AVAYA

Avaya Aura® Contact Center / Avaya Contact Center Select Real-time Data API Programmer’s Guide

Release 7.1.1
Issue 0.1
October 2020

Note to Service Provider

Preventing Toll Fraud

"Toll Fraud" is the unauthorized use of your telecommunications system by an unauthorized party (for example, a person who is not a corporate employee, agent, subcontractor, or is not working on your company's behalf). Be aware that there can be a risk of Toll Fraud associated with your system and that, if Toll Fraud occurs, it can result in substantial additional charges for your telecommunications services.

Avaya Toll Fraud intervention

If you suspect that you are being victimized by Toll Fraud and you need technical assistance or support, call Technical Service Center Toll Fraud Intervention Hotline at +1-800-643-2353 for the United States and Canada. For additional support telephone numbers, see the Avaya Support website: http://support.avaya.com or such successor site as designated by Avaya. Suspected security vulnerabilities with Avaya products should be reported to Avaya by sending mail to: securityalerts@avaya.com.

Trademarks

Downloading Documentation

For the most current versions of Documentation, see the Avaya Support website: http://support.avaya.com or such successor site as designated by Avaya.

Contact Avaya Support

See the Avaya Support website: http://support.avaya.com for Product or Hosted Service notices and articles, or to report a problem with your Avaya Product or Hosted Service. For a list of support telephone numbers and contact addresses, go to the Avaya Support website: http://support.avaya.com (or such successor site as designated by Avaya), scroll to the bottom of the page, and select Contact Avaya Support.

Contents

Chapter 1: Introduction 6

Purpose 6
Intended audience 6
Support 6

Chapter 2: Overview 7

Real-time statistics 7
Obtaining real-time data 8
Update values 8
API summary 8
Data element storage functions 9
Query description functions 9
Data access functions 10
Data request functions 10
Preprocessing and postprocessing functions 11
Communication failure 12
Debug functions 12

Chapter 3: Installation 13

Development Environment 13
Execution Environment 13
Unicode / ANSI Flavours 13
Installing the RTDSDK 14
Migrating Existing RTDAPI Applications 14

Chapter 4: Real-time statistics 15

Introduction 15
Table definitions 15
Application statistics 16
Skillset statistics 19
Agent statistics 23
Nodal statistics 26
IVR statistics 27
Route statistics 28

Chapter 5: Real-time API definition 31

Type definitions 31
Data element storage functions 31
Value functions 31
Name functions 32

Query description functions *33*
Data access functions *40*
Data request functions *42*
Preprocessing and postprocessing functions *46*
Debug functions *51*
Error codes 51

Chapter 6: Sample Application 55

Introduction 55
Application design 55
Basic skeleton *55*
Setup query *56*
Callback function *57*
Source Code Description *58*
Programming tips 60

Chapter 1: Introduction

Purpose

This document provides information about the Avaya Aura Contact Center / Avaya Contact Center Select (AACC/ACCS) Real-time Data API (RTDAPI).

Intended audience

This document is intended for people who want to use the AACC/ACCS RTDAPI. It is primarily aimed at the software designers and developers responsible for developing RTDAPI applications.

Support

Visit the Avaya Support website at http://support.avaya.com for the most up-to-date AACC/ACCS documentation, product notices, and knowledge articles. You can also search for release notes, downloads, and resolutions to issues. Use the online service request system to create a service request. Chat with live agents to get answers to questions, or request an agent to connect you to a support team if an issue requires additional expertise.

Chapter 2: Overview

This programmer's guide describes how to use the Real-time Data (RTD) Application Programming Interface (API). It documents the functions and data structures that enable software developers to develop displays and agent desktop applications.

This document is applicable to the RTDAPI available in AACC/ACCS 7.

