Interaction Concentrator User's Guide

Interaction Concentrator 8.1.5

12/29/2021

Table of Contents

Welcome to the Interaction Concentrator User’s Guide

The *User's Guide* contains articles and information enabling you to understand basic Interaction Concentrator architecture, detailed information about Interaction Concentrator features and functionality, and explanations of how to perform ongoing maintenance and administration of Interaction Concentrator.

For planning materials, prerequisites, and deployment instructions, see the *Interaction Concentrator Deployment Guide*.

This document assumes 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 products deployed in your contact center
- Your real-time and historical reporting objectives

The information in this *Guide* is divided into the following sections:

About Interaction Concentrator

These topics provide detailed information about Genesys Interaction Concentrator data, features, and functionality. They also provide a high-level overview of the basic architecture and components of Interaction Concentrator.

- Overview: Architecture, Components, and Functionality
- Introducing IDB Schema
- How ICON Works
- Identifying Who Released the Call
- Tracking Multi-Site Call Data Via ISCC
- Integrating with Multimedia
- Processing Attached Data

- Monitoring Virtual Queues and Routing Points
- Agent States and Login Sessions
- Integrating with Outbound Contact
- Processing User Events and Custom-Defined States

High Availability

Explains how to implement HA in Interaction Concentrator and how data is processed in HA environments.

- The Interaction Concentrator HA Model
- Extracting Data in an HA Deployment

Administration

Explains how to perform ongoing maintenance and administration of Interaction Concentrator, as well as to troubleshoot startup and runtime problems.

- Monitoring Interaction Concentrator
- Filtering IDB Data
- Resynchronizing Configuration Changes
- Using Special Stored Procedures

- For how to start and stop ICON, see Starting and Stopping in the *Interaction Concentrator Deployment Guide*.
- For troubleshooting tips, see Troubleshooting in the *Interaction Concentrator Deployment Guide*.

Architecture, Components, and Functionality

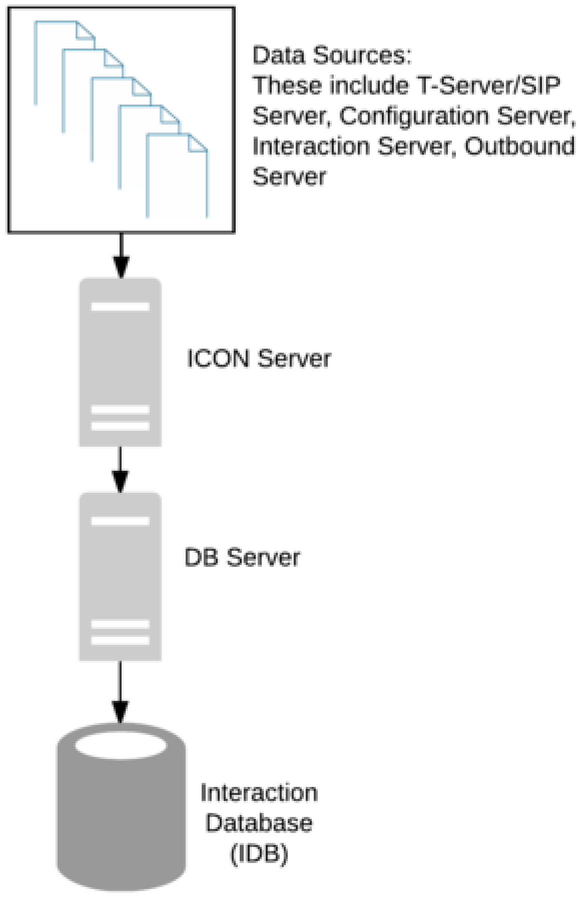
This topic describes the basic architecture and components of Interaction Concentrator. It also provides a high-level overview of Interaction Concentrator functionality. It contains the following sections:

- Basic Architecture
- Secure Connections
- Components and Functions
- Supported Features and Functionality

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.


Data Sources:
These include T-Server/SIP Server, Configuration Server, Interaction Server, Outbound Server
ICON Server
DB Server
Interaction Database (IDB)

Security Features

Interaction Concentrator supports the following security features:

- Encrypted RDBMSs.
- Hiding TEvent or multimedia-specific attached user data in the ICON log. To specify keys for which the values should be hidden in the ICON log file, configure one—or—both of the following options in the ICON `Application` object:
  - Set the **default-filter-type** option in the **[log-filter]** section.
  - Explicitly define one or more **<key-name>** options in the **[log-filter-data]** section.

    If any attached data is configured to be hidden, ICON debug-level messages might hide all attached data values in the log, not just the values of the keys configured to be hidden. If the **default-filter-type** option in the **[log-filter]** section is set to `skip`, ICON prints an empty string for all KVList pairs in the log file, except those keys that are explicitly configured in the **<key-name>** options in the **[log-filter-data]** section.

    If any of the **<key-name>** options in the **[log-filter-data]** section are set to `skip`, ICON might print an empty string, not only for the specified key but also for other KVList pairs in the log file.

Important

The **default-filter-type** option in the **[log-filter]** section and the **<key-name>** options in the **[log-filter-data]** section are configured in the ICON `Application` object. However, the description of the default and valid values, and a full explanation of how these options work, are located in Hide Selected Data in Logs (in the *Genesys Security Deployment Guide*).

Warning

ICON includes a set of predefined routing attached data keys that specify call processing data that is *not* considered sensitive. The list of keys is specified in the "Universal Routing Server Attached Data" section in Configuring for Attached Data. If the keys from this set are configured in the **<key-name>** option in the **[log-filter-data]** section of the ICON `Application` object, the data in the debug level may or may not be hidden in the log file.

Support for Secure Connections

- Starting with release 8.1.1, Interaction Concentrator supports Transport Layer Security (TLS) and TLS-FIPS connections.
- Starting with release 8.1.2, Interaction Concentrator supports client-side port definition, to provide secured connections to T-Server, SIP Server, Configuration Server, and Message Server.

On Windows platforms, support for TLS is integrated into the operating system, and there are no additional requirements to enable Interaction Concentrator to support it. On UNIX-based platforms, you must install the Genesys Security Pack on the Interaction Concentrator host.

In the following scenario, you must configure ICON host security certificate settings at the ICON DAP `Application` object level:

- You need a secure connection between Configuration Server and ICON configured with the `cfg` role or the `all` role.
- The ICON server and the DB Server to which it is connected are on different hosts.

See Securing Connections Using TLS in the *Genesys Security Deployment Guide* for configuration instructions.

Important

For details on all of the supported security features, see the *Genesys Security Deployment Guide*.

Components and Functions

Interaction Concentrator consists of the following elements:

- ICON server
- Interaction Database (IDB)

The following subsections provide brief descriptions of the Interaction Concentrator components. For more details, see How ICON Works.

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. Roles are configured using the role configuration option. For more information, see ICON Roles.
- 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.
- 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 Deployment Guide*

Persistent Queue and Persistent Caches

The persistent queue (**.pq**) 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.

> **Important**
> To reduce the possibility that Interaction Concentrator might lose connection with the **.pq** file, you are required to locate it on a local drive rather than a network drive.

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

Persistent Cache for Agent Login Session Data

For the ICON instances that perform the `gls` role, 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 and Extracting Agent-Specific Data.

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.)
- Message Server, to log messages to the Central Logger.

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 reporting functionality. 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. For a complete table structure and descriptions of all IDB tables and fields, see the *Interaction Concentrator 8.1 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:

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.

> **Important**
> These separate purge procedures were discontinued in release 8.1.503.03. If you are using Interaction Concentrator 8.1.503.03 or higher, use gsysPurge81 or (in Oracle environments) purgePartitions811.

- gsysPurge81—An alternative procedure that safely purges voice, multimedia, and outbound interactions; attached data; and agent login session data from IDB.

> **Important**
> Interaction Concentrator release 8.0.000.32 and earlier includes the prior version of the purge procedure, gsysPurge80, which does not purge Outbound Contact data.

- purgePartitions811—If you are running Oracle 11g or higher and need to purge large amounts of data efficiently, you can create a partitioned IDB that can be purged by truncating partitions.

Custom Dispatchers

- gudCustDISP1 and gudCustDISP2—The stored procedures that customize attached data processing.

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.

> **Important**
> Genesys Info Mart 8.x does not use the Interaction Concentrator merge procedures.

Time-Setting

- gsysInitTimeCode—The stored procedure that populates the G_TIMECODE table, to enable time-interval reporting.

For more information about these stored procedures, see Using Special Stored Procedures.

Supported Features and Functionality

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 How ICON Works.
- 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 Integrating with Multimedia.
- 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 Agent States and Login Sessions.
    - 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. See also the information about configuring agent login session metrics in the *Interaction Concentrator Deployment Guide*.
- Supports custom agent states, for agents handling voice interactions. For more information, see Agent States and Login Sessions.
- For all types of interactions, captures and stores detailed information about virtual queue usage in interaction processing. For more information, see Monitoring Virtual Queues and Routing Points.
- 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 (e-mail and chat) and 3rd Party Media interactions. For more information, see Processing Attached Data.
- Supports customized attached data processing for voice calls. For more information, see Customized Attached Data Processing.
- 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 Integrating with Outbound Contact.

- Provides a configurable filtering mechanism for certain types of data, to enable the optimization of database size and performance. For more information, see Filtering IDB Data.
- Provides the ability to resynchronize the configuration data in IDB with Configuration Database on demand. For more information, see Resynchronizing Configuration Changes.
- 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 The Interaction Concentrator HA Model.
- 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 Using Special Stored Procedures.
- 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 Procedures.
- If configured with the `lrm` role, supports Genesys License Reporting Manager (LRM) by storing LRM-specific data. For detailed information see Configuring for LRM Data.

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 topic provides a summary of the precalculated call and party metrics that are available in the statistical tables.

This page describes the following groups of tables:

- Operational Tables
- Configuration Tables
- Data Source Session Control Tables
- Service and Dictionary Tables
- Log Tables

Important

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

The data filtering feature limits the data stored to IDB. See Filtering IDB Data for more information.

Operational Tables

There are eight subgroups of tables that contain operational data:

- Interaction-related and Party-related tables
- Agent State–related and Login Session–related tables
- Custom State–Related tables
- Attached Data–related tables
- Outbound-related tables
- Active Call and Active Interaction tables
- Virtual Queue–related tables

Interaction-Related and Party-Related Tables

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

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, 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 corresponding core tables. The history tables are named the following:

- G_CALL_HISTORY
- G_PARTY_HISTORY
- G_IR_HISTORY
- 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 interactions and parties, respectively.

Interaction Metrics

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

Available Interaction Metrics

Metric Name	**Description**	**Media Type Supported**			
Voice	**eServices**				
	Email	**Chat**	**3rd Party Media**		
F_CONN	Flag	Yes	Yes	Yes	Yes

Metric Name	Description	Media Type Supported			
	CONNECTED.				
F_CONN_EXTN	Flag CONNECTION 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_LGN	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	Yes	Yes	Yes	Yes

Metric Name	Description	Media Type Supported			
	on any device (either by an agent or IVR).				
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.	n/a	n/a	n/a	n/a

Party Metrics

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

Available Party Metrics

Metric Name	Description	Media Type Supported			
Voice	**eServices**				
	Email	**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

Metric Name	Description	Media Type Supported			
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
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.	n/a	n/a	n/a	n/a

Agent State and Login Session Tables

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

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.

For more information, see Processing Attached Data.

There are three subgroups of attached data–related tables:

Attached Data State Tables for Voice (flat 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.

Attached Data History Tables for Voice (generic 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 Attached Data Processing for Multimedia.

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.

Active Call and Active Interaction Tables

The G_CALL_ACTIVE and G_IR_ACTIVE tables store data about currently active calls and interactions, respectively. When a call or an interaction terminates, the record is removed from the G_*_ACTIVE tables.

These tables are used to help more efficiently clear calls that become stuck after Interaction Concentrator stops. When Interaction Concentrator restarts, it checks the G_*_ACTIVE tables, and terminates all listed calls with the termination timestamp being the time that Interaction Concentrator restarted.

For details on these tables, see the relevant sections of the *Interaction Concentrator Physical Data Model* document for your RDBMS.

Virtual Queue Tables

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.

Data about virtual queue IDs are provided in the GSYS_EXT_VCH1 field of the G_ROUTE_RESULT table for interactions that have been cleared or abandoned from the virtual queue. The G_ROUTE_RES_VQ_HIST table also contains information about the use of virtual queues in interaction processing.

For details on these tables, see the relevant sections of the *Interaction Concentrator Physical Data Model* document for your RDBMS. For more detailed information about stored virtual queue data, see Monitoring Virtual Queues and Routing Points.

Configuration Tables

There are two subgroups of tables that contain configuration data:

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, such as information about the fact that an Agent was removed from an Agent Group.

Data Source Session Control Tables

The Interaction Concentrator IDB schema contains 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.

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` object.

Important
When assigned the `lrm` role, ICON creates records in the G_DSS_GLS_PROVIDER and G_DSS_GOS_PROVIDER tables.

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

- Data Sources
  The identity of the data sources for that particular provider—for example, the identifier that Configuration Server assigns to the primary and backup data source applications (such as DS_DBID and DS_DBID_PRIM).
- Connection
  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
  An important connection-related field, DSCONN_TYPE, describes the type of connection. This indicates the reason that a new record was created in the provider table:
  - 1—Indicates that this is the first connection following a restart of the ICON server.
  - 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 Server

The identity and status of the ICON server—for example, the DBID of the ICON server (ICON_DBID), and the server 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 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.

Handling Very Short Connections

When ICON establishes a connection to a data source, it saves all necessary data in memory and waits for next event on this connection. If the next event is a data event, the data is stored in IDB.

However, if the data source provides no operational data, ICON waits until the expiration of the NoData timeout before processing the NoData event.

If the lifetime of the connection is less than the NoData timeout, no information about this short-lived, inactive, connection is written to IDB.

