PROVIDER tables is updated for all open sessions created by the ICON instance. The value of the NODATA_IUTC field is the time the “no data” record was created—in other words, the timestamp of the ICON confirmation

that no data was received from the data source server in the previous period of time set for the “no data” interval.

Example

ICON-1 performs the gcc role and is connected to three data source servers: T-Server1, T-Server2, and T-Server3. The G_DSS_GCC_PROVIDER table contains records for active sessions for all three data source servers. The value for the “no data” interval is set to the default value of 300 seconds.

- Starting from time t_0 , T-Server1 and T-Server2 have no activity. However, T-Server3 continues to send data.

No change is made to the value of the NODATA_IUTC field in the record for any of the sessions, because ICON is receiving data from a data source.

- Starting from time t_1 , T-Server3 has no activity. T-Server1 and T-Server2 continue to have no activity.

No change is made to the value of the NODATA_IUTC field in the record for any of the sessions, because ICON has not yet identified the “no data” situation.

- At time $t_1 + 300$ seconds, there is still no activity from T-Server1, T-Server2, or T-Server3. ICON creates the “no data” record and sends it to IDB.

The NODATA_IUTC field in the record for all three sessions is updated with the timestamp of the “no data” record.

IDB Availability Analysis

The ETL can evaluate the recent activity of the NODATA_IUTC field value and the LEVENT_ETIME field value (the timestamp for the last event stored on the connection), and use the information to identify if there is a problem between ICON and IDB.

Table 6 summarizes the analysis.

Table 6: Determining IDB Availability

Value of IDB Field During Last Two Minutes		Conclusion
LEVENT_ETIME	NODATA_IUTC	
Changed	Not applicable	IDB is available, and new data is arriving.
Unchanged	Changed	IDB is available, but there is no new data.
Unchanged	Unchanged	IDB is not available.

Determining Data Reliability

Interaction Concentrator provides mechanisms to determine the reliability of available data in two ways:

- Evaluation by ICON of the reliability of the data it records in IDB (see [“Reliability of Data in IDB Records”](#))
- Evaluation by the downstream reporting application of the reliability of the data provided by a particular ICON instance (see [“Reliability of Data from ICON and IDB”](#))

Reliability of Data in IDB Records

Within IDB, ICON uses system fields in various tables to flag the reliability of data in the record. For example, the `GSYS_EXT_INT1` field in the `G_USERDATA_HISTORY` or `G_SECURE_USERDATA_HISTORY` tables indicates the reliability of the change type that is assigned to the user data record. For more information about the reliability flags in IDB, see the *Interaction Concentrator 8.0 Physical Data Model* document for your particular RDBMS.

Reliability of Data from ICON and IDB

From the information in the provider control tables and the analysis of data availability, ICON users can evaluate the reliability of data from a particular ICON instance.

ICON can guarantee the reliability of call data only if the call was visible to ICON for the entire call duration, from the time of call creation until call termination.

- ICON does not store any data for calls that were created before ICON started up.
- ICON does store data for calls that were created after ICON started up but that were not visible to ICON for the entire call duration.

If the event flow that ICON monitors is incomplete (the call is not yet terminated) and the ETL determines that no new data is expected (IDB is not available), then data for all non-terminated calls should be considered unreliable.

For information about further analysis of data reliability to optimize ETL extraction strategies in HA deployments, see [“Extracting HA Data” on page 109](#).

For information about the analysis of data availability, see [“Determining Data Availability” on page 50](#).



Chapter

4

Integrating with Multimedia

Genesys eServices/Multimedia (formerly known as Multi-Channel Routing or MCR) refers to those parts of the Genesys Customer Interaction Management (CIM) platform that work together to manage interactions that involve nontraditional, non-voice media (for example, e-mail and chat) and open (custom-designed) media (for example, fax and web forms).

This chapter describes how Interaction Concentrator (ICON) processes data about eServices/Multimedia and 3rd Party Media (formerly referred to as Open Media) interactions.

This chapter contains the following sections:

- [Multimedia Objects, page 56](#)
- [Populating Multimedia Interaction Data, page 57](#)
- [Handling Active Multimedia Interactions, page 59](#)
- [isOnline Chat Attribute, page 61](#)

For information about ICON configuration and other Configuration Layer settings that make data about multimedia interactions available in Interaction Database (IDB), see the chapter about special configuration requirements in the *Interaction Concentrator 8.0 Deployment Guide*.

Note: This chapter does not cover Session Initiation Protocol (SIP) chat. For ICON processing, the important distinction is the source of the data, not the media type. The source of eServices/Multimedia and 3rd Party Media data is Interaction Server. The source of SIP chat data is SIP Server.

Terminology Note

In this document, the term *multimedia* refers generically to Genesys eServices/Multimedia and 3rd Party Media objects and activities. When it is necessary to distinguish between the two types of multimedia, the terms *eServices/Multimedia* and *3rd Party Media* are used, as applicable.

Multimedia Objects

This section introduces the terminology and elements (objects) that pertain to Interaction Concentrator data about multimedia activities.

This section contains information about the following:

- Endpoints
- DNs

Endpoints

ICON stores reporting data about the following logical endpoints:

- **Interaction Queue**—Configured in the Configuration Layer as a Script object (of type Interaction Queue).
- **Interaction Workbin**—Configured in the Configuration Layer as a Script object (of type Interaction Work Bin).
- **Routing Strategy**—Configured in the Configuration Layer as a Script object (of type Simple Routing).
- **Agent's Place**—Configured in the Configuration Layer as a Place object.

ICON stores configuration-related information about Script objects in the GC_SCRIPT table.

ICON stores configuration-related information about Place objects in the GC_PLACE table.

DNs

The Interaction Server uses a Switch configuration object of type Multimedia Switch. ICON stores information about DNs that are configured under the switch associated with the Interaction Server switch in the same way that it stores information about DNs that are configured under any other type of switch.

ICON stores configuration-related information about DN configuration objects in the GC_ENDPOINT table.

ICON stores configuration-related information about the association between DNs and places in the GCX_ENDPOINT_PLACE table.

Populating Multimedia Interaction Data

ICON stores detailed information about multimedia interaction processing and agent activities that are related to this processing. This section contains information about the following:

- Multimedia Reporting Protocol events (see below)
- Multimedia interactions (see below)
- Supported scenarios (see [page 59](#))

For information about the way in which ICON handles attached data for multimedia interactions, see “Attached Data Processing for Multimedia” on [page 68](#).

For information about how to capture information about agent states and login sessions for multimedia interactions, see Chapter 7 on [page 79](#).

For detailed information about the tables in IDB in which ICON stores multimedia data, see the *Interaction Concentrator 8.0 Physical Data Model* document for your particular RDBMS.

Multimedia Reporting Protocol Events

ICON connects to Interaction Server, and it receives notifications, in the form of Multimedia Reporting Protocol events, about multimedia interaction processing.

ICON processes the following Multimedia Reporting Protocol events for interactions:

- EventInteractionSubmitted
- EventProcessingStopped
- EventPlacedInQueue
- EventTakenFromQueue
- EventPlacedInWorkbin
- EventTakenFromWorkbin
- EventPartyAdded
- EventPartyRemoved
- EventPropertiesChanged
- EventAgentInvited
- EventRejected
- EventRevoked
- Multimedia Reporting custom message (envelope for virtual queue–related TEvents)

For more information about Multimedia Reporting Protocol events, see the *Genesys 7 Events and Models Reference Manual*.

Multimedia Interactions

Each interaction received from Interaction Server contains a media type name that is defined in Configuration Server. When ICON processes reporting events, such as EventInteractionSubmitted, it extracts the media type name from the interaction and maps this string to an integer value.

ICON stores the details about eServices/Multimedia and 3rd Party Media interactions in the same tables in IDB in which it stores voice records:

- Core Tables**
- **G_CALL**—Each interaction with a multimedia media type (e-mail, chat, or open media) is represented as a single record. ICON creates the record when it receives `EventInteractionSubmitted`. ICON updates the record (marks the record as terminated) when it receives `EventProcessingStopped`. ICON sets the value of the `GSYS_EXT_INT1` field to 1 when the record is for a multimedia interaction (in other words, if the interaction record was created as a result of Interaction Server event processing). For 3rd Party Media interactions, ICON also sets the value of the `GSYS_EXT_VCH1` field to a string value containing the name of the media type.
 - **G_PARTY**—Contains information about the parties who participate in a multimedia interaction. ICON creates a new record when Interaction Server reports that a relationship has been established between an endpoint and a multimedia interaction.

Note: No records are stored for external parties who participate in a multimedia interaction.

- **G_IR**—Contains information about the data that is common to all of the interactions in a particular scenario. ICON sets the value of the `GSYS_EXT_INT1` field to 1 when the record is for a multimedia interaction (in other words, if the interaction record was created as a result of Interaction Server event processing).

A new record for a multimedia interaction is created if either of the following conditions is met:

- The interaction is not associated with an existing parent interaction.
- The interaction is associated with an existing parent interaction, but the information about the associated `G_IR` parent record is not available. In this case, ICON stores the ID information that Interaction Server provides for the parent record in the `GSYS_EXT_VCH1` field.
- **G_ROUTE_RESULT**—Refers to the record created in the `G_PARTY` table for the strategy and contains information about the results of routing for the interaction.

Note: No multimedia interaction data is written to the `G_IS_LINK` table.

- History Tables**
- **G_CALL_HISTORY**, **G_PARTY_HISTORY**, and **G_IR_HISTORY**—Contain intermediary (history) states of the data that was previously stored in the core tables.

- Statistical Tables**
- **G_CALL_STAT** and **G_PARTY_STAT**—Contain metrics about multimedia interactions and parties, respectively. For information about the available precalculated metrics, see Table 1 on [page 27](#) and Table 2 on [page 29](#).

Agent Activity

ICON stores the details about agents that are involved in multimedia interaction activity in the same tables in IDB in which it stores voice records. For information about the tables, see “Agent State–Related and Login Session–Related Tables” on [page 30](#).

In all the agent-related tables except `G_LOGIN_SESSION` and `G_DND_HISTORY`, ICON sets the value of the `GSYS_EXT_INT1` field to 1 when the record is for a multimedia interaction (in other words, if the interaction record was created as a result of Interaction Server event processing). For 3rd Party Media interactions, ICON also sets the value of the `GSYS_EXT_VCH1` field to a string value containing the name of the media type.

Supported Scenarios

Interaction Concentrator supports the following scenarios for multimedia interactions:

- Interaction submission
- Interaction distribution
- Transfer
- Conference
- Auto-acknowledgement
- Autoresponse
- Abandonment without handling (for chat only)

Handling Active Multimedia Interactions

The major difference between voice and multimedia interactions is the duration of the interaction—the lifetime of a multimedia interaction might be measured in months, whereas voice calls are measured in minutes or hours. Therefore, unlike voice interactions, ICON processes multimedia interactions as each step occurs, and immediately stores all available data related to the interaction in IDB. Multimedia interactions have a relatively simple state model, and this enables ICON to start processing multimedia interactions at almost any point in the lifecycle.

ICON can handle millions of active multimedia interactions over a sustained period of time without failing because of insufficient memory.

To make this possible, ICON does the following:

- Removes interactions from operational memory (see “Removing Interactions from Memory” on [page 60](#)).
- Reconstructs interaction data later for further processing (see “Reconstructing Interaction Data” on [page 60](#)).

- Tracks user data changes (see “Tracking User Data Changes” on [page 69](#)).
- Filters out strategy data not required for reporting (see “Strategy Activity Data” on [page 139](#)).

Removing Interactions from Memory

When ICON receives an `EventPlacedInQueue` and/or `EventPlacedInWorkbin` reporting event from Interaction Server, it considers this the signal to remove the corresponding multimedia interaction from memory, provided that certain user-defined options have been set:

- The global `om-memory-optimization` option must be set to `true` to allow ICON to optimize memory according to the user-defined values of the other memory options. If this option is set to `false`, no memory optimization will occur regardless of the values of the other options.
- The `om-max-in-memory` option defines the maximum number of keep-in-memory interactions that concurrently reside in an interaction queue or interaction workbin. When this maximum is reached, ICON removes the oldest interaction.
- If the `om-memory-clean` option is set to `true` (on script objects of type `interaction queue`), ICON immediately removes interactions from operational memory as soon as they arrive; it does not wait until the value of the `om-max-in-memory` option is reached before removing interactions.

For more information about these and other options, see the chapter about configuration options in the *Interaction Concentrator 8.0 Deployment Guide*.

Reconstructing Interaction Data

When ICON receives an `EventTakenFromQueue` and/or `EventTakenFromWorkbin` reporting event from Interaction Server, it considers this the signal to reconstruct the corresponding multimedia interaction if the interaction was previously removed from memory.

To reconstruct the interaction, ICON restores the following key data:

- `CALLID` of the interaction
- `PARTYID` of the party for the last interaction in the queue and/or the last interaction in the workbin
- User data

Once the interaction is reconstructed, ICON processes it as a regular multimedia interaction. See “Populating Multimedia Interaction Data” on [page 57](#) for more information.

isOnline Chat Attribute

The `isOnline` attribute supports reporting on non-SIP chat interactions. Genesys Chat Server can notify Interaction Server when a chat session is active by giving the `isOnline` attribute a value of `on`. When a chat session is terminated—that is, the last person leaves the chat session—the value of the `isOnline` attribute changes to `off`, and Chat Server sends notification of this attribute change to Interaction Server.

Chat Server is responsible for monitoring and changing the value of the `isOnline` attribute. If Chat Server disconnects or shuts down, Interaction Server does not send any notification regarding the status of the chat interaction to Interaction Concentrator. [Figure 2](#) illustrates the possible transmissions and states of the `isOnline` chat attribute.

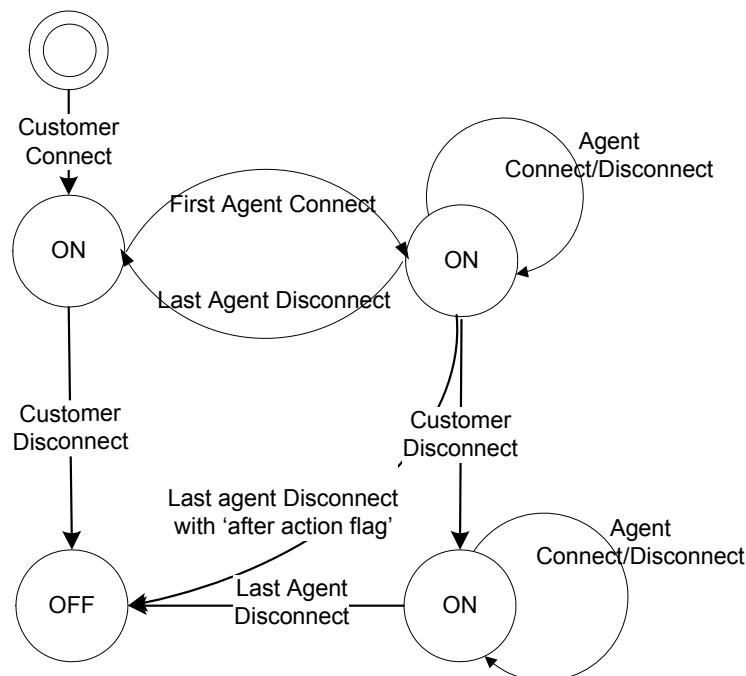


Figure 2: isOnline Attribute of Chat Interactions

Processing the isOnline Attribute

ICON monitors changes in the `isOnline` attribute in all interaction reporting events (`EventInteractionSubmitted`, `EventPropertiesChanged`, and `EventProcessingStopped`). Using the same mechanism to store the `isOnline` attribute as it does to process attached data, ICON creates a new record in the `G_USERDATA_HISTORY` table to store the value of the `isOnline` attribute. Table 7 on [page 62](#) describes the key fields in the `G_USERDATA_HISTORY` table and the possible values for chat interactions.

Table 7: Key Fields in G_USERDATA_HISTORY Table for Chat Interactions

Field Name	Value	Comment
KeyName	_attr_is_online	This value does not change.
ChangeType	1 2 3 4 5	<p>The reason the value of the key was recorded:</p> <ul style="list-style-type: none"> • 1—Created. Indicates that the value of the key was attached to the chat interaction when ICON received an <code>EventInteractionSubmitted</code> event with an <code>isOnline</code> attribute setting of <code>true</code>. • 2—Added. Indicates that the value of the key has been added to an existing interaction. • 3—Updated. Indicates that the value of the key has changed. For chat interactions, ICON receives the information in an <code>EventPropertiesChanged</code> event. • 4—Deleted. Indicates that the key has been deleted from <code>UserData</code>. • 5—Terminated. Indicates that ICON recorded the value of the key when ICON received an <code>EventProcessingStopped</code> event.
Type	1 2 3 4 5	<p>The type of the data source—extensions, reasons, or attached data (userdata)— represented by the following values:</p> <ul style="list-style-type: none"> • 1—userdata • 2—reasons • 3—extensions • 4—attributes (reserved for future use) • 5—mcr_workbin
KeyID	9995	This is a hard-coded value.
Value	0 1	<ul style="list-style-type: none"> • 1—chat session is active • 0—chat session is stopped <p>The value is taken from the event.</p>
Added	<Timestamp value>	This is the time corresponding to when the record was modified.



Chapter

5

Processing Attached Data

This chapter describes how Interaction Concentrator (ICON) processes user data that is attached to voice calls, eServices/Multimedia, and 3rd Party Media interactions. It contains the following sections:

- [Attached Data Specification File, page 63](#)
- [Attached Data Processing for Voice Calls, page 64](#)
- [Customized Attached Data Processing, page 67](#)
- [Attached Data Processing for Multimedia, page 68](#)

Note: The processing and storage of attached data is resource intensive and expensive. Genesys recommends that you carefully consider your reporting and troubleshooting requirements in order to limit the amount of attached data that you configure Interaction Concentrator to capture.

For information about ICON configuration and other Configuration Layer settings that make attached data available in Interaction Database (IDB), see the chapter about special configuration requirements in the *Interaction Concentrator 8.0 Deployment Guide*.

Attached Data Specification File

The attached data specification is an XML file stored in the installation directory that you specify when you install the Interaction Concentrator application.

The attached data specification file (named `ccon_adata_spec.xml` by default) maps the key-value pairs (KVPs) in reporting event attributes to IDB tables and fields.

For information about creating an attached data specification for ICON to use, about the XML schema definition, and for sample attached data specification files, see the *Interaction Concentrator 8.0 Deployment Guide*.

Attached Data Processing for Voice Calls

This section briefly describes how Interaction Concentrator (ICON) processes user data that is attached to voice calls in events from T-Server.

For information about attached user data that is sent in User Events, see Chapter 9 on [page 99](#).

T-Server Interactions

When processing data from T-Server, ICON checks all TEvents for changes to attached data. When attached data changes for a particular interaction, ICON analyzes the change and stores the data in IDB, according to either its application configuration or the attached data specification.

Call-Specific Data

Call-specific data is attached data that is associated only with a call. This data is stored in IDB when the call is cleared after a specified timeout. For information about setting the `call-deletion-timeout` configuration option (on the Switch object) to configure the timeout interval, see the configuration chapter in the *Interaction Concentrator 8.0 Deployment Guide*.

Historical Data

Historical data is associated with the attached data. Historical data can have the following associations:

- **Party**—When an attribute `AttributeCtrlParty` represents a party in `EventCallDataChanged`.
The party with which the historical data is associated is the last party that updated the user data, even if that party had been terminated from the call before the call ended (see “Post-Routing User Data Processing” on [page 65](#)).
- **Endpoint**—When an attribute `AttributeCtrlParty` is specified in `EventCallDataChanged`.
- **Agent**—When an agent is associated with a specific device at the moment when the data is modified.

The stored change type reflects the change type for records with a history type of `all`. For more information about the history types, see the section about configuring for storage of attached data in the chapter about special

configuration requirements in the *Interaction Concentrator 8.0 Deployment Guide*.

[Table 8](#) defines the values for types of attached data changes.

Table 8: Attached Data Change Types

Type of Change	Numeric Equivalent	Description
created	1	Call was created with the specified key.
added	2	Key was added into the existing attached data.
updated	3	Existing key was updated.
deleted	4	Existing key was deleted.
terminated	5	Call was terminated with an existing key.

Post-Routing User Data Processing

Starting with Interaction Concentrator release 8.0, ICON supports the scenario in which a party (strategy or agent) updates user data after the call has left the party. The following are examples of this scenario:

Use Case Examples

- A call reaches a Routing Point, and a strategy is loaded on the Routing Point. The strategy routes the call to an agent. After the call leaves the Routing Point, the strategy updates the user data.
- Agent 1 is handling a call, initiates a consultation call to Agent 2, and then transfers the call to Agent 2. After the call has been transferred, Agent 1 attaches or updates user data.

Provided that the user data is updated within the call deletion timeout, ICON reports the PartyID of the last party on the device that updates user data, even if that party is no longer part of the call.

Implementation

To support this functionality, ICON stores in memory a list of every device that participated in a call and that is a potential source of user data. For each device (EndpointID), ICON stores the PartyID of the last party that was created on the device. The history is updated every time a party is deleted from the call. The history is temporary; it is stored in memory until the call itself is deleted and the user data is written to IDB in the usual way (for example, in the G_USERDATA_HISTORY table). In the IDB record, the EndpointID and PartyID are the most recent device and party associated with a user data update.

Limitations

Note the following limitations for this feature:

- Interaction Concentrator reporting of the party that attached or updated user data is reliable except in the scenario in which multiple parties were created on the same device in the same call, and the endpoint did not send the request(s) for attached data before the next party was created.

For example, a call reaches a Routing Point, and a strategy is loaded on the Routing Point. In a first pass, the strategy attaches user data and returns the call to the Routing Point; in a second pass, the strategy routes the call to an agent; then the Routing Point sends the user data update. ICON will associate the user data update with the second pass through the strategy (the last party on the device).

- The history of recent parties is stored in memory, and it will be lost in the event of a shutdown or restart of ICON before the UserData record has been written to IDB.

Database Schema Extensions for Voice Attached Data

IDB contains two types of tables for UserData collection:

- So-called *flat* tables are used when data that corresponds to a particular key name is stored into a particular field in a table.
- Generic*, or *historical*, tables are used when each attached data value is stored in a separate row with the corresponding context.

[Table 9](#) lists IDB tables that are used to store user data that is attached to voice calls.

Note: The attached data storage tables store user data about each call in a separate record (one record per call).

Table 9: Attached Data Storage Tables for Voice

Table Name	Type of Data	Description
G_CALL_USERDATA	Call	Designed for predefined UserData that is collected during the entire duration of a call.
G_CALL_USERDATA_CUST	Call	Designed for custom UserData that is collected during the entire duration of a call.
G_CALL_USERDATA_CUST1	Call	Designed for custom UserData that is collected during the entire duration of a call.

Table 9: Attached Data Storage Tables for Voice (Continued)

Table Name	Type of Data	Description
G_CALL_USERDATA_CUST2	Call	Designed for custom UserData that is collected during the entire duration of a call.
G_USERDATA_HISTORY	Historical	Designed to store historical changes to UserData.
G_SECURE_USERDATA_HISTORY	Historical	Designed to store historical changes to UserData. Permissions for this table must be set at your particular site.

Attached Data Security

By providing a number of separate tables in which to store attached data, Interaction Concentrator provides a mechanism to secure sensitive user data. The Database Administrator (DBA) can assign user privileges in order to restrict access to the secure user data history table. Similarly, the DBA can restrict access to one or more of the flat tables.

Multi-Site and Distributed Environments

In a multi-site environment or a geographically distributed environment, when attached data is propagated between two T-Servers, each of these T-Servers stores the attached data in its own IDB. As a result, attached data is duplicated in IDBs across the sites.

Customized Attached Data Processing

You can create a custom stored procedure, or *custom dispatcher*, to handle user data that is attached to voice calls and to store the attached data in custom tables in IDB. ICON calls the custom dispatcher when the call ends.

Data Types ICON can process two types of attached data values:

- String
- Integer

Key Groups The values of the key names, specified in the same group, are provided by the same call to the custom dispatcher stored procedure. Within the attached data configuration file, you can specify groups of keys, with each group containing a maximum of 17 string key-value pairs (KVPs) and 17 integer KVPs. You can configure the maximum number of key groups that ICON will process (see the description of the `gud-cust-disp-groups` configuration option in the *Interaction Concentrator 8.0 Deployment Guide*).

For an example of the XML specification for customized attached data processing, see the appendix that provides sample attached data specification files in the *Interaction Concentrator 8.0 Deployment Guide*.

Custom Dispatchers

The IDB initialization scripts create two custom dispatcher stored procedures:

- gudCustDISP1
- gudCustDISP2

The gud-cust-disp configuration option in the ICON Application object specifies which custom dispatcher ICON calls (see the option description in the configuration option chapter in the *Interaction Concentrator 8.0 Deployment Guide*).

While ICON is running, you can switch from one dispatcher to the other. This enables you to make changes to the custom dispatcher configuration without interrupting the processing of attached data.

The default custom dispatchers do nothing. You must modify the scripts in order to create the custom dispatchers that store the attached data that you require. You must also create scripts that, in turn, create the required custom tables in IDB.

Sample Scripts

In addition to the IDB initialization scripts, the Interaction Concentrator installation package contains a sample SQL script, `SampleProc_<db_type>.sql`, to create a custom dispatcher stored procedure and a custom attached data storage table in your IDB schema. The sample script is partially reproduced in an appendix in the *Interaction Concentrator 8.0 Deployment Guide*.

Attached Data Processing for Multimedia

This section briefly describes how ICON processes user data that is attached to multimedia interactions (Genesys eServices/Multimedia and 3rd Party Media).

Interaction Server Interactions

When processing data from Interaction Server, ICON checks all Multimedia Reporting Protocol events for changes to attached data. When attached data changes for a particular interaction, ICON analyzes the change and stores the data in IDB, according to its application configuration and the attached data specification.

For an example of the XML specification for multimedia attached data processing, see the appendix that provides sample attached data specification files in the *Interaction Concentrator 8.0 Deployment Guide*.

Multimedia Interaction–Specific Data

Multimedia interaction–specific data is attached data that is associated only with a multimedia interaction. ICON stores this data in predefined fields in separate multimedia attached data tables in IDB.

Note: By default, Interaction Server does not automatically attach the keys that are required in order to report all the multimedia attached data that Interaction Concentrator can support. You might need to modify your routing strategies so that Interaction Server attaches data for the required keys (for example, Suggested Response Name).

Database Schema Extensions for Multimedia Attached Data

IDB contains two tables for multimedia-specific user data collection:

- **GM_L_USERDATA**—Stores the values of attached data keys for suggested and auto responses and acknowledgements, customer IDs, and reasons for stopping processing. ICON writes information to this table when Interaction Server reports that the interaction has finished.
 - ICON captures the names of the responses and acknowledgements from the value that the applicable keys had when ICON first received the report about the interaction from Interaction Server.
 - ICON captures the customer IDs and reason information from the final value of the applicable keys when Interaction Server reports that the interaction has ended.
- **GM_F_USERDATA**—Stores information about predefined logical keys in the attached data of multimedia interactions. Information includes:
 - Sender (“From” name and e-mail address)
 - Called back
 - Subject
 - Type and subtype
 - Origination source (webform or e-mail)
 - Time the interaction was received

ICON writes information to this table when Interaction Server first sends reporting events about the interaction to ICON. Therefore, the values that are stored by ICON are the first values that are presented for the applicable keys.

Tracking User Data Changes

The memory optimization feature enables ICON to process a large number of active multimedia interactions. In order to do so, ICON removes interactions from operational memory if they are not in the active stage of processing, and later reconstructs the interaction and attached user data for further processing.

With this feature enabled, ICON tracks changes to multimedia user data in the following ways:

- Reconstructs the user data attached to interactions that were removed from operational memory. As part of the reconstruction of an interaction, ICON also reconstructs the user data content from the `EventTakenFromQueue` reporting event that was received from Interaction Server. Any changes in user data are then processed using the same rules implemented in release 7.6.
- Processes user data for interactions in the Interaction queue. While an interaction is in the Interaction Queue, Interaction Server sends `EventPropertiesChanged` reporting events to ICON for storage in IDB. Because ICON does not know whether the user data key is new or changed, ICON stores this information in IDB with an attribute changed.



Chapter

6

Monitoring Virtual Queues and Routing Points

This chapter describes how Interaction Concentrator monitors virtual queues and routing points, and stores this routing information in the Interaction Database (IDB). It discusses the two modes that Interaction Concentrator (ICON) supports for virtual queue data storage.

This chapter contains the following sections:

- [Monitoring Route Results on Virtual Queues, page 71](#)
- [Monitoring Route Results on Routing Points, page 75](#)

For information about ICON configuration and other Configuration Layer settings that are related to virtual queue functionality, see the chapter about special configuration requirements in the *Interaction Concentrator 8.0 Deployment Guide*.