Real-time statistics

During normal operation, AACC/ACCS generates a variety of real-time statistics. The RTDAPI provides these statistics to applications using a C programming interface. The Basic Status Reporting package contains data from the following tables:

1. Application statistics
2. Skillset statistics
3. Agent statistics
4. Nodal statistics
5. IVR statistics
6. Route statistics

Each type of statistic is collected in two different ways—interval-to-date and moving window. The interval used for interval-to-date calculations is user-configurable in 15-minute increments from 15 minutes to 24 hours. The interval used for moving window calculations is predefined to be 10 minutes.

API architecture

The RTDAPI supports building 32-bit applications that run on Microsoft Windows operating systems.

- An application built using the API is expected to run on any Windows platform that supports running 32-bit applications. The Windows platform running the RTD client application is referred to as the API client.
- The API has been tested using Microsoft Visual Studio 2015 on Windows 10. While the API is not tested against all releases of Microsoft Visual Studio, it is expected to be compatible with all recent and future releases of Microsoft Visual Studio.

The API client accesses an AACC/ACCS by way of a TCP/IP connection. The API allows a single application to connect to a single server. To display a continuous stream of data from multiple servers, applications must connect to each server through a different process.

On AACC/ACCS, a Real-time Data session appears as a user logon. The server allows for a total of 100 client sessions. For instance, if there are 100 different client applications logged on to the server, then the next workstation client or application attempting to log on is rejected.

Application programs can obtain real-time data in two different ways:

- They can make one-time requests for data.
- They can register with a server for a continuous stream of data updates.

Obtaining real-time data

If a continuous stream of updates is requested, you must specify an update rate. Updates do not occur more frequently than the rate specified. Depending on system load, however, updates may occur less frequently than requested.

Each data request (one-time or continuous) obtains data from a single table on a single server. To request data, perform a Real-time Data query (conceptually similar to an SQL select) specifying:

- a table ID
- a list of columns (statistics)
- an optional where clause (selection criteria). If a where clause is not specified, the API returns the specified columns for all of the rows in the table.

For a one-time request, use the `NIrtd_singleDataRequest()` function. To start a continuous stream of data updates, use the `NIrtd_startDataStream()` function. In either case, the data is returned to the API client in three formats:

1. **newValues**—These are new rows of data that should be added to the application's data image (new rows of data have been added in the server data image). These rows should be added to the data menu of the API client.
2. **deletedValues**—These are rows of data that should be removed from the application's data image (rows have been removed from the server data image). Note that the returned values consist of keys to the data. The API client should take each key value, find its place in the data menu, and delete the row entry for that key.
3. **deltaValues**—These are rows of data that already exist in the application's data image but whose column data should be updated. In other words, existing rows have been modified in the server data image. Each key value should be used to locate the row in the application's data menu and the relevant column data should be updated.

The functions `NIrtd_allocateRow()` and `NIrtd_getCol()` allow you to progress through each table.

Update values

The first update from a data request (one-time or continuous) holds only the newValues table, which contains all of the rows that satisfy the request. Subsequent updates contain table rows that indicate new, deleted, or changed rows since the last received request. Within each table, key ID columns are returned to the application. These IDs are used when applying delta information.

Returned internal ID values can be converted to displayable names through a call to `NIrtd_getName()`. Examples of displayable names include agent telset login IDs, supervisor user IDs, skillset IDs, and application IDs. A name cache performs the translation.

API summary

The RTDAPI contains the following functional groups:

1. data element storage functions
2. query description functions
3. data access functions
4. data request functions
5. preprocessing and postprocessing functions
6. debug functions

Data element storage functions

The RTDAPI uses the following data element storage functions:

1. `NIrtd_allocateValue()`
2. `NIrtd_allocateName()`
3. `NIrtd_freeValue()`
4. `NIrtd_freeName()`
5. `NIrtd_cpValue()`
6. `NIrtd_cpName()`

Before calling any of the API functions where `NIrtd_stValue` or `NIrtd_stName` are used as parameters, the string space required by these structures must first be allocated by calling `NIrtd_allocateValue` and `NIrtd_allocateName`.