If two or all three provider roles (`gcc`, `gls`, and `gud`) are assigned to ICON, when the NoData condition becomes true for any one provider, NoData records are created for all other providers from that particular data source. This occurs because all providers use the same DSS connection.

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 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.

Important

When ICON starts up, the `gcc`, `gls`, and `gud` providers start automatically in order to perform certain internal functions, regardless of the role that has been configured in the ICON Application object. The respective provider control tables might therefore contain partial records (for example, containing information about the start time or end time of the ICON session), even if that ICON instance does not perform the particular provider role. However, the ICON provider stores data in IDB only if the ICON instance 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, Determining Data Availability and Reliability and Extracting Data in an HA Deployment.

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.

How ICON Works

This page describes basic Interaction Concentrator functioning—how the Interaction Concentrator server (ICON) collects, processes, and stores configuration and interaction data in Interaction Database (IDB). It contains the following sections:

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.

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 configured in 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. See also The Persistent Queue (PQ) file.

For more information about the ICON configuration options, see the Interaction Concentrator Options Reference.

Important

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 correctly shows the disabled state (the STATE field in the GC_ENDPOINT table), but any activity on that DN continues 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 is set to `cfg` or `all`.

Important

Interaction Concentrator, configured with the `cfg` role, treats an object as deleted if you change the permission on that object so that ICON can no longer access it. In this scenario, if you restore the permissions, ICON does process interactions for the objects moved back to visibility, but ICON does not update the tables storing configuration data. To fully restore the objects, you must reinitialize your Interaction Concentrator configuration IDB. You might also need to manually resynchronize your Genesys Info Mart data.

Annex Tab Data

Starting in release 8.1.4, Interaction Concentrator can be configured to collect and store data about changes on the Annex tab of the following objects:

- Persons
- Agent Groups
- DNs
- DN Groups
- Switches

This information enables Genesys Interactive Insights to control visibility of certain data and reports based on attributes such as geographical location, business line, or organization structure. This functionality is available only when ICON has the `cfg` role and the cfg-annex option configured.

To have Interaction Concentrator track **Annex** tab data, set the value of the **cfg-annex** option to 1 (the default value is `0`).

Interaction Concentrator stores data for the following occurrences:

- Option created
- Option deleted
- Option renamed
- Option value changed
- Section renamed
- Section deleted
- Configuration object deleted

When an **Annex** tab configuration option is deleted and then created again, the CREATED and LASTCHANGE fields in the GC_ANNEX table are set to the current timestamp.

For efficient processing, Interaction Concentrator collects and stores section and option change data only for certain sections on the **Annex** tab. See the Monitored Annex Tab Sections table, below, for the sections that are tracked.

Monitored Annex Tab Sections

Configuration Object	Monitored Sections
Person	Any section starting with RPT
Agent Group	Any section starting with RPT
DN	gim-etl Any section starting with RPT Any section starting with agg-
DN Group	Any section starting with RPT
Switch	gim-etl Any section starting with agg-

Occasions When ICON Collects Configuration Data

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.
- When the persistent cache is unavailable. For more information, see Persistent Cache Is Not Available.
- Through real-time object change notifications. For more information, see ICON Receives Dynamic Notifications.
- Upon receipt of the Configuration Server’s history log file. For more information, see ICON Reads the

Configuration History Log.

- Upon user request for resynchronization. For more information, see User Request For Resynchronization.

Important

If you have configured ICON with the **cfg** role while the Configuration Server **history-log-active** option is set to `false`, you must manually resynchronize to pick up any configuration changes that occur during ICON downtime.

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

Important

ICON stores information in the GC_APPLICATION table 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 Physical Data Model* document for your RDBMS.

Persistent Cache for Configuration Data

The persistent cache enables ICON to synchronize configuration data in IDB with current Configuration Server information. 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 renamed. 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 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 or Genesys Administrator, 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**

Important

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 *Management Framework Deployment Guide* and the history log section in the *Framework 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 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 Integrating with Multimedia.
- For information about the way in which ICON handles attached data for voice calls, see Attached Data Processing for Voice Calls.

T-Server TEvents

ICON connects to T-Server and 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_ACTIVE	G_CALL_USERDATA
G_IR_ACTIVE	G_CALL_HISTORY	G_CALL_USERDATA_CUST
G_IR_HISTORY	G_CALL_STAT	G_CALL_USERDATA_CUST1
G_IS_LINK	G_PARTY	G_CALL_USERDATA_CUST2
G_IS_LINK_HISTORY	G_PARTY_HISTORY	G_USERDATA_HISTORY
G_ROUTE_RESULT	G_PARTY_STAT	G_SECURE_USERDATA_HISTORY
G_CALL		

For detailed information about the tables in IDB in which ICON stores interaction data, see the *Interaction Concentrator 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 the G_PARTY table, 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.

Interaction Concentrator stores data on active voice interactions in the G_CALL_ACTIVE and G_IR_ACTIVE tables. After restarting, any interactions remaining in either table will be marked as terminated, with the termination timestamp being the time that Interaction Concentrator restarted.

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

Important

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 Physical Data Model* document for your particular RDBMS.

Identifying the DNIS for Outbound Softphone Calls

By default, Interaction Concentrator takes the value of the DNIS (dialed number information service, which is the number that initiates a call) from the AttributeDNIS field in T-Server events and stores it in the CallDNIS field of the G_CALL table. However, because various switches operate differently, the value of the DNIS might be distributed in other attributes and/or in different TEvents. In some cases, the standard AttributeDNIS field may be empty or not present.

By appropriately configuring the value of the **gts-dnis-detection** configuration option in the configuration object for the associated Switch, you can choose to have ICON take the value of the DNIS from a related subset of TEvents.

> Important
>
> Once defined, the value of the DNIS cannot be changed.

Determining Data Availability and Reliability

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.

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 Data in an HA Deployment.

For information about the IDB inconsistencies that can result from unavailable data, see Consequences of Failures.

Determining ICON Responsiveness

Interaction Concentrator supports a mechanism that enables Local Control Agent (LCA) to determine whether the ICON server process has become unresponsive.

- An unresponsive process is one that appears to be running but for some reason is unable to provide service to its clients or peers. LCA can alert you to ICON’s status and enable you to take corrective action, such as restarting ICON. For more information on unresponsive process detection, see the *Framework Management Layer User’s Guide*.

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.

The following table, Scenario Example—G_DSS_GCC_Provider Table Field Values, 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.

Important

G_DSS_GCC_PROVIDER table fields not related to connection data are not shown in the table. For the meaning of the DSCONN_TYPE values, see Connection Type.

Events	G_GCC_ Provider Table Record	DSCONN_TYPE	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	No change when ICON restarts	4	t0	t5	Null	Null
ICON shuts down gracefully at t6.	Updated when ICON restarts	4	t0	t5	t6	t6

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 the table above, 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_ITIME) 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.

Important

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.

1. Starting from time t0, 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.
2. Starting from time t1, 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.
3. At time t1 + 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_ITIME 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.

The following table summarizes the analysis. It shows the value of the specified IDB fields during last two minutes.

LEVENT_ITIME	NODATA_IUTC	Conclusion
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 the following two ways:

- Evaluation by ICON of the reliability of the data it records in IDB.
- Evaluation by the downstream reporting application of the reliability of the data provided by a particular ICON instance.

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 GC_AGENT and GC_PLACE tables indicates the reliability of the record timestamps. For more information about the reliability flags in IDB, see the Interaction Concentrator 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.

Setting Alarms for Call Processing Failures

When Interaction Concentrator receives an incomplete event flow, it might skip processing the calls involved or even destroy them. To guard against this, you can set an alarm that indicates when ICON does not receive segments of call data events.

> **Important**
>
> - This functionality applies only to calls received via T-Server/SIP Server. It is not applicable to Outbound or multimedia interactions.
> - ICON does not generate an error message for transactions that were terminated by a call deletion event, such as EventCallDeleted.

- To enable this functionality, set an appropriate value for the log-call-failure option.
- The log event on which you can set an alarm is 09-20039. For instructions on how to set up an alarm condition based on log event, see Alarm-Signaling Functions in the *Framework Management Layer User's Guide*.

Call-Processing Errors Leading to Alarms

The following is a list of scenarios in which call-processing is disrupted or calls are destroyed (that is, they are not recorded in IDB), and therefore can trigger an alarm.

Failed call-match transactions

In failed call-match scenarios, ICON receives EventCallCreated, but either no subsequent EventDialing, EventRinging, EventQueued, EventCallDeleted event arrives or else these subsequent events are received after the default timeout period. Such calls are not written to IDB and all record of them is destroyed when the timeout expires.

Failed to restore call after reconnect/switchover (Standalone mode)

The following sample scenario demonstrates how calls might be destroyed after a disconnection:

1. ICON receives EventCallCreated, EventCallPartyAdded, EventDialing, or another applicable event and creates a call.
2. ICON loses the connection to T-Server.
3. While ICON is disconnected from T-Server, the call is released.
4. When ICON reconnects to T-Server, it sends TRegisterAddress() requests for each monitored DN.
5. After the last request is sent, ICON sets a timer for ten minutes.
6. After the timer expires, all calls that are not synchronized are deleted.
   An *unsynchronized call* is one that was released while ICON was disconnected from T-Server. If ICON receives an EventCallCreated event corresponding to a call that was created before the disconnect, ICON can *synchronize* the call data and complete call processing.

> **Important**
> If ICON reconnects to T-Server and starts the timer, but then another disconnection takes place, ICON resets the timer again. In this case, ICON might not remove some unsynchronized calls.

Call-processor errors

The following are examples of call-processor errors:

- AttributeThisDN is missing
- AttributeRefConnID is missing
- Call was not found

Low-level finite state machine transition failures

These errors may occur when ICON encounters elements of interactions that it cannot resolve due to incomplete event flow.

Failed single-step transfer, conference, and redirect transactions

These are transactions of the following types that were not completed when the timer expired.

- Single-step transfers
- Two-step transfers
- Conferences
- Call redirects

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.
- 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.

For information about the implications for multi-site deployments, see Multi-Site Deployments.

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 table.
- 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.

Important

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.

G_PARTY_STAT Values

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:

- `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.

Important

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

The following tables summarize 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.

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

Important

The *releasing party* is in relation to the T-Server or switch monitored by the ICON instance.

Call type = Inbound or Outbound

Releasing Party	AttributeCtrlParty in EventCallDeleted	G_CALL_STAT Table, GSYS_EXT_VCH1 Field	G_CALL_STAT Table, GSYS_EXT_VCH2 Field
External	No data	Empty string	Empty string
External	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	Internal DN	Internal DN	*PartyID associated with EndpointDN = "Internal DN" and CallID of this call*

Call type = Internal

Releasing Party	AttributeCtrlParty in EventCallDeleted	G_CALL_STAT Table, GSYS_EXT_VCH1 Field	G_CALL_STAT Table, GSYS_EXT_VCH2 Field
Internal	No data	Empty string	Empty string
Internal	Internal DN	Internal DN	*PartyID associated with EndpointDN = "Internal DN" and CallID of this call*

Call type = Consultation

Releasing Party	AttributeCtrlParty in EventCallDeleted	G_CALL_STAT Table, GSYS_EXT_VCH1 Field	G_CALL_STAT Table, GSYS_EXT_VCH2 Field
External or Internal	No data	Empty string	Empty string
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.			

Call type = Conference (one internal party, all other parties external)

Releasing Party	AttributeCtrlParty in EventCallDeleted	G_CALL_STAT Table, GSYS_EXT_VCH1 Field	G_CALL_STAT Table, GSYS_EXT_VCH2 Field
Internal	Internal DN	Internal DN	*PartyID associated with EndpointDN = "Internal DN" and CallID of this call*

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 of Call-Release Reporting in a Multi-Site Deployment

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.

The following table summarizes the values that ICON will report for the party that released the call.

IDB	G_CALL_STAT Table, Data Source	G_CALL_STAT Table, GSYS_EXT_VCH1 Field	G_CALL_STAT Table, GSYS_EXT_VCH2 Field	G_PARTY_STAT Table, Data Source	G_PARTY_STAT Table, GSYS_EXT_INT1 Field
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/SIP Server Support

To implement reporting on which party released the call, ICON relies on T-Server to provide the required data.

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

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 Important note for Call-Based Reporting and the Important note 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.

Tracking Multi-Site Call Data Via ISCC

Multi-site calls are tracked using an intersite link, which enables you to connect the information regarding two calls that originated on two different sites.

Intersite link (IS link) data is stored in the G_IS_LINK and the G_IS_LINK_HISTORY tables. It enables ICON to link all multi-site calls, determine the correct TEvent sequence, and check whether a multi-site call is completed.

Determining the Correct Event Sequence

To track the order in which events come from SIP Server, ICON writes the SIP Server event sequence in the GSYS_EXT_VCH1 field in the G_IS_LINK table. This TEvent sequence can be the same value for different data source sessions (DSSs).

ICON also writes the value for LastTransferOrigDN from the TEvent into the GSYS_EXT_VCH2 field in the G_PARTY table and the value for LastTransferHomeLocation into the GSYS_EXT_VCH1 field in the G_PARTY table.

ICON populates the TEvent attribute values for the ENDPOINTDN in the GSYS_EXT_VCH2 field and LastTransferHomeLocation in the GSYS_EXT_VCH1 field in the G_PARTY table whenever SIP Server provides this information. As a result, in multi-site interactions, ICON stores the necessary data to link ISCC calls, provide the correct event sequence, and determine when the call is completed.

Determining When a Multi-Site Call is Completed

A multi-site call 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 Interaction Roots in the G_IR table are closed.

Post-Mortem IS_LINK Capability

Post-mortem IS links occur when the information about the external site to which a call was transferred arrives after the call left the switch and ICON deleted it. When ICON receives EventTransactionStatus with the iscc.is-link-creation.post-mortem attribute set to true, ICON creates records in the G_IS_LINK table. The timestamp of the event is recorded as the link creation time, although that might be several seconds after call deletion.