Monitoring Route Results on Virtual Queues

ICON is capable of:

- Monitoring virtual queue objects that are configured in the contact center and that are used for routing purposes. The related data is provided to Interaction Concentrator by Universal Routing Server (URS) through T-Server.
- Storing, as separate records in a special table in IDB, associations between virtual queues and interactions that are being queued.

This section describes how Interaction Concentrator processes TEvents that pertain to a virtual queue, and also what virtual queue data Interaction Concentrator stores, and how.

Note: For detailed information about virtual queue data that is available in IDB, see the *Interaction Concentrator 8.0 Physical Data Model* document for your particular RDBMS.

Storage Modes

Interaction Concentrator supports two modes for virtual queue data storage:

- One-step storage
- Two-step storage

One-Step Storage

The one-step storage mode, which is the default, forces Interaction Concentrator to combine, into a single database transaction, all data for a single record that otherwise would be stored during separate steps for record creation and record update. In this mode, ICON creates a record in the `G_VIRTUAL_QUEUE` IDB table when the association between a virtual queue and an interaction ends—that is, after ICON receives either the `EventDiverted` or the `EventAbandoned` TEvent. This is the recommended storage mode, because it minimizes the number of inserts to IDB, thus improving performance. Keep in mind, however, that interactions must be promptly delivered from a virtual queue to the agents' DNs.

Two-Step Storage

In the two-step storage mode, ICON first creates a record after receiving `EventQueued`, and the ICON updates the record after receiving either `EventDiverted` or `EventAbandoned`. This storage mode is particularly useful in an environment where interactions remain in a virtual queue for long periods of time.

Data Processing Steps

For the purpose of explaining the details of virtual queue data processing, this section describes the two-step storage mode.

Step One—Record Creation

When an `EventQueued` TEvent arrives that pertains to a particular virtual queue that is configured to be monitored by Interaction Concentrator, a new row is

inserted into the `G_VIRTUAL_QUEUE` IDB table. The stored data includes information, taken from both the `TEvent` and Configuration Database, about:

- The interaction, in the form of a T-Server-reported `CallUUID`, which later identifies the original interaction for reporting purposes.
- The switch through which the interaction arrived, in the form of a database identifier that Configuration Server assigned to the corresponding `Switch` object, if available.
- The virtual queue at which the interaction is being queued, in the form of both the number reported in the `ThisDN` attribute of `EventQueued` and the database identifier that Configuration Server assigned to the DN object corresponding to this virtual queue.

In addition, Interaction Concentrator stores:

- The status of the association. The value is 8, which signifies `queued`.
- The cause of the change in the virtual queue state. The value is 1, which signifies `normal`.
- The time at which the association was created. The value is the time at which `EventQueued` arrived.

Step Two—Record Update

When either the `EventDiverted` or the `EventAbandoned` `TEvent` arrives and it pertains to the same virtual queue and to the same interaction for which a record has already been created, ICON updates that record in the `G_VIRTUAL_QUEUE` IDB table.

ICON updates the following data in the `G_VIRTUAL_QUEUE` IDB table:

- The status of the association, stored in the `STATUS` field, which changes to one of the following:
 - 13—Signifies `diverted`, if `EventDiverted` arrived with `CallState=0`, indicating that the call was diverted from this virtual queue.
 - 1—Signifies `connection cleared`, if either `EventDiverted` arrived with `CallState=22`, indicating that the call was diverted from another virtual queue, or `EventAbandoned` arrived for this virtual queue.
- The cause of the change in the virtual queue state, stored in the `CAUSE` field, which in the case of `EventDiverted`, is one of the following:
 - 1—Signifies `normal`. The interaction was routed to the target destination defined by the target selection object in the strategy.

- 3—Signifies stuck. The record was processed after the interaction was stuck in a virtual queue.

Note: To use the following cause-of-change values (101, 102, 103, 104, 105, 133, 134), the `extended-route-result` configuration option must be set to 1 on the `ICON Application` object to store this value in the `CAUSE` field. Universal Routing Server (URS) release 7.6 (or higher) and Interaction Server release 7.6.000.18 (or higher) are also required.

- 101—The interaction was routed in a parallel virtual queue to the target destination.
- 102—The interaction was routed by URS to the default destination as defined by the URS configuration options.
- 103—The interaction was routed by the switch to the default destination.
- 104—The interaction was cleared from the virtual queue by the URS strategy `ClearTarget` function.
- 105—Signifies other (not classified) causes reported by URS as others.
- 133—The routing interaction timeout, configured on Interaction Server, expired.
- 134—The interaction was removed (pulled out) from the strategy by Interaction Server.
- The cause of the change in the virtual queue state, stored in the `CAUSE` field, which in the case of `EventAbandoned`, changes to:
 - 2—Signifies abandoned.

ICON also adds the following data to the record:

- The identifier of the interaction that has been either diverted or abandoned, in the form of a T-Server-reported `CallUUID` that the interaction has on a physical device at the moment of distribution or abandonment, if available. This value later identifies the distributed or abandoned interaction for reporting purposes.
- The switch to which the interaction has been delivered or at which it was abandoned, in the form of the database identifier that Configuration Server assigned to the corresponding `Switch` object, if available.
- The DN to which the interaction is being distributed, in the form of a number reported in the `ThirdPartyDN` attribute of `EventDiverted`, if the interaction is diverted from this virtual queue and if the information about that DN is available.
- The time at which the association ended, which is equal to the time at which either `EventQueued` or `EventAbandoned` arrived.

- Information about the original interaction, the switch through which it arrived, the virtual queue, and the start time of the association remain unchanged at the time of update.

Monitoring Route Results on Routing Points

If configured to do so, URS distinguishes the routing results from interactions that are distributed from routing points or routing queues. ICON can then store these “extended” routing results, which are received from URS through T-Server, in the **RESULT** field of IDB table **G_ROUTE_RESULT** (see [Table 10](#)).

Note: To support the extended routing feature, certain URS and ICON configuration options must be set. For more information, see the section about configuring for storage of virtual queue data and extended routing results, in the chapter about special configuration requirements in the *Interaction Concentrator 8.0 Deployment Guide*.

Table 10: Summary of Values Stored in G_ROUTE_RESULT

Reporting Event	Value of RESULT Field	Description of Stored Results
When extended-route-result = 0 (ICON release 7.5 functionality)		
EventRouteUsed	1 (normal, ROUTE_RESULT_SUCCESS)	The call/interaction was routed.
EventAbandoned or EventPartyRemoved	2 (abandoned, ROUTE_RESULT_FAIL)	The call was abandoned or the interaction was removed.
When extended-route-result = 1 (ICON release 7.6 functionality)		
EventRouteUsed	1 (normal, ROUTE_RESULT_SUCCESS)	The call/interaction was routed by the URS strategy.
	102 (timeout)	The call was either routed to the default destination after the timeout expired, or function TRoute[DN] was called in the strategy to route the call to a specific DN.
	103 (routed by switch)	The call was routed by the switch to the default location.
EventAbandoned	2 (abandoned, ROUTE_RESULT_FAIL)	The call was abandoned.

Table 10: Summary of Values Stored in G_ROUTE_RESULT (Continued)

Reporting Event	Value of RESULT Field	Description of Stored Results
EventPartyRemoved	1 (normal, ROUTE_RESULT_SUCCESS)	The interaction was routed.
	2 (abandoned, ROUTE_RESULT_FAIL)	The interaction was stopped
	134	The interaction was pulled out by Interaction Server.
EventPartyRemoved (continued)	133	Routing timeout, defined on Interaction Server, expired.
	105	While URS attempted to route interaction, the connection between Interaction Server and URS was lost.

Reliability Flag

ICON uses a reliability flag stored in the GSYS_EXT_INT1 field in the G_ROUTE_RESULT table. This flag indicates the reliability of routing data stored in the G_ROUTE_RESULT table (see [Table 11](#)).

Table 11: Reliability Flag Values

Value	What Value Indicates for Voice Calls/Interactions
1 (ok)	Routing data stored in G_ROUTE_RESULT is valid.
2 (valid in past)	Routing data stored in G_ROUTE_RESULT is valid in the past.
0 (unknown)	There is no data in routing-related notifications about strategy targets chosen for routing.

For information about what routing data is stored in the G_ROUTE_RESULT table and how that data relates to values provided in T-Server notification events regarding routing (EventRouteUsed, EventAbandoned, and EventPartyRemoved), refer to the *Interaction Concentrator 8.0 Physical Data Model* for your RDBMS.

Virtual Queue DBID

If a virtual queue is involved in routing an interaction, and if URS provides the information, ICON stores the database identifier that is assigned to the virtual queue by Configuration Server (the DBID of the virtual queue) in the GSYS_EXT_VCH2 field in the G_ROUTE_RESULT table.

ICON obtains the DBID from the `RVQDBID` parameter in the call `UserData`. If a virtual queue is configured and reported by URS, URS attaches this parameter when it routes the call.

Note: Universal Routing Server (URS) release 8.0.000.18 or later is required in order to provide this data.



Chapter

7

Agent States and Login Sessions

Agent state data in Interaction Concentrator provides detailed information about agent activity, within the context of agent login sessions. This chapter describes how Interaction Concentrator (ICON) processes agent activity.

This chapter contains the following sections:

- [Agent and Login Session Models, page 79](#)
- [Available Agent State and Login Session Data, page 85](#)
- [After-Call Work and Not-Ready Agent States, page 87](#)
- [Populating Agent Login Session Data, page 88](#)

For information about ICON configuration and other Configuration Layer settings that make data about agent activity available in Interaction Database (IDB), see the chapter about special configuration requirements in the *Interaction Concentrator 8.0 Deployment Guide*.

Agent and Login Session Models

This section introduces the terminology, elements (objects), and models that pertain to Interaction Concentrator (ICON) data about agent activities.

This section contains information about the following:

- Agent and login session objects (see [page 80](#))
- The agent state model (see [page 81](#))
- The login sessions model (see [page 84](#))

Agent and Login Session Objects

The following terms refer to the agent and login session objects and elements about which ICON stores reporting data:

- *ACDQ*—An Automatic Call Distribution (ACD) queue object (ACD device or ACD group).
- *Agent state*—The state that an agent may take in relation to the interactions that are associated with the agent. For voice interactions, it is also the state that an agent may take in relation to an ACDQ.

ICON obtains information about agent state changes through agent state event reporting. ICON tracks agent states against different media types independently.

- *Login session*—The period of time during which the agent was logged in to an ACDQ or other endpoint on a particular switch.
- *Media type*—The communication medium applicable for the agent's interactions (for example, e-mail or chat).
- *Pending agent state*—The agent state to which the agent will transition after the Busy state.
 - When the agent transitions from any state to Busy, ICON keeps the pre-transition state as the pending state.
 - If ICON receives information about an agent state change while the agent state is Busy, ICON keeps the new state as the pending state.
 - When the Busy state ends, ICON restores the agent's state from the pending state.
- *Reason*—The reason for the agent state change.

For voice calls, ICON obtains reasons for agent states from T-Server reporting, from any of the following sources:

- The workmode parameter
- TEvent Attribute Extensions (known as hardware reasons), from the value of predefined ReasonCode keys
- TEvent Attribute Reason (known as software reasons), from the values of one or more key-value pairs

For eServices/Multimedia or 3rd Party Media interactions, the Interaction Server Multimedia Reporting events may include attributes that contain information about the reason for the agent state change. ICON stores this reason code information in the same way that it stores hardware reasons. There can be only one reason for each agent state-related event that Interaction Server reports.

Agent State Model—Voice and Multimedia

Agent-Orientated Agent State Model Interaction Concentrator supports an agent-orientated agent state model. In this model, the switching function maintains a single state for the agent for each media type (for example, voice, e-mail, or chat) within the agent login session, regardless of the number of endpoints or ACDQ objects with which the agent is associated. The agent can have a different state in relation to each media type.

Agent States Table 12 lists the agent states, described in relation to voice calls. The ACDQ is not applicable to multimedia, but the agent states for multimedia media types are equivalent. Table 12 also provides the T-Server or Interaction Server (MM) reporting events that represent entry into each state.

Table 12: Agent States and State Transition Events

Agent State	Description / State Entry Event
Null	<p>The agent is not logged on to the ACDQ at a particular device. Logging on to the ACDQ causes a transition from this state. Similarly, logging off from the ACDQ causes a transition to this state.</p> <p>TEvent</p> <ul style="list-style-type: none"> EventAgentLogout—Reports the agent state change for all ACDQ objects to which the agent was logged on. EventQueueLogout—Reports the agent state change for a particular ACDQ. <p>MM Event</p> <ul style="list-style-type: none"> EventAgentLogout—Reports the agent state change for all endpoints and media types at the place to which the agent was logged on. EventMediaRemoved—Reports that the agent is no longer logged on for the specified media type.
Login	<p>The agent is logged on to the ACDQ at a particular device and is ready to contribute to the activities of the ACD queue. The state does not indicate that the agent is ready to accept ACD calls (see Ready).</p> <p>TEvent</p> <p>EventAgentLogin</p> <p>Note: The presence or absence of the <code>ThisQueue</code> parameter in the TEvent specifies whether the agent has logged on to an ACD queue.</p> <p>MM Event</p> <p>EventAgentLogin</p> <p>Note: If the event includes information about media types, a separate record is created for each media type, and the appropriate agent state for each media type is set.</p>

Table 12: Agent States and State Transition Events (Continued)

Agent State	Description / State Entry Event
NotReady	<p>The agent is logged on to the ACDQ at a particular device but is not ready to handle interactions distributed from the ACD queue. An agent in this state can receive calls that are not ACD calls.</p> <p>TEvent EventAgentNotReady, with the workmode parameter not equal to AgentAfterCallWork</p> <p>MM Event • EventNotReadyForMedia • EventMediaAdded—Provides information about a media type that was added to the agent’s login session.</p> <p>Note: ICON obtains information about the agent state from the event attribute. In Multimedia, the agent state specified for the EventMediaAdded event is always NotReady.</p>
Ready	<p>The agent is logged on to the ACDQ at a particular device and is ready to handle ACD interactions, even though the agent might be involved in non-ACD calls.</p> <p>TEvent EventAgentReady</p> <p>MM Event EventReadyForMedia</p>
Busy	<p>The agent is logged on to the ACDQ at a particular device and is involved in an existing call at the device. The call may be on hold at the device.</p> <p>There is no specific entry event for this agent state. This state covers the period during which the agent is involved with the interaction, until the agent is disconnected.</p> <p>In addition to ACD interactions, calls between agents, calls between supervisors and agents, and private calls can also cause transition to Busy state.</p>
ACW	<p>The agent is no longer connected to a call but is still occupied with work related to the previous call (for example, the agent might be performing administrative duties, such as updating a business order form). In this state, the agent cannot receive ACD calls but may be able to receive non-ACD calls.</p> <p>Note: ICON cannot obtain call information from the T-Server event. ICON keeps information about the last call represented on the agent’s device before the transition from Busy state to any other state. In cases where ACW state follows Busy state, ICON assigns the last call on the agent’s device as related to ACW state.</p> <p>TEvent EventAgentNotReady, with the workmode parameter equal to AgentAfterCallWork</p> <p>MM Event Not applicable</p>
Unknown	<p>An agent’s login session exists, but ICON has no information about the agent state (for example, because of disconnection from T-Server).</p>

Note: Interaction Concentrator release 7.5 and higher also supports custom agent states. For more information, see Chapter 9 on [page 99](#).

Agent State FSM

Figure 3 illustrates the finite state machine (FSM) for the agent state for voice calls, within a login session. Except for the *AfterCallWork* state and the names of the state transition events, the FSM for the agent state for multimedia is identical.

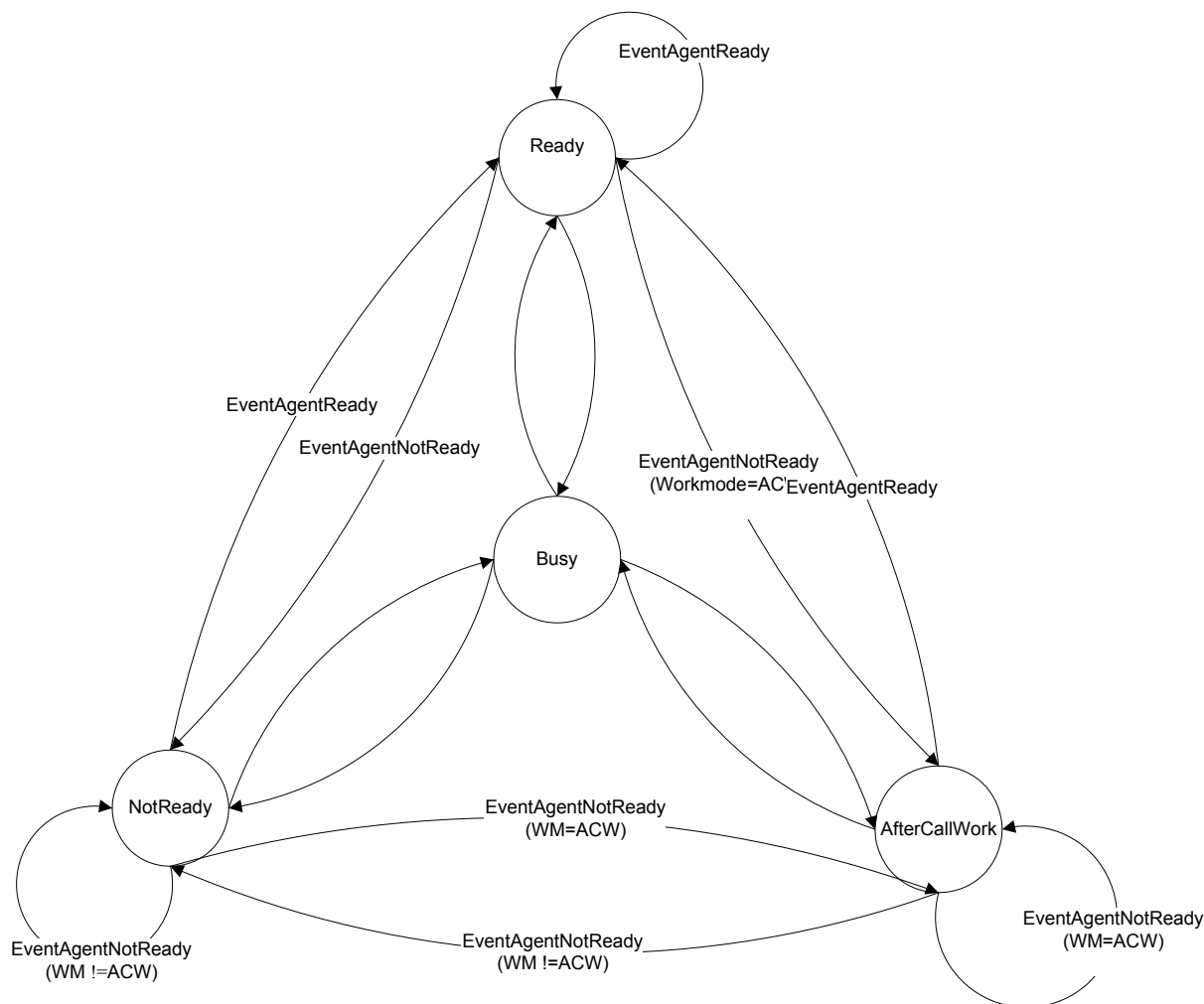


Figure 3: Agent State FSM

Agent State Restoration

On startup or reconnection, when ICON registers with T-Server to start monitoring agent states, the *EventRegistered* TEvent provides ICON with a snapshot of the current agent state for voice calls, as well as a list of the calls that were distributed by T-Server and presented on the agent's desktop.

In an eServices/Multimedia solution, when ICON registers with Interaction Server to start monitoring agent states, the *EventPlaceAgentState* reporting event similarly provides ICON with a snapshot of the current agent states for

the various media, as well as a list of the interactions that were distributed by Interaction Server and presented on the agent's desktop.

Agent State Model—SIP Chat

ICON processes the `chat` media type in agent state-related events received from SIP Server. Data extracted from these events is stored in existing IDB tables. Within these tables, the media type is stored as an integer code in the `GSYS_EXT_INT1` field.

State-Related Events

Interaction Concentrator supports a simple agent model within each media type. An agent on a DN can have a different state for each media type (for example, the agent state can be `Busy` for voice but `Ready` for chat).

SIP Server does not provide information about the media type in the agent's state-related events (`EventAgentLogin`, `EventAgentReady`, `EventAgentNotReady`, `EventAgentLogout`). As result, ICON cannot determine the media type from an agent's state-related event and therefore processes information from received events as applicable to *all* media types associated with agent's login session.

Party State Changes

Interaction Concentrator can determine the media type from the calls related to the call party. ICON will process party state changes against agent's state created for this media type only. Agent's state for any other media types is not affected by this party state change.

DND State Changes

SIP Server does not provide information about the media type in the `EventDNDOn` and `EventDNDOff` events. Interaction Concentrator processes and stores information about changes in DND states regardless of the media type. If more than one media type is configured on a DN, and the agent state is different for each media type, the agent state in the related record will be `LoggedIn`.

ThisQueue

Interaction Concentrator processes and stores information about the `ThisQueue` attribute for voice media interactions. If the voice media type is disabled on a DN, this attribute will be processed for the chat media type instead.

Login Sessions Model

An agent's login session starts when ICON receives the first `EventAgentLogin` event for that agent, either after a period of time during which the agent was not logged on to the switch at all, or after a configurable period of time during which information about the agent was not available (see the description of the `gls-max-inactivity` configuration option in the *Interaction Concentrator 8.0 Deployment Guide*).

If an agent is already logged on to the switch and then either logs on to additional ACDQ or endpoint objects, or a media type is added or removed, ICON continues the existing login session.

An agent's login session ends when the agent is no longer logged on to the ACDQ, or if there has been no agent-related activity within the configured maximum inactivity interval.

Figure 4 represents the login session FSM.

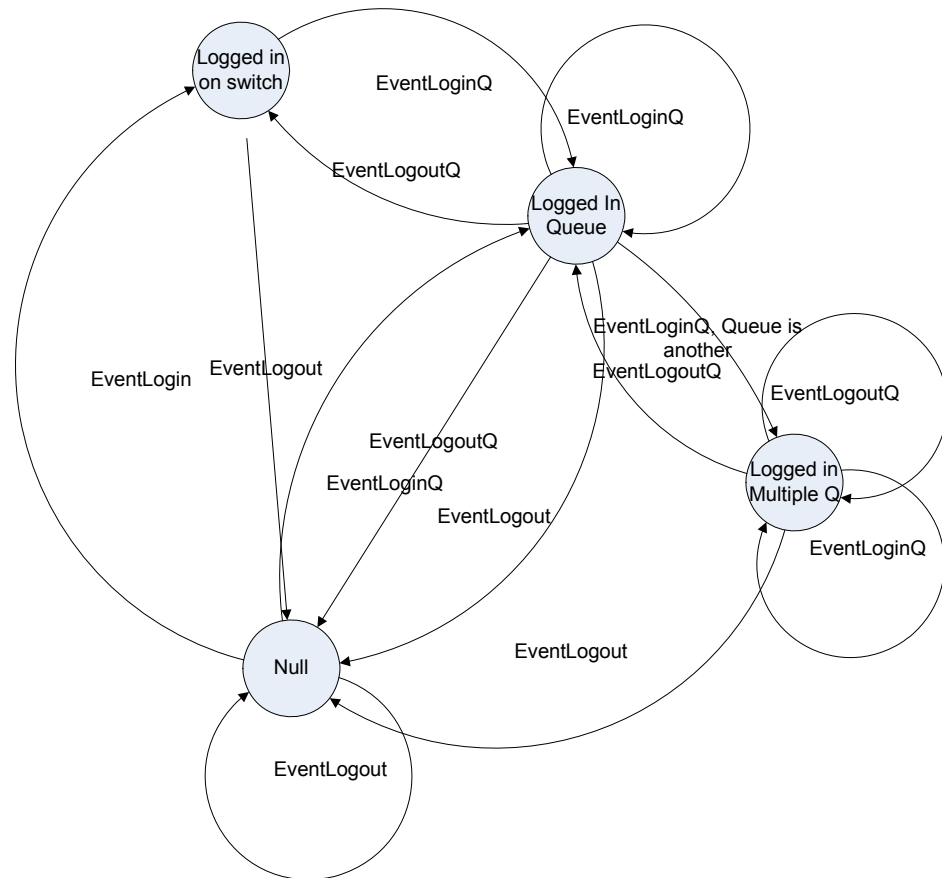


Figure 4: Login Session Finite State Machine

Available Agent State and Login Session Data

This section describes the kinds of agent state and login session data that ICON captures, provided that ICON has been configured to perform this role.

For a high-level summary of the IDB tables in which ICON stores data about agent states and login sessions, see “Agent State–Related and Login Session–Related Tables” on [page 30](#). For more detailed information, see the *Interaction Concentrator 8.0 Physical Data Model* document for your particular RDBMS.

Agent State and Login Session Data

ICON stores both actual and historical data about login sessions and agent state changes, as well as the associations between agent states, login sessions, and endpoints. ICON tracks changes against different media types independently.

The following data is available:

- Current and historical information about agent login sessions.
- Current and historical information about the associations between login sessions and endpoints.

Note: Multimedia Reporting events include information about the agent place, but not about agent endpoint DNs. Therefore, ICON records in the agent activity–related tables do not include meaningful endpoint information for multimedia interactions.

- Detailed information about agent state changes during the agent login session, including:
 - Changes to the agent’s state, pending state, workmode, and hardware reason code.
 - Time of the state change.
 - Other party connections and disconnections.
- Detailed information about the hardware and software reason code changes during the agent login session.

Note: If Multimedia Reporting events include information about reason codes for agent state changes, ICON stores the reason code in the same way that it stores hardware reason codes that are provided by T-Server reporting for voice. Hardware reason codes do not apply to multimedia.

- Detailed information about usage of the DND feature within the agent login session. For voice, DND can be activated and deactivated for individual DNs. For multimedia, DND can be activated and deactivated only for an entire place.

Precalculated Agent State Metrics

In addition to raw object data, ICON stores the following agent state–related statistics:

- Duration of agent state:
 - `Duration_ready`
 - `Duration_notready`
 - `DurationACW` (not applicable to Multimedia)
 - `DurationBusy`

- Duration of agent workmode (not applicable to Multimedia):
 - `Duration_UNKNOWN`
 - `Duration_AUX`
 - `Duration_LegalGuard`
 - `Duration_GoAway`
 - `Duration_ReturnBack`

After-Call Work and Not-Ready Agent States

After-call work (ACW) is the work that is required of an agent immediately following an inbound call, such as entering data, or making internal calls to complete the transaction. The agent is considered unavailable to receive another inbound call while in this mode. This section describes the functionality of the after-call work and not-ready agent states in Interaction Concentrator.

Uninterrupted ACW or Not-Ready Duration

Interaction Concentrator provides the capability to continue the after-call work or NotReady agent state without any interruption due to the agent placing or receiving calls during the after-call work or not-ready period. ICON recognizes completion of after-call work or the NotReady agent state when any of the following occur:

- The agent logs out.
- The agent places himself/herself in Ready mode.
- The agent goes NotReady for any reason other than to perform after-call work. (This includes indirect work mode changes such as when the agent walks away from his desk for a period of time.)

To enable this uninterrupted ACW or NotReady agent state functionality, the `gls-enable-acw-busy` configuration option must be set to `false` on the switch Annex tab.

Interrupted ACW or Not-Ready Duration

ICON also supports the capability to interrupt the after-call work or not-ready agent state when the agent places or receives calls during the ACW or NotReady period. By default, ICON interrupts ACW or NotReady agent states while an agent is handling another call. You can also set the `enable-acw-busy` configuration option to `true` on the switch Annex tab to enable this behavior. See the configuring options chapter in the *Interaction Concentrator 8.0 Deployment Guide* for more information on setting this option.

Associating ACW with an Interaction

ICON can associate after-call work with the voice interaction that immediately precedes the *start* of the after-call work (the first voice interaction), or with the last interaction, which is the default behavior.

In the first case, subsequent voice interactions that occur during the period of after-call work are considered as related to ACW processing and do not interrupt measurement of ACW-related metrics.

To support this functionality, the `gls-acw-first` configuration option can be set either at the switch level or on the `ICON Application`. ICON uses the value set on the `Application` for all switches except the SIP switch. For SIP switches, the option must be specified on the switch level. See the configuring options chapter in the *Interaction Concentrator 8.0 Deployment Guide* for more information.

Populating Agent Login Session Data

ICON instances that perform the `gcc`, `gls`, and `gud` roles maintain a persistent cache for agent login session data. A configuration option, `agent-pstorage-name`, enables you to specify the name of this persistent cache. The default file name is `apstorage.db`.

When it receives data about login sessions, ICON writes the data from its in-memory queue to the persistent cache as well as to the persistent queue.

- If the reported `AgentSessionID` does not already exist in IDB or the persistent cache, ICON writes the data from the persistent queue into the `G_LOGIN_SESSION` table in IDB.
- If the reported `AgentSessionID` already exists in IDB or the persistent cache:
 - In high availability (HA) deployments, the login session is considered to be a duplicate and is discarded.
 - In non-HA deployments, the existing login session is marked as closed (for example, the reason is `stuck`), and a new session is created.

