Interaction Concentrator User's Guide

Monitoring Virtual Queues and Routing Points

5/3/2025

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.