ICON creates the new record in the G_IS_LINK table even if the referenced call has not been recorded in IDB. As a result, IDB might store records in the G_IS_LINK table that do not have a matching record in the G_CALL table.

The post-mortem IS link functionality can handle even complex scenarios. For example, ICON can correctly link and sequence call records for a consultation call that is transferred to another site and, after the transfer is complete, is merged with the main call.

Integrating with Multimedia

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

- Multimedia Objects
- Populating Multimedia Interaction Data
- Handling Active Multimedia Interactions
- isOnline Chat Attribute
- Chat Session Attributes that Indicate Who Ended the Session

For information about ICON configuration and other Configuration Layer settings that make data about multimedia interactions available in Interaction Database (IDB), see the Configuring for Multimedia Data in the *Interaction Concentrator Deployment Guide*.

Important

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 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 both Genesys eServices and 3rd Party Media objects and activities. When it is necessary to distinguish between the two types of multimedia, the terms eServices 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 or Enhanced 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 and 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 and 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
- Multimedia Interactions
- Supported Scenarios

For information about the way in which ICON handles attached data for multimedia interactions, see Attached Data Processing for Multimedia.

For information about how to capture information about agent states and login sessions for multimedia interactions, see Agent States and Login Sessions.

For detailed information about the tables in IDB in which ICON stores multimedia data, see the *Interaction Concentrator 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 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 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.

Important

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.
- 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 the available Interaction Metrics and Available Party Metrics tables in Interaction-Related and Party-Related Tables.

Agent Activity Tables

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 and Login Session Tables.

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).
- Reconstructs interaction data later for further processing (see Reconstructing Interaction Data).
- Tracks user data changes (see Tracking User Data Changes).
- Filters out strategy data not required for reporting (see Strategy Activity Data).

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, which is configured on `Script` objects of type Interaction Queue, is set to `true`, 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 *Interaction Concentrator Options Reference*.

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:

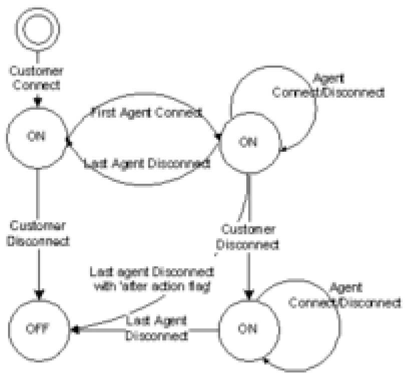
- CALLID of the interaction
- PARTYID of the party for the last interaction in the queue and/or the last interaction in the workbin
- User data

Once the interaction is reconstructed, ICON processes it as a regular multimedia interaction. See Populating Multimedia Interaction Data.

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. The figure below illustrates the possible transmissions and states of the isOnline chat attribute.



isOnline Attribute of Chat Interactions

Processing the isOnline Attribute

ICON monitors changes to 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. The table below describes the key fields in the G_USERDATA_HISTORY table and the possible values for chat interactions.

Field Name	Value	Comment
KeyName	_attr_is_online	This value does not change.
ChangeType	1|2|3|4|5	The reason the value of the key was recorded: • 1—Created. Indicates that the value of the key was attached to the chat interaction when ICON received an EventInteractionSubmitted event with an isOnline attribute setting of true. • 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 EventPropertiesChanged event. • 4—Deleted. Indicates that the key has been deleted from UserData. • 5—Terminated. Indicates that ICON recorded the value of the key when ICON received an EventProcessingStopped event.
Type	1|2|3|4|5	The type of the data source—extensions, reasons, or attached data (userdata)—represented by the following values: • 1—userdata • 2—reasons • 3—extensions

Field Name	Value	Comment
		• 4—attributes (reserved for future use) • 5—mcr_workbin
KeyID	9995	This is a hard-coded value.
Value	NULL|0|1	• `1`—Chat session is active. • `0`—Chat session is stopped. • `NULL`—Interaction Concentrator records this value whenever it receives EventProcessingStopped, regardless of any other information from Interaction Server. The value is taken from the event.
Added	*Timestamp value*	This is the time corresponding to when the record was modified.

Chat Session Attributes that Indicate Who Ended the Session

ICON release 8.1.507.06 and higher, with the gud role set, can store data provided by Chat Server in the EventPropertiesChanged or EventProcessingStopped event that identifies which party ended a chat interaction. A chat session ends when a party closes their chat window or the agent ends the chat interaction by clicking the **End Chat** button. In conference chat scenarios, one agent clicking **End Chat** does not end the session as long as another agent remains on the chat session with the customer.

ICON stores the following two attributes that enable you to report on who ended the chat session and when:

- **ChatServerSessionClientLeftAt**—The timestamp when the customer left the chat session.
- **ChatServerSessionClosedAt**—The timestamp when the chat session was actually closed.

By default, once ICON receives either or both of these attributes, it writes records into the G_USERDATA_HISTORY table using the key(s) **ChatServerSessionClientLeftAt** and **ChatServerSessionClosedAt**, with the values taken from the event (UTC datetime string). Only one value for each key is written per chat interaction; by default, this is the first value.

You can configure ICON to write these attributes to a different table or to disregard them entirely using the attached data specification XML file. You can also specify whether ICON writes the first value or the final value depending on how you configure the `history` parameter in the attached data specification XML file.

Important

- Chat Server 8.5.103.22 or later is required for this functionality.
- The information from Chat Server does not include Place or Party data. As a result, the PARTYID and ENDPOINTDN fields in the G_USERDATA_HISTORY table will contain a `NULL` value for records with the **ChatServerSessionClientLeftAt** and **ChatServerSessionClosedAt** keys.
- If by chance you have already been using keys named **ChatServerSessionClientLeftAt** and **ChatServerSessionClosedAt**, rename your keys. The keys **ChatServerSessionClientLeftAt** and **ChatServerSessionClosedAt** are now reserved.

Processing Attached Data

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

- Attached Data Specification File
- Attached Data Processing for Voice Calls
- Customized Attached Data Processing
- Attached Data Processing for Multimedia

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 Configuring for Attached Data in the *Interaction Concentrator 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 (by default, named **ccon_adata_spec.xml**) 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 Attached Data Specification File in the *Interaction Concentrator Deployment Guide*.

Important

In releases prior to 8.1.5, you must restart ICON after you make changes to the attached data specification file for the changes to take effect.

When ICON reads the attached data specification file, it creates a dictionary, which then becomes active. All records for newly-created interactions are associated with this active dictionary and all data processing for this interaction is implemented in accordance with it.

When you update the adata-spec-name option with the name of a new attached data specification file, ICON reads the new file and creates a new active dictionary. All new interactions are associated with this new active dictionary. Data processing of old interactions (those created before the new dictionary became active) is continued based on old dictionary. The new dictionary is used only for new interactions.

After you edit the old attached data specification file, ensure that ICON reloads it by changing the file name and updating the value of the **adata-spec-name** option accordingly.

After reloading the attached data specification file, ICON compares the active (older) dictionary with newly-loaded one. If these two dictionaries are identical (that is, they contain the same key names and definitions) ICON continues to use the currently active dictionary and does not create a new one.

The result of the reloading procedure is then written in a message in the Interaction Concentrator log file.

Example:

1. ICON currently uses an attached data specification file named **ccon_adata_spec.xml**.
2. After you modify this file, you save it as **ccon_adata_spec2.xml**.
3. To enforce the attached data specification file reload procedure, you change the value of the **adata-spec-name** option from **ccon_adata_spec.xml** to **ccon_adata_spec2.xml**.
4. As a result, ICON rereads the attached data specification from the file named **ccon_adata_spec2.xml**.
5. Noting differences between **ccon_adata_spec.xml** and **ccon_adata_spec2.xml**, ICON creates a new active dictionary that is used to process all new interactions. Interactions that ICON started processing based on the old dictionary continue to be processed using the old dictionary.
6. ICON writes a message in the Interaction Concentrator log recording the result of the attached data specification file reload.

The Number of Dictionaries ICON Simultaneously Maintains in Active Memory

ICON keeps an old dictionary until all the interactions using it are completed. However, by default the maximum number of dictionaries ICON keeps in active memory is twelve. ICON cannot process additional dictionaries once it reaches this limit. Therefore, Genesys recommends that you do not change the attached data specification more than ten times during the usual lifetime of the longest-living interaction type. If you need additional dictionaries in memory, contact Genesys Customer Care for assistance.

Limitations

- Consultation calls and call merging:

    During a long call that involves a consultation, the main call and the consultation call might have different dictionaries. For example, the consultation call might be created after the attached data specification file was changed. As a result, some key names from the consultation call might not be transferred to the main call because of the differences in their dictionaries.

    When the merge is completed, only one call is still alive (usually the main call). Data processing for that call continues according to the dictionary assigned for that call.

- Number of simultaneously loaded active dictionaries in ICON memory:

By default, the number of dictionaries ICON keeps in memory at once is twelve. If you need to increase the maximum number of active dictionaries loaded in memory, contact Genesys Customer Care for assistance.

- High availability:

  In high availability (HA) environments, two ICON instances work with the same T-Server/SIP Server/Interaction Server. It is possible that the one of the two ICON instances might load an update to the attached data specification file slightly before the other.

  Because there is no synchronization mechanism between the HA ICON pair, ICON-1 might finish processing the updated attached data specification file at one time (T-0) and ICON-2 later, at another time, T-1.

  All interactions created on ICON-1 after T-0 use dictionary-2, but until T-1 ICON-2 still continues to use dictionary-1. This might produce some discrepancy in IDB tables related to user data. Only after T-1 do both ICON instances use dictionary-2 and unquestionably store the same data.

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.

> **Important**
> For information about attached user data that is sent in User Events, see Processing Attached Data.

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.

> **Important**
> A key-value pair is identified by its unique combination of the key name and the datatype specified for the value. When the datatype for the value changes, ICON creates new key-value pair and marks the key with the previous data type as deleted

in the G_USERDATA_HISTORY table.

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. Use the call-deletion-timeout Switch-level configuration option to configure the timeout interval.

Historical Data

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

- Party—When 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).
- Endpoint—When 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 Configuring for Attached Data in the *Interaction Concentrator Deployment Guide*.

The following table defines the values for types of attached data changes.

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.

The following table lists IDB tables that are used to store user data that is attached to voice calls. The attached data storage tables store user data about each call in a separate record (one record per call).

Attached Data Storage Tables for Voice

Table Name	Type of Data	Description
G_CALL_USERDATA	Call	Designed for predefined UserData that is collected during

Table Name	Type of Data	Description
		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.
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 in the gud-cust-disp-groups configuration option.

For an example of the XML specification for customized attached data processing, see Attached Data Specification File in the *Interaction Concentrator 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.

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.

Important
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 Sample Script for Custom Attached Data in the *Interaction Concentrator Deployment Guide*.

Attached Data Processing for Multimedia

This section briefly describes how ICON processes user data that is attached to multimedia interactions (Genesys eServices 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.

Important

A key-value pair is identified by its unique combination of the key name and the datatype specified for the value. When the datatype for the value changes, ICON creates new key-value pair and marks the key with the previous data type as deleted in the G_USERDATA_HISTORY table.

For an example of the XML specification for multimedia attached data processing, see the Multimedia Sample in the *Interaction Concentrator 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.

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.
- 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.

- 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 indicating that it was changed.

Monitoring Virtual Queues and Routing Points

This page 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 page contains the following sections:

- Monitoring Route Results on Virtual Queues
- Monitoring Route Results on Routing Points

For information about ICON configuration and other Configuration Layer settings that are related to virtual queue functionality, see the Configuring for Virtual Queue Data in the *Interaction Concentrator Deployment Guide*.

Monitoring Route Results on Virtual Queues

ICON can do the following:

- Monitor 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.
- Store, 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.

For detailed information about virtual queue data that is available in IDB, see the *Interaction Concentrator 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.

  Important

  For the following seven values (101 through 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 EventDiverted 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 G_ROUTE_RESULT table.

The results stored depend on the value set for the extended-route-result configuration option, as shown in the two following tables.

Important

To support the extended routing feature, certain URS and ICON configuration options must be set. For more information, see Configuring for Virtual Queue Data in the *Interaction Concentrator Deployment Guide*.

Summary of Values Stored in the G_ROUTE_RESULT Table
When **extended-route-result** = `0` (ICON release 7.5 functionality)

Reporting Event	Value of RESULT Field	Description of Stored Results
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 and higher functionality)

Reporting Event	Value of RESULT Field	Description of Stored Results
EventRouteUsed	1 (normal, ROUTE_RESULT_SUCCESS)	The call/interaction was routed by the URS strategy.
EventRouteUsed	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.
EventRouteUsed	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.
EventPartyRemoved	1 (normal, ROUTE_RESULT_SUCCESS)	The call/interaction was routed.

Reporting Event	Value of RESULT Field	Description of Stored Results
EventPartyRemoved	2 (abandoned, ROUTE_RESULT_FAIL)	The interaction was stopped.
EventPartyRemoved	134	The interaction was pulled out by Interaction Server.
EventPartyRemoved	133	Routing timeout, defined on Interaction Server, expired.
EventPartyRemoved	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, as shown in the following table.

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 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.

Important

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

Reporting Virtual Queue IDs Associated with a Route Result

Interaction Concentrator can store data on a virtual queue (VQ) ID associated with a routing point for T-Server, or with a routing strategy for Interaction Server in the G_ROUTE_RES_VQ_HIST table. It can store this data both for a call or interaction successfully routed by a strategy and for a call or interaction that is abandoned while being routed by a strategy

Important

This functionality requires URS release 8.1.100.08 or higher.

Universal Routing Server (URS) provides the VQ ID when the call or interaction is on the routing point. At that time, the strategy identifies the VQ associated with the target selected by the routing strategy.

The G_ROUTE_RESULT table stores the endpoint DN data, which can include routing points, but also other types of endpoint DNs on which a strategy is loaded and running, such as routing queues or service numbers. For convenience, the term *routing point* is used to describe the way this feature functions and how to set up an association of a Party with the VQ ID specified in the strategy loaded on the associated endpoint DN.