Data in the persistent cache survives a shutdown and restart of ICON.



Chapter

8

Integrating with Outbound Contact

This chapter describes how Interaction Concentrator (ICON) processes data from the Genesys Outbound Contact solution.

This chapter contains the following sections:

- [Outbound Objects and Models, page 89](#)
- [Available Outbound Data, page 91](#)
- [Outbound Contact Deployment Scenarios, page 95](#)
- [Extracting Outbound Contact Data, page 97](#)

For information about ICON configuration and other Configuration Layer settings that make data about Outbound Contact activity available in Interaction Database (IDB), see the chapter about special configuration requirements in the *Interaction Concentrator 8.0 Deployment Guide*.

Outbound Objects and Models

This section introduces the terminology, elements (objects), and models that pertain to Genesys Outbound Contact.

Outbound Contact Objects

The following terms refer to the Outbound Contact objects and elements about which ICON stores reporting data:

- *Campaign*—A master plan for managing customer data, generating calls, and monitoring and handling call results. Outbound Contact uses campaigns to automate outbound dialing.

- *Campaign Group*—A runtime association among a campaign, a calling list, and a group of resources (such as agents) that is assigned to handle the campaign activity.
- *Calling List*—A database table made up of records that contain customer data, which could have dialing filters applied.
- *Record*—The basic unit of customer data. Each record represents one customer phone number.
- *Field*—A single piece of data (for example, a phone number) within a record. A calling list includes mandatory fields, which are required in order for the Outbound Contact solution to operate. A calling list can also include custom fields.
- *Chain*—Records that are logically united, usually representing the same customer contact information. For example, three records with the home, business, and cell phone numbers for the same customer comprise a chain of records.
- *Format*—A template that organizes the data in a calling list. When calling lists are created and populated with records, the customer data in the records fills the fields and conforms to the field properties that are established in the format, according to either the system default setting or a user-defined setting.

Note: For more detailed information about Outbound Contact objects, see the *Outbound Contact 8.0 Deployment Guide*.

Campaign Model

Figure 5 shows the finite state machine (FSM) for a campaign group.

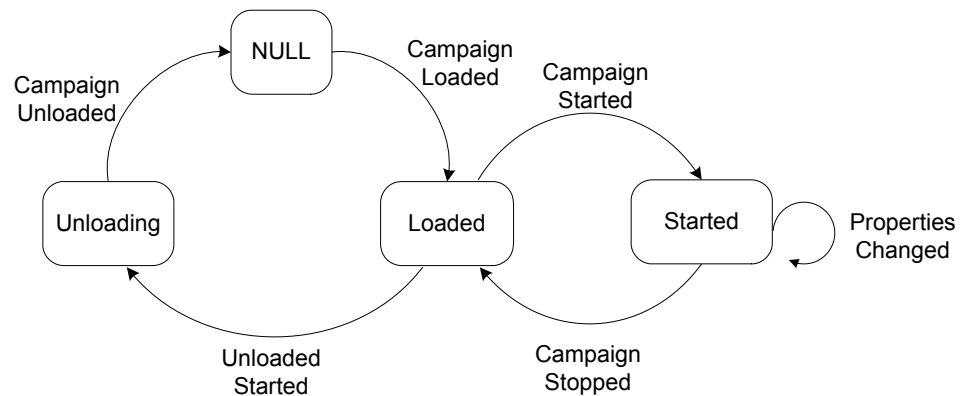


Figure 5: Finite State Machine for a Campaign Group

Chain Model

Figure 6 shows the FSM for a chain.

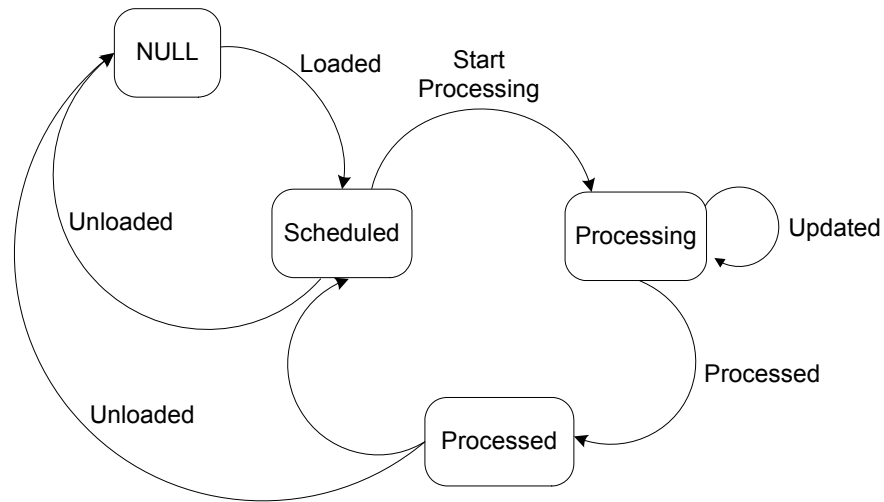


Figure 6: Finite State Machine for a Chain

Chain States In Figure 6, the chain states are as follows:

- *NULL*—The initial state, in which no records for this chain have yet been selected from the database. This is also the final state, in which all records in the chain are either finalized or unloaded, and updated in the database.
- *Processing*—The state in which the processing of a record from the chain has started, and resources for this processing have been allocated.
- *Processed*—The state in which a record is processed. Depending on the actual processing result, the record(s) in the chain are finalized or rescheduled.
- *Scheduled*—The state in which a record in the chain is scheduled for processing, either because the chain has been loaded from the calling list or because of the previous processing attempt. The chain can be scheduled for processing either at a specified date and time or *for now*—that is, as soon as resources for processing the chain are available to Outbound Contact Server (OCS).

Available Outbound Data

This section describes how ICON communicates with Outbound Contact Server (OCS) and what kind of data OCS can send to ICON, provided that both OCS and ICON are properly configured.

ICON and OCS Communications

ICON connects to OCS at the port that OCS opens for regular client connections. Unlike other OCS clients, ICON uses a special protocol to receive reporting-related data. The OCS data includes runtime information about OCS objects, as well as certain statistics that OCS calculates specifically for reporting purposes.

If two or more OCS instances are running simultaneously in a given contact center, a single ICON instance can process and store data from either a single OCS instance or multiple OCS instances.

Outbound Objects Data

ICON stores campaign and chain information, including both their history within a given campaign session and the associations between calls and parties, when this data is available.

Specifically, you can integrate ICON with OCS so that the following data is stored in IDB for outbound reporting purposes:

- History of changes in campaign configuration properties, and the target values of optimization parameters.
- History and results of chain processing.
- History of changes that OCS makes in calling list records during chain processing, and that ICON is configured to track.
- Results of each attempt to generate an outbound call, whether successful or not.
- Call progress detection (CPD) call results.
- Call results assigned by an agent, which are stored independently from the CPD call results, and which do not overwrite them.

Note: For the full list and descriptions of IDB tables that store outbound data, see the *Interaction Concentrator 8.0 Physical Data Model* document for your RDBMS.

Precalculated OCS Metrics

In addition to raw object data, you can configure OCS to calculate, and ICON to store, a set of outbound-related statistics, which are also referred to as *precalculated metrics*.

[Table 13](#) lists the precalculated metrics that OCS provides to ICON for storage in the GO_METRICS IDB table.

Table 13: Precalculated OCS Metrics

Metric	Description
OCS calculates the following subset of metrics for all currently active campaign groups, and for all calling lists that participate in the active campaign groups. OCS delivers these metrics to ICON in a single data packet every 10 minutes.	
Total number of records per calling list	Number of physical records in the calling list database view (for example, records from the database table and, possibly, the WHERE clause from the dialing filter).
Total number of records per campaign group	Number of physical records in all database views whose calling lists are assigned to the campaign that participates in the campaign group.
Total number of chains per calling list	Number of logical chains in the calling list database view (for example, records from the database table and, possibly, the WHERE clause from the dialing filter). A <i>logical chain</i> is defined as one or more database table records with the same Chain ID value.
Total number of chains per campaign group	Number of logical chains in all database views, whose calling lists are assigned to the campaign that participates in the campaign group.
Current number of records not processed per calling list	Number of physical records in the calling list database view (for example, records from the database table and, possibly, the WHERE clause from the dialing filter) that have the type GENERAL and the status READY.
Current number of records not processed per campaign group	Number of physical records in all database views that have the type GENERAL and the status READY, and whose calling lists are assigned to the campaign that participates in the campaign group.
Current number of chains not processed per calling list	Number of logical chains in the calling list database view (for example, records from the database table and, possibly, the WHERE clause from the dialing filter), in which at least one record of the chain has the type GENERAL and the status READY.
Current number of chains not processed per campaign group	Number of logical chains in all database views, in which at least one record of the chain has the type GENERAL and the status READY, and whose calling lists are assigned to the campaign that participates in the campaign group.
Current number of busy dialer ports per campaign group	Current number of busy (that is, occupied with outbound calls that are in the process of being placed) CPD Server ports or Switch call classifier ports.

Table 13: Precalculated OCS Metrics (Continued)

Metric	Description
Current number of engaged (busy) ports per campaign group	<p>Current number of ports used by engaged calls that CPD Server already placed.</p> <p>Note: OCS calculates this metric only for campaign groups that are activated in Active Switching Matrix (ASM) dialing modes with CPD Server.</p>
Outbound Calls Overdialed	<p>Indicates that Outbound Dialing Engine is to consider a given call overdialed.</p> <p>For information about overdialed calls, see the Outbound Contact documentation.</p> <p>Note: OCS calculates this metric for all currently active campaign groups. When a call is overdialed, OCS delivers this metric to ICON. OCS does not deliver this metric at a preset interval, but instead calculates it and passes it to ICON on a <i>per-occurrence</i> basis (that is, for each overdialed call).</p>
<p>OCS calculates the following subset of metrics for all currently active campaign groups. As a result of each successful dial attempt, OCS delivers these metrics to ICON in the form of a single snapshot event. OCS does not deliver these metrics at a preset interval, but instead calculates them and passes them to ICON on a <i>per-occurrence</i> basis (that is, for each successfully dialed outbound call).</p> <p>Note: OCS bases calculations related to call times on FTC trail (audit log) calculations. To enable OCS to provide the following metrics, ensure that audit logging has been enabled in OCS (the <code>log_call_stats</code> option is set to true or yes).</p>	
Outbound Call Dialing Time	<p>The time, in milliseconds, between (a) initiation of dialing and (b) the moment when the call was answered by the called party or when a call that did not reach the called party was released.</p> <p>Note: The time that the call was answered by the called party is available only when CPD Server is used for dialing. Therefore, if calls are not dialed through CPD Server, this metric will be available only for unsuccessful calls.</p>
Outbound Call Transfer Time	<p>The time, in milliseconds, between completion of CPD and the moment when the call was established on the Agent or IVR DN.</p> <p>Note: The time that CPD completed is available only when CPD Server is used for dialing. Therefore, if calls are not dialed through CPD Server, this metric is not available.</p>
CPD Time	<p>The time, in milliseconds, from the moment when the call was answered by the called party until the moment when CPD was done.</p> <p>Note: Both time stamps are available only when CPD Server is used for dialing. Therefore, if calls are not dialed through CPD Server, this metric is not available.</p>

Outbound Contact Deployment Scenarios

Interaction Concentrator supports all types of Outbound Contact deployments and, most importantly, stores outbound data consistently, regardless of the deployment scenario.

This section discusses two main deployment scenarios:

- Single-site deployment in which one ICON instance is dedicated to handling outbound data.
- Multi-site deployment in which a single OCS instance serves all sites, and in which there is a single ICON instance for each site.

Diagram Conventions

To simplify the deployment diagrams in this section:

- DB Server, which enables a connection between ICON and IDB, is omitted from the diagrams, although it is required in actual deployments (see Figure 1 on [page 18](#)).
- Storage of configuration data is not shown, although it is required in actual deployments.

Single-Site Deployment Example

The simplest deployment scenario, which is suitable for single-site contact centers, consists of a single ICON instance that is dedicated to storing all outbound data to a single IDB instance (see [Figure 7](#)).

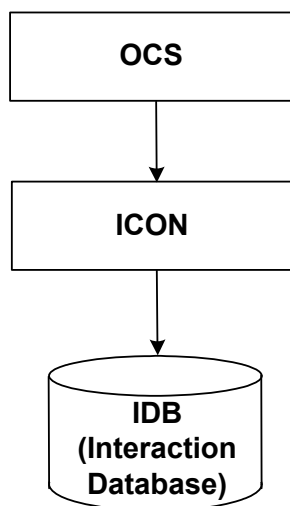


Figure 7: Single-Site Deployment: A Single OCS Instance and a Dedicated ICON Instance

In this scenario, ICON is dedicated to storing outbound data, including both object information and precalculated metrics. This ICON instance does not have a connection to T-Server or, in a Multimedia solution, to Interaction Server, nor does it gather interaction data from T-Server or Interaction Server.

It is assumed that another ICON instance stores CTI-related data and interaction data to either the same IDB or a separate IDB.

Multi-Site Deployment Example

The simplest multi-site deployment scenario consists of ICON instances that run at various contact center sites. All of these ICON instances store outbound data that they receive from the same OCS instance, and each of them also stores CTI data that it receives from its particular T-Server or Interaction Server.

Figure 8 shows a simple multi-site deployment scenario. To simplify the diagram, the scenario shows a deployment for voice calls only, and it omits the IDB deployment. The relationship between the OCS instance and the Interaction Server in a Multimedia solution is the same as the relationship between the OCS instance and the T-Server in a voice solution.

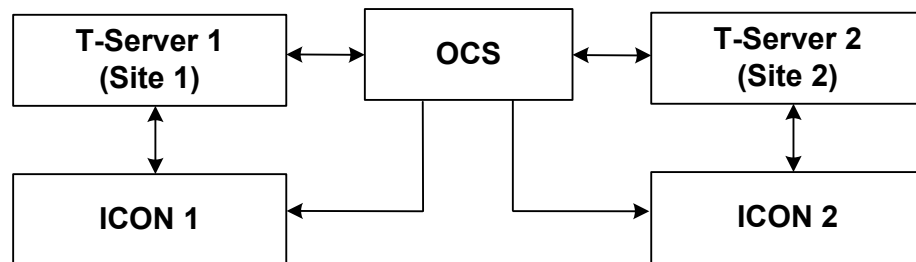


Figure 8: Multi-Site Deployment: A Single OCS Instance and Multiple ICON Instances

In this scenario:

- ICON 1 stores:
 - Data related to outbound activity generated by OCS at Site 1 only.
 - CTI data related to call activity reported by T-Server 1, which is serving Site 1.
- ICON 2 stores:
 - Data related to outbound activity generated by OCS at Site 2 only.
 - CTI data related to call activity reported by T-Server 2, which is serving Site 2.

Note that if a call originates at Site 1 but it is being handled by an outbound agent at Site 2, ICON 1 reports the outbound-specific data, whereas ICON 2 reports the voice (CTI) activity that is associated with this call.

Matching Outbound and CTI Data in IDB

To match CTI or interaction data (reported by T-Server or Interaction Server) and outbound data (reported by OCS) for the same interaction, use the call attempt identifier (CallAttId) that ICON stores in the GOX_Chain_Call table.

Note: This functionality is not available with Outbound Contact Server (OCS) release 7.2.

OCS Reporting Limitation

OCS is not able to track certain types of outbound calls when the Campaign Group is running in the following dialing modes:

- Preview mode—Calls that are dialed by the Agent Desktop
- Power GVP mode—Calls that are dialed by GVP Dialer
- Push Preview mode—Interactions that are pushed to the Agent Desktop from Interaction Server

Chain-related events for these types of calls do not contain the unique identifier (Call GUID) of the outbound call(s) dialed during chain processing. As a result, ICON does not receive the information it needs to create a record in the GOX_Chain_Call table.

Workaround As a workaround, you can use the call attempt GUID to bridge data about outbound chain activity with data about telephony activity. OCS reports the call attempt GUID in chain-related events, and T-Server and Interaction Server include it in the user data of outbound calls.

Extracting Outbound Contact Data

To extract Outbound data from IDB, Genesys recommends an approach similar to that used for voice data interactions:

1. Use records from the GO_CAMPAIGN and GO_CHAIN tables as the base records. The records from these two tables can be extracted independently.

Note: Only terminated records are considered for extraction.

Use a combination of the values in the following fields as the unique record identifier:

SessID	OCSID
CallingListID	RECORDHANDLE
CHAINGUID	TYPE
CALLATTID	ADDED_TS

- | | |
|--|--|
| Extraction of Campaign-Level Data | 2. Use the value of the SessID field as a unique identifier to extract terminated records from the GO_CAMPAIGN, GO_CAMPAIGNHISTORY, and GO_CAMPPROP_HIST tables. |
| Extraction of Chain-Level Data | 3. Use the value of the CHAINGUID field as a unique identifier to extract terminated records from the GO_CHAIN, GO_CHAINREC_HIST, GO_RECORD, GO_CUSTOM_FIELDS, GO_FIELDHIST, GO_SECURE_FIELDS, GO_SEC_FIELDHIST and GO_CHAIN_CALL tables. |
| Extraction of Precalculated Metrics | 4. OCS precalculates some metrics and sends this data to ICON. OCS does not provide a unique identifier for these metrics, but you can use the object's identifier (for the metric calculated) and the metric's timestamp to identify records. |



Chapter

9

Processing User Events and Custom-Defined States

Interaction Concentrator support for customer-defined states is designed to provide compatibility with Call Concentrator functionality in legacy deployments. This feature enables customers to use existing investments in customized desktop applications.

This chapter describes how Interaction Concentrator (ICON) processes user data from User Events. It contains the following sections:

- [Custom States in Interaction Concentrator, page 99](#)
- [Storing Data from EventUserEvent, page 100](#)

For information about ICON configuration and agent desktop application settings that make data about custom states and common data available in Interaction Database (IDB), see the chapter about special configuration requirements in the *Interaction Concentrator 8.0 Deployment Guide*.

Custom States in Interaction Concentrator

The term *custom states* refers to customer-defined states of endpoints. If it has been configured to do so, Interaction Concentrator (ICON) supports the processing of data from the T-Server EventUserEvent, in order to store associations between:

- Common data and the voice call (or the call party, when applicable).
- Custom states and the voice call (or the call party, when applicable).

Both an active call and the last call (the call that ended immediately before the start of the custom state or the arrival of EventUserEvent) on the device can participate in the association.

Storing Data from EventUserEvent

ICON processes data from EventUserEvent separately from call user data.

Common Data If it has been configured to support custom states, ICON stores common data from EventUserEvent in two tables that are created by the ICON installation script: G_CUSTOM_DATA_S and G_CUSTOM_DATA_P. You can configure unique keys to store values of customer-defined keys. Duplicate key names in the attached data is not supported.

Note: Interaction Concentrator release 7.5 and 7.6 do not support storage of common data in customer-created tables or additional customer-created fields.

Custom States ICON stores data related to custom states from EventUserEvent in the G_CUSTOM_STATES table, which is created by the ICON installation script. ICON writes information to IDB when the custom state is finished. To avoid problems arising from stuck custom states, ICON clears all active custom states when the agent's login session is terminated.

For more information about the custom data and custom state tables, see the *Interaction Concentrator 8.0 Physical Data Model* document for your particular RDBMS.



Part

2

Implementing High Availability

This part of the document contains information about the high availability (HA) model in Interaction Concentrator, and about how you can implement HA for Interaction Concentrator.

This information appears in the following chapters:

- Chapter 10, “The Interaction Concentrator HA Model,” on [page 103](#)
- Chapter 11, “Extracting Data in an HA Deployment,” on [page 109](#)

For information about ICON configuration and other Configuration Layer settings that are required in an HA deployment, see the chapter about special configuration requirements in the *Interaction Concentrator 8.0 Deployment Guide*.



Chapter

10

The Interaction Concentrator HA Model

This chapter describes the design and implementation of high availability (HA) in Interaction Concentrator. It describes the Interaction Concentrator HA model and the consequences of dropped server connections and other failures as they relate to data consistency in Interaction Database (IDB).

This chapter contains the following sections:

- [Overview, page 103](#)
- [ICON HA Model, page 104](#)
- [Consequences of Failures, page 106](#)

Overview

You can deploy Interaction Concentrator in an HA configuration to ensure redundancy of your reporting data for voice and multimedia interactions, voice attached data, agent states and login sessions, virtual queues, and Outbound Contact solution details.

In an HA configuration, if a component or network outage prevents one of the Interaction Concentrator (ICON) processes from storing data in its IDB, data is not lost as long as the other ICON process is able to store those interactions in its IDB.

A downstream reporting application such as Genesys Info Mart can continue to extract, transform, and load interaction, voice attached data, virtual queue, agent-related, and Outbound Contact data, as long as data is available in at least one of the IDBs in the HA pair.

Furthermore, starting with release 8.0, the ICON processes track and store control information about connections and data source events. Downstream reporting applications can use this information to determine the availability and reliability of data in HA pairs of IDBs for a given timeframe, so that they

can determine the best IDB to use as the source of reporting data for that time period.

ICON HA Model

Different from Standard Genesys HA Model

The HA model used in Interaction Concentrator differs significantly from the Genesys standard HA model that is implemented in a majority of Genesys servers.

Unlike a typical Genesys server (for example, T-Server), two Interaction Concentrator instances (ICONS) that are deployed as a redundant (HA) pair do not operate in either primary or backup mode, nor does their mode switch from backup to primary when the other server fails. Rather, both ICONs in the HA pair operate in parallel as stand-alone servers and process incoming data independently.

A redundant pair of ICONs receives interaction-related events from the same T-Server, Interaction Server, or Outbound Contact Server (OCS)—either a stand-alone server or the primary server from a redundant pair—and stores data in two independent IDBs. The downstream reporting applications are responsible for managing extraction of reliable data.

In addition to data from interaction-related events, the ICON HA pair also store configuration and connection-related information from Configuration Server. Downstream reporting applications can use this information to determine the availability and reliability of IDB data for a given timeframe.

Figure 9 on [page 105](#) illustrates the major components of a full Interaction Concentrator HA solution.

Note: For simplicity, DB Servers and the connections between Configuration Server and the other Genesys components are not included in Figure 9. The Network T-Servers shown in the diagram are operating in load-balancing mode.

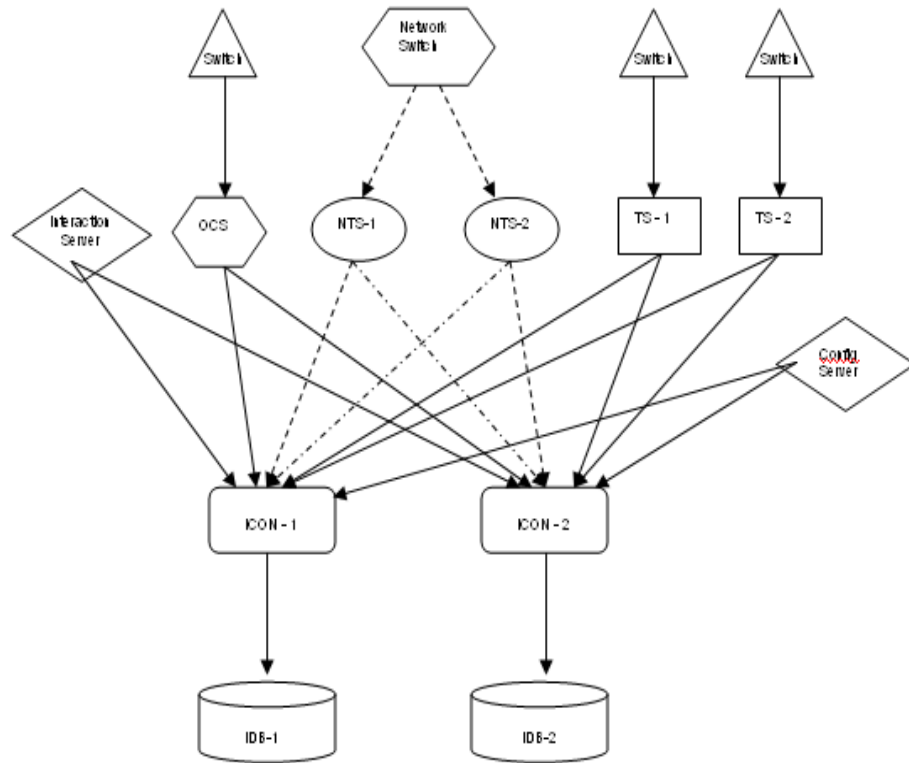


Figure 9: ICON HA Model

Sources of Data Interruption

Data source session control tables in IDB enable ICON and the downstream reporting application to identify and handle the following interruptions:

- On the data source server side—T-Server or Interaction Server, Outbound Contact Server (OCS), and Configuration Server, depending on the ICON role:
 - Disconnection of the data source server from ICON
 - Shutdown of the data source server
 - No interaction activity on T-Server/Interaction Server or OCS
 - Reconnection of T-Server to the switch or OCS to T-Server
- On the ICON side:
 - Disconnection of ICON from its data sources
 - Shutdown of ICON
 - Disconnection of ICON from IDB

- On the IDB side:
 - RDBMS server disconnection
 - RDBMS server shutdown

Note: On the IDB side, connection problems between the Genesys DB Server and the RDBMS server are a potential source of failure that is not covered in the Interaction Concentrator release 8.0 HA design.

For more information about the data source session control tables, see “Data Source Session Control Tables” on [page 34](#).

For more information about using the data source session control tables to identify data interruptions, see “Determining Data Availability and Reliability” on [page 49](#) and “Extracting HA Data” on [page 109](#).

Consequences of Failures

Dropped server connections and other failures that interrupt data result in data inconsistencies in IDB. [Table 14](#) summarizes the consequences of these kinds of failures in an ICON HA configuration.

Table 14: IDB Inconsistencies Resulting from Failures

Point of Failure	Resulting Data Inconsistency
ICON connection to T-Server	<ul style="list-style-type: none"> • Incorrect party transitions from ICON’s point of view resulting in missing transition or party error records • Calls deleted at unknown times • Missing information about: <ul style="list-style-type: none"> • Agent states • Attached data • Changes in attached data • Parties • Call linkages
Computer-telephony integration (CTI) link	<ul style="list-style-type: none"> • Stuck calls • Incorrect transitions in call- and agent-related events • Stuck parties—parties which ICON reports as active (that is, not deleted) when the associated call interaction has in fact been deleted

Table 14: IDB Inconsistencies Resulting from Failures (Continued)

Point of Failure	Resulting Data Inconsistency
Active T-Server in an HA pair	<ul style="list-style-type: none"> • Incorrect transitions in TEvents • Stuck parties • Missing attached data • Other inconsistencies due to incomplete eventflows
ICON connection to Interaction Server	<ul style="list-style-type: none"> • Incorrect party transitions from ICON's point of view resulting in missing transition or party error records • Interactions deleted at unknown times • Stuck interactions • Missing information about: <ul style="list-style-type: none"> • Agent state • Attached data
ICON connection to OCS	<ul style="list-style-type: none"> • Missing information about: <ul style="list-style-type: none"> • Campaign processing • Changes in calling lists • Linkages between OCS and voice call data • Missing OCS precalculated metrics
ICON server	<p>As a result of data loss for the particular ICON instance as well as possible failure in restoring queued data from the persistent queue:</p> <ul style="list-style-type: none"> • Stuck calls • Stuck parties • Missing records



Chapter

11

Extracting Data in an HA Deployment

Through the use of data source session control tables, Interaction Concentrator provides information about the availability and reliability of data in Interaction Database (IDB). This chapter describes how downstream reporting applications can use that information to optimize their extraction, loading, and transformation (ETL) processes to extract the most reliable data. This chapter also describes additional considerations for specific types of high availability (HA) data.

Note: This chapter is intended for users who develop their own ETL engine or who use Genesys Info Mart 8.0 or later to create reports that are based on data extracted from IDB.

This chapter contains the following sections:

- [Extracting HA Data, page 109](#)
- [Extracting Multimedia Data, page 115](#)
- [Extracting Virtual Queue-Specific Data, page 116](#)
- [Extracting Configuration Data, page 116](#)
- [Extracting Agent-Specific Data, page 116](#)

Extracting HA Data

Downstream reporting applications can leverage information about data availability in order to optimize ETL processes for extracting data from an HA pair of IDBs. Before it extracts data from a particular IDB for a particular time period, the downstream reporting application can use information about gaps in the data flow to determine the reliability of data in that IDB. Adjusted ETL

processes can then avoid reading the unreliable data from one IDB, and switch over to extracting data from the other IDB for that particular time period.

For information about identifying gaps in the data flow from a particular Interaction Concentrator (ICON) instance to a particular IDB, see “Determining Data Availability and Reliability” on [page 49](#).

For information about the data source session control tables that support this functionality, see “Data Source Session Control Tables” on [page 34](#).

