This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Interaction Concentrator User's Guide

How ICON Works

12/19/2025

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