Important

The PARTYID and CALLID taken from when the call or interaction was queued and recorded in the G_ROUTE_RES_VQ_HIST table can be different from the PARTYID and CALLID noted when the call or interaction is distributed from or cleared from the VQ.

To store all VQ IDs associated with routing points (for T-Server) or routing strategies (for Interaction Server), URS must attach or update the UserData for the call or interaction with the RPVQID key each time a new VQ is specified in a strategy.

With the `gcc` role assigned, ICON monitors changes to the RPVQID UserData key. When it notes a change, it checks the Control Party that caused the change and searches for the related Party. If the Party is found with the Routing Point type for T-Server or the Routing Strategy type for Interaction Server, the changed value associated with the RPVQID key is stored in the G_ROUTE_RES_VQ_HIST table.

Scenario: Connection to T-Server or Interaction Server is Lost
VQ IDs reported in UserData while ICON is disconnected from T-Server or Interaction Server are not stored in IDB. After reconnection, the initial value associated with the RPVQID key is stored in memory, but not written to the G_ROUTE_RES_VQ_HIST table. From this starting point, any changes are then written to the G_ROUTE_RES_VQ_HIST table.

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 page describes how Interaction Concentrator (ICON) processes agent activity.

This page contains the following sections:

- Agent and Login Session Models
- Available Agent State and Login Session Data
- After-Call Work and Not-Ready Agent States
- Populating Agent Login Session Data

For information about ICON configuration and other Configuration Layer settings that make data about agent activity available in Interaction Database (IDB), see Configuring for Agent State and Login Data in the *Interaction Concentrator 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
- Agent State Model—Voice and Multimedia
- Login Sessions Model

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 the agent will change to when the Busy state is ended. When the agent changes from any state to Busy, ICON keeps the pre-change 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.
    - When the agent transitions from any state to Busy, ICON keeps the pre-transition state as 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 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.

> **Important**
> The software key name ReasonCode is reserved. Do not use this key name for your own reason codes.

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

The table below 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. The table also provides the T-Server or Interaction Server (MM) reporting events that represent entry into each state.

Agent States and State Transition Events Table

Agent State	Description/State Entry Event
Null	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. • TEvent Event • AgentLogout—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. • MM Event • 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	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). • TEvent • EventAgentLogin **Note:** The presence or absence of the ThisQueue parameter in the TEvent specifies whether the agent has logged on to an ACD queue. • MM Event • EventAgentLogin **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.
NotReady	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. • TEvent • EventAgentNotReady, with the workmode parameter *not* equal to AgentAfterCallWork. • MM Event

Agent State	Description/State Entry Event
	• EventNotReadyForMedia • EventMediaAdded—Provides information about a media type that was added to the agent's login session. **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.
Ready	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. • TEvent • EventAgentReady • MM Event • EventReadyForMedia
Busy	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. • 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. • In addition to ACD interactions, calls between agents, calls between supervisors and agents, and private calls can also cause transition to the Busy state.
ACW	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. **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 the ACW state follows the Busy state, ICON assigns the last call on the agent's device as related to the ACW state. • TEvent

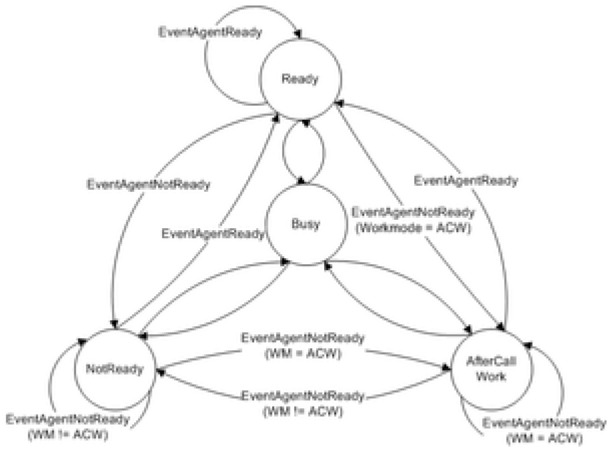
Agent State	Description/State Entry Event
	• EventAgentNotReady, with the workmode parameter equal to AgentAfterCallWork • MM Event • Not applicable
Unknown	An agent’s login session exists, but ICON has no information about the agent state (for example, because of disconnection from T-Server).

Important

Interaction Concentrator also supports custom agent states. For more information, see Configuring for Agent State and Login Data in the *Interaction Concentrator Deployment Guide*.

Agent State FSM

The figure below 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.



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 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.

Handling Stuck SIP Chat Interactions

Interaction Concentrator stores data about all interactions in the G_CALL and G_IR tables. Active voice interactions that were received from T-Server or SIP Server are also stored in the G_CALL_ACTIVE and G_IR_ACTIVE tables s long as they are active (not terminated).

When Interaction Concentrator is restarted, all interactions that are still stored in the G_CALL_ACTIVE and G_IR_ACTIVE tables will be marked as terminated in the G_CALL and G_IR tables.

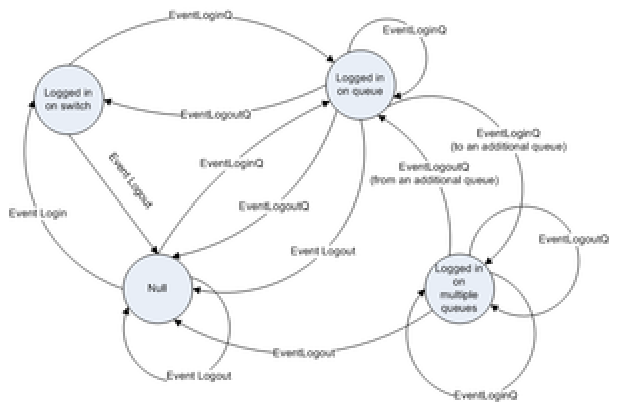
Login Sessions Model

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 gls-max-inactivity configuration option).

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.

The following figure represents the login session FSM.



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 and Login Session Tables. For more detailed information, see the *Interaction Concentrator 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 it stores hardware reason codes 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 ACW 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 end of 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 gls-enable-acw-busy configuration option to `true` on the Switch **Annex** tab to enable this behavior.

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 object. 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.

Populating Agent Login Session Data

ICON instances that perform the `gls` role maintain a persistent cache for agent login session data. The agent-pstorage-name configuration option 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.

Integrating with Outbound Contact

This page describes how Interaction Concentrator (ICON) processes data from the Genesys Outbound Contact solution. It contains the following sections:

- Outbound Objects and Models
- Available Outbound Data
- Outbound Contact Deployment Scenarios
- Extracting Outbound Contact Data

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

Outbound Objects and Models

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

Outbound Contact Objects

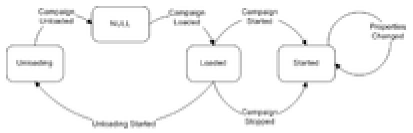
The following terms refer to the Outbound Contact objects and elements about which ICON stores reporting data:

- **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.

For more detailed information about Outbound Contact objects, see the Outbound Contact Deployment Guide.

Campaign Model

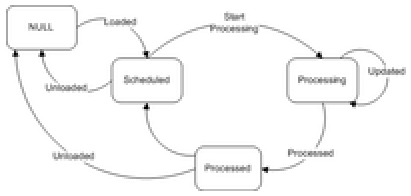
The following figure shows the finite state machine (FSM) for a campaign group.



Finite State Machine for a Campaign Group

Chain Model

The following figure shows the FSM for a chain.



Finite State Machine for a Chain

Chain States

In the figure "Finite State Machine for a Chain", 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.

For the full list and descriptions of IDB tables that store outbound data, see the *Interaction Concentrator 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.

The following table lists the precalculated metrics that OCS provides to ICON for storage in the GO_METRICS IDB table.

Precalculated OCS Metrics Arriving at Set Intervals
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.

Metric	Description
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).

Metric	Description
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 logical chain 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.
Current number of engaged (busy) ports per campaign group	Current number of ports used by engaged calls that CPD Server already placed. **Note:** OCS calculates this metric only for campaign groups that are activated in Active Switching Matrix (ASM) dialing modes with CPD Server.
Outbound Calls Overdialed	Indicates that Outbound Dialing Engine is to consider a given call overdialed. For information about overdialed calls, see the Outbound Contact documentation. **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 per-occurrence basis (that is, for each overdialed call).

Precalculated OCS Metrics Arriving at Each Occurrence
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 per-occurrence basis (that is, for each successfully dialed outbound call).

Important

OCS bases calculations related to call times on FTC trail (audit log) calculations.

Metric	Description
Outbound Call Dialing Time	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. **Note:** The time that the call was answered by the called party is available only when SIP Server or CPD Server is used for dialing. Therefore, if calls are not dialed through SIP Server or CPD Server, this metric will be available only for unsuccessful calls.
Outbound Call Transfer Time	The time, in milliseconds, between completion of CPD and the moment when the call was established on the Agent or IVR DN. **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.
CPD Time	The time, in milliseconds, from the moment when the call was answered by the called party until the moment when CPD was done. **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.

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. In this environment, you can use a single ICON instance for each site, or a single ICON instance for all sites.

Important

Each OCS instance can connect to multiple switches. If you simply add OCS to your **Connections** tab, ICON takes Outbound data from all switches to which the specified OCS instance(s) are connected. To take Outbound data only from some of the total pool of switches, add T-Server(s) associated with the desired switch(es) to the ICON `Application` object **Connections** tab. ICON then takes data only from the switches that are associated with the specified T-Servers.

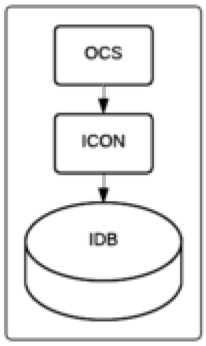
Diagram Conventions

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.
- 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 (shown in the figure below).



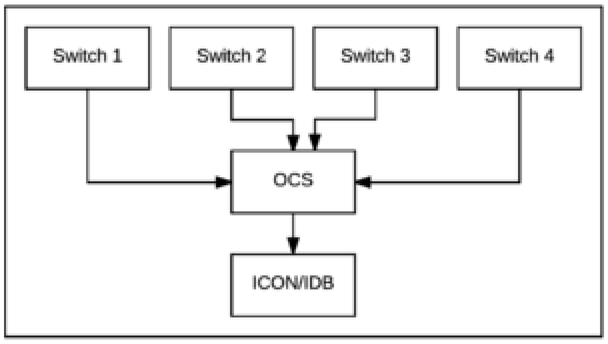
Single-Site Deployment: One OCS–One ICON

In this scenario, ICON is dedicated to storing outbound data, including both object information and precalculated metrics, which comes ultimately from the switch to which OCS is connected. 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 Examples

There are a number of options available if you want to collect Outbound Contact data from multiple sites/switches.

Example 1
The simplest multi-site deployment scenario consists of a single ICON instance connected to a single OCS that gathers data from multiple switches/sites. In the following architecture, ICON collects data from all of the switches connected to the OCS instance which is specified on the ICON `Application` object **Connection** tab.

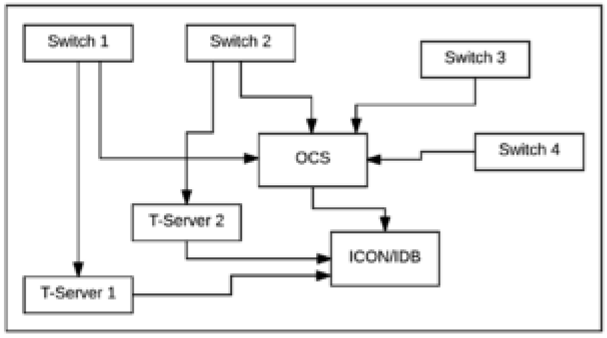


Multi-Site Deployment: One ICON Connected to OCS

Example 2
You might want to use a similar architecture, with a single OCS taking in data from multiple switches, but prefer to have ICON store data from only some of the switches. To limit the Outbound Contact data that ICON stores, add connections to the T-Servers associated with the desired switches to the ICON `Application` object **Connection** tab. This architecture is shown in the graphic below, in which ICON is connected to an instance of OCS that is then connected to four switches. To have ICON store Outbound Contact data only from switches 1 and 2, add connections to T-Servers 1 and 2, which are connected to the desired switches.

Tip

You must *also* add the associated OCS instance to the ICON `Application` object **Connection** tab or else ICON does not collect outbound data.

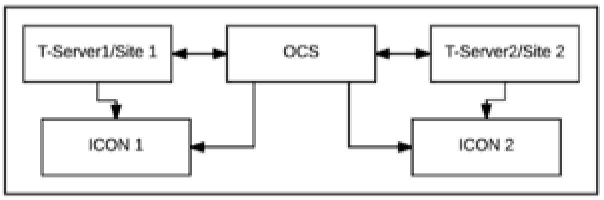


Multi-Site Deployment: One ICON Connected to OCS and T-Server

Example 3
The figure below shows a variation on the multi-site deployment scenario shown in Example 2. In this case, Outbound Contact data is split between two ICONs, though it is all processed by the same OCS instance.

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.



Multi-Site Deployment: Multiple ICONs Connected to OCS and T-Server

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.

Important
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:

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

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

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

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

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.

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
- Storing Data from EventUserEvent
- Processing Data from EventCustomReporting

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 Configuring for Agent State and Login Data.

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, and Interaction Server EventCustomReporting, to store associations between:

- Common data and the voice call/multimedia interaction (or the call/interaction party, when applicable).
- Custom states and the voice call (or the call party, when applicable). Note that Interaction Concentrator only tracks custom states for voice calls, not for multimedia interactions.

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

By default, ICON stores CallID and PartyID information for the sixteen most recent calls/interactions and parties associated with a device to be able to associate them with information from EventUserEvent and EventCustomReporting events. You can change the number of calls/interactions and parties stored by setting a different value in the max-party-info configuration option.