This section uses a sample scenario to illustrate two methods of optimizing data extraction:

- [Scenario: Interrupted Data Flow with Calls, page 110](#)
- [Extracting HA Data: Approach 1, page 114](#)
- [Extracting HA Data: Approach 2, page 114](#)

Scenario: Interrupted Data Flow with Calls

Consider the following scenario:

- One data source (T-Server) and one provider (GCC provider).
- Connection and call events occur at the following times:

Time	Event
t0	ICON-1 starts. Calls 1, 2, and 3 start.
t1	ICON-2 starts. Calls 4 and 5 start.
t2	ICON-1 disconnects/terminates unexpectedly. Calls 6, 7, and 8 start.
t3	ICON-1 reconnects/restarts. Calls 9 and 10 start.
t4	ICON-2 disconnects/terminates unexpectedly.
t5	No new events. Calls continue.

- All timestamps use T-Server time.
- Reporting data is required for the interval t0–t5.

Figure 10 on [page 111](#) illustrates the relationship between the calls and the interrupted data flow in this scenario.

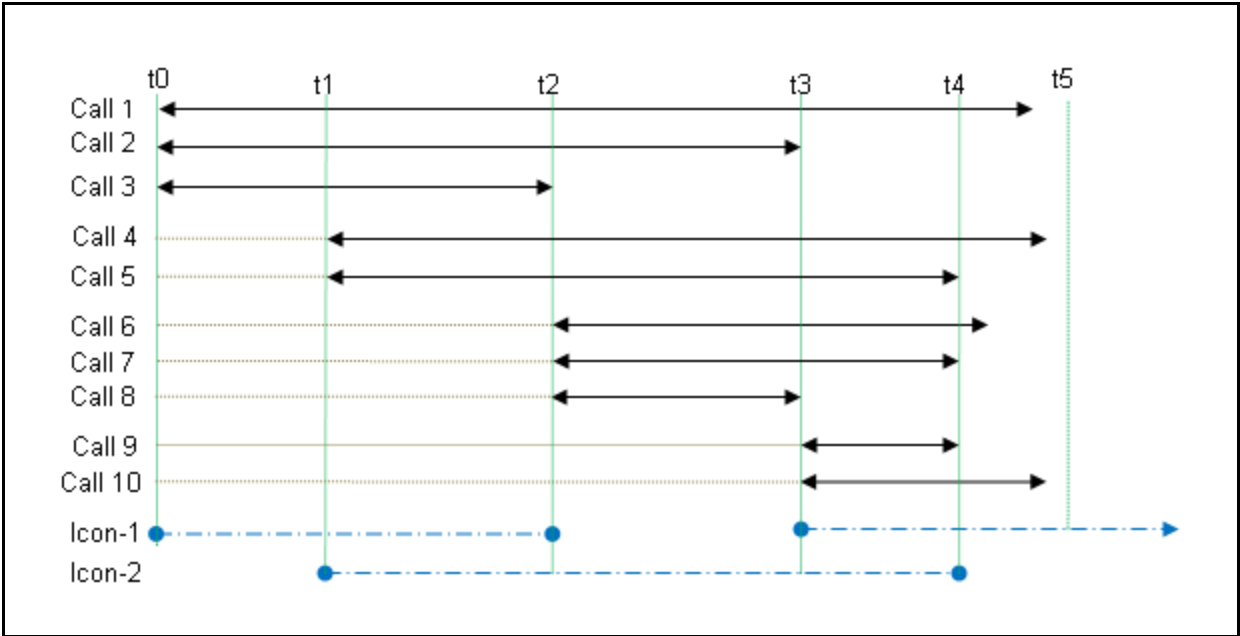


Figure 10: Calls in Relation to Interrupted Data Flow

Table 15 shows the values of selected fields in the records that the ICON instances create in the G_DSS_GCC_PROVIDER table in IDB-1 and IDB-2. For more information about the G_DSS_GCC_PROVIDER table fields, see the *Interaction Concentrator 8.0 Physical Data Model* for your RDBMS.

Table 15: Data Flow Interruption —G_DSS_GCC_PROVIDER Table Field Values

Record	DSS_ID	ICON_STIME	DSCONN_STIME	DSCONN_ETIME	FEVENT_DSTIME	LEVENT_DSTIME
ICON-1, G_DSS_GCC_PROVIDER table in IDB-1						
1	37	t0	t0	t2	t0	t2
2	38	t3	t3		t3	t5
ICON-2, G_DSS_GCC_PROVIDER table in IDB-2						
1	3767	t1	t1	t4	t1	t4

Legend (G_DSS_GCC_PROVIDER column names):

DSS_ID = Data source session ID

ICON_STIME = ICON startup time

DSCONN_STIME = Start of the connection to the data source server

DSCONN_ETIME = End of the connection to the data source server

FEVENT_DSTIME = Timestamp of the first event stored on the connection (T-Server time)

LEVENT_DSTIME = Timestamp of the last event stored on the connection (T-Server time)

Analysis

Table 15 shows that:

- ICON-1 has a failure timeframe from t2 to t3.
- ICON-2 was collecting data only from t1 to t4.

Alternatively, from the timestamps of the first and last saved events, the table also shows that:

- There was an uninterrupted data source session on ICON-1 from t0 to t2, and another uninterrupted data source session on ICON-1 from t3 to t5.
- There was an uninterrupted data source session on ICON-2 from t1 to t4.

Conclusions

Based on the connection information, the maximum timeframes of reliable data from each ICON are:

- [t0–t2]: ICON-1
- [t2–t4]: ICON-2
- [t4–t5]: ICON-1

Alternatively, the downstream reporting application can determine the maximum timeframes of reliable data from each ICON based on the durations of the data source sessions, and switch over from one IDB to the other at the break points.

HA Merge Results

Tables 16 and 17 compare the reliability of data from ICON-1, from ICON-2, and from both IDBs using a simple HA merge mechanism. Table 16 compares data for the case where ICON-1 terminates at t2. Table 17 compares data for the case where ICON-1 terminates at t2 and then restarts.

Table 16: Comparison of Data Reliability—ICON-1 Terminates

Call	ICON-1 (terminates at t2)		ICON-2		HA (ICON-1 + ICON-2)	
	Data	Reliable	Data	Reliable	Data	Reliable
1	No tail	No	No data	No	No tail	No
2	No tail	No	No data	No	No tail	No
Legend: No tail—ICON has not stored any data related to the end of the call. No data—There is no data about the call in IDB at all. All data—ICON stored all data about the call.						

Table 16: Comparison of Data Reliability—ICON-1 Terminates (Continued)

Call	ICON-1 (terminates at t2)		ICON-2		HA (ICON-1 + ICON-2)	
	Data	Reliable	Data	Reliable	Data	Reliable
3	All data	Yes	No data	No	All data	Yes
4	No tail	No	No tail	No	No tail	No
5	No tail	No	All data	Yes	All data	Yes
6	No data	No	No tail	No	No tail	No
7	No data	No	All data	Yes	All data	Yes
8	No data	No	All data	Yes	All data	Yes
9	All data	Yes	All data	Yes	All data	Yes
10	All data	Yes	No tail	No	All data	Yes
Legend: No tail—ICON has not stored any data related to the end of the call. No data—There is no data about the call in IDB at all. All data—ICON stored all data about the call.						

Table 17: Comparison of Results—ICON-1 Terminates then Restarts

Call	ICON-1 (terminates at t2, restarts at t3)		ICON-2		HA (ICON-1 + ICON-2)	
	Data	Reliable	Data	Reliable	Data	Reliable
1	No interim	No	No data	No	No interim	No
2	No tail	No	No data	No	No tail	No
3	All data	Yes	No data	No	All data	Yes
4	No interim	No	No tail	No	No interim	No
5	No interim	No	All data	Yes	All data	Yes
Legend: No interim—ICON missed part of the call. No tail—ICON has not stored any data related to the end of the call. No data—There is no data about the call in IDB at all. All data—ICON stored all data about the call.						

Table 17: Comparison of Results—ICON-1 Terminates then Restarts (Continued)

Call	ICON-1 (terminates at t2, restarts at t3)		ICON-2		HA (ICON-1 + ICON-2)	
	Data	Reliable	Data	Reliable	Data	Reliable
6	No data	No	No tail	No	No tail	No
7	No data	No	All data	Yes	All data	Yes
8	No data	No	All data	Yes	All data	Yes
9	All data	Yes	All data	Yes	All data	Yes
10	All data	Yes	No tail	No	All data	Yes
Legend: No interim—ICON missed part of the call. No tail—ICON has not stored any data related to the end of the call. No data—There is no data about the call in IDB at all. All data—ICON stored all data about the call.						

Extracting HA Data: Approach 1

This section describes one approach that enables the downstream reporting application to optimize the extraction and merging of data from the HA pair of IDBs. For the scenario described above (see “Scenario: Interrupted Data Flow with Calls” on [page 110](#)), the ETL:

1. Determines the timestamps of data flow interruption and the timeframes of reliable data. For the analysis applicable to this scenario example, see [page 112](#).
2. Extracts all data from IDB-1 related to calls that were terminated during the period [t0–t2].
3. Switches over to IDB-2, and extracts all data related to calls that were terminated during the period [t2–t4].
4. Switches back to IDB-1, and extracts all data related to calls that were terminated during the period [t4–t5].

Extracting HA Data: Approach 2

The following is an alternative approach that enables the downstream reporting application to optimize the extraction and merging of data from the HA pair

of IDBs. For the scenario described above (see “Scenario: Interrupted Data Flow with Calls” on [page 110](#)), the ETL:

1. Determines the timestamps of data flow interruption and the timeframes of reliable data. For the analysis applicable to this scenario example, see [page 112](#).
2. Extracts all data from IDB-1 until the timestamp of the ICON-1 break (t2).
3. In a staging area, removes all data related to non-terminated calls.
4. Switches over to IDB-2, and extracts all data for calls in IDB-2, starting with the call with the earliest timestamp that is greater than the timestamp of the last terminated call from IDB-1.

Note: This approach does not support late-arriving EventUserEvent data. EventUserEvent data that arrives after a call was terminated might be ignored, because it has a timestamp that falls between the last terminated call in IDB-1 and the next call in IDB-2.

Extracting Multimedia Data

Except for multimedia-specific attached data, data related to multimedia activity is stored in the same IDB tables as voice data. Starting with release 8.0, Interaction Concentrator supports HA of interaction-related data for active and completed multimedia interactions.

Extraction Procedure

To extract and merge multimedia interactions from an HA pair of IDBs with minimum overlap, follow the approach for voice calls described in “Extracting HA Data: Approach 1” on [page 114](#):

1. Determine the timestamp of the data flow interruption (see “Extracting HA Data” on [page 109](#)).
2. Extract all data from the first IDB until the break. This means that the ETL will extract all data for which the timestamp of last modification from the first ICON is earlier than the timestamp of the break.
3. Switch over to the second IDB, and extract all data for which the timestamp of last modification from the second ICON is earlier than the timestamp of the break in the second ICON’s data flow.
4. Switch back to the first IDB, and so on.

For historical tables (for example, G_IR_HISTORY), in which ICON only inserts records, you can use the last modification timestamp as the marker for switching from the historical table in one IDB to the same table in the other

IDB, and you can use a simple merge mechanism to reconstruct the multimedia data in the table.

However, for tables in which ICON can update as well as create records (for example, G_IR), you cannot simply add records from another IDB in a simple merge. The downstream reporting application must analyze the data to determine how to combine related records. This analysis will likely be deployment-specific.

Limitation

If some IDB data is calculated by adding a new value to a previously existing value, and if one of the records contains data that does not cover the interaction from the beginning, the data received after a merge will not be reliable. For example, 3rd Party Media statistics are processed in this cumulative manner; for a multimedia interaction scenario similar to the call scenario in Figure 10 on [page 111](#), if the ETL switches over to IDB-2 after the Icon-1 break at time t2, accumulated statistics for Call 1 and Call 2 will have incorrect values.

Extracting Virtual Queue-Specific Data

The general approach for extracting virtual queue data is the same as for other real-time interaction data. However, because virtual queue records have the same unique ID in both IDB instances (the VQID field in the G_VIRTUAL_QUEUE table), you can use the value of the STATUS field in the G_VIRTUAL_QUEUE table for additional data validation. For example, a virtual queue record with a value of stuck (integer value = 2) in the STATUS field can be ignored in favor of a virtual queue record with a normal or abandoned status.

Extracting Configuration Data

The general approach for extracting configuration data is the same as for interaction data. However, because ICON uses database identifiers that are assigned by Configuration Server (DBIDs) to identify configuration objects stored in IDB, you can use IDB data for additional data validation. ICON flags the reliability of configuration data in the GSYS_EXT_INT1 field of the GC_* and GCX_* tables. When the reliability value of the same configuration record differs between two HA IDBs, extract the data from the more reliable record. For more information, see “Reliability Flag” on [page 143](#).

Extracting Agent-Specific Data

ICON supports high availability of agent-specific data (login sessions, agent states and reasons, and after-call work).

The primary T-Server or Interaction Server creates `AgentSessionIDs` from the initial agent login and timestamp. It does not propagate the `AgentSessionID` to the backup T-Server or Interaction Server. Instead, when the backup T-Server or Interaction Server becomes primary (after a switchover), new `AgentSessionIDs` are assigned to all known agent sessions. To support HA of agent data, a particular ICON instance processes only `AgentSessionIDs` that are received from the primary T-Server or Interaction Server.

In addition, ICON uses the mechanism of a persistent cache to verify agent login session data and to prevent storage of duplicate login sessions. For more information, see “Populating Agent Login Session Data” on [page 88](#).



Part

3

Administering Interaction Concentrator

Part Two of this *Interaction Concentrator User's Guide* contains information that is required to perform ongoing maintenance and administration of Interaction Concentrator, as well as to troubleshoot startup and runtime problems. It also contains detailed information about Interaction Concentrator features and functionality.

This information appears in the following chapters:

- Chapter 12, “Starting and Stopping Interaction Concentrator,” on [page 121](#)
- Chapter 13, “Monitoring Interaction Concentrator,” on [page 131](#)
- Chapter 14, “Filtering IDB Data,” on [page 133](#)
- Chapter 15, “Resynchronizing Configuration Changes,” on [page 141](#)
- Chapter 16, “Using Special Stored Procedures,” on [page 151](#)
- Chapter 17, “Troubleshooting ICON Installation and Deployments,” on [page 181](#)



Chapter

12

Starting and Stopping Interaction Concentrator

This chapter describes the prerequisites for Interaction Concentrator startup, and it provides instructions for starting and stopping Interaction Concentrator (ICON). It contains the following sections:

- [Overview, page 121](#)
- [Before You Begin, page 122](#)
- [Command-Line Parameters, page 123](#)
- [Starting ICON, page 123](#)
- [Stopping ICON, page 127](#)

Overview

You can start and shut down Interaction Concentrator components by using the Management Layer, a startup file, a manual procedure, or Services Manager.

All of these methods usually require command-line parameters for a server application as well as an executable file name. The next section describes the command-line parameters that are common to most Genesys server applications. Subsequent sections describe the startup and shutdown procedures.

Note: For information about using the Management Layer, startup files, and Services Manager for startup, see the *Framework-8.0 Deployment Guide*.

Before You Begin

The following issues are important for you to consider before you attempt to start ICON.

Starting the `cfg` Role for an Oracle IDB

If the relational database management system (RDBMS) of the Interaction Database (IDB) that stores configuration-related data is Oracle, Genesys strongly recommends that you collect statistics on your IDB schema before you start the ICON instance that performs the `cfg` role.

Collecting statistics before starting ICON significantly shortens the amount of time it takes to start up.

Verifying ICON Connections and Configuration

Before you attempt to start ICON, confirm that the connections and configuration options that have been configured for your ICON Application are correct for your deployment.

- Connections** In general, do not change any connections on the `Connections` tab of the ICON Application during startup or runtime. Furthermore, do not disconnect from Configuration Server during startup.
- If ICON disconnects from Configuration Server during startup, ICON initialization will fail.
 - If you remove or change other connections during startup, ICON might fail to initialize correctly.
 - If you remove or change any connections during runtime, ICON functioning might be affected.

For more information about configuring connections, see the step about configuring the `Connections` tab, in the procedure to configure the ICON application object in the *Interaction Concentrator 8.0 Deployment Guide*.

Note: You must restart Interaction Concentrator after a backup instance is configured of any application, such as OCS or T-Server, for which Interaction Concentration has a connection configured on the `Connections` tab. If you do not restart Interaction Concentrator, data from the affected application is not written to the database.

- Configuration Options** Do not make changes to ICON configuration options during startup. You can make changes to ICON configuration options during runtime, but in some cases you must restart ICON for the changes to take effect. For more information, see the chapter about configuration options in the *Interaction Concentrator 8.0 Deployment Guide*.

Command-Line Parameters

The following startup command-line parameters are supported by Interaction Concentrator:

-host	The name of the host on which Configuration Server is running.
-port	The communication port that client applications must use to connect to Configuration Server.
-app	The exact name of an application as configured in the Configuration Database.
-v	The version of a component. Note that specifying this parameter does not start an application, but instead returns its version number. You can use either an uppercase letter (V) or lowercase letter (v).
-lmspath	The full path to the log messages files that an application uses to generate log events. (These files are the common file named <code>common.lms</code> and the application-specific file with the extension <code>*.lms</code> .) Use this parameter when the common and application-specific log message files are located in a directory other than the application's working directory—for example, when the application's working directory differs from the directory to which the application was originally installed. Note that if the full path to the executable file is specified in the startup command line (for instance, <code>c:\gcti\multiserver.exe</code>), the path that is specified for the executable file is used to locate the <code>*.lms</code> files, and the value of the <code>lmspath</code> parameter is ignored.

Warning! An application that does not locate its `*.lms` file at startup cannot generate application-specific log events and send them to Message Server.

Starting ICON

This section provides manual startup instructions for ICON server. You can start ICON in any of the following ways:

- From SCI (see [page 124](#)).
- Manually on UNIX (see [page 125](#)).
- Manually on Windows (see [page 126](#)).
- As a Windows Service on Windows (see [page 127](#)).

Starting ICON with Solution Control Interface

Complete the following procedure to start ICON with Solution Control Interface (SCI).

Procedure: Starting ICON with SCI

Prerequisites

Genesys recommends that the following applications be running before you start ICON:

- The DB Server that provides access to IDB.
- The relational database management system.
- T-Server.
- Outbound Contact Server, if ICON is configured to collect data from OCS.
- Interaction Server, if ICON is configured to collect data from Multimedia.

If you have configured ICON to store attached data, ensure that there is a proper attached data specification file in ICON's working directory. (By default, ICON uses the `ccon_adata_spec.xml` file.)

For a short period of time after starting or restarting, ICON may produce [cp:...] or FSM errors in the log. These errors occur when ICON encounters elements of interactions that it cannot resolve because the interactions were already in progress when ICON was started or restarted. You can safely ignore these errors.

For detailed instructions about starting the Genesys components on which Interaction Concentrator depends, see:

- *Framework-8.0 Deployment Guide*
- *Framework-8.0 T-Server Deployment Guide* for your particular T-Server type
- *Framework-8.0 DB Server User's Guide*
- *Outbound Contact 8.0 Deployment Guide*
- *~~Multimedia-8.0~~ eServices Deployment Guide*

Start of procedure

1. On the list pane in the SCI Applications view, select your ICON Application object.
2. Do one of the following:
 - On the toolbar, click the Start button.
 - From the Action menu, select Start.

- Right-click the `Application` object to access the shortcut menu, and then select `Start`.
3. In the confirmation box that appears, click `Yes`.
SCI starts your Interaction Concentrator application.

End of procedure

Next Steps

- You have completed all the steps necessary to start ICON using SCI.

Starting ICON Manually

Complete the following procedure to start ICON manually on UNIX.

Procedure: Starting ICON manually on UNIX

Start of procedure

1. Go to the directory to which you have installed ICON.
2. Enter the name of the ICON executable, followed by the appropriate command-line parameters, using the following syntax:

```
./icon -host <hostname> -port <portno> -app <application>
```

Where:

- *hostname* is the name of the host on which Configuration Server is running.
- *portno* is the communication port that client applications must use to connect to Configuration Server.
- *application* is the name of the Interaction Concentrator `Application` object, as defined to Configuration Server.

Note: If the host name or application name contains spaces or hyphens (-), enclose them in double quotation marks.

For example, to start ICON with command-line parameters that specify the host as `cs-host`, the port as `2020`, and the name as `ICON 03`, enter the following:

```
./icon -host "cs-host" -port 2020 -app "ICON 03"
```

End of procedure

Next Steps

- You have completed all the steps necessary to start ICON manually on UNIX.

Starting ICON on Windows

Complete the following procedure to start ICON on Windows.

Procedure:
Starting ICON on Windows

Purpose: To start ICON from the Start > Programs menu, or from the console window.

Start of procedure

1. Open a console window.
2. Go to the directory to which you installed Interaction Concentrator.
3. Enter the following command line:

```
icon.exe -host <hostname> -port <portno> -app <application>
```

Where:

- *hostname* is the name of the host on which Configuration Server is running.
- *portno* is the communication port that client applications must use to connect to Configuration Server.
- *application* is the name of the Interaction Concentrator Application object, as defined to Configuration Server.

Note: If the host name or application name contains spaces or hyphens (-), enclose them in double quotation marks.

For example, to start ICON with command-line parameters that specify the host as *cs-host*, the port as *2020*, and the name as *ICON 03*, enter the following:

```
icon.exe -host "cs-host" -port 2020 -app "ICON 03"
```

End of procedure**Next Steps**

- You have completed all the steps necessary to start ICON on Windows.

Starting ICON as a Windows Service

On Microsoft Windows platforms, by default, the installation process installs Interaction Concentrator as a Windows Service. If you stopped ICON from running as a Windows Service and need to start it again as a Windows Service, complete the following procedure.

Procedure: Starting ICON as a Windows service

Start of procedure

1. Open the Windows Control Panel, and then double-click the Services icon. The Services dialog box opens.
2. In the Services list box, select your ICON service, and then click Start. (If you disabled Interaction Concentrator from operating as a Windows Service, the Start option for this application will not be available.)

Note: You can install the Local Control Agent (LCA) as a Windows Service with the user interface disabled. In this case, all servers that are started through SCI are started without a console, unless you specifically select the Allow Service to Interact with Desktop check box for both LCA and ICON.

End of procedure

Next Steps

- You have completed all the steps necessary to start ICON as a Windows service.

Stopping ICON

You can stop ICON in any of the following ways:

- From SCI (see [page 128](#)). (This is the recommended method.)
- Manually on UNIX (see [page 128](#)).
- Manually on Windows (see [page 129](#)).
- As a Windows Service on Windows (see [page 130](#)).

Note: To prevent ICON from self-starting, make sure that you clear the autorestart property in the ICON Application object in Configuration Manager.

Stopping ICON with Solution Control Interface

If you are using LCA and SCS, complete the following procedure to stop ICON with SCI.

Procedure: Stopping ICON using SCI

Start of procedure

1. On the list pane in the SCI Applications view, select your ICON Application object.
2. Do one of the following:
 - On the toolbar, click Stop.
 - From the Action menu, select Stop.
 - Right-click the Application object to access the shortcut menu, and then select Stop.
3. In the confirmation box that appears, click Yes.
SCI stops your Interaction Concentrator application.

End of procedure

Next Steps

- You have completed all the steps to stop ICON using SCI.

Stopping ICON on UNIX

Stop ICON on UNIX by using one of the following procedures.

Procedure: Stopping ICON on UNIX from the command line

Start of procedure

- On the command line, enter the following:
`kill -SIGTERM <processid>`
Where <processid> is the application's UNIX process ID.

End of procedure

Next Steps

- You have completed all the steps to stop ICON from the command line.

Procedure:
Stopping ICON on UNIX from the console window**Start of procedure**

- From the active console window, press CTRL+C.

End of procedure**Next Steps**

- You have completed all the steps to stop ICON from the console window.

Note: If you are using LCA and SCS, you can also use SCI to stop ICON (see “Stopping ICON with Solution Control Interface” on [page 128](#)).

Stopping ICON on Windows

If ICON is running as an application—not as a Windows Service—stop it using the following procedure.

Procedure:
Stopping ICON on Windows from the console window**Start of procedure**

- From the application’s console window, press CTRL+C.

End of procedure**Next Steps**

- You have completed all the steps to stop ICON from the console window.

Note: If you are running ICON as a Windows Service, you should stop it only from the Services Control Manager (see “[Stopping ICON as a Windows Service](#)” below).

Stopping ICON as a Windows Service

To stop Interaction Concentrator running as a Windows Service, use the following procedure.

Procedure: **Stopping ICON running as a Windows service**

Start of procedure

1. Open the Control Panel, and then double-click the Services icon. The Services dialog box opens.
2. In the Services list box, select your ICON service, and then click Stop.

End of procedure

Next Steps

- You have completed all the steps to stop ICON running as a Windows Service.



Chapter

13

Monitoring Interaction Concentrator

This chapter describes how to access the HTTP Listener to monitor and report on Interaction Concentrator performance. It contains the following section:

- [Accessing the Performance Counter, page 131](#)
- [Performance Counter Pages, page 132](#)

Accessing the Performance Counter

You can configure an HTTP Listener to access a performance counter page from your browser. You can use this performance counter to report on ICON performance in real time and to identify performance and usage patterns which you can then use to fine-tune your ICON configuration. You can also access the performance counter pages as needed for troubleshooting purposes.

Note: Accessing the performance counter pages can significantly impact ICON performance. Therefore, Genesys recommends that you do not routinely access these pages.

To configure an HTTP Listener, create a `listeners` section on the `ICON Application` object that describes the HTTP listening port. Once the HTTP Listener is configured, you can access the performance counter pages when ICON is running, as follows:

- Open a browser window and go to the following address:
`http://<host>:<port>`
where:
 - `host` is the name of the ICON server host

- port is the value configured for the port option in the <http-connection> section on the Options tab of the ICON Application object.

For more information about configuring the HTTP Listener, see the chapter about configuring options in the *Interaction Concentrator 8.0 Deployment Guide*.

Performance Counter Pages

The performance counter pages provide the following types of information:

- Configuration object statistics—for example, the number of DN, Person, Place, and Agent Login objects read by ICON.
- CTI statistics—for example, detailed information about active calls, parties, and agent sessions; information about active T-Server connections; statistics about CTI events (such as the number of call processing errors and the average volume of calls and events); and accumulated statistics about agent login sessions.

Similar information is provided for Multimedia interactions as well.

- ICON statistics—information about memory allocation for current activity involving key ICON entities, such as calls and agent sessions.
- Database-writing statistics—information and statistics about database-writing activity for the ICON instance as a whole as well as by DAP role.

Figure 11 shows a sample performance counter page—the main Database Writer page.

Note: Figure 11 is only an example of the web page. Your actual screen might look quite different from this one.

Wicon_Enterprise_75	
Database Writer	
Status	Normal
Status time	2007-01-12 12:34:53
Error message	
Output queue backlog	0
Total data records written	50000
Total data records written/sec	0
Total records queued	50000
Records queued from calculation time	0
Records queued in previous 15 minutes	0
Last calculation time	2007-01-12 13:04:54
Components	
Configuration History	
Call/party	
User data	
Agent login/session	
Outbound	

Figure 11: Main Database Writer Page



Chapter

14 Filtering IDB Data

This chapter describes the Interaction Concentrator (ICON) data filtering capability and how to configure it by setting options on the `ICON Application` and `script` objects.

The data filtering feature provides users with the ability to configure what data will *not* be stored in IDB in order to maintain a smaller IDB and improve database performance.

This chapter contains the following sections:

- [Overview, page 133](#)
- [What Data Can Be Filtered?, page 134](#)

Overview

The data filtering capability of ICON allows you to control what type of data is stored in IDB based on your reporting requirements. Excluding certain types of data from IDB storage may be helpful in your environment, by:

- Saving database space (maintaining a smaller IDB)
- Improving the performance of your IDB
- Improving the performance of downstream reporting applications (such as Genesys Info Mart)

This feature is implemented by setting configuration options in the `filter-data` section of your `Interaction Concentrator Application` object. Because setting these options can cause ICON to cease writing data to the corresponding IDB tables, carefully evaluate whether your reports require each type of data described in the following sections before setting any options. Refer to Chapter 2, “Introducing IDB Schema,” on [page 25](#) for an overview of IDB schema and its tables.

ICON also supports the ability to filter out multimedia interaction activity related to strategies, where such activity data is not required for reporting

purposes. To implement this feature, set configuration options in the `callconcentrator` section of your `ICON Application` and `script` configuration objects. For more information about the `ICON Application` and relevant `script` configuration options, see the chapter about configuration options in the *Interaction Concentrator 8.0 Deployment Guide*.

Note: By default, the filtering configuration options are set to `false` and all available data is stored in the IDB.

What Data Can Be Filtered?