> **Important**
> If you remove a device from the configuration layer, all calls/parties associated with

> the device are lost. As a result, ICON cannot restore associations if an associated device was removed from the configuration before the arrival of EventUserEvent/ EventCustomReporting.

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 IDB initialization 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 are not supported.

Custom States

ICON stores data related to custom states from EventUserEvent in the G_CUSTOM_STATES table, which is created by the IDB initialization 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 8.1 Physical Data Model* document for your particular RDBMS.

Processing Data from EventCustomReporting

Interaction Concentrator stores EventCustomReporting data from any multimedia application that uses the Interaction Server protocol. At this time, the main use case for this functionality is to receive focus time data, via Interaction Server, from Genesys Workspace Desktop Edition v8.5.111 and higher, and provide this data for use by downstream Reporting applications.

Using EventCustomReporting from Interaction Server for Focus Time Reporting

ICON stores data from EventCustomReporting to enable reporting on how much time a particular interaction was in focus (that is, actively being processed) on the agent desktop. ICON tries to identify the interaction from the data included in the event. If ICON identifies the interaction, it writes the user data key-value pairs (KVPs) listed in **attr_event_content** into the G_CUSTOM_DATA_S table.

To enable ICON to process EventCustomReporting events:

1. Include the gud role in the ICON configuration.
2. Set the value of the store-event-data option to `all` or `conf`. If you set the value to `conf`, ICON stores the values of the keys that are configured in the **EventData** option.
3. To configure the EventData option, configure **R_TimeInFocus** (char), **R_AgentDBID** (int), **R_PlaceDBID** (int), and **R_InteractionId** (char) as special keys. These specify that you want to store focus time data and that identify the associated Agent, Place, or Interaction respectively.

Important

If you want to *exclude* new records from the G_CUSTOM_DATA_S table and ignore focus time information, you can filter out the relevant keys using the EventData option.

Mandatory EventCustomReporting Attributes

ICON processes EventCustomReporting only if it includes the following attributes:

- **attr_event_time**
- **attr_itx_id**; alternatively, ICON can use the optional **R_InteractionId** key.
- **attr_event_content** (a non-empty list), with the mandatory **R_PlaceDBID** key.
- **ReportingEventSequenceNumber** in **attr_extension**; ICON might try to process the data without this attribute, but the recorded data will lack the sequence number.

ICON ignores EventCustomReporting events with missing mandatory attributes.

EventCustomReporting Processing Logic

1. Keys and values are taken from the **attr_event_content** attribute of the EventCustomReporting event.
2. ICON uses the value of the **attr_itx_id** attribute to identify the interaction and checks whether the interaction is still alive.
3. ICON attempts to identify the endpoint Place using **R_PlaceDBID**. ICON does not process EventCustomReporting if **R_PlaceDBID** is missing.
4. ICON attempts to identify the agent using **R_AgentDBID**. If this attribute is not specified, ICON searches for an agent last logged into the Place found in the previous step. If the agent cannot be identified, ICON writes a `0` (zero) value for the agent identification attributes.
5. ICON identifies the Party based on the Place ID. Since EventCustomReporting may arrive after a Party left the Place, ICON keeps a history of Parties for a specific Place. For the specified Place, ICON searches for the interaction mentioned in **attr_itx_id** and uses the Party assigned to this interaction in this Place.
6. ICON writes data into the G_CUSTOM_DATA_S table, depending on the values you set for the store-event-data option and (optionally) the EventData option.

The following are differences in how ICON processes EventCustomReporting data compared with EventUserEvent data:

- The **EndPointDN** field in the G_CUSTOM_DATA_S table stores the Place name rather than DN name.
- The **SWITCH** field in the G_CUSTOM_DATA_S table stores the DBID of the Interaction Server Application object.
- ICON treats **R_TimeInFocus**, **R_AgentDBID**, **R_PlaceDBID**, and **R_InteractionId** as special keys and uses them only to identify the associated focus time value, Agent, Place, or Interaction. Unless these keys are configured explicitly using the **EventData** configuration option, ICON does not write these keys and their values into the G_CUSTOM_DATA_S table.
- ICON does not track custom states via EventCustomReporting.

The Interaction Concentrator HA Model

This topic and the following topic contain information about the high availability (HA) model in Interaction Concentrator, and about how you can implement HA for Interaction Concentrator.

- How Interaction Concentrator Achieves High Availability
- Extracting Data in an HA Deployment

For information about ICON configuration and other Configuration Layer settings that are required in an HA deployment, see Configuring for High Availability in the *Interaction Concentrator Deployment Guide*.

How Interaction Concentrator Achieves High Availability

This section 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 page contains the following sections:

- Overview
- ICON HA Model
- Consequences of Failures

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 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

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

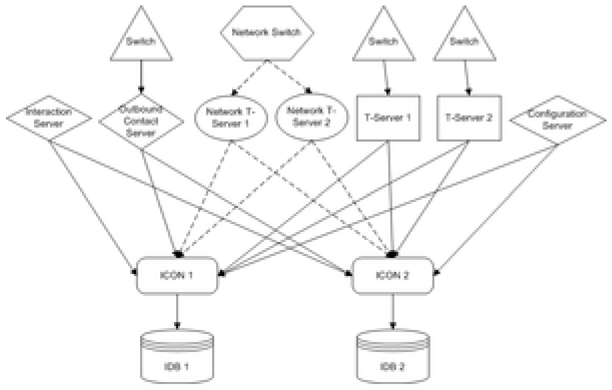
Different from Standard Genesys HA Model

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.

The figure below illustrates the major components of a full Interaction Concentrator HA solution. For simplicity, DB Servers and the connections between Configuration Server and the other Genesys components are not included. The Network T-Servers shown in the diagram are operating in load-balancing mode.



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

Important

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 HA design.

For more information about the data source session control tables, Data Source Session Control Tables.

For more information about using the data source session control tables to identify data interruptions, see Determining Data Availability and Reliability.

Consequences of Failures

Dropped server connections and other failures that interrupt data result in data inconsistencies in IDB. The table below summarizes the consequences of these kinds of failures in an ICON HA configuration.

IDB Inconsistencies Resulting from Failures

Point of Failure	Resulting Data Inconsistency
ICON connection to T-Server	• 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: • Agent states • Attached data • Changes in attached data • Parties

Point of Failure	Resulting Data Inconsistency
	• Call linkages
Computer-telephony integration (CTI) link	• 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
Active T-Server in an HA pair	• Incorrect transitions in TEvents • Stuck parties • Missing attached data • Other inconsistencies due to incomplete eventflows
ICON connection to Interaction Server	• 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: • Agent state • Attached data
ICON connection to OCS	• Missing information about: • Campaign processing • Changes in calling lists • Linkages between OCS and voice call data • Missing OCS precalculated metrics
ICON server	• As a result of data loss for the particular ICON instance as well as possible failure in restoring queued data from the persistent queue: • Stuck calls • Stuck parties

Point of Failure	Resulting Data Inconsistency
	• Missing records

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 page 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.

This page 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 page contains the following sections:

- Extracting HA Data
- Extracting Multimedia Data
- Extracting Virtual Queue-Specific Data
- Extracting Configuration Data
- Extracting Agent-Specific Data

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.

For information about the data source session control tables that support this functionality, Data Source Session Control Tables.

Interrupted Data Flow with Calls Scenario

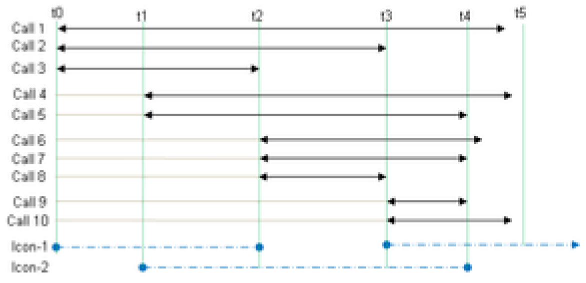
This section uses a sample scenario to illustrate two methods of optimizing data extraction:

- Extracting HA Data: Approach 1
- Extracting HA Data: Approach 2

Consider the following scenario:

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

The figure below, Calls in Relation to Interrupted Data Flow, illustrates the relationship between the calls and the interrupted data flow in this scenario.



Calls in Relation to Interrupted Data Flow

The following tables show 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 an explanation of the column headers, see the legend in the last row of the table.
- For more information about the G_DSS_GCC_PROVIDER table fields, see the Interaction Concentrator Physical Data Model for your RDBMS.

Record	DSS_ID	ICON_STIME	DSCONN_STIME	DSCONN_ETIME	FEVENT_DSTIME	LEVENT_DSTIME
ICON-1: Data Flow Interruption—G_DSS_GCC_PROVIDER Table Field Values in IDB-1						
1	37	t0	t0	t2	t0	t2
2	38	t3	t3		t3	t5
ICON-2: Data Flow Interruption—G_DSS_GCC_PROVIDER Table Field Values in IDB-2						
1	3767	t1	t1	t4	t1	t4
Legend (G_DSS_GCC_PROVIDER column names):						

Record	DSS_ID	ICON_STIME	DSCONN_STIME	DSCONN_ETIME	FEVENT_DSTIME	LEVENT_DSTIME
• 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

The tables above show 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

Important
The merge procedure is not supported on PostgreSQL RDBMSs.

The two following tables compare the reliability of data from ICON-1, from ICON-2, and from both

IDBs using a simple HA merge mechanism. The first table compares data for the case where ICON-1 terminates at t2. The second table compares data for the case where ICON-1 terminates at t2 and then restarts.

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
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.**

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
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

Call	ICON-1 (terminates at t2, restarts at t3)	ICON-2	HA (ICON-1 + ICON-2)
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 (Interrupted Data Flow with Calls Scenario), 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 Analysis.
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 Interrupted Data Flow with Calls Scenario), 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 Analysis.
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.

Important

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. 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:

1. Determine the timestamp of the data flow interruption (the process is described above in Extracting HA Data).
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 Interrupted Data Flow with Calls Scenario, 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 CAUSE field in the

G_VIRTUAL_QUEUE table for additional data validation. For example, a virtual queue record with a value of stuck (integer value = 3) in the CAUSE 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.

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.

Monitoring Interaction Concentrator

Important

From release 8.1.512.08 and higher, the functionality described on this page is no longer supported.

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

- Accessing the Performance Counter
- Performance Counter Pages

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.

Important

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 in 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:

1. 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 configuration option descriptions in the **[listeners]** section.

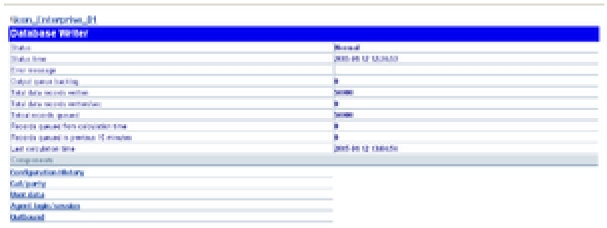
Performance Counter Pages

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 categorized as by DAP role.

The figure below shows a sample performance counter page—The main Database Writer page. Note that this figure is only an example of the web page. Your actual screen might look quite different from this one.



*Icon_Enterprise_01

Database Writer	
Status	Normal
Status time	[illegible]
Error message	
Output queue backlog	0
Total data records written	50000
Total data records [illegible]	0
Total records queued	50000
Records queued from calculation time	0
Records queued in previous 15 minutes	0
Last calculation time	[illegible]
Components	
Configuration history	
Call/party	
User data	
Agent login/session	
Outbound	

Main Database Writer Page

Filtering IDB Data

This page describes the Interaction Concentrator (ICON) data filtering capability and how to configure it by setting options on the ICON Application object and Script objects.

The data filtering feature enables users to configure data that will not be stored in IDB in order to maintain a smaller IDB and improve database performance.

This page contains the following sections:

Overview

The data filtering capability of ICON enables 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. See Introducing IDB Schema for an overview of IDB schema and its tables.

ICON also supports the ability to filter out multimedia interaction activity related to strategies if such activity data is not required for reporting purposes. To implement this feature, set configuration options in the **[callconcentrator]** section of your ICON `Application` object and the relevant `Script` configuration objects. For more information about the ICON `Application` object and relevant `Script` configuration options, see the *Interaction Concentrator Options Reference*.

Important

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
- Party Metrics
- User Data History
- Call Metrics
- Call History
- Interaction Record History
- Agent Activity Data
- Service Observer Data
- Strategy Activity Data

Important

In addition to the information contained here, refer also to **|[filter-data]** configuration options in the *Interaction Concentrator Options Reference* for information about setting configuration options in ICON, and the *Interaction Concentrator 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, to 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:

- Set the acd-party-history configuration option to `true` on the ICON `Application` object to 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, to choose not to store party metrics data for distribution devices in IDB:

- Set the acd-party-metrics configuration option to `true` in 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

To choose not to store external-party data into IDB storage:

- Set the external-party configuration option to `true` in 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

To have ICON not store the call–termination value of UserData keys to IDB:

- Set the udata-history-terminated configuration option to `true` in the ICON `Application` object.

ICON does 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. The following table describes the information stored in these IDB tables.

Types of Change Information Stored in IDB

Value	Column Change Type	Description
1	created	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 eventInteractionSubmitted event with an isOnline attribute of `true`.
2	added	Indicates that the value of the key has just been added.

Value	Column Change Type	Description
3	updated	Indicates that the value of the key has changed. For chat interactions, ICON assigns a change type of 3 upon receipt of an eventPropertiesChanged event with an isOnline attribute setting of either `true` or `false`.
4	deleted	Indicates that the key has been deleted from UserData.
5	terminated	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` in 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` in 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` in 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

To choose not to store *any* agent activity information in IDB:

- Set the gls-all configuration option to `true` in 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

To exclude agent state information:

- Set the gls-metrics configuration option to `true` in 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

To exclude data about agent activity at IVR endpoints from IDB storage:

- Set the gls-ivr configuration option to `true` in 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

To exclude data from IDB storage about agent activity for any agent whose login ID is not associated with any Person configuration object:

- Set the gls-no-person configuration object to `true` in 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 agent work modes and changes in agent work modes in the following IDB tables:

- G_AGENT_STATE_HISTORY
- G_AGENT_STATE_RC
- GS_AGENT_STAT_WM

To exclude data about changes in agent work mode that do not coincide with changes in agent state:

- Set the gls-wm configuration option to `true` on the ICON `Application` object.

ICON instead records a value of `unknown` in the IDB tables.

Important
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

To filter out information about the queues where agents are logged in:

- Set 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 does, 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

To choose not to store data about the party with the role observer in IDB:

- Set the observer-party configuration option to `true` in 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 does not store to IDB 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` in the ICON `Application` object 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.

Resynchronizing Configuration Changes

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

> **Important**
>
> 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 page 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

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.

Important

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.

The following table 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 Reliability Flag for a description of the values).

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 pre-existing value for the associated field.

The following table 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.

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 , as shown in the following table.

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.

Important

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). 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.
- 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 has detected an inconsistency in configuration data and logged a message accordingly. (However, the first inconsistency message logged following IDB upgrade is normal and should be ignored.)
- Your downstream reporting application (such as Genesys Info Mart) has detected 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...`

ICON 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 Genesys Administrator Extension. For information about alarm conditions and how to set them, see Alarm Conditions in the Genesys Administrator Extension Help.

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 (Recommendations for Resynchronization). You can also set an alarm Clearance Timeout which will clear the alarm condition after the specified time (in hours) has expired.

How to Resynchronize Configuration Data

By default, Interaction Concentrator resynchronizes with Configuration Server every time it starts. However, you may need to manually resynchronize ICON and Configuration Server, as discussed in the following section.

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 Genesys Administrator Extension, open the ICON `Application` and set the start-cfg-sync option to `0` on the **Options** tab.
3. Save your update.
4. Now change the option value set for the start-cfg-sync option to 1 on the **Options** tab.
5. Apply 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.

Important

It does not suffice to pre-set the value of the **start-cfg-sync** option to 1 or to set it 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.

After ICON has retrieved all the data necessary for resynchronization from Configuration Server, 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).

If you are using downstream reporting applications that rely on 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:

1. Check ICON logs for log event: `09-25017, Configuration objects are reloaded in IDB`.

2. Execute the following SQL statement against IDB:
   ```
   select eventid from G_SYNC_CONTROL where providertag = 5
   ```
   - If the SQL statement returns no records or eventID = `0`, the resynchronization is still in progress.
   - If the above statement returns a non-zero value for 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:

1. Restore the Configuration Database from a backup copy (see Restoring the Configuration Database from Backup).
2. Restart Interaction Concentrator.
3. Perform manual resynchronization of configuration data in IDB as described in How to Resynchronize Configuration Data.

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 that is 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, below).
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.

Important

These steps must be repeated for each type of configuration object stored by ICON (see the Configuration Server Object Types and IDB table.

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
   ```
   - Replace *IDB_TABLE_NAME* in the SQL statement with the correct table name (see the Configuration Server Object Types and IDB table).

2. If the result of the statement 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 the Configuration Server Object Types and IDB table.
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
   ```
   - Replace *cfg_object_type* in the SQL statement with the integer code of the Configuration Server object type (see the Configuration Server Object Types and IDB table).
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)
   ```
   - Replace *cfg_object_type* in the SQL statement with the integer code of the Configuration Server object type (see the Configuration Server Object Types and IDB table).
   - 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
   ```
   - Replace *cfg_object_type* in the SQL statement with the integer code of the Configuration Server object type (see the Configuration Server Object Types and IDB table).
   - 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.

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
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

Object Type	IDB Table Name: ***IDB_TABLE_NAME***	Config Server Object Type: ***cfg_object_type***
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

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 section describes the ICON stored procedures you will use, including instructions on how to set up and execute them. Examples for each supported RDBMS are also provided.

This section contains the following pages:

- Purge Stored Procedures
- Custom Dispatchers

The following stored procedures are provided for use in environments without Genesys Info Mart 8.x. Use these procedures *only* if you are running Genesys Info Mart 7.6 or earlier or are running Interaction Concentrator without Genesys Info Mart.

- Merge Stored Procedure
- gsysInitTimeCode Stored Procedure

Important

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

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 Deploying IDB, the wrapper script was executed as part of the IDB initialization or upgrade.

Purge 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.

The Interaction Concentrator purge stored 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.

You can execute only one instance of a purge procedure at any one time. The purge procedures can run in the background while ICON is writing data to the IDB.

Interaction Concentrator 8.1 provides three sets of purge procedures to purge data safely from IDB:

- gsysPurge81—Interaction Concentrator v8.1.x and higher provides the gsysPurge81 stored procedure. (Interaction Concentrator release 8.0.000.32 and earlier includes the prior version of the purge procedure, gsysPurge80, which does not purge Outbound Contact data.)
  - Purges data from IDB voice, multimedia, attached data, agent login sessions, and outbound data tables.
  - Use the gsysPurge81 purge procedure if one of the following applies:
    - You are a new ICON 8.1 customer.
    - You need to purge large amounts of data.
    - You are concerned about performance.
- purgePartitions811—If you are running Oracle 11g or higher and need to purge large amounts of data efficiently, you can create a partitioned IDB that can be purged by truncating partitions.
- gsysPurgeIR, gsysPurgeUDH, gsysPurgeLS, and gsysPurgeOS

> **Important**
>
> These separate purge procedures were discontinued in release 8.1.503.03. If you are using Interaction Concentrator 8.1.503.03 or higher, use gsysPurge81 or (in Oracle environments) purgePartitions811.

The separate stored procedures purge the following data from IDB:

- gsysPurgeIR—Purges interaction records (IRs).
- gsysPurgeUDH—Purges user data history (UDH) data.
- gsysPurgeLS—Purges Agent login and session (ALS) data.
- gsysPurgeOS—Purges Outbound Contact data.

Tables Purged by the Purge Stored Procedures

Not all IDB tables are purged. For efficiency, purging is reserved for tables liable to accumulate the greatest amount of data, while avoiding those that tend to have long-lived data with low accumulation rates. The following list of tables applies to all of the purge stored procedures:

GS_AGENT_STAT	GS_AGENT_STAT_WM	G_LOGIN_SESSION	G_AGENT_STATE_RC
G_AGENT_STATE_HISTORY	G_DND_HISTORY	G_CUSTOM_DATA_P	G_CUSTOM_DATA_S
G_CUSTOM_STATES	G_SECURE_UDATA_HIST	G_USERDATA_HISTORY	GM_F_USERDATA
GM_L_USERDATA	G_IR	G_IR_HISTORY	G_VIRTUAL_QUEUE
G_ROUTE_RESULT	G_IS_LINK	G_IS_LINK_HISTORY	GX_SESSION_ENDPOINT
G_PARTY	G_PARTY_HISTORY	G_PARTY_STAT	GO_SECURE_FIELDS
GOX_CHAIN_CALL	G_ROUTE_RES_VQ_HIST	G_CALL	G_CALL_HISTORY
G_CALL_STAT	G_CALL_USERDATA	G_CALL_USERDATA_CUST	G_CALL_USERDATA_CUST1
G_CALL_USERDATA_CUST2	GO_CHAIN	GO_CAMPAIGNHISTORY	GO_CAMPPROP_HIST
GO_CHAINREC_HIST	GO_CUSTOM_FIELDS	GO_FIELDHIST	GO_METRICS
GO_RECORD	GO_SEC_FIELDHIST		

Retention Period

All the purge procedures accept an input parameter, *number_of_days* or *count_days*, which specifies the retention period—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).

delete_all_flag Parameter

The gsysPurge81 purge procedure accepts a second input parameter, *delete_all_flag*, which further defines which data to purge in IDB.

Scheduling Considerations

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 7.x). 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.

Important

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.

If ICON experiences an issue that causes it to stop writing data to IDB, ensure that you suspend your purge procedure until ICON is running correctly. If you continue to run the purge procedure while there is a backlog of data ready to be written to IDB, depending on your settings, data may be purged before the downstream reporting application has a chance to run its ETL cycle and capture that data.

Using the gsysPurge81 Stored Procedure

The gsysPurge81 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 gsysPurge81 procedure uses two user-specified input parameters, *number_of_days* and *delete_all_flag*, to purge IDB data.

Important

The purge procedure has been renamed to GSYSPurge81Common; however, the wrapper name, GSYSPurge81, remains the same as in previous releases, so you do not need to change scripts as a result of this update.

number_of_days Input Parameter

Use the *number_of_days* parameter to calculate the retention period for data. Enter the number of days, including the current one, for which data should be retained. For example, if you run this stored procedure with *number_of_days* = 1, and *delete_all_flag*= 1 (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.

delete_all_flag Input Parameter

Use the *delete_all_flag* input parameter to define which data to purge in IDB. The only valid values for this parameter are `0` and `1`, where:

- 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.
- 1—ICON deletes all records (voice and multimedia) older than *number_of_days* day(s).

Important

The purge procedure deletes from the GO_CAMPAIGN and GO_CHAIN tables all records older than the number of days specified in the *number_of_days* parameter and does not take account of the value of the *delete_all_flag* parameter.

gsysPurge81 Purge Optimization

The gsysPurge81 stored procedure differs from the gsysPurgeIR procedure in that it does not consider merged interactions when purging data. Instead, gsysPurge81 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.

This purge procedure also provides the option to purge only terminated records using the *delete_all_flag* parameter. This will affect the time it takes to perform the purge operation, because the purge procedure 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.

Purge Lock Mechanisms

The lock mechanisms available depend on the release of Interaction Concentrator you are running.

Functionality in Release 8.1.503 and Higher

There are two purge locking options:

- Use of the the PSTATE record (supported only on Oracle and DB2).
- Native RDBMS lock mechanisms (supported only on Oracle, PostgreSQL, and Microsoft SQL).

Native Lock Mechanisms

RDBMS native lock mechanisms are also used to prevent more than one instance of the purge stored procedure from running simultaneously within a single data schema. Using the native lock mechanisms avoids the need to manually unlock the purge procedure if the process is halted by an outside command. Use the appropriate one of the following RDBMS-specific lock mechanisms:

- Oracle: `DBMS_LOCK package (DBMS_LOCK.REQUEST / DBMS_LOCK.RELEASE)`
- PostgreSQL: `Advisory Locks (PG_TRY_ADVISORY_LOCK / PG_ADVISORY_UNLOCK)`
- Microsoft SQL: `Application Locks (SP_GETAPPLOCK / SP_RELEASEAPPLOCK)`

Important

These native locks are automatically released when a session terminates.

Running the Purge Lock Mechanisms

On Microsoft SQL, PostgreSQL, and DB2 RDBMSs, you have only one option for purge locking. The available mechanism runs automatically when you start the purge procedure.

Special Considerations for Oracle

If you are using an Oracle RDBMS, you can chose to run whichever lock mechanism you prefer.

- By default, Oracle uses the PSTATE lock mechanism. If you choose to use this form of locking, you do not need to perform any extra steps when calling the purge procedure.
- If you choose to use the native lock mechanism, perform the following steps:

1. Make sure that you have the Oracle DBMS_LOCK package installed.

   Important

   Do not initialize IDB until after you install the Oracle DBMS_LOCK package.

2. Make sure that you have granted the EXECUTE privilege to the users who will be executing the purge stored procedure. For example, execute the following command:
   - `GRANT EXECUTE ON DBMS_LOCK TO USER1;`.
3. Modify GSYSPurge81 by changing the GSYSPURGE81PState() call to GSYSPURGE81NativeLock(). The example below shows how the command should appear:

```
CREATE OR REPLACE PROCEDURE GSYSPURGE81 (
                        NUM_OF_DAYS$           INTEGER,
                        DELETE_ALL_FLAG$     INTEGER)
 AS
 BEGIN
                GSYSPURGE81NativeLock (
                        NUM_OF_DAYS$,
                        DELETE_ALL_FLAG$);
 END;
```

Functionality in Releases Prior to 8.1.503

Releases prior to 8.1.503.xx use only the PSTATE lock mechanism.

The PSTATE Lock Mechanism

When the purge procedure starts, it tries to insert a special record, PSTATE, in the G_PURGE_STATE table.

- If the insert is successful, no other purge instances are running, and the purge procedure continues. The purge procedure has switched to *lock mode*.
- If the insert fails, the purge procedure halts because another purge process is already active.

After the purge procedure has successfully completed, the special PSTATE record is deleted from the G_PURGE_STATE table. Purging is now set to *unlocked mode*.

If the purge procedure stops unexpectedly, or if purging is halted by an external command, you must manually delete the PSTATE record from the G_PURGE_STATE table before you can run another purge instance.

gsysPurge81 Logging and Error Handling

The gsysPurge81 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 gsysPurge81 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 gsysPurge81 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)
- Input parameters are incorrect (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:gsysPurge81`

G_LOG_ATTRS

The G_LOG_ATTRS table stores parameters of log messages, generated by the gsysPurge81 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`*.

The table below shows all possible logging messages (MESSAGES_TEXT) and attributes (ATTR_NAME,ATTR_VALUE) that the gsysPurge81 purge procedure (APPNAME) generates and stores in the G_LOG_MESSAGES and G_LOG_ATTRS tables.

Important
You can read information from the G_LOG_MESSAGES and G_LOG_ATTRS tables in their entirety by using Genesys Solution Control Interface (SCI).