The data described in this section can be filtered— included or excluded from storage in IDB—by setting various configuration options. It includes the following sections:

- [Party History Data, page 134](#)
- [Party Metrics, page 135](#)
- [External Parties, page 135](#)
- [User Data History, page 135](#)
- [Call Metrics, page 136](#)
- [Call History, page 136](#)
- [Interaction Record History, page 136](#)
- [Agent Activity Data, page 136](#)
- [Service Observer Data, page 138](#)
- [Strategy Activity Data, page 139](#)

Note: In addition to the information contained here, refer also to the *Interaction Concentrator 8.0 Deployment Guide* for information about setting configuration options in `ICON`, and the *Interaction Concentrator 8.0 Physical Data Model* for your RDBMS type, for details about data stored in the IDB tables mentioned.

Party History Data

The `G_PARTY_HISTORY` table contains call party information related to when the history of a party state changed. For distribution devices only, this table contains information about queuing on a device and distribution from a device only (for example, it is not possible to have a ringing or hold state on a distribution device).

For SIP and voice interactions only, you can choose not to store party history information in IDB about distribution devices such as ACD queues, routing points, virtual routing points, and External Routing Points. Setting the `acd-party-history` configuration option to `true` on the `ICON Application`

will prevent ICON from writing party-related information to the `G_PARTY_HISTORY` table.

Party Metrics

ICON collects precalculated party metrics for distribution devices, such as ACD queues, routing points, and virtual routing points, and stores this information in the `G_PARTY_STAT` table.

For SIP and voice interactions only, you can choose not to store party metrics data for distribution devices in IDB by setting the `acd-party-metrics` configuration option to `true` on the `ICON Application` object.

External Parties

ICON collects information about external parties (for example, interaction participants outside a given switch domain) and stores this information in the following IDB tables:

- `G_PARTY`
- `G_PARTY_HISTORY`
- `G_PARTY_STAT`

You can choose not to store external-party data from IDB storage by setting the `external-party` configuration option to `true` on the `ICON Application` object.

User Data History

When ICON is configured in a way that it should store an entire history of `UserData` values for certain keys, ICON collects data about every change in value for those keys and, at interaction termination, stores this information in the following IDB tables:

- `G_USERDATA_HISTORY`
- `G_SECURE_USERDATA_HISTORY`

By setting the `udata-history-terminated` configuration option to `true` on the `ICON Application` object, ICON will not store the call-termination value of `UserData` keys to IDB. ICON will, however, continue to store the history of `UserData` changes—the creation, addition, and removal of key-value pairs—to these tables as changes occur during the call's life time. [Table 18](#) describes the information stored in these IDB tables.

Table 18: Change Information Types Stored in IDB

Value	Column Change Type	Description
1	created	1 indicates that the value of the key was attached to the call at the moment the call was created. For chat interactions, ICON assigns a change type of 1 upon receipt of an <code>eventInteractionSubmitted</code> event with an <code>isOnline</code> attribute of <code>true</code> .
2	added	2 indicates that the value of the key has just been added.
3	updated	3 indicates that the value of the key has changed. For chat interactions, ICON assigns a change type of 3 upon receipt of an <code>eventPropertiesChanged</code> event with an <code>isOnline</code> attribute setting of either <code>true</code> or <code>false</code> .
4	deleted	4 indicates that the key has been deleted from <code>UserData</code> .
5	terminated	5 indicates that ICON records the value of the key upon call termination.

Call Metrics

ICON stores information about call metrics in the `G_CALL_STAT` table. To exclude call metrics from IDB storage, set the `call-metrics` configuration option to `true` on the `ICON Application` object.

Call History

ICON stores call history information in the `G_CALL_HISTORY` table. To exclude call history information from IDB storage, set the `call-history` configuration option to `true` on the `ICON Application` object.

Interaction Record History

ICON collects interaction record history and stores this information in the `G_IR_HISTORY` table. To exclude data about the interaction record history from IDB storage, set the `ir-history` configuration option to `true` on the `ICON Application` object.

Agent Activity Data

ICON collects information about agent activity, such as login sessions and agent states. Unless certain types of data are configured to be excluded (see the subsections below), ICON stores this information in the following IDB tables:

- `G_LOGIN_SESSION`

- GX_SESSION_ENDPOINT
- G_AGENT_STATE_HISTORY
- G_AGENT_STATE_RC
- G_DND_HISTORY
- GS_AGENT_STAT
- GS_AGENT_STAT_WM

You can choose not to store any agent activity information in IDB by setting the `gls-all` configuration option to `true` on the `ICON Application` object. `ICON`, however, continues writing each agent's ID to the `G_PARTY` table.

Agent Metrics

`ICON` collects and stores agent state information in the following IDB tables:

- GS_AGENT_STAT
- GS_AGENT_STAT_WM

You can choose to exclude agent state information by setting the `gls-metrics` configuration option to `true` on the `ICON Application` object.

Agent Activity From IVR Devices

`ICON` collects data about agent activity when agent login sessions are initiated from IVR endpoints, and stores this information in the following IDB tables:

- G_LOGIN_SESSION
- GX_SESSION_ENDPOINT
- G_AGENT_STATE_HISTORY
- G_AGENT_STATE_RC
- G_DND_HISTORY
- GS_AGENT_STAT
- GS_AGENT_STAT_WM

You can choose to exclude data about agent activity at IVR endpoints, from IDB storage, by setting the `gls-ivr` configuration option to `true` on the `ICON Application` object. `ICON` verifies whether the DN at which an agent logs in is an IVR device, and in this case, does not store information about this agent's activity to IDB. Furthermore, for parties associated with an IVR device, `ICON` does not record the agent's ID in the `G_PARTY` table.

Agent Activity For Persons Not Configured

`ICON` collects data about all agent activity and stores this information in IDB tables (see "Agent Activity Data" on [page 136](#)). You can choose to exclude data from IDB storage about agent activity for any agent whose login ID is not associated with any Person configuration object by setting the `gls-no-person` configuration object to `true` on the `ICON Application` object. If so configured, `ICON` verifies whether the LoginID that was reported in events regarding

agent states is assigned to any Person object configured in the Configuration Database. If this is not the case, ICON does not store information about this agent's activity to these tables.

Agent Work Mode

ICON collects and stores data about agents' work mode and changes in agents' work modes in the following IDB tables:

- G_AGENT_STATE_HISTORY
- G_AGENT_STATE_RC
- GS_AGENT_STAT_WM

You can choose to exclude data about changes in agent work mode that do not coincide with changes in agent state by setting the `gls-wm` configuration option to `true` on the `ICON Application` object. ICON instead records a value of `unknown` in the IDB tables.

Note: This option does not affect ICON's ability to track after-call work.

Agent Queue

ICON collects and stores information about agents' queue(s) in the following IDB tables:

- G_AGENT_STATE_HISTORY
- G_AGENT_STATE_RC
- GS_AGENT_STAT
- GS_AGENT_STAT_WM
- GX_SESSION_ENDPOINT

You can filter out information about the queues where agents are logged in, by setting the `gls-queue` configuration option to `true` on the `ICON Application` object. In this case, ICON ceases writing queue-related data to the first four tables (above). ICON will, however, continue writing information to the `GX_SESSION_ENDPOINT` table about the queues where agents are logged in.

Service Observer Data

ICON collects data related to parties with role `observer` on a call and stores this information in the following IDB tables:

- G_PARTY
- GS_PARTY_STAT

You can choose not to store data about the party with the role `observer` in IDB, by setting the `observer-party` configuration option to `true` on the `ICON Application` object.

Strategy Activity Data

ICON can filter out activity related to a particular strategy—information such as parties or user data changes made by the strategy. Such data may not be required for reporting purposes, and therefore you can choose to exclude it from storage in IDB.

To filter out strategy activity data, set the following configuration options to the values specified:

- `om-activity-report`—Set this option to `false` on the Annex tab of script configuration objects of type `simple routing` (for a routing strategy). ICON will not store to IDB any data that is related to any Multimedia interaction handled by this strategy, provided that the `om-check-filter-flag` option is also set to 1.
- `om-check-filter-flag`—Set this option to 1 on the ICON Application object (`callconcentrator` section) to allow ICON to store strategy activity according to the value of the `om-activity-report` option. If set to 0, ICON continues to store all strategy activity regardless of the value of `om-activity-report`.



Chapter

15

Resynchronizing Configuration Changes

Under certain conditions, Interaction Database (IDB) data can fall out of synchronization with Configuration Server data. Operating Interaction Concentrator (ICON) instances and downstream reporting applications that rely on ICON data under these circumstances can lead to results that are difficult to interpret. Under such conditions, you might consider resynchronizing IDB with the current configuration data.

Note: If you configure ICON to store configuration changes, it is important to ensure that ICON runs continually (without stopping). This will limit the chance of IDB losing synchronization with Configuration Server data.

This chapter describes the conditions under which you might consider invoking a forced resynchronization of IDB data, and the steps required to resynchronize your IDB. This chapter also describes how to restore Configuration Database should you need to recover its data from a crash. It includes the following sections:

- [How Resynchronization Works, page 142](#)
- [How Long Does Resynchronization Take?, page 144](#)
- [When to Resynchronize IDB, page 144](#)
- [How to Resynchronize Configuration Data, page 146](#)
- [Recommendations for Resynchronization, page 147](#)
- [Restoring the Configuration Database from Backup, page 148](#)

How Resynchronization Works

ICON stores configuration data in two types of tables, one for configuration objects and the other for configuration object relationships. Each configuration data record is therefore defined by a pair of attributes: configuration object type and database identifier (DBID).

During resynchronization, ICON requests all configuration data from Configuration Server and stores it to the local cache. For each pair of attributes (object type and DBID), ICON checks whether it exists in the IDB:

- If it does *not* exist, a new record is added to the IDB.
- If it does exist, ICON checks if any stored properties are different for this object, and if so, the IDB record is updated.

Note: Any active objects that exist in the IDB but not in the Configuration Database, are marked for deletion.

During resynchronization, ICON suspends receipt of real-time notifications about new configuration object updates from Configuration Server. Configuration Server queues the notifications and delivers them after synchronization completes.

Table 19 shows some of the information that ICON records in IDB for new, updated, and deleted configuration objects when resynchronization occurs:

- The `Field` column indicates the column in the targeted `GC_*` IDB table that ICON will write to when ICON transfers data to IDB.
- Values in the `GSYS_EXT_INT1` field indicate the reliability of the timestamps flag (see Table 21 on [page 143](#) for a description of the values).

Table 19: IDB Stores Object Changes

Field	New Object	Updated Object	Deleted Object
STATUS	1 (for active)	1 (for active)	2 (for inactive)
CREATED	sync time		
DELETED	null	null	sync time
LASTCHANGED	sync time	sync time	sync time
GSYS_EXT_INT1	1		2 or 3

Where no value is provided in the table above, ICON makes no change to the value that pre-exists for the associated field.

Table 20 shows some information that ICON records in IDB for new, updated, and deleted object relationships when resynchronization is run. LASTCHANGED

information does not pertain to relationships and thus is not included in the table.

Table 20: IDB Stores Object Relationship Changes

Field	New Object Relationship	Updated Object Relationship	Deleted Object Relationship
STATUS	1 (for active)	1 (for active)	2 (for inactive)
CREATED	sync time		
DELETED	null	null	sync time
GSYS_EXT_INT1	1		2 or 3

Reliability Flag

ICON uses a reliability flag to provide downstream reporting applications (such as Genesys Info Mart) the ability to extract the most reliable data from two separate instances of IDB. The GSYS_EXT_INT1 field stores a value, in all IDB configuration records, which indicates the reliability of the data (see [Table 21](#)).

Table 21: Reliability Flag Values

Value	What Value Indicates
0	The creation or deletion timestamp is reliable. The timestamp was provided by either real-time Configuration Server notifications or the configuration history log.
1	The creation timestamp is not reliable. The timestamp corresponds to the initial data load or resynchronization time.
2	The deletion timestamp is not reliable. The timestamp corresponds to the initial data load or resynchronization time.
3	Neither the creation nor the deletion timestamp is reliable. Both timestamps correspond to the time of the initial data load or resynchronization.

How Long Does Resynchronization Take?

The amount of time required for the synchronization process to complete depends on a number of factors including:

- Performance of Configuration Server
- Network performance
- Size of the Configuration Database
- Performance of the database that hosts the IDB.

Under normal conditions, synchronization should take several minutes.

When to Resynchronize IDB

In some cases, ICON recognizes that the configuration data in the IDB is suspect and produces a special log message (09-25131) to that effect. In most cases, however, no such message will be generated and the decision to resynchronize configuration data is left to your discretion.

Note: If ICON generates this special log message, it will stop monitoring configuration changes, and move to a waiting state where it will remain until it receives the user command to start resynchronization. To avoid any loss of configuration data changes, Genesys strongly recommends setting an alarm condition (see “Setting an Alarm Condition” on [page 145](#)). ICON immediately resumes monitoring configuration changes upon completion of resynchronization.

The following *exceptional* circumstances are guidelines you can use to determine if resynchronization is required:

- The RDBMS for the Configuration Database has crashed and you must recover it from backup. See “Restoring the Configuration Database from Backup” on [page 148](#).
- Unavoidable events have prevented ICON from running for a period of time, while Configuration Server was operating and changes were made to configuration objects or their relationships to other objects.
- Configuration tracking for ICON was mistakenly turned off for a period of time while changes were made to configuration-related data.
- ICON ran configuration tracking against the wrong Configuration Database. (This is a user-driven error.)
- ICON detects an inconsistency in configuration data and logs a message accordingly. (The first message of inconsistency logged following IDB upgrade is normal and should be ignored.)

- Your downstream reporting application (such as Genesys Info Mart), detects missing or inconsistent configuration data in IDB. For Genesys Info Mart users, refer to the Genesys Info Mart documentation for more information on this exceptional circumstance.

Setting an Alarm Condition

ICON generates the following Standard-level log event if it suspects that the configuration data in the IDB is inconsistent:

```
09-25131: Configuration data inconsistency is detected; reason:
[reason]. Waiting for customer command...
```

It will then stop monitoring configuration changes, and move to a waiting state where it will remain until it receives the user command to start resynchronization.

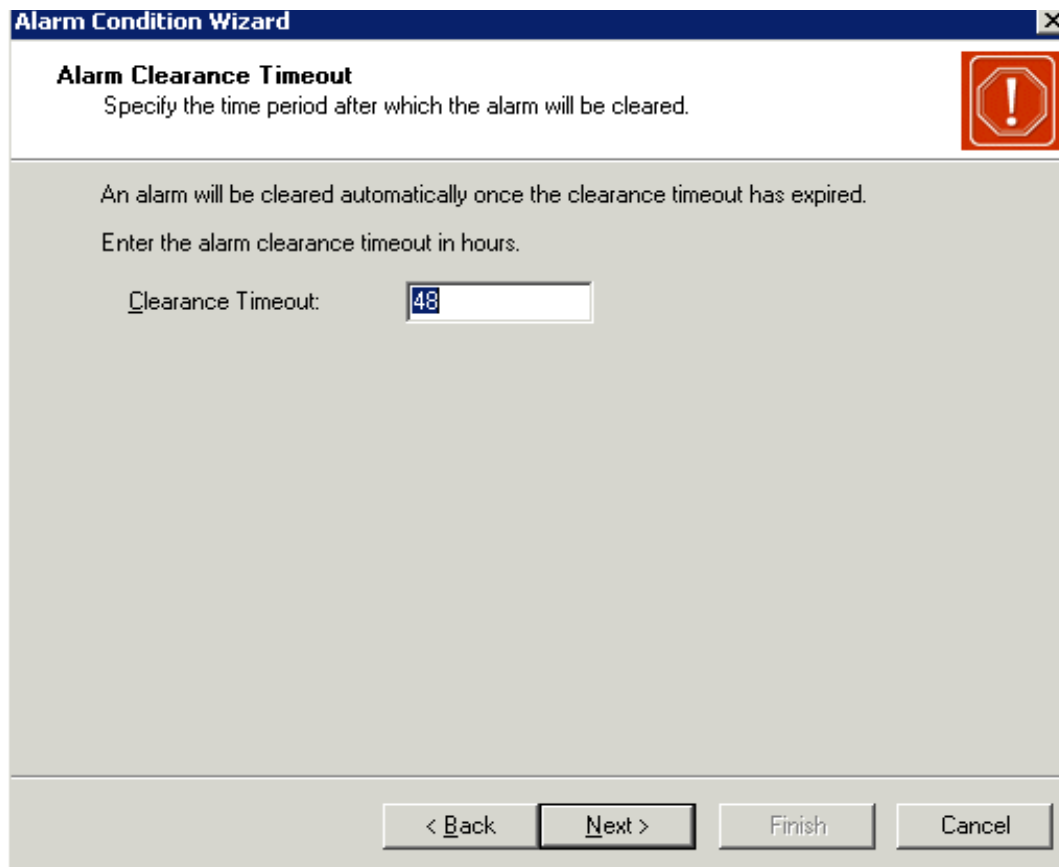
To avoid any loss of data, Genesys *strongly* recommends that you set an alarm condition using the Solution Control Interface based on this triggering log event. *Framework 8.0 Solution Control Interface Help* explains how to create alarm conditions using the Alarm Condition Wizard.

When you arrive at the Alarm Detection page in the wizard, specify a Detect log event as shown in [Figure 12](#).

Figure 12: Specifying Alarm Detection Event for ICON Within SCI

Although there is no corresponding Cancel (or clearing) event, ICON generates log event 09-25017, when all the data necessary for resynchronization has been retrieved. This log event can be used as a clearance event (see “Recommendations for Resynchronization” on [page 147](#)). You can also set an

alarm Clearance Timeout which will clear the alarm condition after the specified time (in hours) has expired (see [Figure 13](#)).



Alarm Condition Wizard

Alarm Clearance Timeout
Specify the time period after which the alarm will be cleared.

An alarm will be cleared automatically once the clearance timeout has expired.
Enter the alarm clearance timeout in hours.

Clearance Timeout:

< Back Next > Finish Cancel

Figure 13: Specifying Clearance Timeout for ICON Within SCI

How to Resynchronize Configuration Data

If you must resynchronize your configuration data in IDB, do the following while ICON is running:

1. Verify that the ICON application, with the `role` configuration option set to `cfg`, is started.
2. In Configuration Manager, set the `start-cfg-sync` option value to 0 in the Application Properties window of this ICON application.
3. Click OK in the option properties window.
4. Click Apply in the ICON Application Properties window to enable this setting.
5. Change the option value to 1.
6. Click OK in the Option Properties window.

7. Click Apply in the ICON Application Properties window to enable this new setting.

This action prompts Interaction Concentrator to start the resynchronization process. Once started, the resynchronization process continues until completion regardless of any subsequent changes to the option value. ICON logs a message when synchronization begins and when it completes.

Note: It does not suffice for the start-cfg-sync option to be preset to 1 or for it to be set to any other nonzero value and then to 1. ICON starts synchronization only when it detects the change from 0 to 1.

Recommendations for Resynchronization

- Consider changing the Configuration Server operation mode to Read-only for the time period during which data resynchronization is performed. When ICON retrieves from Configuration Server all the data necessary for resynchronization, ICON generates the following Standard-level log event:

09-25017 Configuration objects are reloaded in IDB

At this point, you can change the Configuration Server operation mode back to normal. Consider using this log event as a clearance event when configuring the alarm condition for resynchronization (see “Setting an Alarm Condition” on [page 145](#)).

- If you are using downstream reporting applications that use Interaction Concentrator as their data source, do the following to verify that the data in IDB is ready before you start your ETL engine for the first time after the resynchronization process:
 - Check ICON logs for log event: 09-25017, Configuration objects are reloaded in IDB
 - Execute the following SQL statement against IDB:

```
select eventid from G_SYNC_CONTROL where providertag = 5
```

If the above statement returns no records or eventID = 0, the resynchronization is still in progress.

If the above statement returns a non-zero value of eventID, the resynchronization is completed, and it is safe to run your ETL engine.

If you are restoring data from a backup Configuration Database (after your primary Configuration Database is corrupted, for example), perform the following steps:

- a. Restore the Configuration Database from a backup copy (see [“Restoring the Configuration Database from Backup”](#)).
- b. Restart Interaction Concentrator.

- c. Perform manual resynchronization of configuration data in IDB as described in “How to Resynchronize Configuration Data” on [page 146](#).

Restoring the Configuration Database from Backup

To restore your Configuration Database from backup, do the following:

1. Stop Configuration Server.
2. Stop the Interaction Concentrator instance tracking configuration data changes (`role = cfg`).
3. Restore the Configuration database from backup.
4. Make modifications in the Configuration Database to prevent Configuration Server from reusing the same DBID previously reported to ICON (see “[Modifying the Configuration Database](#)”).
5. Start Configuration Server.

Modifying the Configuration Database

To prevent Configuration Server from reusing the same (already reported) DBIDs, make the following modifications to the Configuration Database:

Note: These steps must be repeated for each type of configuration object stored by ICON (see Table 22 on [page 149](#)).

1. Check the last object DBID reported to ICON by Configuration Server by executing the following SQL statement against the IDB (and saving the result):

```
select max(id) from <IDB_TABLE_NAME>
```

Note: Replace <IDB_TABLE_NAME> in the SQL statement with the correct table name (see [Table 22](#)).

2. If the result of the statement executed in step 1 is *not NULL*—that is to say, some records exist in the IDB table—go to [Step 3](#). If the result is *NULL*, repeat [Step 1](#) for the next type of configuration object in [Table 22](#).
3. Check the maximum value of the last object DBID by executing the following SQL statement against the Configuration Database:

```
select MAX_DBID from CFG_MAX_DBID where object_type =  
<cfg_object_type>
```

Note: Replace `<cfg_object_type>` in the SQL statement with the integer code of the Configuration Server object type (see [Table 22](#)).

4. If the result of the statement executed in [Step 3](#) is any value—that is to say, *not NULL*—go to [Step 6](#). If the result is `NULL`, go to [Step 5](#).
5. Insert a record in the `CFG_MAX_DBID` Configuration Database table by executing the following SQL statement against the Configuration Database:

```
insert into CFG_MAX_DBID (object_type,max_dbid) values
(<cfg_object_type>, <max_id> + 1)
```

Notes: Replace `<cfg_object_type>` in the SQL statement with the integer code of the Configuration Server object type (see [Table 22](#)).

Replace `<max_id>` in the SQL statement with the value of `max(id)` from [Step 1](#).

6. Update the record in the `CFG_MAX_DBID` Configuration Database table by executing the following SQL statement against the Configuration Database:

```
update CFG_MAX_DBID set max_dbid = <max_id> + 1 where object_type =
<cfg_object_type>
```

Notes: Replace `<cfg_object_type>` in the SQL statement with the integer code of the Configuration Server object type (see [Table 22](#)).

Replace `<max_id>` in the SQL statement with the value of `max(id)` from [Step 1](#).

7. Repeat [Steps 1](#) through [6](#) for each configuration object type stored by ICON.

Table 22: Configuration Server Object Types and IDB Tables

Object Type	IDB Table Name: <IDB_TABLE_NAME>	Config Server Object Type: <cfg_object_type>
Switch	GC_SWITCH	1
Endpoint (DN)	GC_ENDPOINT	2
Person (agent)	GC_AGENT	3
Place	GC_PLACE	4
Groups	GC_GROUP	5
Tenant	GC_TENANT	7

Table 22: Configuration Server Object Types and IDB Tables (Continued)

Object Type	IDB Table Name: <IDB_TABLE_NAME>	Config Server Object Type: <cfg_object_type>
Application	GC_APPLICATION	9
Scripts	GC_SCRIPT	12
Skills	GC_SKILL	13
Action codes	GC_ACTION_CODE	14
Agent Login	GC_LOGIN	15
Folder	GC_FOLDER	22
Table Field	GC_FIELD	23
Table Formats	GC_FORMAT	24
Table access	GC_TABLE_ACCESS	25
Calling List	GC_CALLING_LIST	26
Campaigns	GC_CAMPAIGN	27
Treatments	GC_TREATMENT	28
Filters	GC_FILTER	29
Time Zones	GC_TIME_ZONE	30
Voice Prompts	GC_VOICE_PROMPT	31
IVR Ports	GC_IVRPORT	32
IVRs	GC_IVR	33
Business Attributes	GC_BUS_ATTRIBUTE	35
Attribute Values	GC_ATTR_VALUE	36
Objective Table	GC_OBJ_TABLE	37



Chapter

16

Using Special Stored Procedures

Aside from the stored procedures that Interaction Concentrator (ICON) uses internally to perform its functions, Interaction Concentrator provides a number of stored procedures that are relevant for users.

This chapter describes the stored procedures that are of special relevance to ICON users, including instructions on how to set up and execute them. Examples for each supported RDBMS are also provided.

This chapter contains the following sections:

- [Merge Stored Procedure, page 152](#)
- [Purge Stored Procedures, page 162](#)
- [Stored Procedure gsysInitTimeCode, page 178](#)
- [Custom Dispatchers, page 179](#)

Warning! SQL-like statements in this chapter are not true SQL statements; they are intended only to demonstrate the logic of the described stored procedures.

Note: Starting with Interaction Concentrator 8.0, the names of the stored procedures that are called internally start with a schema-specific prefix. The Interaction Concentrator installation package (IP) includes a wrapper script that links the generically named stored procedures that are described in this chapter to the equivalent, schema-specific stored procedures. The information in this chapter assumes that, as recommended in the installation instructions in the *Interaction Concentrator 8.0 Deployment Guide*, the wrapper script was executed as part of the IDB initialization or upgrade.

Merge Stored Procedure

The merge stored procedure merges voice interaction records in the Interaction Database (IDB) in order to finalize data processing of closed single-site and multi-site interactions. The procedure merges voice interaction records that are stored within a single IDB (*intra-IDB merge*). The procedure does not merge records that are distributed across more than one IDB (*inter-IDB merge* or *multi-IDB merge*).

The merge procedure can run in the background while instances of the ICON process are actively writing data to IDB.

Scheduling It is the responsibility of the customer (usually the Database Administrator) to run the merge procedure as required. Genesys recommends that you schedule the merge stored procedure to run on a regular basis. If you do not have this stored procedure scheduled to run regularly (not less than every 15 minutes), ensure that you run it before any data is extracted from IDB.

Note: If you are using Genesys Info Mart 7.6, run this procedure every five minutes. Genesys Info Mart 8.0 does not use this stored procedure. Instead, it merges interactions automatically as a part of the scheduled ETL process.

If you run the merge procedure occasionally, on an ad hoc basis, execution can take significantly longer than if you run it regularly. Also, in this case, ICON performance can be adversely affected while the procedure is executing.

This section contains the following information about the merge procedure:

- How the `gsysIRMerge` and `gsysIRMerge2` stored procedures function.
- Setting up the merge procedure (see [page 156](#)).
- Merge procedure parameters (see [page 158](#)).
- Executing the merge procedure (see [page 160](#)).

`gsysIRMerge` and `gsysIRMerge2`

Starting with release 7.5, Interaction Concentrator supports two stored procedures for voice data merge:

- `gsysIRMerge`—A wrapper for `gsysIRMerge2`.
- `gsysIRMerge2`—The stored procedure that actually performs the merge.

`gsysIRMerge` was originally provided to ensure backward compatibility with Interaction Concentrator 7.2. Calls to earlier versions of `gsysIRMerge` and `gsysIRMerge2` are compatible with calls to `gsysIRMerge` and `gsysIRMerge2` in Interaction Concentrator release 8.0.

Tables Involved in the Procedure

Figure 14 shows the tables that are involved in the merge procedure.

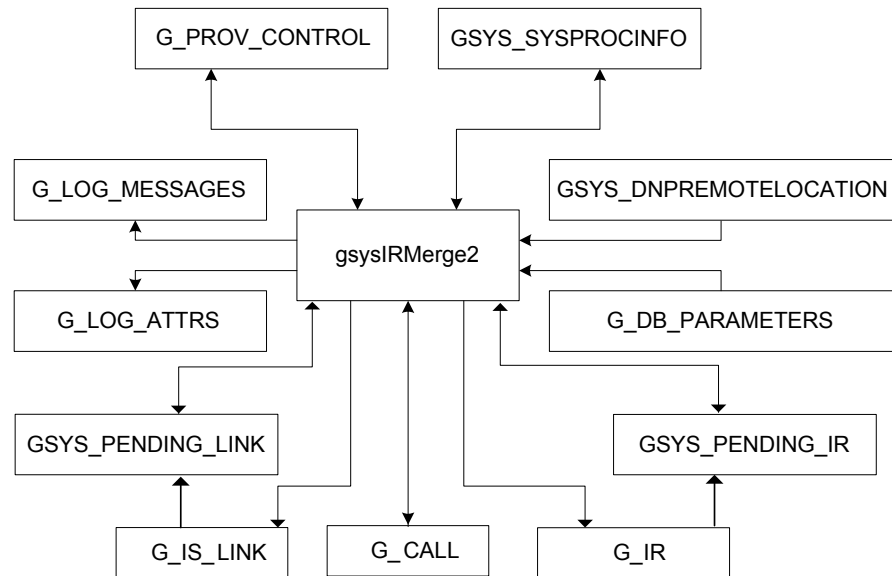


Figure 14: Tables Involved in the Merge Procedure

The following subsections describe each of these tables, and how they are used in the merge procedure.