Messages and Attributes Generated by gsysPurge81

G_LOG_MESSAGES Table	G_LOG_ATTRS Table	
MESSAGETEXT	**ATTR_NAME**	**ATTR_VALUE**
Purge procedure *purge_procedure_version* started	MAX_PARTITION_TO_PURGE	*yyyymmdd*
	NUMBER_OF_DAYS_LEFT	*positive_integer*
	DELETE_ALL	*0* or *1*
Purge initialization completed	PartitionID	*0*
	Record_Count	*0*
GSYS_MARK_PARTITION mark partition started...	PARTITION_TO_MARK From	*yyyymmdd*
	PARTITION_TO_MARK To	*yyyymmdd*
purging in progress...	MAX_PARTITION_TO_PURGE	*yyyymmdd*
	PARTITION_COUNT	*number_of_terminated_and_non-terminated_records_in_partition*
	DELETE_ALL	*0* or *1*
GSYS_PURGE_PARTITION purge started...	PartitionID	*yyyymmdd*
	Record_Count	*0*
IDB:Purge-table:*table_name* initiated...	PartitionID	*yyyymmdd*
	Record_Count	*number_of_terminated_and_non-terminated_records_ in_table_that_belongs_to_partition*
IDB:Purge-table:*table_name* completed...	PartitionID	*yyyymmdd*
	Record_Count	*actual_number_of_deleted_records*
GSYS_PURGE_PARTITION purge completed	PartitionID	*yyyymmdd*
GSYSPurge81 purge completed	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 *delete_all_flag*.

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.MESSAGETEXT,G_LOG_ATTRS.attr_name,G_
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 gsysPurge81

It is the responsibility of the customer (usually the Database Administrator) to run the gsysPurge81 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 gsysPurge81 purge procedures.

Setting up gsysPurge81

The G_DB_PARAMETERS table stores parameters that the purge procedures use to control their operation. To set up the gsysPurge81 procedure, ensure that the parameter settings in IDB are suitable for your deployment.

The table below lists the parameters and their default values.

gsysPurge81 Parameters in the G_DB_PARAMETERS Table

G_DB_PARAMETERS Table		Description
OPT Column	**VAL Column**	
rowspertransaction	0	gsysPurge81 purges all records in a single table with the same partition ID in one transaction. This is the default value.

G_DB_PARAMETERS Table		Description
		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, gsysPurge81 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 = `'gsyspurge81'`.		

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 `'gsyspurge81'`), OPT (always `'rowspertransaction'`), and VAL.

Executing gsysPurge81

Genesys recommends that you execute the gsysPurge81 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 gsysPurge81.

Calling gsysPurge81

You must supply the *number_of_days* and *delete_all_flag* input parameters when you call the gsysPurge81 purge procedure. To execute the gsysPurge81 stored procedure, use the statement that corresponds to your RDBMS:

On Microsoft SQL
```
EXEC gsysPurge81 number_of_days, delete_all_flag
```
On Oracle
```
EXEC gsysPurge81 (number_of_days, delete_all_flag);
commit
```
On DB2
```
CALL gsysPurge81 (number_of_days, delete_all_flag);
```
On PostgreSQL
```
SELECT gsysPurge81 (number_of_days, delete_all_flag);
commit
```

Examples The following examples illustrate the syntax required to purge terminated multimedia interaction records older than 30 days:

Microsoft SQL
```
EXEC gsysPurge81 30,0
```
Oracle
```
EXEC gsysPurge81 (30,0);
```

commit
DB2
```
CALL gsysPurge81 (30,0);
```
PostgreSQL
```
SELECT gsysPurge81 (30,0);
commit;
```

Important

In these examples, gsysPurge81 retains any non-terminated records older than 30 days.

Purging by Truncating Partitions

Environments with large amounts of data to maintain can choose to partition their database and then purge efficiently by truncating entire partitions using the purgePartitions811 stored procedure. During this purge, all records in the specified partitions—both terminated and non-terminated—are truncated unconditionally.

If you need to purge only non-terminated records, use the gsysPurge81 purge procedure instead.

Important

This purge partitions functionality is supported only for Oracle 11g and higher.

To use the purgePartitions811 stored procedure, you must:

1. Set up a new IDB using the appropriate SQL scripts. For instructions, see the Configuring a Partitioned Oracle IDB in the *Interaction Concentrator Deployment Guide*.
2. Set a value of 1 or 2 for the partition-type configuration option. Note that Genesys Info Mart requires that you set the value to 2.

How purgePartitions811 Works

The number of partitions is set to 14 when you create your IDB.

Data for each individual day, as defined by UTC timestamps stored in the GSYS_TS field, is stored together in the same partition.

Depending on the value of the ICON partition-type option, the media type, and the specific table, the value stored in the GSYS_TS field might be the same as the created_ts field, might contain the timestamp when the interaction was submitted, or might contain the timestamp when the interaction was terminated.

The purge procedure does not require that a partition contain a certain amount of data before moving on to write into the next one.

ICON writes each day's data sequentially into the partitions starting with Partition 1 on Day 1 and continuing through Partition 14 on Day 14. When Partition 14 is filled, ICON returns to Partition 1 to record Day 15 data.

Important

If a partition is not purged before its next turn in the writing cycle, it could contain data from multiple days. For example, if on Day 15, Partition 1 has not yet been purged, Partition 1 will then contain data from both Day 1 and Day 15. In this scenario, when Partition 1 is purged, data from both Day 1 and Day 15 is truncated completely.

In general, ICON writes data from the current day into the current partition as it is gathered. However, if there happens to be a backlog of data for some reason, ICON uses the UTC timestamp values to write data to the appropriate partitions.

For example, let us assume that today is Day 3 and ICON is writing to Partition 3. However, if there is a backlog in writing data to IDB, and data from yesterday (Day 2) arrives, ICON writes the Day 2 data in Partition 2.

When you run the purge procedure, you specify the number of partitions to keep in the *number_of_days* parameter. Partitions are eligible to be purged based on where they are in the writing cycle relative to the partition into which ICON is currently writing.

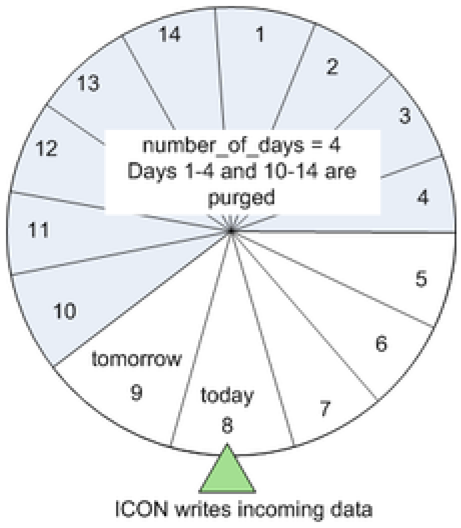
How to Configure the purgePartitions811 Procedure

When you run the purge procedure, you will specify the number of *completed* or *whole* days you want to keep. The purge procedure only considers whole days (as defined by UTC time) when calculating which days to keep and which to purge. ICON retains *number_of_days*+1 days/partitions. It keeps *number_of_days* partitions—including the one for the current day—and one future or *tomorrow* partition. The *tomorrow* partition provides a safeguard in case the purge procedure execution should overlap from one UTC day to the next. Without it, if the purge execution continued past the day boundary, data would start being written into a day that was being purged, with possible loss of data.

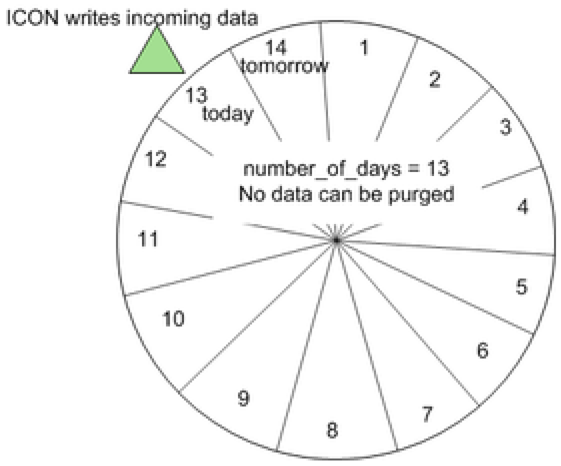
Example

There are 14 partitions, each of which contains the data for a day. The *number_of_days* parameter is set to 4. Today's partition is Number 8. When you run the purge procedure, it keeps four partitions, including today: Partitions 5-8. It also keeps Partition 9. Partitions 1-4 and 10-14 are truncated/ purged.

If the tomorrow partition, Partition 9, was truncated, and the purge process started on Day 8 and went past the day boundary to Day 9, then it would still be running while data started being written to Partition 9. If Partition 9 were then truncated, you might lose data.



The following graphic shows why 13 is an invalid value for the *number_of_days* parameter. All days are retained, no data is purged, and an error message is logged in the G_LOG_MESSAGES table.



Executing the purgePartitions811 Procedure

To execute the purge stored procedure, use the following statement, entering a *number_of_days* value of 12 or less:

```
EXEC purgePartitions811 (number_of_days);
commit;
```

Scheduling the purgePartitions811 Procedure

Genesys recommends the following guidelines for scheduling the purge procedure:

- Schedule execution of the stored procedure daily so as to consistently leave the number of partitions set in the *number_of_days* parameter in IDB.

- Refer to the documentation for your downstream reporting application for additional considerations and recommendations. For example, the Genesys Info Mart documentation recommends that you retain IDB data for at least 7 days to ensure that extract, transform, and load (ETL) processing, which might be delayed by network problems, ETL outage, or other issues, has time to complete before the data is purged.

If ICON Data Writing is Partially or Completely Interrupted

If for some reason, ICON is stopped or not writing data from some sources, this gap in writing data does not stop the purge procedure. It continues each day to advance the partition it considers to be the current one.

The purge procedure does not take into account any aspect of the data itself. For example, if records for a non-terminated interaction are located in a partition that is scheduled for purging, that partition will be truncated, regardless of the state of the data.

Important

For this reason, do not schedule automatic daily purging unless you also schedule an automated task to check that ICON is correctly writing all required data to IDB. For example, you might check DSS table information to make sure all source streams are current and that ICON has not fallen behind.

If, for some reason, purging is suspended so long that the partition into which ICON is now writing data already contains data from the previous writing cycle, the existing data is not lost.

However, it means that obsolete data is retained because that partition is not eligible to be purged until the value set in the *number_of_days* parameter has elapsed. This is also why the *number_of_days* parameter value must be smaller than the number of partitions, or else data is never purged.

Logging and Error Handling

The G_LOG_MESSAGES table logs the following information about the state of a purge procedure operation:

- `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.

The G_LOG_ATTRS table contains the following information about a purge procedure operation:

- Error descriptions (code and message).
- The value of the *number_of_days* input parameter.
- Information about the processed data, including the number of partitions purged from the each table subject to purging by truncating of partitions.

You can read information from the G_LOG_MESSAGES and G_LOG_ATTRS tables by using Genesys Solution Control Interface (SCI).

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.

Important

These separate purge procedures were discontinued in release 8.1.503.03. If you are using Interaction Concentrator 8.1.503.03 or higher, use gsysPurge81 or (in Oracle environments) purgePartitions811.

Purging Outbound Sessions Data

You can use either gsysPurgeOS or gsysPurge81 to purge outbound-session (OS) data. If you choose to use gsysPurge81, review the information in Using the gsysPurge81 Stored Procedure. The present section describes gsysPurgeOS.

The gsysPurgeOS procedure calls the gsysPurge_GOS stored procedure, which 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 are not deleted.

Tables Purged by the OS Purge Procedure

GO_CAMPAIGN	GO_CAMPPROP_HIST	GO_METRICS	GO_CHAIN
GO_CAMPAIGNHISTORY	GOX_CHAIN_CALL	GO_RECORD	GO_CHAINREC_HIST
GO_CUSTOM_FIELDS	GO_SECURE_FIELDS	GO_SEC_FIELDHIST	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

Genesys recommends using the gsysPurge81 purge procedure if you are purging large amounts of data, or if you are concerned about ICON's performance. See Using the gsysPurge81 Stored Procedure.

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.

The gsysPurgeIR procedure deletes only merged IRs.

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 Purged by the gsysPurgeIR Purge Procedure

G_IR	G_IR_HISTORY	G_CALL	G_CALL_HISTORY
G_PARTY	G_PARTY_HISTORY	G_IS_LINK	G_IS_LINK_HISTORY
G_ROUTE_RESULT	G_ROUTE_RES_VQ_HIST	G_VIRTUAL_QUEUE	G_CALL_STAT
G_PARTY_STAT	G_CALL_USERDATA	G_CALL_USERDATA_CUST	G_CALL_USERDATA_CUST1
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 does not contain references to a party that has been purged from the G_PARTY table.

Purging User Data History

Genesys recommends using the gsysPurge81 purge procedure if you are purging large amounts of data, or if you are concerned about ICON's performance. See Using the gsysPurge81 Stored Procedure.

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 Purged by 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

Genesys recommends using the gsysPurge81 purge procedure if you are purging large amounts of data, or if you are concerned about ICON’s performance. See Using the gsysPurge81 Stored Procedure.

The gsysPurgeLS procedure, which is used to purge Agent Login Session (ALS) data, calls the following stored procedures:

- gsysPurge_GLS—Deletes sessions.

The gsysPurgeLS procedure deletes only closed sessions.

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 Purged by the ALS Purge Procedure

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

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

The G_LOG_MESSAGES table logs the following information about the state of a purge procedure operation:

- `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.

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).

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.

Scheduling the Separate Purge Procedures

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 interaction records)
- gsysPurgeUDH (purge user data and user data handling data)
- gsysPurgeLS (purge agent login and session data)
- gsysPurgeOS (purge Outbound Contact 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 should 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.

Setting Up the Separate Purge Procedures

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. The table below lists the parameters and their default values.

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 are used for data lookup.
gsysPurgeUDH	UDH_time_step	300	Specifies the chunk size, in seconds, that is 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 is used to calculate the number of rows to purge in a single database

Applicable Procedure	G_DB_PARAMETERS Table		Description
			transaction.
gsysPurgeOS	OS_time_step	500	Specifies the chunk size, in seconds, that is 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 is allowed to execute.
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 Purge Parameters in the G_DB_PARAMETERS table), 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>’
```

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
- Example for Microsoft SQL
- Example for PostgreSQL
- Example for DB2

In these examples, the IR purge and ALS purge procedures are executed sequentially in the same session, and 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 PostgreSQL

```
declare COUNT_DAYS int;
declare COUNT_DAYS1 int;
begin
    COUNT_DAYS:= 14;
    perform gsysPurgeIR(COUNT_DAYS);
    COUNT_DAYS1:= 15;
    perform gsysPurgeLS (COUNT_DAYS1);
end;
```

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;
```

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 Customized Attached Data Processing.