G_LOG_MESSAGES and G_LOG_ATTRS

G_LOG_MESSAGES

The G_LOG_MESSAGES table contains information about the start (MESSAGE_ID = 5020) and end (MESSAGE_ID = 5030) of the merge procedure.

G_LOG_ATTRS

The G_LOG_ATTRS table contains the error descriptions (code and message), and detailed information about the processed data.

You can read information from these two tables by using Genesys Solution Control Interface (SCI), which is the graphical user interface (GUI) component of the Genesys Management Layer.

G_PROV_CONTROL and GSYS_SYSPROCINFO

G_PROV_CONTROL

From the G_PROV_CONTROL table, the merge procedure reads information about the ICON instances that are writing data to the database, and the corresponding number of the latest transaction.

The procedure executes the following query against the G_PROV_CONTROL table:

```

SELECT PRIMARYID SYS_ID          /* ICON instance identity*/
      SEQCURRENT                  /* number of latest transaction*/
FROM G_PROV_CONTROL
WHERE PROVIDERTAG = 1 AND SEQCURRENT IS NOT NULL
  
```

In addition, the merge procedure reads from this table the number of its own latest transaction, using the following query:

```
SELECT SEQCOUNTER FROM G_PROV_CONTROL
WHERE DOMAINID = 0
AND PRIMARYID = 99
AND PROVIDERTAG = 0
```

After this, the merge procedure starts data processing. After data processing is complete, the merge procedure updates the number of its own transaction to a new value.

GSYS_ SYSPROCINFO

From the GSYS_SYSPROCINFO table, the merge procedure reads information about the number of the latest ICON transaction that was processed, using the following queries for each ICON instance:

```
SELECT SEQPROCESSED FROM GSYS_SYSPROCINFO
WHERE PROVIDERTAG = 1 AND PROCNAME = 'gsysirmerge'
```

After data processing is complete, the merge procedure updates the number of the transaction to the processed one.

GSYS_DNPREMOTELOCATION and G_DB_PARAMETERS

The data from the GSYS_DNPREMOTELOCATION and G_DB_PARAMETERS tables is used to process *stuck* inter-server (IS) links (that is, links for which there is no information about a corresponding IS-Link at another site).

The normal situation for the merge procedure is to merge interactions when IDB contains complete information about all parts of the interaction. However, for interactions that cross multiple sites, if all the sites are not monitored by ICON instances writing to the same IDB, the IDB will never have enough information to conclude that the interaction has ended and to identify the constituent parts of the interaction. The IS-Link is stuck.

GSYS_ DNPREMOTE LOCATION

From the GSYS_DNPREMOTELOCATION table, the merge procedure reads the names of the switches that are not monitored by any ICON instance that writes to this particular instance of IDB. From the point of view of this database, unmonitored switches are considered to be remote locations.

Records in the GSYS_DNPREMOTELOCATION table indicate to the merge procedure that IS-Links that point to remote locations are unpaired within this instance of IDB. Therefore, the merge procedure does not expect any further information about the interaction on the other end of the IS-Link. In this way, data from the GSYS_DNPREMOTELOCATION table enables the merge procedure to minimize the amount of time required in order to process stuck IS-Links, because the procedure will not wait for the IS-Link timeout to expire. (For more information about the IS-Link timeout, see “[G_DB_PARAMETERS](#)”. See also [stuckthreshold](#) on [page 156](#).)

G_DB_ PARAMETERS

From the G_DB_PARAMETERS table, the merge procedure reads the time interval, in seconds, at which information about the IS-Link at another site is expected to be reported, using the following query:

```
SELECT VAL FROM G_DB_PARAMETERS
  WHERE SECT = 'merge' AND
        OPT = 'stuckthreshold'
```

The merge procedure also uses other parameters from the G_DB_PARAMETERS table. For more information, see “IS-Link Timeout and Other Parameters” on [page 156](#).

GSYS_PENDING_IR and GSYS_PENDING_LINK

The GSYS_PENDING_IR and GSYS_PENDING_LINK tables track the merge state of the interaction records and IS-Links while they are being processed. The merge procedure populates the GSYS_PENDING_IR and GSYS_PENDING_LINK tables with temporary data.

G_IS_LINK, G_IR, and G_CALL

G_IS_LINK The merge procedure uses the G_IS_LINK table to determine multi-site interactions. A selected multi-site interaction is considered to be completed when one of the following occurs:

- All IS-Links have a corresponding IS-Link from another site.
- An IS-Link has been opened to an unmonitored switch.
- The timeout for an IS-Link to arrive from another site has expired, and all corresponding IRs are closed.

When a multi-site interaction is completed, the procedure makes the following updates:

```
UPDATE G_IS_LINK SET MERGESTATE = 3
  WHERE LINKID in (ALL IS LINK IN INTERACTION)

UPDATE G_IR SET MERGESTATE = 3
  ROOTIRID = (FIRST IR IN INTERACTION)
  GSYS_MSEQ = (the procedure's current TRANSACTION ID)
  GSYS_MSEQ_TS = (time of TRANSACTION)
  WHERE ROOTIRID in (ALL ROOT IRs IN INTERACTION)

UPDATE G_CALL
  SET ROOTIRID = (FIRST IR IN INTERACTION)
  WHERE ROOTIRID in (ALL ROOT IRs IN INTERACTION)
```

For a single-site interaction, if all corresponding IRs are closed, the interaction is completed, and the procedure makes the following update:

```
UPDATE G_IR SET MERGESTATE = 3
  GSYS_MSEQ = (the procedure's current TRANSACTION ID)
  GSYS_MSEQ_TS = (time of TRANSACTION)
  WHERE IRID in (IRs IN INTERACTION)
```

Setting Up the Merge Procedure

To set up the merge procedure, ensure that the following information in IDB is correct for your purposes:

- Unmonitored Switches**
- The GSYS_DNPREMOTELOCATION table contains the names of the switches that are not monitored by any ICON instance that writes to this IDB. If necessary, manually update the GSYS_DNPREMOTELOCATION table, using a standard INSERT statement. The REMOTELOCATION column in the GSYS_DNPREMOTELOCATION table stores the names of unmonitored switches.

Example For example, suppose that you have two switches, each of which is monitored by a separate ICON that writes to its own IDB:

- ICON1 monitors switch SITE1_sw1 and writes to IDB1.
- ICON2 monitors switch SITE2_sw2 and writes to IDB2.

For optimal performance of the merge stored procedure, add the following records to the respective IDBs:

- In IDB1, set GSYS_DNPREMOTELOCATION.REMOTELOCATION='SITE2_sw2'
- In IDB2, set GSYS_DNPREMOTELOCATION.REMOTELOCATION='SITE1_sw1'

- IS-Link Timeout and Other Parameters**
- The G_DB_PARAMETERS table stores parameters that the merge procedure uses to control its operation. [Table 23](#) lists the parameters and their default values that the merge procedure uses.

Table 23: Merge Parameters in the G_DB_PARAMETERS Table

OPT Column	VAL Column			Description
	Oracle	Microsoft SQL	DB2	
nodnrl	0	0	0	Specifies whether the merge procedure disregards remote locations. Valid values: 1 Disregard REMOTELOCATION. 0 Do not disregard REMOTELOCATION.
stuckthreshold	28860	28860	28860	Specifies the time interval, in seconds, at which IS-Link information is expected from another site. After this time period expires, the IS-Link is considered to be stuck.
Note: All entries have column SETID = 0 and column SECT = 'merge'.				

Table 23: Merge Parameters in the G_DB_PARAMETERS Table (Continued)

OPT Column	VAL Column			Description
	Oracle	Microsoft SQL	DB2	
step	75	75	25	<p>Specifies the number of transactions that the merge procedure selects at the same time (when it is reading in new IRs and new links).</p> <p>A low value limits exposure to locks on core tables. However, a very low value can result in too many iterations.</p> <p>For additional information about large-scale deployments using Microsoft SQL, see “Note for Large-Scale Deployments Using Microsoft SQL”.</p>
limit	1000	2000	1000	<p>Specifies the number of root interactions that the merge procedure considers for closure at the same time (when it updates the G_IS_LINK, G_IR, and G_CALL tables).</p> <p>This setting effectively limits the number of IRs per MSEQ. A very low value can result in too many iterations.</p> <p>For additional information about large-scale deployments using Microsoft SQL, see “Note for Large-Scale Deployments Using Microsoft SQL”.</p>
limit2	10000	10000	10000	<p>Specifies the number of new links and interaction records that the merge procedure reads before it begins the closure phase (when it updates the G_IS_LINK, G_IR, and G_CALL tables).</p> <p>A very high value can result in suboptimal performance.</p>
Note: All entries have column SETID = 0 and column SECT = 'merge'.				

Note for Large-Scale Deployments Using Microsoft SQL

With Microsoft SQL, the default values of the `step` and `limit` parameters are not optimal for IDBs with large amounts of data (in the order of millions of interactions). For better performance, Genesys recommends the following as a first step:

- Increase the value of the `step` parameter to 200.
- Increase the value of the `limit` parameter to 3000.

However, you will likely need to experiment to find the optimal values for your large-scale deployment.

Updating the G_DB_PARAMETERS Table

If necessary, update the G_DB_PARAMETERS table. Interaction Concentrator provides a stored procedure, `svcUpdateDBParameters`, to perform this function. The stored procedure requires you to specify values for SECT (always 'merge'), OPT (see the OPT Column in Table 23 on [page 156](#)), and VAL.

For example, to write the IS-Link timeout to the database, execute the following statement (the exact syntax depends on the RDBMS):

```
EXEC svcUpdateDBParameters
0,
'merge',
'stuckthreshold',
'<TIMEOUT>'
```

gsysIRMerge2 Parameters

The `gsysIRMerge2` stored procedure accepts five parameters: two input parameters and three output parameters.

When the `gsysIRMerge2` stored procedure is invoked by a call to the `gsysIRMerge` stored procedure, `gsysIRMerge` supplies the required parameters.

The initial implementation of `gsysIRMerge2` required the parameters for concurrency control, to ensure that no more than one instance of the procedure was running at any given time. Subsequent changes to the merge procedure have relaxed the restrictions for database locking. All the `gsysIRMerge2` parameters have been retained for backward compatibility. However, there is no meaningful difference in the way the merge procedure executes, regardless of the values you enter for the input parameters.

[Table 24](#) lists the `gsysIRMerge2` input and output parameters, as well as the values that are used for the input parameters when `gsysIRMerge2` is called from `gsysIRMerge`.

Table 24: Merge Procedure Parameters

Parameter	Data Type	Description
Input		
OVERRIDE	int	Specifies the lock override setting. Valid values: 0 Do not override lock. 1 Override if PREVCALLER==CALLER. 2 Always override. gsysIRMerge value: 0
CALLER	varchar	Specifies the name of the caller of the stored procedure. Valid values: Any string (for example, Infomart_DBID_1001, DCA6) gsysIRMerge value: 'singleIDBMerge'
Output		
RESULT	int	Specifies the result of the stored procedure call. Valid values: 0 Success. 2 IDB locked. 1 Other error. gsysIRMerge value: RESULT
PREVCALLER	varchar	If RESULT==2, specifies the name of the previous caller. gsysIRMerge value: PREVCALLER
PREVAGE	int	If RESULT==2, specifies the number of seconds since the previous caller obtained the lock. gsysIRMerge value: PREVAGE

The RESULT parameter provides information about the status of execution of the procedure.

Note: The `gsysIRMerge` stored procedure discards the output parameters (RESULT, PREVCALLER, and PREVAGE).

Executing the Merge Procedure

Genesys recommends that you execute the merge procedure periodically, in order to reduce the amount of time that is required for data processing and for the delivery of correct data to other applications—for example, downstream reporting systems.

You can call the merge stored procedure in two ways:

- By calling `gsysIRMerge`
- By calling `gsysIRMerge2`

The following subsections discuss each of these methods in turn.

In Interaction Concentrator 8.0, the only meaningful difference between the two methods is that calling `gsysIRMerge2` returns the result of the merge procedure, as well as other output parameters (see Table 24 on [page 159](#)).

Note: The merge procedure can be executed on voice interaction data only.

Calling `gsysIRMerge`

To execute the merge procedure, use the following statement (the exact syntax depends on the RDBMS):

```
EXEC gsysIRMerge
```

The `gsysIRMerge` stored procedure then, in turn, calls `gsysIRMerge2`, using the following parameters: 1, 'singleIDBMerge', RESULT, PREVCALLER, and PREVAGE.

Calling `gsysIRMerge2`

Invoking the `gsysIRMerge2` stored procedure directly enables you to specify your own values for the input and resulting output parameters. You must supply the required parameters when you call the procedure. (For more information about the input and output parameters, see “`gsysIRMerge2` Parameters” on [page 158](#).)

For each supported RDBMS, there are different syntax requirements for the script that invokes the `gsysIRMerge2` stored procedure directly. This section provides the following examples:

- Example for Oracle
- Example for Microsoft SQL
- Example for DB2

Example for Oracle

```

declare
  OVERRIDE$ int := 1;
  CALLER$ varchar2(40) := 'singleIDBMerge';
  RESULT$ int;
  PREVCALLER$ varchar2(40);
  PREVAGE$ int;
begin
  gsysIRMerge2( OVERRIDE$, CALLER$, RESULT$,
    PREVCALLER$, PREVAGE$ );
  dbms_output.put_line(''||RESULT$);
end;

```

Example for Microsoft SQL

```

declare @OVERRIDE          int
declare @CALLER            varchar(40)
declare @RESULT            int
declare @PREVCALLER        varchar(40)
declare @PREVAGE           int
set @OVERRIDE = 1
set @CALLER = 'singleIDBMerge'
exec gsysIRMerge2 @OVERRIDE, @CALLER, @RESULT, @PREVCALLER, @PREVAGE

```

Example for DB2

```

create procedure gsysIRMergeTest
DYNAMIC RESULT SETS 1
language SQL
begin
  declare OVERRIDE          int default 1;
  declare CALLER            varchar(40) default 'singleIDBMerge';
  declare RESULT            int;
  declare PREVCALLER        varchar(40);
  declare PREVAGE           int;

  call gsysIRMerge2( OVERRIDE, CALLER, RESULT,
    PREVCALLER, PREVAGE );
  begin
    declare c_cur cursor with return for
      select RESULT, PREVCALLER, PREVAGE from sysibm.sysdummy1;
    open c_cur;
  end;

end;
call gsysIRMergeTest;
drop procedure gsysIRMergeTest;

```

Improving Merge Procedure Performance

Database configuration, database settings, and merge procedure scheduling can significantly impact merge procedure performance.

Computing Statistics

Periodically gathering statistics is a generic requirement for good database performance. For optimal performance of the merge procedure, ensure that the available database statistics are representative (in other words, relatively fresh).

Troubleshooting the Merge Procedure

For information about merge procedure execution problems and their solutions, see “Merge Procedure Problems” on [page 187](#).

Note: To minimize the occurrence of deadlocks and to optimize merge procedure performance, carefully review the information in “Setting Up the Merge Procedure” on [page 156](#) and in “Merge Procedure Performance Is Slow or Unstable” on [page 189](#). However, deadlocks are inevitable, and the downstream reporting application must be capable of handling them.

If the procedure raises any exceptions, issue a rollback statement before performing any other actions on the connection.

Resetting the Merge Procedure

For convenience, Interaction Concentrator provides a stored procedure, `gsysIRMergeReset`, that resets the merge procedure to a clean state. Execute the reset procedure if the merge procedure does not complete successfully and you want to rerun it.

To execute the merge procedure, use the following statement (the exact syntax depends on the RDBMS):

```
EXEC gsysIRMergeReset
```

Purge Stored Procedures

The size of IDB is one of the most significant factors that affect Interaction Concentrator performance. To maintain a manageable IDB, you must implement a suitable purging strategy.

Interaction Concentrator provides two sets of stored procedures to purge IDB voice and multimedia data. The purge procedures provide the following functionality:

- Safely purge voice multimedia and outbound interaction data from IDB. Also safely purge agent login sessions and attached data. All logically related information is deleted at the same time.
- Do not affect ICON performance while operating.
- Accept input parameters to limit the range of deleted data.

Note: Interaction Concentrator 7.5 and 7.6 does not support purging of data by Tenant ID.

This section contains the following information about the purge procedures:

- Descriptions of the purge procedures (see “Purge Procedures”)
- Logging and error handling (see [page 166](#) and [page 173](#))
- Scheduling the purge procedures (see [page 174](#))
- Setting up the purge procedures (see [page 175](#))
- Executing the purge procedures (see [page 177](#))

Purge Procedures

Interaction Concentrator 8.0 provides two sets of purge procedures to purge data safely from IDB:

- `gsysPurge80`—Interaction Concentrator v8.0.000.31 and higher provides the `gsysPurge80` stored procedure. (Earlier versions provided `gsysPurge76`.)
The `gsysPurge80` procedure purges from IDB voice, multimedia, attached data, agent login sessions, and in v8.0.000.33 and higher, outbound data.
- `gsysPurgeIR`, `gsysPurgeUDH`, `gsysPurgeLS`, and `gsysPurgeOS`

Notes: Interaction Concentrator release 8.0.000.32 and earlier includes the prior version of the purge procedure, which does not purge Outbound Contact data

Genesys recommends using the `gsysPurge80` purge procedure if one of the following applies:

You are a new ICON 8.0 customer.

You need to purge large amounts of data.

You are concerned about performance.

These stored procedures purge the following data from IDB:

- **gsysPurge80**—Purges voice, eServices/Multimedia, Outbound Contact, 3rd Party Media, attached data, and agent login sessions (see “Using the gsysPurge80 Stored Procedure” on [page 164](#)).
- **gsysPurgeIR**—Purges voice interaction records (IRs) (See “Purging Voice and Logically Related Interaction Data” on [page 171](#)).
- **gsysPurgeUDH**—Purges User Data History (UDH) data (see “Purging User Data History” on [page 172](#)).
- **gsysPurgeLS**—Purges Agent Login Session (ALS) data (see “Purging Agent Login Sessions” on [page 173](#)).
- **gsysPurgeOS**—Purges Outbound data (see “Purging Outbound Sessions Data” on [page 170](#)).

The purge stored procedures are protected from concurrent use. (In other words, you can execute only one instance of a particular primary procedure at any one time).

The purge procedures can run in the background while instances of the ICON process are actively writing data to the IDB.

Retention Period	All the purge procedures accept an input parameter, <code><number of days></code> or <code><count days></code> , which specifies the <i>retention period</i> (the number of days, including the current date, for which data will be left in the database after the purge). Data for days preceding the retention period is eligible for purging. The input parameter must be a positive integer (minimum value = 1, no maximum).
Purge Type	The gsysPurge80 purge procedure accepts a second input parameter, <code><deleteAllFlag></code> , which further defines which data to purge in IDB.

Using the gsysPurge80 Stored Procedure

The **gsysPurge80** procedure purges voice, multimedia, outbound, attached data, and agent login sessions from IDB. Genesys recommends that you use this procedure as your primary purge procedure if you are purging large amounts of data or if you are concerned about ICON performance. The **gsysPurge80** procedure uses two user-specified input parameters, `<number of days>` and `<deleteAllFlag>`, to purge IDB data.

First Input Parameter	The first input parameter, <code><number of days></code> , is used to calculate the retention period for data. It specifies the number of days, including the current one, for which data will be retained. For example, if you run this stored procedure with <code><number of days> = 1</code> , and <code><deleteAllFlag> = 1</code> (delete all records), then all voice, multimedia, open media, attached data, and agent login session interaction data older than one day will be purged from IDB. The valid values for this parameter are positive integers greater than, or equal to, 1.
Second Input Parameter	The second input parameter, <code><deleteAllFlag></code> , further defines which data to purge in IDB. The only valid values for this parameter are 0 and 1, where:

- `<deleteAllFlag> = 0`—ICON deletes only terminated records older than `<number of days>` day(s). Non-terminated records—records that ICON may update in the future—are retained in IDB.
- `<deleteAllFlag> = 1`—ICON deletes all records (voice and multimedia) older than `<number of days>` day(s).

**Optimizes
Performance**

The new `gsysPurge80` stored procedure differs from the `gsysPurgeIR` procedure in that it does not consider merged interactions when purging data. Instead, `gsysPurge80` collects information about existing records in IDB, and then uses this data to optimize its performance in subsequent purges. This data may take considerable time to acquire the first time the purge procedure is executed.

The new purge procedure also provides the option to purge only terminated records using the `<deleteAllFlag>` parameter. This will affect the time it takes to perform the purge operation, because ICON first conditionally deletes terminated records in IDB and then rechecks all records not deleted on the first pass in case previously non-terminated records are now terminated.

Tables Involved in the `gsysPurge80` Procedure

The `gsysPurge80` purge procedure affects data in the following tables:

- | | |
|--|--------------------------------------|
| • <code>GS_AGENT_STAT</code> | • <code>G_VIRTUAL_QUEUE</code> |
| • <code>GS_AGENT_STS_WM</code> | • <code>G_ROUTE_RESULT</code> |
| • <code>G_LOGIN_SESSION</code> | • <code>G_IS_LINK</code> |
| • <code>G_AGENT_STATE_RC</code> | • <code>G_IS_LINK_HISTORY</code> |
| • <code>G_AGENT_STATE_HISTORY</code> | • <code>GX_SESSION_ENDPOINT</code> |
| • <code>G_DND_HISTORY</code> | • <code>G_PARTY</code> |
| • <code>G_CUSTOM_DATA_P</code> | • <code>G_PARTY_HISTORY</code> |
| • <code>G_CUSTOM_DATA_S</code> | • <code>G_PARTY_STAT</code> |
| • <code>G_CUSTOM_STATES</code> | • <code>G_CALL</code> |
| • <code>G_SECURE_USERDATA_HISTORY</code> | • <code>G_CALL_HISTORY</code> |
| • <code>G_USERDATA_HISTORY</code> | • <code>G_CALL_STAT</code> |
| • <code>GM_F_USERDATA</code> | • <code>G_CALL_USERDATA</code> |
| • <code>GM_L_USERDATA</code> | • <code>G_CALL_USERDATA_CUST</code> |
| • <code>G_IR</code> | • <code>G_CALL_USERDATA_CUST1</code> |
| • <code>G_IR_HISTORY</code> | • <code>G_CALL_USERDATA_CUST2</code> |
| • <code>GO_CAMPAIGN</code> | • <code>GO_CAMPPROP_HIST</code> |
| • <code>GO_CHAINREC_HIST</code> | • <code>GO_FIELDHIST</code> |
| • <code>GO_RECORD</code> | • <code>GO_SECURE_FIELDS</code> |
| • <code>GO_CAMPAIGNHISTORY</code> | • <code>GO_CHAIN</code> |
| • <code>GO_CUSTOM_FIELDS</code> | • <code>GO_METRICS</code> |
| • <code>GO_SEC_FIELDHIST</code> | • <code>GOX_CHAIN_CALL</code> |

gsysPurge80 Logging and Error Handling

The gsysPurge80 purge procedure provides logging information about the state and the results of the purge procedure, including the number of records that are flagged for deletion in a partition and the actual number of purged records in a partition.

The gsysPurge80 purge procedure stores information about the purge process in two IDB tables:

- **G_LOG_MESSAGES**— Stores log messages in the MESSAGETEXT and APPNAME fields.
- **G_LOG_ATTRS**—Stores parameters of log messages as attribute pairs in the ATTR_NAME and ATTR_VALUE fields.

G_LOG_MESSAGES

The MESSAGETEXT field in G_LOG_MESSAGES contains information about the action performed by the gsysPurge80 procedure, including:

- The start of the purge procedure (MESSAGE_ID=25922)
- The end of the purge procedure (MESSAGE_ID=25927)
- The start of the next purge procedure action (MESSAGE_ID=25920)
- The end of the purge procedure action (MESSAGE_ID=25930)
- Other purge instance in progress (MESSAGE_ID = 25925)
- Any execution errors (MESSAGE_ID=25925)

The APPNAME field in G_LOG_MESSAGES contains information about the name of the stored procedure that created the record in G_LOG_MESSAGES. It has the format: ICON DB:gsysPurge80

G_LOG_ATTRS

The G_LOG_ATTRS table stores parameters of log messages, generated by the gsysPurge80 procedure, as attribute pairs in the ATTR_NAME and ATTR_VALUE fields of the G_LOG_ATTRS table. The primary key (ID) of the G_LOG_MESSAGES table is used to identify related records in the G_LOG_ATTRS table, where LRID is a foreign key. For example, a record in G_LOG_MESSAGES where ID = <id_value> relates to the corresponding records in G_LOG_ATTR that have the same LRID value: LRID = <id_value>.

[Table 25](#) shows all possible logging messages (MESAGES_TEXT) and attributes (ATTR_NAME, ATTR_VALUE) that the gsysPurge80 purge procedure (APPNAME) generates and stores in the G_LOG_MESSAGES and G_LOG_ATTRS tables.

Note: You can read information from the G_LOG_MESSAGES and G_LOG_ATTRS tables in their entirety by using Genesys Solution Control Interface (SCI), which is the graphical user interface (GUI) component of the Genesys Management Layer.

Table 25: Messages and Attributes Generated by gsysPurge80

G_LOG_MESSAGES Table		G_LOG_ATTRS Table	
MESSAGETEXT	APPNAME	ATTR_NAME	ATTR_VALUE
Purge procedure <purge procedure version> started	ICON DB: GSYSPurge80	MAX_PARTITION_TO_PURGE	<yyyymmdd>
		NUMBER_OF_DAYS_LEFT	<positive integer>
		DELETE_ALL	<0 or 1>
Purge initialization completed	ICON DB: GSYSPurge80	PartitionID	<0>
		Record_Count	<0>
GSYS_MARK_PARTITION mark partition started...	ICON DB: GSYSPurge80	PARTITION_TO_MARK From	<yyyymmdd>
		PARTITION_TO_MARK To	<yyyymmdd>
purging in progress...	ICON DB: GSYSPurge80	MAX_PARTITION_TO_PURGE	<yyyymmdd>
		PARTITION_COUNT	<the number of terminated and non-terminated records in the partition>
		DELETE_ALL	<0 or 1>
GSYS_PURGE_PARTITION purge started...	ICON DB: GSYSPurge80	PartitionID	<yyyymmdd>
		Record_Count	<0>
IDB:Purge-table:<table name> initiated...	ICON DB: GSYSPurge80	PartitionID	<yyyymmdd>
		Record_Count	<the number of terminated and non-terminated records in the table that belongs to the partition>
IDB:Purge-table:<table name> completed...	ICON DB: GSYSPurge80	PartitionID	<yyyymmdd>
		Record_Count	<the actual number of deleted records>
GSYS_PURGE_PARTITION purge completed	ICON DB: GSYSPurge80	PartitionID	<yyyymmdd>
GSYSPurge80 purge completed	ICON DB: GSYSPurge80	MAX_PARTITION_TO_PURGE	<yyyymmdd>
		NUMBER_OF_DAYS_LEFT	<positive non-zero integer>
		DELETE_ALL	<0 or 1>

Consider the following examples:

Example 1 The `G_LOG_ATTRS` table contains two attribute pairs, (`PartitionID/20080627`) and (`Record_Count/2673`), corresponding to the log message, “`GSYS_PURGE_PARTITION purge started...`” in the `G_LOG_MESSAGES` table.

In this example:

- The first attribute pair, (`PartitionID/20080627`), provides the ID of the partition to be purged.
- The second set of attributes, (`Record_Count/2673`), returns the total number of records found in the partition. In this case, 2673 records were found in partition 20080627. The actual number of records purged depends on the value of the `<deleteAllFlag>`.

Example 2 The following is a sample SQL statement which you might use to extract information from `G_LOG_MESSAGES` and `G_LOG_ATTRS` tables:

```
SELECT
G_LOG_MESSAGES.id, G_LOG_MESSAGES.TIMEWRITTEN, G_LOG_MESSAGES.MESSAGE
TEXT, G_LOG_ATTRS.attr_name, G_LOG_ATTRS.attr_value
FROM G_LOG_MESSAGES, G_LOG_ATTRS
WHERE G_LOG_MESSAGES.ID=G_LOG_ATTRS.lrid and G_LOG_MESSAGES.id>0
order by G_LOG_MESSAGES.id;
```

Scheduling gsysPurge80

It is the responsibility of the customer (usually the Database Administrator) to run the `gsysPurge80` procedure as required. Genesys recommends that you run the purge procedure at a time when contact center activity is low.

If you run the purge procedure occasionally, on an ad hoc basis, execution can take significantly longer than if you run it regularly. Also, in this case, ICON performance can be adversely affected while the procedure is executing.

There are no specific restrictions about the order in which you must run the `gsysPurge80` purge procedures.

Setting up gsysPurge80

The `G_DB_PARAMETERS` table stores parameters that the purge procedures use to control their operation. To set up the `gsysPurge80` procedure, ensure that the parameter settings in IDB are suitable for your deployment.

[Table 26](#) lists the parameters and their default values.

Table 26: gsysPurge80 Parameters in the G_DB_PARAMETERS Table

Applicable Procedure	G_DB_PARAMETERS Table		Description
	OPT Column	VAL Column	
gsysPurge80	rowspertransaction	0	gsysPurge80 purges all records in a single table with the same partition ID in one transaction. This is the default value. Note: Values from 1 to 100,000 are considered invalid and are treated the same as a value of 0.
		>100000	Specifies, in number of records, the maximum size of one transaction.
		no value	If no value has been stored in the G_DB_PARAMETERS table, gsysPurge80 sets a default transaction size of 200000 when it is executed, and stores the corresponding record in the G_DB_PARAMETERS table.
Note: All entries have column SETID = 0, and column SECT = 'gsyspurge80'.			

Updating the G_DB_PARAMETERS Table

If necessary, update the G_DB_PARAMETERS table. Interaction Concentrator provides a stored procedure, svcUpdateDBParameters, to perform this function. The stored procedure requires you to specify values for SECT (always 'gsyspurge80'), OPT (always 'rowsperttransaction'), and VAL.

Executing gsysPurge80

Genesys recommends that you execute the gsysPurge80 procedure on a daily basis, in order to reduce the amount of time that is required for data processing and for the delivery of correct data to other applications—for example, downstream reporting systems. For additional scheduling considerations, see “Scheduling gsysPurge80” on [page 168](#).

Calling gsysPurge80

You must supply the <number of days> and <deleteAllFlag> input parameters when you call the gsysPurge80 purge procedure (see “Using the gsysPurge80 Stored Procedure” on [page 164](#)).

To execute the gsysPurge80 stored procedure, use the statement that corresponds to your RDBMS:

On Microsoft SQL

```
EXEC GSYPurge80 <number of days>, <deleteAllFlag>
```

On Oracle

```
EXEC GSYPurge80 (<number of days>, <deleteAllFlag>);  
commit
```

On DB2 `CALL GSYPurge80 (<number of days>, <deleteAllFlag>);`

Examples

The following examples illustrate the syntax required to purge terminated multimedia interaction records older than 30 days:

Microsoft SQL `EXEC GSYPurge80 30, 0`

Oracle `EXEC GSYPurge80 (30, 0);`
 `commit`

DB2 `CALL GSYPurge80 (30, 0)`

Note: In these examples, `gsysPurge80` retains any non-terminated records older than 30 days.

Using Separate Purge Procedures

This section explains how to use the separate `gsysPurgeIR`, `gsysPurgeUDH`, `gsysPurgeLS`, and `gsysPurgeOS` procedures to purge voice interactions, user data history, agent login session, and Outbound Contact data, respectively, from IDB.

Purging Outbound Sessions Data

You can use either `gsysPurgeOS` or `gsysPurge80` to purge outbound-session (OS) data. If you choose to use `gsysPurge80`, review the information in “Using the `gsysPurge80` Stored Procedure” on [page 164](#). The present section describes `gsysPurgeOS`.

The `gsysPurgeOS` procedure calls the following stored procedure:

- `gsysPurge_GOS`—Deletes data related to outbound sessions.

Input Parameter The `gsysPurgeOS` input parameter, `<count days>`, is used to calculate the retention period for outbound-session data. The purge procedure deletes all session-related data for closed outbound campaign sessions that have a termination date earlier than the current date minus `<count days>`. The current date starts at midnight, and the time segment of the current date is ignored.

`gsysPurgeOS` deletes outbound session-related data for closed campaign sessions only. If the parent campaign session is not closed, all corresponding chains will not be deleted.

Tables Involved in the OS Purge Procedure

The following tables contain OS-related data:

- | | |
|----------------------|--------------------|
| • GO_CAMPAIGN | • GO_RECORD |
| • GO_CAMPPROP_HIST | • GO_CHAINREC_HIST |
| • GO_METRICS | • GO_CUSTOM_FIELDS |
| • GO_CHAIN | • GO_SECURE_FIELDS |
| • GO_CAMPAIGNHISTORY | • GO_SEC_FIELDHIST |
| • GOX_CHAIN_CALL | • GO_FIELDHIST |

After the OS purge procedure is completed, there are no references to non-existent data in the tables, except for possible references to non-existent interaction records.

Purging Voice and Logically Related Interaction Data

Note: Genesys recommends using the `gsysPurge80` purge procedure if you are purging large amounts of data, or if you are concerned about ICON's performance. See "Using the `gsysPurge80` Stored Procedure" on [page 164](#).

Purging Voice Interaction Data

The `gsysPurgeIR` procedure, which is used to purge IRs, calls the following stored procedures:

- `gsysPurge_GCC`—Deletes IR data.
- `gsysPurge_GUD`—Deletes user data.

`gsysPurgeIR` deletes merged IRs only.

Input Parameter The `gsysPurgeIR` input parameter, `<count days>`, is used to calculate the retention period for IR-related data. The purge procedure deletes all IR-related data for merged IRs that have a merge date earlier than the current date minus `<count days>`. The current date starts at midnight, and the time segment of the current date is ignored.

For example, if the date and time at which the stored procedure is invoked is 05/05/2007 13:15 and the input parameter is 1, all IRs that have a merge date earlier than 05/04/2007 00:00 will be deleted.

Tables Involved in the gsysPurgeIR Purge Procedure

The following tables contain IR-related data. Depending on the deployment, all the tables may not be populated:

- | | |
|---------------------|-------------------------|
| • G_IR | • G_ROUTE_RESULT |
| • G_IR_HISTORY | • G_VIRTUAL_QUEUE |
| • G_CALL | • G_CALL_STAT |
| • G_CALL_HISTORY | • G_PARTY_STAT |
| • G_PARTY | • G_CALL_USERDATA |
| • G_PARTY_HISTORY | • G_CALL_USERDATA_CUST |
| • G_IS_LINK | • G_CALL_USERDATA_CUST1 |
| • G_IS_LINK_HISTORY | • G_CALL_USERDATA_CUST2 |

After the IR purge procedure is completed, there are no references to non-existent data in the tables. For example, the G_PARTY_STAT table will not contain references to a party that has been purged from the G_PARTY table.

Purging User Data History

Note: Genesys recommends using the gsysPurge80 purge procedure if you are purging large amounts of data, or if you are concerned about ICON's performance. See "Using the gsysPurge80 Stored Procedure" on [page 164](#).

The gsysPurgeUDH procedure is used to purge User Data History (UDH) records. The procedure deletes records that are related to merged IRs only.

Input Parameter

The gsysPurgeUDH input parameter, <count days>, is used to calculate the retention period for UDH data. The purge procedure deletes all UDH records that satisfy one of the following conditions:

- The merge date of the corresponding IR is earlier than the current date minus <count days>.
- No corresponding IR exists, and the date the history record was added is earlier than the current date minus <count days>.

The current date starts at midnight, and the time segment of the current date is ignored.

Tables Involved in the UDH Purge Procedure

The following tables contain UDH data. Depending on the roles and the attached data specification configured for the ICON application(s) writing to IDB, the tables may not be populated.

- G_SECURE_USERDATA_HISTORY
- G_USERDATA_HISTORY

After the UDH purge procedure is completed, there are no references to non-existent data in the tables.

Purging Agent Login Sessions

Note: Genesys recommends using the `gsysPurge80` purge procedure if you are purging large amounts of data, or if you are concerned about ICON's performance. See "Using the `gsysPurge80` Stored Procedure" on [page 164](#).

The `gsysPurgeLS` procedure, which is used to purge Agent Login Session (ALS) data, calls the following stored procedure:

- `gsysPurge_GLS`—Deletes sessions.

`gsysPurgeLS` deletes closed sessions only.

Input Parameter The `gsysPurgeLS` input parameter, `<count days>`, is used to calculate the retention period for ALS-related data. The purge procedure deletes all session-related data for closed agent login sessions that have a termination date earlier than the current date minus `<count days>`. The current date starts at midnight, and the time segment of the current date is ignored.

Tables Involved in the ALS Purge Procedure

The following tables contain ALS-related data:

- | | |
|--------------------------------------|------------------------------------|
| • <code>G_LOGIN_SESSION</code> | • <code>GS_AGENT_STAT</code> |
| • <code>G_AGENT_STATE_HISTORY</code> | • <code>GS_AGENT_STAT_WM</code> |
| • <code>G_AGENT_STATE_RC</code> | • <code>GX_SESSION_ENDPOINT</code> |
| • <code>G_DND_HISTORY</code> | |

After the ALS purge procedure is completed, there are no references to non-existent data in the tables, except for possible references to non-existent IRs.

Logging and Error Handling

G_LOG_MESSAGES

The `G_LOG_MESSAGES` table logs the following information about the state of a purge procedure operation:

- `purge stored procedures Started`—The primary procedure has started.
- `ERROR`—A database error has occurred. (The `G_LOG_ATTRS` table provides a description of the error.)
- `Completed [OK|ERROR|Locked]`—The primary procedure has ended or stopped. `Completed: Locked` means that the primary procedure was terminated because a prior instance of the procedure was still running.

G_LOG_ATTRS

The `G_LOG_ATTRS` table contains the following information about a purge procedure operation:

- Error descriptions (code and message).
- The value of the `<count days>` input parameter.

- Information about the processed data, including the number of records purged from the main table (for each primary purge procedure) and the number of records purged from all tables.

You can read information from the G_LOG_MESSAGES and G_LOG_ATTRS tables by using Genesys Solution Control Interface (SCI), which is the graphical user interface (GUI) component of the Genesys Management Layer.

Auto-Recovery

The purge procedures use a number of temporary tables and also update additional indexes during execution. If an error occurs during execution of a purge procedure, the procedure is terminated, but no special action is required to reset the procedure. The next time the procedure is started, the entry point is calculated from minimum merge sequence or timestamp values in the applicable IDB tables.

Timeout Interval A maximum transaction time parameter sets a timeout interval for the procedure lock. If a purge procedure terminates because of an unhandled error, the lock will be overridden if the next instance of the procedure starts after the timeout interval has expired. For more information about the maximum transaction time parameter, see “Setting Up the Separate Purge Procedures” on [page 175](#).

Scheduling the Purge Procedures

Note: To schedule the new gsysPurge80 purge procedure, refer to “Scheduling gsysPurge80” on [page 168](#).

It is the responsibility of the customer (usually the Database Administrator) to run the purge procedures as required. Genesys recommends that you run the purge procedures at a time when contact center activity is low.

If you run the purge procedure occasionally, on an ad hoc basis, execution can take significantly longer than if you run it regularly. Also, in this case, ICON performance can be adversely affected while the procedure is executing.

There are no specific restrictions about the order in which you must run the purge procedures. However, for best performance and to maintain data consistency throughout the process, Genesys recommends that you run the purge procedures sequentially, in the following order:

- gsysPurgeIR (purge IRs)
- gsysPurgeUDH (purgeUDH data)
- gsysPurgeLS (purge ALS data)
- gsysPurgeOS (purge OS data)

In particular, if you are purging more than one day’s worth of data, execute the purge procedures sequentially.

Note the following additional considerations and recommendations:

- If you are running the purge procedures sequentially, you may need to adjust the `<count days>` input parameter for purge procedures that execute later in the sequence, because the procedures may start after midnight. For example, if the procedure to purge IRs starts at 05/05/2007 20:00 (08:00 PM) and the `<count days>` input parameter is 1, set the input parameter for all procedures that will start after 05/05/2007 23:59:59 (11:59:59 PM) to 2.
- Because the `gsysPurgeIR` purge procedures purge merged interactions only, consider your merge procedure schedule when scheduling the purge procedures. In particular, note that, even in a single-site deployment, you must run the merge procedure before you will be able to purge any records from IDB.
- Consider the extraction, transformation, and loading (ETL) cycle of your downstream reporting application, to ensure that you do not delete data before your downstream reporting application has successfully extracted the data.

In general, schedule the purge procedures to run after your downstream reporting application has completed its regular ETL cycle. Keep an appropriately large sliding window of data, and delete only the oldest ETL cycle's worth of data each time you run the purge procedures.

Your downstream reporting application may provide functionality that spreads the extraction of a backlog of data across multiple ETL cycles (for example, the `limit-extract-data` configuration option in Genesys Info Mart). If you are exercising this functionality, be aware that, at the end of a particular ETL cycle, your downstream reporting application may not have extracted all the data that was available at the time of extraction.

Note: With Genesys Info Mart as your downstream reporting application, it is possible to run the purge procedures in parallel with the Genesys Info Mart `Job_AggregateGIM` and `Job_MaintainGIM` jobs.

- If your downstream reporting application does not successfully complete an ETL job cycle, ensure that you stop invoking the purge procedures until the problem is fixed and the ETL job cycle starts running successfully again. Otherwise, you may delete IDB source data before the downstream reporting application has extracted it.

Setting Up the Separate Purge Procedures

Note: To set up the `gsysPurge80` purge procedure, refer to “Setting up `gsysPurge80`” on [page 168](#).

The G_DB_PARAMETERS table stores parameters that the purge procedures use to control their operation. To set up the purge procedures, ensure that the parameter settings in IDB are suitable for your deployment.

Table 23 on [page 156](#) lists the parameters and their default values.

Table 27: Purge Parameters in the G_DB_PARAMETERS Table

Applicable Procedure	G_DB_PARAMETERS Table		Description
	OPT Column	VAL Column	
gsysPurgeIR	IR_seq_step	75	Specifies the chunk size, in terms of number of ICON transactions, that will be used for data lookup.
gsysPurgeUDH	UDH_time_step	300	Specifies the chunk size, in seconds, that will be used to calculate the number of rows to purge in a single database transaction.
gsysPurgeLS	LS_time_step	500	Specifies the chunk size, in seconds, that will be used to calculate the number of rows to purge in a single database transaction.
gsysPurgeOS	OS_time_step	500	Specifies the chunk size, in seconds, that will be used to calculate the number of rows to purge in a single database transaction.
gsysPurgeIR	IR_max_tran_time	600	Specifies the timeout interval for the procedure lock, in seconds. The maximum transaction time is the amount of time between the most recent purge transaction and the current time, after which a new instance of the procedure will be allowed to execute.
gsysPurgeUDH	UDH_max_tran_time		
gsysPurgeLS	LS_max_tran_time		
gsysPurgeOS	OS_max_tran_time		
Note: All entries have column SETID = 0 and column SECT = 'purge'.			

Updating the G_DB_PARAMETERS Table

If necessary, update the G_DB_PARAMETERS table. Interaction Concentrator provides a stored procedure, svcUpdateDBParameters, to perform this function. The stored procedure requires you to specify values for SECT (always 'purge'), OPT (see the OPT Column in Table 23 on [page 156](#)), and VAL.

For example, to change the value of the procedure lock timeout for the IR purge procedure, execute the following statement (the exact syntax depends on the RDBMS):

```
EXEC svcUpdateDBParameters
0,
'purge',
```



```
'IR_max_tran_time',
'<TIMEOUT>'
```

Executing the Separate Purge Procedures

Note: To execute the `gsysPurge80` purge procedure, refer to “Executing `gsysPurge80`” on [page 169](#).

Calling Purge Stored Procedures

You must supply the `<count days>` input parameter when you call the `gsysPurgeIR`, `gsysPurgeUDH`, `gsysPurgeLS`, and `gsysPurgeOS` purge procedures.

For each supported RDBMS, there are different syntax requirements for the script that invokes a purge procedure. This section provides the following examples:

- Example for Oracle (see [page 177](#))
- Example for Microsoft SQL (see [page 177](#))
- Example for DB2 (see [page 177](#))

In these examples, the IR purge and ALS purge procedures will be executed sequentially in the same session, and will use different values for the retention period.

Example for Oracle

```
declare
  COUNT_DAYS$ int := 14;
begin
  gsysPurgeIR( COUNT_DAYS$ );
end;

declare
  COUNT_DAYS1$ int := 15;
begin
  gsysPurgeLS( COUNT_DAYS1$ );
end;
```

Example for Microsoft SQL

```
declare @COUNT_DAYS int
set @COUNT_DAYS = 14
exec gsysPurgeIR @COUNT_DAYS

declare @COUNT_DAYS1 int
set @COUNT_DAYS1 = 15
exec gsysPurgeLS @COUNT_DAYS1
```

Example for DB2

```
language SQL
begin
  declare COUNT_DAYS int default 14;
```

```

        call gsysPurgeIR( COUNT_DAYS );

end;

language SQL
begin
    declare COUNT_DAYS1 int default 15;

    call gsysPurgeLS( COUNT_DAYS1 );

end;

```

Stored Procedure gsysInitTimeCode

ICON uses the `gsysInitTimeCode` stored procedure to populate the `G_TIMECODE` table. [Figure 15](#) shows the three tables that are involved in the time-setting procedure.

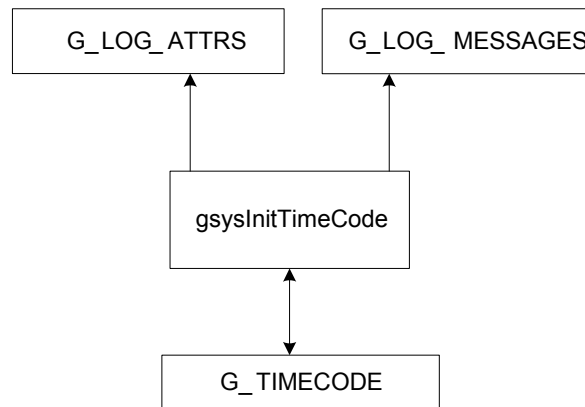


Figure 15: Tables Involved in the Time-Setting Procedure

The following subsections describe each of these tables, and how they are used in the time-setting procedure.

G_LOG_MESSAGES and G_LOG_ATTRS

G_LOG_MESSAGES

The `G_LOG_MESSAGES` table contains information about the start (`MESSAGE_ID = 5040`) and end (`MESSAGE_ID = 5050`) of the time-setting procedure, as well as information about any errors (`MESSAGE_ID = 5045`) that occurred while the procedure was running.

G_LOG_ATTRS

The `G_LOG_ATTRS` table contains the error descriptions (code and message) and detailed information about the processed data.

You can read information from these two tables by using Genesys SCI.

G_TIMECODE

After the time-setting procedure is executed, the G_TIMECODE table is filled, based on the input parameters for the procedure. You can use the G_TIMECODE table to create time-interval reports. The ID field in this table is related to the *_TCODE fields in other tables, and it represents the amount of time, in seconds, counted in five-minute intervals, since January 1, 1970.

Note: The gsysInitTimeCode procedure does not populate the TC_WEEKDAY, TC_WEEK, TC_DAYNAME, TC_WEEKNAME, and TC_MONTHNAME columns in the G_TIMECODE table.

Setting Up the Time-Setting Procedure

No setup is required in order to execute the time-setting procedure.

Executing the Time-Setting Procedure

Execute the time-setting procedure as often as required, using the following input parameters:

- BEGIN_DATE—The date of the first interval.
- END_DATE—The date of the last interval.

To execute the time-setting procedure, use the following statement (the exact syntax depends on the RDBMS):

```
EXEC gsysInitTimeCode  
getdate(),  
getdate()+365
```

Custom Dispatchers

To support customized attached data processing, IDB initialization scripts create two custom dispatcher stored procedures:

- gudCustDISP1
- gudCustDISP2

You must modify the scripts in order to create custom dispatchers that store the attached data that you require.

For more information about the custom dispatchers, see “Custom Dispatchers” on [page 68](#).



Chapter

17

Troubleshooting ICON Installation and Deployments

This chapter describes problems that you might encounter when starting or running your Interaction Concentrator (ICON) application, and how to resolve them. It contains the following sections:

- [Restrictions, page 181](#)
- [Startup Problems, page 181](#)
- [Runtime Problems, page 183](#)
- [Merge Procedure Problems, page 187](#)

Restrictions

To avoid a wide range of startup and runtime problems, observe the following restrictions:

- Do not disconnect ICON from Configuration Server during startup.
- Do not change any connections on the Connections tab of the ICON Application during runtime.

Startup Problems

The following are the most common startup problems:

- ICON does not connect to the Configuration Server (see [“No Connection to the Configuration Server”](#) below).
- ICON closes at startup (see [“ICON Exits at Startup”](#) on [page 182](#)).

No Connection to the Configuration Server

Possible causes of this problem are as follows:

- Command-line parameters on the ICON Application object's Server Info tab incorrectly specify the Configuration Server host and port.

Solution: Correct the command-line parameters and restart the application. For more information about the command-line parameters, see "Command-Line Parameters" on [page 123](#).

- Configuration Server is not running, or it is inaccessible over the network.

Solution: Start Configuration Server or re-establish the network connection.

ICON Exits at Startup

See the ICON log file for the reasons for the startup failure. Possible reasons include:

- The application name specified in the ICON startup command line does not correspond to any existing Application object in the Configuration Layer.

Solution: Create the Application object. For more information about creating and configuring the ICON Application, see the installation and configuration chapter in the *Interaction Concentrator 8.0 Deployment Guide*.

- The application name specified in the ICON startup command line refers to an Application object that is not of the Call Concentrator application type.

Solution: Remove the Application object of the incorrect type, and then use the correct template to create a new Application object of the Call Concentrator type. For more information about creating and configuring the ICON Application object, see the installation and configuration chapter in the *Interaction Concentrator 8.0 Deployment Guide*.

- There is no assignment to a Database Access Point (DAP) Application object on the Connections tab of the ICON Application object.

Solution: Add to the ICON Application object's Connections tab any DAP Application objects through which this ICON instance will access Interaction Databases (IDBs).

- The DAP Application object assigned on the ICON Application object's Connections tab does not have an associated DB Server application.

Solution: Associate a DB Server with the DAP Application object. For more information, see the installation and configuration chapter in the *Interaction Concentrator 8.0 Deployment Guide*.

- The ICON instance has been configured to process call attached data (`role = gud`), but ICON cannot open the file specified in the `adata-spec-name` configuration option. The following error message in the log file indicates the existence of this condition:

```
Std 02016 Unable to open attached data file '<attached data
specification file name>', error code XXX
```

Solution: Verify the following and correct as required.

- The file specified in the `adata-spec-name` configuration option exists. If the file does not exist, create a new one or use the default attached data specification file (`ccon_adata_spec.xml`) provided in the Interaction Concentrator installation package.
- The Interaction Concentrator user (the account under which ICON has been started) has the required permissions to read the attached data specification file.
- The persistent queue file has become corrupted.

Solution: Force ICON to create a new persistent queue file by doing one of the following:

- Using operating system commands, move or rename the corrupted `.pq` file. On restart, ICON will create a new `.pq` file with the original file name in the original location.
- Reset the `pq-dbname` configuration option in the `ICON Application` object. On restart, ICON will create a new `.pq` file with the new file name in the specified location. For more information about the `pq-dbname` configuration option, see the configuration chapter in the *Interaction Concentrator 8.0 Deployment Guide*.

In both cases, all unprocessed data in the old `.pq` file will be lost to ICON and IDB.

- There is no free disk space on the disk where the `apstorage.db` file resides.

Solution: Free up memory on the disk or add more disk memory. For more information about the `apstorage.db` file, see “Populating Agent Login Session Data” on [page 88](#).

Runtime Problems

The following are the most common runtime problems:

- ICON does not connect to T-Server or Interaction Server (see “No Connection to T-Server or Interaction Server” on [page 184](#)).
- ICON does not receive call-related events from T-Server (see “ICON Does Not Receive Call-Related Events from T-Server” on [page 185](#)).
- ICON does not write information to the database (see “ICON Does Not Write Information to the Database” on [page 185](#)).

- ICON has lost synchronization with the Configuration Database (see “ICON Has Lost Synchronization with the Configuration Database” on [page 187](#)).
- ICON functionality is not available as configured, or ICON functioning becomes unpredictable (see “ICON Does Not Function Correctly” on [page 187](#)).

No Connection to T-Server or Interaction Server

Possible causes of this problem are as follows:

- There is no assignment to the T-Server Application object or the Interaction Server Application object on the ICON Application object's Connections tab.

Solution: Add to the ICON Application object's Connections tab any T-Server or Interaction Server Application objects from which this ICON instance will receive interaction-related information.

- The T-Server or Interaction Server application is not running, or it is not accessible over the network.

Solution: Start the application or re-establish the network connection.

- The T-Server or Interaction Server Application object cannot connect to its Switch link.

Solution: See the applicable troubleshooting guide for your particular T-Server or Multimedia Interaction Server.

- The release of the T-Server or Interaction Server Application object is not compatible with Interaction Concentrator. T-Server release 7.2 is the minimum version required by any release of Interaction Concentrator. Multimedia Interaction Server release 7.5 is the minimum version required for Interaction Concentrator support of eServices/Multimedia. For more information about Interaction Concentrator compatibility and interoperability with other Genesys components, see the section about compatibility in the planning chapter in the *Interaction Concentrator 8.0 Deployment Guide*.

Solution: Upgrade the T-Server or Interaction Server Application object to a compatible release.

- The Switch object associated with the T-Server Application object does not have all the necessary DN objects configured.

Solution: Create the DN objects. For more information, see the *Deployment Guide* for your particular T-Server.

ICON Does Not Receive Call-Related Events from T-Server

Possible causes of this problem are as follows:

- ICON was not restarted after changes were made on the ICON Application object's Connections tab.

Solution: Stop ICON, then restart. For more information, see “Starting and Stopping Interaction Concentrator” on [page 121](#).

- ICON was not restarted after a backup instance was configured of a T-Server to which Interaction Concentrator has a connection configured on the Connections tab.

Solution: Stop ICON, then restart. For more information, see “Starting and Stopping Interaction Concentrator” on [page 121](#).

- There is no connection between the ICON Application object and T-Server. See “No Connection to T-Server or Interaction Server” on [page 184](#).

ICON Does Not Write Information to the Database

Possible causes of this problem are as follows:

- The database parameters are incorrectly specified on the DAP Application object. These parameters include the user name and password.

Solution: Specify the correct values on the DAP Application object's DB Info tab, then restart ICON. For more information, see the configuration and installation chapter in the *Interaction Concentrator 8.0 Deployment Guide*.

- DB Server is not running, or it is inaccessible over the network.

Solution: Start DB Server or re-establish the network connection.

- The RDBMS server is not available, or the IDB to which DB Server is trying to connect is not available.

Solution: Take the necessary steps to make the database server and database available.

- The DAP Application object has been configured for a role that prevents it from writing certain classes of information to the database.

Solution: Reconfigure the role option for the DAP Application object. Restart ICON. For more information about configuring a DAP, see the configuration and installation chapter in the *Interaction Concentrator 8.0 Deployment Guide*. For a description of the role option, refer to the chapter about configuration options in the *Interaction Concentrator 8.0 Deployment Guide*.

- IDB has not been initialized by the Interaction Concentrator initialization scripts.

Solution: Run the Interaction Concentrator initialization scripts. For more information, see the configuration and installation chapter in the *Interaction Concentrator 8.0 Deployment Guide*.

- ICON was not restarted after changes were made on the ICON Application object's Connections tab.

Solution: Stop ICON, then restart. For more information, see “Starting and Stopping Interaction Concentrator” on [page 121](#).

- ICON was not restarted after a backup instance was configured of a T-Server to which Interaction Concentrator has a connection configured on the Connections tab.

Solution: Stop ICON, then restart it. For more information, see “Starting and Stopping Interaction Concentrator” on [page 121](#).

- Records are accumulating in the in-memory queue and are not being written to IDB.

Solution: This might not be a problem. Configuration options control whether a size threshold or timeout triggers the transfer of records from the in-memory queue to the persistent queue, from which the records are then written to IDB. Wait for the event that triggers the transfer, and re-evaluate your configuration as necessary. For more information about in-memory queue configuration options, refer to the chapter about configuration options in the *Interaction Concentrator 8.0 Deployment Guide*.

- The program logic consistently produces an error because of incorrect RDBMS settings. For example, there may be insufficient free space available on the RDBMS for data storage, or the rollback segment may be too small.

Solution: Review the error messages reported in the ICON log file. If you have configured an HTTP Listener, you can also view the error messages on the Database Writer performance counter web page (for more information, refer to “Monitoring Interaction Concentrator” on [page 131](#)). Provide the appropriate fix on the RDBMS side. For example, if the error messages cite no free space available for data storage, increase the table space.

If the error was entirely related to the RDBMS problem, you do not need to restart ICON or perform any manipulation of the persistent queue (.pq file). However, if the .pq file has become corrupted and there are additional errors in the program logic, you must replace the .pq file (see the problem about a corrupted persistent queue file on [page 183](#)).

ICON Has Lost Synchronization with the Configuration Database

There are a number of reasons why ICON might lose synchronization with the Configuration Database, especially following a shutdown of ICON.

Loss of synchronization has the following impact on IDB:

- ICON fails to capture data about configuration objects created while ICON was stopped.
- ICON does not mark configuration data as deleted in cases where the applicable configuration objects were deleted while ICON was stopped.
- ICON fails to capture changes in associations between objects (while it is stopped).