The Merge Stored Procedure

Important

- The merge stored procedure described on this page is necessary only if you are running Genesys Info Mart 7.6 or earlier or you are running Interaction Concentrator *without* Genesys Info Mart. Genesys Info Mart provides a merge procedure that supplants the one documented on this page.
- This merge procedure is not supported on PostgreSQL RDBMSs.

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.

Important

If you are using Genesys Info Mart 7.6, run this procedure every five minutes. Genesys Info Mart 8.x 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
- gsysIRMerge2 Parameters

- Executing the Merge Procedure

gsysIRMerge and gsysIRMerge2

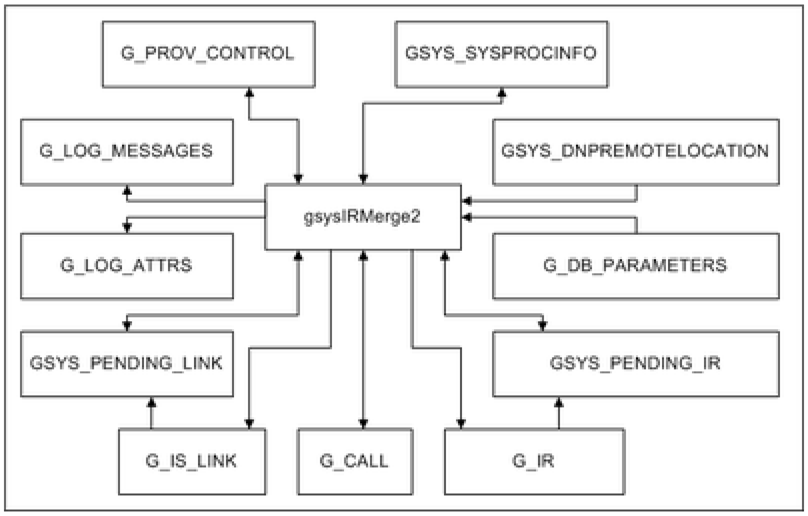
For backward compatibility and uncommon environments, Interaction Concentrator retains support for two stored procedures for voice data merge:

- 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.x.

Tables Used by gsysIRMerge2

The following figure shows the 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—Contains information about the start (MESSAGE_ID = 5020) and end (MESSAGE_ID = 5030) of the merge procedure.
- G_LOG_ATTRS—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

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_DNPREMOTELOCATION

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.

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.

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

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. The table below lists the parameters and their default values that the merge procedure uses.

Merge Parameters in the G_DB_PARAMETERS Table

OPT Column	VAL Column			Description
Oracle	**Microsoft SQL**	**DB2**		
loopresolver	0	0	0	Specifies whether the merge procedure should process the loopresolver parameter, which handles calls that form a call topology having loops. Valid values:
				1—Handle calls

OPT Column	VAL Column			Description
				that form a call topology with loops.
				0—Do not handle calls that form a call topology with loops. Setting VAL to 0 is equivalent to not configuring the parameter at all.
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.
brokenthreshold	14460	14460	14460	Specifies the time interval, in seconds, for which the merge procedure will wait for IS-Link information for a failed Inter Server Call Control (ISCC) transfer to another site. In other words, this parameter specifies the stuck link timeout for broken links.
step	75	75	25	Specifies the number of transactions that the merge

OPT Column	VAL Column			Description
				procedure selects at the same time (when it is reading in new IRs and new links). A low value limits exposure to locks on core tables. However, a very low value can result in too many iterations. For additional information about large-scale deployments using Microsoft SQL, see Note for Large-Scale Deployments Using Microsoft SQL.
limit	1000	2000	1000	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). This setting effectively limits the number of IRs per MSEQ. A very low value can result in too many iterations. For additional information about large-scale deployments using Microsoft SQL, see Note for Large-Scale Deployments Using Microsoft SQL.
limit2	10000	10000	10000	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).

OPT Column	VAL Column			Description
				A very high value can result in suboptimal performance.
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 the Merge Parameters in the G_DB_PARAMETERS table, 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>’
```

The brokenthreshold Parameter

The brokenthreshold parameter is used to streamline the processing of calls when an inter-site transfer fails. For example, consider the scenario in which ICON1 monitors Site1 and ICON2 monitors Site2, and an agent on Site1 attempts to transfer a call to Site2, but the attempted ISCC transfer does not reach Site2 or the agent on Site2 does not answer. In this situation:

ICON1 will receive information from T-Server that the link has failed, and the IS-Link information that ICON1 stores in IDB will be marked as failed (G_IS_LINK.STATE=3).

For Site2:

If the attempted ISCC transfer does not reach Site2, ICON2 will never receive information about the attempted transfer, and ICON2 will never store any corresponding IS-Link information in IDB.

If the call reaches Site2 but the agent does not answer, ICON2 will receive information from T-Server and will store corresponding IS-Link information in IDB, but the information might be delayed.

In both cases, by default, the merge procedure will wait up to 4 hours 1 minute for missing IS-Link information from ICON2 (instead of the default 8 hours 1 minute for other stuck links) before finalizing the merge.

To modify the value of the broken links timeout, insert a row into the G_DB_PARAMETERS table with the following attributes: SETID=`0`, SECT=`merge`, OPT=`brokenthreshold`, and VAL=*`new_timeout_value`*. Valid values are any positive integer. Changes take effect on the next run of the merge stored procedure.

Important

- If you set the brokenthreshold parameter to a value greater than the stuck link timeout (the stuckthreshold parameter), the stuckthreshold parameter will control the timeout for broken links as well.
- If you are migrating from an Interaction Concentrator release earlier than 8.0.000.42, the brokenthreshold parameter does not control the timeout for broken links that are already in the GSYS_PENDING_LINK table. The broken links timeout applies only to new broken links that are processed after you upgrade to release 8.0.000.42 or later.

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.

Important

Genesys Info Mart 8.x does not use the Interaction Concentrator merge stored procedures.

The table below 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.

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.

Important
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.x, 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, as shown in the Merge Procedure Parameters table.

> **Important**
> 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.)

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 Script for Oracle
- Example Script for Microsoft SQL
- Example Script for DB2

Example Script 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 Script 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 Script 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 Troubleshooting in the *Interaction Concentrator Deployment Guide*.

To minimize the occurrence of deadlocks and to optimize merge procedure performance, carefully review the information in Setting Up the Merge Procedure and in Troubleshooting in the *Interaction Concentrator Deployment Guide*. 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):

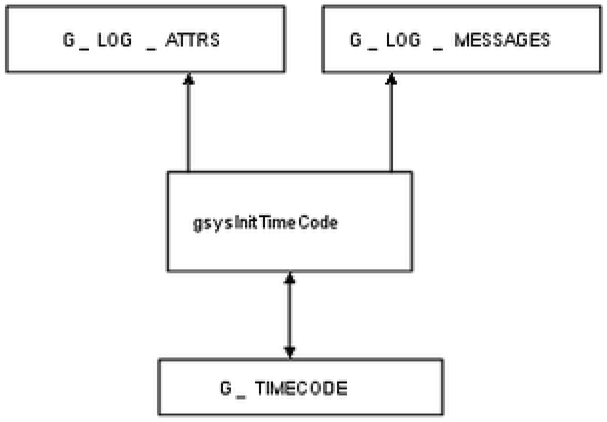
```
EXEC gsysIRMergeReset
```

The gsysInitTimeCode Stored Procedure

Important

The gsysInitTimeCode stored procedure is necessary only if you are running Interaction Concentrator *without* Genesys Info Mart. Genesys Info Mart provides functionality that supplants that documented on this page.

ICON uses the gsysInitTimeCode stored procedure to populate the G_TIMECODE table. The figure below shows the three tables that are 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

- 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.
- 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.

Important

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
```

Document Change History

This section lists content that is new or that has changed significantly since the first release of this document. The most recent changes appear first.

New in Document Version 8.1.514.03

- To support the addition of functionality enabling you to set an alarm for situations in which Interaction Concentrator call handling rules result in unprocessed or destroyed calls, added a new section, Setting Alarms for Call Processing Failures, to the How ICON Works topic.
- Corrected a note explaining how ICON handles situations when a configuration object is deleted while ICON is not connected.
- Specified the data types used to configure the EventData configuration option.

New in Document Version 8.1.514.02

- Added information about the the newly-introduced ability to configure how many last calls/interactions and parties associated with a device should be stored in the G_CUSTOM_DATA_P, G_CUSTOM_DATA_S, and G_CUSTOM_STATES tables. See Custom States in Interaction Concentrator.

New in Document Version 8.1.512.00

- Revised the discussion of supported Outbound Contact deployment scenarios to clarify how to set up Outbound Contact-related connections on the ICON Application object **Connections** tab.
- Added a note to the Monitoring Interaction Concentrator page indicating that, starting from release 8.1.512.08, the functionality documented on that page is no longer supported.

New in Document Version 8.1.510.00

- Added a note explaining how ICON, running with **role** = `cfg`, handles objects that were removed from permissions so that they became invisible to ICON.
- Moved two configuration-related sections from the How ICON Works page to the *Deployment Guide*: Configuring for Multi-Language Support and Configuring Conferencing and Transfer Options.

New in Document Version 8.1.509.00

- Added a section that lists the configuration options relevant to collecting conference and transfer data (section moved to the *Deployment Guide* in release 8.1.510.00).

New in Document Version 8.1.507.00

- Interaction Concentrator stores EventCustomReporting data from any multimedia application that supports the Interaction Server protocol. You can use this data to identify how long a particular interaction was in focus (that is, actively being processed) on the agent desktop. For details, see Processing Data from EventCustomReporting.
- ICON stores data provided by Chat Server that enables you to determine who ended a chat session. For details, see Chat Session Attributes that Indicate Who Ended the Session.
- The explanation of how the purgePartitions811 purge stored procedure works has been corrected. For details, see How to Configure the purgePartitions811 Procedure.

New in Document Version 8.1.506.00

- The explanation of how to identify who released a call has been updated to specify that ICON supports this functionality for all switches that provide the necessary information. To determine whether your switch supports this functionality, check your T-Server/SIP Server documentation.

New in Document Version 8.1.505.00

- ICON now hides sensitive attached data that is printed in the ICON log file. For details, see Security Features and Hide Selected Data in Logs (in the *Genesys Security Deployment Guide*).

New in Document Version 8.1.504.00

- Added the value `NULL` for the isOnline attribute, which is recorded when EventProcessingStopped is received. For a complete description of the isOnline attribute, see isOnline Chat Attribute.

New in Document Version 8.1.502.00

- Converted the document to the online wiki format, with the option of creating a PDF, from a PDF-only format.
- Added a section on the new purge lock mechanism available for the GSYSPurge81 procedure (Purge Lock Mechanisms).
- Added notes to indicate that the separate purge stored procedures was discontinued in this release (Using Separate Purge Procedures).
- Corrected the TEvents that can trigger ICON to record the end of the association between an interaction and a virtual queue to EventDiverted and EventAbandoned from EventQueued and EventAbandoned (Monitoring Virtual Queues and Routing Points > Data Processing Steps > Step Two—Record Update).

New in Document Version 8.1.401.00

- Added a description of the Interaction Concentrator ability to gather and store **Annex** tab data for certain types of configuration objects for use by Genesys Interactive Insights (see Annex Tab Data).
- Added a description of the Interaction Concentrator ability to store G_IS_LINK records within a call in the order that SIP Server initially added them (Out-of-Signaling-Path feature) (see Tracking Multi-Site Call Data Via ISCC).
- Added an explanation of the Interaction Concentrator ability to track multi-site ISCC interactions even when the external site data arrives after the call has been deleted (see Post-Mortem IS_LINK Capability).
- Added a section noting ICON support for Management Framework's ability to detect unresponsive server processes (see Determining ICON Responsiveness).
- Expanded the section on Security Features and moved it to a different section within the Overview page. Added database encryption support and support for hiding TEvent attached data in logs (see Security Features).

New in Document Version 8.1.201.00

- Added example PostgreSQL scripts to use when invoking the purge stored procedure (see On PostgreSQL and see Calling Purge Stored Procedures).
- Added notes informing users that Genesys Info Mart 8.x does not use the Interaction Concentrator merge stored procedures (see the Important notes in the Merge section and at the top of Merge Stored Procedure).
- Added a note indicating support for various security improvements, including TLS, TLS-FIPS, and client-side port definition (see Support for Secure Connections).
- Included a list of tables purged by the ICON purge stored procedures (see Tables Purged by the Purge Stored Procedures).
- Updated the name of the Genesys eServices product by changing it from *eServices/Multimedia* simply

to *eServices*, its current name. *Multimedia* was the former product name.

- Added a note advising users that they must restart ICON after making changes to the attached data specification file before the changes take effect (see the Important note under Attached Data Specification File. Note that starting in 8.1.5, Interaction Concentrator supports dynamic updates to the attached data specification file.)
- Removed an incorrectly-added section on database locking from the description of the purgePartitions811 stored procedure.

New in Document Version 8.1.101.00

- Added a section to the purge stored procedures chapter describing the new purgePartitions811 stored procedure, which is used to purge an Oracle 11g or higher RDBMS by truncating partitions (see Purging by Truncating Partitions).
- Added a note explaining behavior of certain Outbound Contact tables depending on the deleteAllFlag settings (see the Important note under delete_all_flag Input Parameter).
- Added G_ROUTE_RES_VQ_HIST table to lists of tables affected by gsysPurge81 and gsysPurgeIR (Tables Purged by the Purge Stored Procedures).
- Added a note informing users of the requirement to locate persistent queue files on a local drive (see the Important note under Persistent Queue and Persistent Caches).