Solution: If you suspect that your configuration data in IDB is inconsistent with Configuration Database, perform a manual resynchronization. See “How to Resynchronize Configuration Data” on [page 146](#) for detailed instructions.

ICON Does Not Function Correctly

Possible causes of this problem are as follows:

- A connection configured on the **Connections** tab of the ICON Application was removed or changed while ICON was operating.

Solution: Stop ICON. Verify that the connections that have been configured on the **Connections** tab of the ICON Application object are as required for the deployment, then restart ICON. For more information about configuring connections, see the step about configuring the **Connections** tab, in the procedure to configure the ICON application object in the *Interaction Concentrator 8.0 Deployment Guide*.

Merge Procedure Problems

The most common problems encountered in executing the merge procedure (gsysIRMerge or gsysIRMerge2) are as follows:

- The procedure fails to complete (see “[Merge Procedure Does Not Complete Successfully](#)” below).
- The procedure does not execute (see “[Merge Procedure Does Not Execute](#)” on [page 188](#)).
- Merge procedure performance is slow or constantly aborts (see “[Merge Procedure Performance Is Slow or Unstable](#)” on [page 189](#)).

For more information about the merge procedure, see “[Merge Stored Procedure](#)” on [page 152](#).

Merge Procedure Does Not Complete Successfully

In general, the most likely reason the merge procedure fails is an inconsistency in IDB. The database inconsistency might be introduced by ICON, by the downstream reporting application, through manual intervention, or in some other way. For example, if ICON writes a duplicate `G_IS_LINK` record while the merge procedure is executing, the RDBMS might report a primary key violation. Describing the possible causes of this problem in detail is beyond the scope of this document.

The following tables store information about the state of the merge procedure:

- `GSYS_PENDING_IR`
- `GSYS_PENDING_LINK`
- `GSYS_SYSPROCINFO`

Solution: Review the error messages reported in the ICON log file, and take appropriate action to resolve the cause of the failure. You might also have to reset the merge procedure so that it recovers from its failed state (see [“Merge Procedure Recovery”](#)). Then restart the merge procedure.

Merge Procedure Recovery

~~Starting with release 8.0,~~ Interaction Concentrator provides a stored procedure, `gsysIRMergeReset`, to simplify the steps to reset the merge procedure to recover from a failed state.

To invoke the procedure, use an SQL statement like the following (the exact syntax depends on the RDBMS):

```
EXEC gsysIRMergeReset
```

Note: Under some circumstances, merge procedure recovery is not required. For example, the merge procedure may fail to complete successfully as a result of a deadlock condition. In this case, no special action is required other than to run the procedure again. However, if an error is discovered in the merge procedure, execute the stored procedure to reset the merge procedure.

Merge Procedure Does Not Execute

Possible causes of this problem are as follows:

- The stored procedure was called incorrectly.

Solution: Verify the syntax of the call to execute `gsysIRMerge` or `gsysIRMerge2`, and correct the execution command as required. For more information, see [“Executing the Merge Procedure”](#) on [page 160](#).

- There is an error in the database or in database performance that is not specifically related to the merge procedure or to ICON—for example, insufficient disk space or insufficient privileges.

Solution: Review the error messages reported in the ICON log file. Provide the appropriate fix that the RDBMS requires, then restart the merge procedure.

The database error might be related to an inconsistency in IDB, in the sense that it was exposed or induced by an inconsistency in IDB, or resulted in an inconsistency in IDB. In these cases, reset the merge procedure (see [“Merge Procedure Recovery”](#) above), then restart the merge procedure. If the merge procedure still fails to execute, contact Genesys Technical Support.

Merge Procedure Performance Is Slow or Unstable

Possible causes of this problem are as follows:

- On a DB2 platform, default values of certain database parameters result in an excessive number of deadlocks.

Solution: Significantly increase the values of the LOCKLIST and MAXLOCKS database configuration parameters.

Alternatively, execute ALTER TABLE...LOCKSIZE TABLE statements against the G_IR, G_CALL, and G_IS_LINK tables.

- On a Microsoft SQL platform, in large-scale deployments, default values of certain merge procedure parameters are not optimal.

Solution: Significantly increase the values of the step and limit parameters in the G_DB_PARAMETERS table. For more information, see [“Note for Large-Scale Deployments Using Microsoft SQL”](#) on [page 157](#).

- There is an inconsistency in IDB that does not cause the merge procedure to fail, but that significantly interferes with merge procedure performance.

Solution: Reset the merge procedure (see [“Merge Procedure Recovery”](#) on [page 188](#)), then restart the merge procedure. If the problem persists, review database settings and try general database tuning adjustments. If the problem still persists, contact Genesys Technical Support.



Supplements

Related Documentation Resources

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

Interaction Concentrator

- The *Interaction Concentrator 8.0 Deployment Guide*, which will help you install and configure Interaction Concentrator.
- The *Interaction Concentrator 8.0 Physical Data Model* for your relational database management system (RDBMS) type, which will help you learn about IDB tables.
- The *Interaction Concentrator 8.0 Database Size Estimator*, which will help you estimate the size of your IDB when you are planning your deployment. The estimator is a Microsoft Excel spreadsheet available from the Genesys Technical Support website.

Optional Supplementary Solutions

- The documentation set for Genesys Info Mart release 8.0, if you intend to use Interaction Concentrator as a source of data for Genesys Info Mart.
- The documentation set for Genesys Outbound Contact release 8.0, if you intend to store outbound-related data in IDB.
- The documentation set for Genesys Universal Routing release 8.0, if you intend to store information about virtual queue usage in interaction processing in IDB.
- The documentation set for Genesys Multimedia release 8.0, if you intend to store interaction-related and related data about Multimedia interactions in IDB.

Genesys

- *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.
- *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.
- 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:

- *Genesys Hardware Sizing Guide*, which contains information about recommended hardware architectures and additional information related to database size estimation for Genesys 8.x releases.
- *Genesys 8.0 Combined Log Events Help*, which describes the log events generated by every Genesys server application, including Interaction Concentrator.

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:

80fr_ref_06-2008_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, may 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.

Type Styles

[Table 28](#) describes and illustrates the type conventions that are used in this document.

Table 28: Type Styles

Type Style	Used For	Examples
Italic	<ul style="list-style-type: none"> Document titles Emphasis Definitions of (or first references to) unfamiliar terms Mathematical variables <p>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 194).</p>	<p>Please consult the <i>Genesys Migration Guide</i> for more information.</p> <p>Do <i>not</i> use this value for this option.</p> <p>A <i>customary and usual</i> practice is one that is widely accepted and used within a particular industry or profession.</p> <p>The formula, $x + 1 = 7$ where x stands for . . .</p>

Table 28: Type Styles (Continued)

Type Style	Used For	Examples
Monospace font (Looks like teletype or typewriter text)	<p>All programming identifiers and GUI elements. This convention includes:</p> <ul style="list-style-type: none"> The <i>names</i> 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. <p>Also used for any text that users must manually enter during a configuration or installation procedure, or on a command line.</p>	<p>Select the Show variables on screen check box.</p> <p>In the Operand text box, enter your formula.</p> <p>Click OK to exit the Properties dialog box.</p> <p>T-Server distributes the error messages in EventError events.</p> <p>If you select true for the inbound-bsns-calls option, all established inbound calls on a local agent are considered business calls.</p> <p>Enter exit on the command line.</p>
Square brackets ([])	<p>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.</p>	<pre>smcp_server -host [/flags]</pre>
Angle brackets (< >)	<p>A placeholder for a value that the user must specify. This might be a DN or a port number specific to your enterprise.</p> <p>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.</p>	<pre>smcp_server -host <confighost></pre>



Index

Symbols

[] (square brackets)	194
< > (angle brackets)	194

A

accessing performance counter pages	131
accumulator	39
ACDQ, defined	80
ACW (after-call work) agent state	21, 82
After Call Work (agent state)	21, 82
agent login session	
See login session	84
agent states	
ACW	21, 82
and media types	81, 86
busy	82
custom	21, 22
data	85
defined	80
finite state machine	83
for multimedia interactions	81
for voice calls	81
IDB tables	30
login	81
model	81
multimedia reporting	80
NotReady	21, 82
null	81
pending	80
ready	82
reasons	80
restoration	83
statistics	86
transitions	80
T-Server reporting	80
unknown	82
See also agent metrics	
Agent's Place endpoint	56
AgentSessionIDs	117

alarms	19
angle brackets	194
apstorage.db	19, 88
attached data	
call-specific	64
historical	64
IDB tables	31, 66
multimedia	32
post-routing user data processing	65
replicated in multi-site deployments	67
security	67
attached data processing	
customized	67
multimedia	68
voice	64
attached data specification file	63
AttributeCallUUID attribute	45
AttributeCtrlParty attribute	45
attributes	
AttributeCallUUID	45
AttributeCtrlParty	45
audience, for document	12
availability	
of data	50
of IDB	51

B

brackets	
angle	194
square	194
busy (agent state)	82

C

call	
data in IDB tables	26
identifier, globally unique	21
metrics	27
Call Concentrator	99

- Call Concentrator application type 182
- CALLID field 44
- CAUSE field
 - and dictionary information 37
 - in G_VIRTUAL_QUEUE table 73
- cfg-sync.db 19, 41
- character encoding, multi-byte 22
- chat interactions
 - See multimedia
- commenting on this document 12
- common data
 - call/party associations 99
 - data 100
 - IDB tables 100
- configuration data
 - DNS 56
 - HA extraction 116
 - IDB tables 56
 - persistent cache 19, 41
 - places 56
 - script objects 56
- configuration options
 - gls-max-inactivity 84
 - gud-cust-disp 68
 - gud-cust-disp-groups 67
 - releasing-party-report 46
 - store-releasing-party 24
- Configuration Server
 - and IDB synchronization 19, 41
 - data source 18
 - interface with ICON server 19
- configuration synchronization
 - persistent cache 19, 41
 - troubleshooting 187
- configuration tuning 131
- connections
 - dropped, consequences 106
 - during runtime 122, 187
 - during startup 122
 - monitoring information 132
 - Outbound Contact Server 92
- console window
 - stopping ICON (Windows) 129
- conventions
 - in document 193
 - type styles 193
- custom dispatchers
 - about 20, 67, 68
 - sample scripts 68
- custom states
 - call/party associations 99
 - data 100
 - defined 99
 - feature 21
 - IDB tables 31, 100
 - new in release 7.5 22

- stuck 100
- customer-created
 - fields, not supported 100
 - tables, not supported 100
- customer-defined
 - keys 100
 - states 99
- customized attached data
 - IDB tables 68
 - processing 67
 - See *also* custom dispatchers

D

- data
 - availability, determining 50
 - Do Not Disturb usage 86
 - HA merge results 112
 - interrupted 110
 - reliability, determining 53, 116
 - security 67
 - sources of interruption 105
 - storage, of multimedia interactions 58
- Database Writer Page 132
- database writing
 - monitoring 132
 - troubleshooting 185
- DB Server 17
- deleteAllFlag input parameter 164
- deployment scenarios, Outbound Contact 95
- DND
 - See Do Not Disturb
- DNIS
 - identifying 45
- DNS
 - configuration data 56
 - multimedia switch 56
- Do Not Disturb
 - history 31
 - state changes, SIP chat 84
 - usage data 86
- document
 - audience 12
 - conventions 193
 - errors, commenting on 12
 - version number 193
- dropped server connections 106

E

- e-mail interactions
 - See multimedia
- encoding, multi-byte characters 22
- end message
 - merge procedure 153

purge procedures	173
time-setting procedure	178
endpoints	
Agent's Place	56
and login sessions	86
custom states	99
Interaction Queue	56
Interaction Workbin	56
multimedia	56
Routing Strategy	56
error messages	
merge procedure	153
purge procedures	173
time-setting procedure	178
eServices	
See multimedia	
EventAbandoned	46
EventCallDeleted	45
EventReleased	46
events (Multimedia Reporting Protocol)	57
EventUserEvent	99, 100
executing	
merge procedure	160
time-setting procedure	179

F

features, Interaction Concentrator	21
fine-tuning ICON configuration	131
finite state machine	
agent state	83
campaign group	90
chain	91
errors in log	124
login session	85
font styles	
italic	193
monospace	194
FSM	
See finite state machine	
functionality, Interaction Concentrator	21
functions	
of ICON server	18
of persistent caches	19
of persistent queue	19
of stored procedures	20

G

G_AGENT_STATE_HISTORY table	31
G_AGENT_STATE_RC table	30
G_AGENT_STATE_RC_A table	30
G_CALL table	26, 43
G_CALL_HISTORY table	26, 43
G_CALL_STAT table	26, 27, 43, 45

G_CALL_USERDATA table	32, 43
G_CALL_USERDATA_CUST table	32, 43
G_CALL_USERDATA_CUST1 table	32, 43
G_CALL_USERDATA_CUST2 table	32, 43
G_CUSTOM_DATA_P table	31
G_CUSTOM_DATA_S table	31
G_CUSTOM_STATES table	31
G_DB_PARAMETERS table	37
in merge procedure	154
merge procedure parameters	156
purge procedure parameters	168, 176
updating	158, 169, 176
G_DICT_TYPE table	37
G_DICTIONARY table	37
G_DND_HISTORY table	31
G_IR table	26, 43
G_IR_HISTORY table	26, 43
G_IS_LINK table	26, 43
G_IS_LINK_HISTORY table	26, 43
G_LOG_ATTRS table	37
G_LOG_GETIDRANGEREQ table	37
G_LOG_MESSAGES	
table	37, 153, 166, 173, 178
G_LOGIN_SESSION table	30, 88
G_PARTY table	26, 43
G_PARTY_HISTORY table	26, 43
G_PARTY_STAT table	26, 29, 43, 45, 46
G_PROV_CONTROL table	37
G_ROUTE_RESULT table	23, 26, 43
G_SECURE_USERDATA_HISTORY	
table	32, 43, 67
G_SYNC_CONTROL table	37
G_TIMECODE table	37
G_USERDATA_HISTORY table	32, 43
G_VIRTUAL_QUEUE table	33
and HA data extraction	116
CAUSE field	73
record update	73
STATUS field	73
virtual queue ID	116
GC_* tables	116
GCX_* tables	116
global call identifier	21
glx-max-inactivity option	84
GM_F_USERDATA table	32
GM_L_USERDATA table	32
GO_CAMPAIGN table	97, 98
GO_CAMPAIGNHISTORY table	98
GO_CAMPPROP_HIST table	98
GO_CHAIN table	97, 98
GO_CHAIN_CALL table	98
GO_CHAINREC_HIST table	98
GO_CUSTOM_FIELDS table	98
GO_FIELDHIST table	98
GO_RECORD table	98
GO_SEC_FIELDHIST table	98

GO_SECURE_FIELDS table 98
 GS_AGENT_STAT table 31
 GS_AGENT_STAT_WM table 31
 GSYS_DNPREMOTELOCATION table
 about 37
 in merge procedure 154, 156
 updating 156
 GSYS_EXT_INT1 field
 in configuration tables 116
 in G_CUSTOM_DATA_* tables 23
 in G_IR table 58
 in G_PARTY_HISTORY table 23
 in G_PARTY_STAT table 24, 46
 GSYS_EXT_VCH1 field 24, 45, 58
 GSYS_EXT_VCH2 field 24, 44, 45
 GSYS_PENDING_IR table, in merge
 procedure 155
 GSYS_PENDING_LINK table, in merge
 procedure 155
 GSYS_SYSPROCINFO table 37
 gsysInitTimeCode 20
 gsysIRMerge
 about 20
 and gsysIRMerge2 152
 calling 160, 169
 See *also* merge stored procedure
 gsysIRMerge2
 about 20
 and gsysIRMerge 152
 calling 160
 parameters 158
 See *also* merge stored procedure
 gsysIRMergeReset 20, 24, 162, 188
 gsysPurge80
 about 20, 164
 tables involved 165
 gsysPurgeIR
 about 20, 171
 input parameter 171
 gsysPurgeLS 20
 gsysPurgeOS 20
 gsysPurgeUDH 20
 gud-cust-disp option 68
 gudCustDISP1
 See custom dispatchers
 gudCustDISP2
 See custom dispatchers
 gud-cust-disp-groups option 67
 GX_SESSION_ENDPOINT table 30

H

hardware reasons 80, 86
 high availability
 and Interaction Concentrator 103
 data merge results 112

extracting configuration data 116
 extracting data 109, 114
 extracting multimedia data 115
 extracting virtual queue data 116
 feature support 23
 model 104
 of agent-specific data 116

I

ICON

 See Interaction Concentrator

ICON provider 34

ICON server

 functions 18
 interfaces 19
 starting as a Windows service 127
 starting on UNIX 125
 starting on Windows 126
 starting with SCI 124
 stopping as a Windows service 130
 stopping on UNIX 128
 stopping on Windows 129
 stopping, from SCI 128

IDB

 See Interaction Database

IDB fields

 CAUSE 37, 73
 GSYS_EXT_INT1 23, 24, 46, 58, 116
 GSYS_EXT_VCH1 24, 45, 58
 GSYS_EXT_VCH2 24, 44, 45
 NODATA_IUTC 51
 STATUS 73

IDB tables

 and gsysPurge80 procedure 165
 customer-created, not supported 100
 data source session control 24, 34
 for customized attached data 68
 for voice attached data 66
 G_AGENT_STATE_HISTORY 31
 G_AGENT_STATE_RC 30
 G_AGENT_STATE_RC_A 30
 G_CALL 26, 43, 58, 155
 G_CALL_HISTORY 26, 43, 58
 G_CALL_STAT 26, 27, 43, 45, 58
 G_CALL_USERDATA 32, 43, 66
 G_CALL_USERDATA_CUST 32, 43, 66
 G_CALL_USERDATA_CUST1 32, 43, 66
 G_CALL_USERDATA_CUST2 32, 43, 67
 G_CUSTOM_DATA_P 31
 G_CUSTOM_DATA_S 31
 G_CUSTOM_STATES 31
 G_DB_PARAMETERS 37, 154
 G_DICT_TYPE 37
 G_DICTIONARY 37
 G_DND_HISTORY 31

- G_IR 26, 43, 58, 155
 - G_IR_HISTORY 26, 43, 58
 - G_IS_LINK 26, 43, 58, 155
 - G_IS_LINK_HISTORY 26, 43
 - G_LOG_ATTRS 37, 153, 166, 173, 178
 - G_LOG_GETIDRANGEREQ 37
 - G_LOG_MESSAGES 37, 153, 166, 173, 178
 - G_LOGIN_SESSION 30, 88
 - G_PARTY 26, 43, 58
 - G_PARTY_HISTORY 26, 43, 58
 - G_PARTY_STAT 26, 29, 43, 45, 46, 58
 - G_PROV_CONTROL 37, 153
 - G_ROUTE_RESULT 23, 26, 43, 58
 - G_SECURE_USERDATA_HISTORY 32, 43, 67
 - G_SYNC_CONTROL 37
 - G_TIMECODE 37, 179
 - G_USERDATA_HISTORY 32, 43, 67
 - G_VIRTUAL_QUEUE 33, 72, 73, 116
 - GC_prefix 33
 - GC_* 116
 - GCX_prefix 33
 - GCX_* 116
 - GM_F_USERDATA 32, 69
 - GM_L_USERDATA 32, 69
 - GO_prefix 33
 - GO_CAMPAIGN 97, 98
 - GO_CAMPAIGNHISTORY 98
 - GO_CAMPPROP_HIST 98
 - GO_CHAIN 97, 98
 - GO_CHAIN_CALL 98
 - GO_CHAINREC_HIST 98
 - GO_CUSTOM_FIELDS 98
 - GO_FIELDHIST 98
 - GO_METRICS 93
 - GO_RECORD 98
 - GO_SEC_FIELDHIST 98
 - GO_SECURE_FIELDS 98
 - GOX_prefix 33
 - GOX_Chain_Call 97
 - GS_AGENT_STAT 31
 - GS_AGENT_STAT_WM 31
 - GSYS_SYSPROCINFO 154
 - GSYS_DNPREMOTELOCATION 37, 154
 - GSYS_SYSPROCINFO 37
 - GX_SESSION_ENDPOINT 30
 - in time-setting procedure 178
 - provider control 34
 - Identifying the DNIS 45
 - in-memory queue 39
 - input parameters
 - gsysPurgeIR 171
 - merge procedure 158
 - purge procedures 164
 - time-setting procedure 179
 - intended audience 12
 - Interaction Concentrator
 - alarms 19
 - basic architecture 17
 - components 17, 18
 - features 21
 - functionality 21
 - metrics, call 27
 - time formats 22
 - Interaction Database
 - about 20
 - agent state-related tables 30
 - attached data-related tables 31
 - availability, determining 51
 - call-related tables 26
 - common data 100
 - custom state-related tables 31, 100
 - customer-created tables, not supported 100
 - interaction-related tables 26
 - login session-related tables 30
 - outbound-related tables 33
 - party-related tables 26
 - stored procedures 20
 - Virtual Queue table 33
 - See also IDB tables
 - Interaction Queue endpoint 56
 - Interaction Server
 - and agent states 80
 - data source 18, 55
 - Interaction Workbin endpoint 56
 - interrupted data 105, 110
 - inter-server links
 - See IS-Links
 - Inter-Site Call Linkage
 - See IS-Links
 - intra-day reporting 22
 - IRID field 44
 - IS-Links
 - in Interaction Concentrator 22
 - merge procedure timeout parameter 156
 - stuck 154
 - timeout 158
 - italics 193
- ## K
- keys, customer-defined 100
 - key-value pairs
 - and attached data specification 63
 - for customized attached data 67
- ## L
- LCA
 - See Local Control Agent
 - Local Control Agent 19, 127, 128
 - lock timeout, purge procedures 176

login (agent state)	81
login sessions	
and media types	86
and multimedia endpoints	86
data	85
data, persistent cache	19, 88
defined	80
end	85
finite state machine	85
IDB tables	30
model	84
start	84

M

MCR

See multimedia

media type, defined	80
media types	
and agent states	81, 86
and login sessions	86
types	81
merge stored procedure	
about	20, 152
calling gsysIRMerge	160, 169
calling gsysIRMerge2	160
end message	153
error messages	153
executing	160
function	22
G_DB_PARAMETERS table parameters	156
improvements	24
IS-Link timeout parameter	156
log information on SCI	153
parameters	158
recovery	20, 24, 188
resetting	162
scheduling	152, 174
setup	156
start message	153
tables involved	153
troubleshooting	187
updating parameters	158, 169, 176
Message Server, interface with ICON server	19
metrics	
agent state	31, 86
call	26, 27
outbound	92
party	26
models	
agent state	81
high availability	104
login session	84
outbound campaign	90
outbound chain	91
monitoring performance	19

monospace font	194
multi-byte character encodings, supported	22
multimedia	
agent state reason codes	86
agent states	81
and agent states	80
attached data processing	68
attached data tables	32
data in Interaction Concentrator	21
data storage	58
defined	55
DNs	56
endpoints	56, 86
extracting HA data	115
supported scenarios	59
switch	56
Multimedia Reporting Protocol events	57
Multimedia Switch	56

N

network	
call parking	21
environment	21
network-based Contact Solution	21
NODATA_IUTC field	51
NotReady (agent state)	21, 82
null (agent state)	81
number of days (input parameter)	164

O

Oracle, statistics	122
Outbound Contact deployment scenarios	95
Outbound Contact Server	
connections	92
data source	18
reporting limitations	97
outbound data	
IDB tables	33
metrics	92
OCS limitation	97
types stored in IDB	92
output parameters, for merge procedure	158

P

parameters	
merge procedure	158
purge procedure	164
time-setting procedure	179
party	
data in IDB tables	26
metrics	26

pending agent state, defined 80
 performance counter pages
 accessing 131
 Database Writer Page 132
 types of information 132
 usage 131
 performance information
 See performance counter pages
 performance monitoring 19
 persistent cache
 for agent login session data 19, 88, 117
 for configuration data 19, 41
 persistent queue
 and configuration synchronization . . . 19, 41
 corrupted 183
 functions 19
 replacing 183
 troubleshooting 183
 place objects configuration data 56
 provider
 control tables 34
 defined 34
 purge stored procedures
 about 20
 and G_DB_PARAMETERS table . . . 168, 176
 calling 177
 described 162–178
 end message 173
 error messages 173
 gsysPurge76 input parameter 164
 gsyspurge80 164
 input parameter 164
 lock timeout 176
 log information on SCI 166, 174
 optimizing performance 165
 retention period 164
 setup 175

Q

queues
 See in-memory queue, persistent queue,
 virtual queues

R

ready (agent state) 82
 real-time reporting 22
 reason codes
 and workmode parameter 80
 for multimedia agent states 86
 IDB table 30
 reasons
 agent state 80
 hardware 80, 86

multimedia reporting 80
 software 80, 86
 ReleasingParty extension key 46
 releasing-party-report configuration option . . 46
 reliability of data 53
 resolving stuck calls 44
 restoring agent states 83
 retention period, for purge procedures . . . 164
 Routing Strategy endpoint 56

S

sample scripts 68
 scheduling, merge procedure 152, 174
 SCI
 See Solution Control Interface
 script objects configuration data 56
 scripts, sample (for customer dispatcher) . . 68
 SCS
 See Solution Control Server
 security, of attached data 67
 setting up
 merge procedure 156
 purge procedures 175
 time-setting procedure 179
 SIP Server 18, 55
 software reasons 80, 86
 Solution Control Interface
 merge procedure logs 153
 purge procedure logs 166, 174
 starting ICON 124
 stopping ICON 128
 time-setting procedure logs 178
 Solution Control Server 128
 Solution Control, interface with ICON server . 19
 square brackets 194
 start message
 merge procedure 153
 time-setting procedure 178
 starting ICON
 as a Windows service 127
 on UNIX 125
 on Windows 126
 with SCI 124
 statistics
 agent state 31, 86
 call 26, 27
 login session 31
 Oracle 122
 party 26
 See also performance counter pages
 STATUS field, in G_VIRTUAL_QUEUE table . 73
 stopping ICON
 as a Windows service 130
 from console window 129
 on UNIX 128

- on Windows 129
- using SCI 128
- Stopping ICON using SCI 128
- stored procedures
 - gsysInitTimeCode 178
 - gsysIRMerge 152
 - gsysIRMerge2 152
 - in IDB schema 20
 - merge reset 162
 - stuck call. 44
 - svcUpdateDBParameters 158, 169, 176
 - time-setting 178
 - See *also* merge stored procedure, purge
 - stored procedures, time-setting procedure,
 - custom dispatchers
- store-releasing-party configuration option . . . 24
- stuck
 - calls, resolution 44
 - custom states 100
 - IS-Links 154
 - records 44
- supported features
 - agent login session data. 21
 - agent state data 21
 - attached data 21
 - attached data, customized 22
 - configuration data 21
 - custom states 21, 22
 - customized attached data 22
 - high availability 23
 - identify data availability 23
 - identify data reliability 23
 - Interaction Concentrator. 21
 - interaction data 21
 - login session data 21
 - multi-byte character encodings 22
 - multimedia interactions 21, 59
 - network 21
 - network call parking 21
 - outbound campaigns data. 22
 - post-routing user data processing. 23, 65
 - real-time reporting 22
 - releasing party. 24
 - virtual queue DBID 23
 - virtual queues 21
- svcUpdateDBParameters
 - stored procedure 158, 169, 176
- switch
 - Multimedia 56
 - Multimedia, DNS. 56
- synchronization
 - troubleshooting 187
 - with Configuration Database 19, 41

T

- time format. 22
- time-interval reports 179
- timeouts
 - IS-Links 158
 - purge procedure lock 176
- time-setting procedure
 - about 20
 - end message 178
 - error messages 178
 - executing 179
 - G_TIMECODE table 179
 - gsysInitTimeCode 178
 - IDB tables involved 178
 - input parameters 179
 - log information on SCI 178
 - setup not required. 179
 - start message. 178
- troubleshooting
 - configuration synchronization. 187
 - database writing 185
 - merge procedure 187
 - persistent queue 183
 - runtime problems 183
 - startup problems 181
 - using the performance counter pages . . . 131
- T-Server
 - and agent states 80
 - data source 18
- type styles
 - conventions 193
 - italic 193
 - monospace 194
- typographical styles 193

U

- UNIX
 - starting ICON 125
 - stopping ICON 128
- unknown (agent state). 82
- updating
 - G_DB_PARAMETERS table 158
 - merge procedure parameters. 158
- updating G_DB_PARAMETERS table. 169, 176

V

- version numbering, document 193
- Virtual Queue data
 - and Multimedia Reporting custom
 - message 57
 - DBID 76
 - HA extraction 116

IDB table.	33
in Interaction Concentrator	21
processing.	72
records	116
storage	72

W

Windows	
starting ICON	126
stopping ICON.	129
Windows Service	
starting ICON	127
stopping ICON.	130
Windows Service Control Manager	129
workbin, interaction endpoint.	56
workmode	86, 87
workmode parameter, and reason codes . . .	80