When you finish with either data element, call `NIrtd_freeValue` or `NIrtd_freeName` to deallocate the space.

`NIrtd_cpValue` and `NIrtd_cpName` are provided to make copies of the structures. Values typically represent data column values and names represent the displayable names for skillsets, applications, or agents.

Query description functions

The RTDAPI uses the following query description functions:

1. `NIrtd_allocateQuery`
2. `NIrtd_selectColumn`
3. `NIrtd_allocateConjunction`
4. `NIrtd_addCondition`
5. `NIrtd_addConjunction`
6. `NIrtd_getValue`
7. `NIrtd_freeConjunction`
8. `NIrtd_freeQuery`

A Real-time Data query consists of:

- a table ID
- a list of columns
- an optional where clause

The command `NIrtd_allocateQuery` allocates a query structure for the specified table. The command `NIrtd_selectColumn` is called to select the desired columns in the query.

The where clause is built from conjunctions, which are built from conditions. A condition refers to a logical expression that must be evaluated to be true. A conjunction refers to a list of conditions that are joined by a logical and operation. The command `NIrtd_allocateConjunction` allocates a conjunction to work with, the `NIrtd_addCondition` function adds a condition to a conjunction, and the command `NIrtd_addConjunction` adds a conjunction to the where clause of a query.

When using conditions, the API client must specify a key value to query for. The command `NIrtd_getValue` is useful for obtaining a key value for a condition. Given the textual name of the desired component (for example, an agent with the name John Doe), the routine returns an ID value that can be passed to the `NIrtd_addCondition` routine.

When done with a conjunction, issue the `NIrtd_freeConjunction` function to free memory allocated to the conjunction.

When you finish with the query (after calling `NIrtd_singleDataRequest` or `NIrtd_startDataStream`), issue the command `NIrtd_freeQuery` to free memory allocated to the query.

Data access functions

The RTDAPI uses the following functions for data access:

1. `NIrtd_allocateRow`
2. `NIrtd_getCol`
3. `NIrtd_freeRow`
4. `NIrtd_freeTableGroup`

The server returns data to the API client in a table group structure. The table group structure consists of three values table structures: a deleted values table, a new values table, and a delta values table. Each values table structure contains the number of rows and columns present in the table. When data in the tables are returned from the server, check the return code to verify that the data was received properly.

The command `NIrtd_allocateRow` retrieves and allocates access to a row in a values table. Use the function `NIrtd_getCol` to obtain the desired column value in a row.

When you finish with a row, issue the command `NIrtd_freeRow` to free memory allocated to the row. When you are done with the entire table group structure, issue the command `NIrtd_freeTableGroup` to free the memory allocated to the table group.

Data request functions

The RTDAPI uses the following functions to request data:

1. `NIrtd_login`
2. `NIrtd_singleDataRequest`
3. `NIrtd_startDataStream`
4. `NIrtd_stopDataStream`
5. `NIrtd_logout`

The command `NIrtd_login` establishes a session on the server and returns the authorization to the application program.

The command `NIrtd_singleDataRequest` is issued for a one-time data request. This function returns the requested data in the table group structure.
To receive regular updates for the same data fields, the proper command is `NIrtd_startDataStream`. This function registers a request for data with the server and returns a RequestID to the calling program.

Data is returned in a table group structure, which is passed to the callback function. The callback function is executed when data arrives at the API client. The callback function must be written within the third-party application and is a parameter passed to `NIrtd_startDataStream`.

Parameters returned in the callback function (`* NIrtd_funCallback) (ULONG return_code, NIrtd_tRequestId, requestid, NIrtd_stTableGroup * tableGroup, void * yourpointer)` include a return code, the request ID (originally returned from `NIrtd_startDataStream`), the table group structure, and the application pointer (originally passed to `NIrtd_startDataStream`).

Use the function `NIrtd_stopDataStream` to stop receiving updates. This command cancels the request that was initiated by a previous call to `NIrtd_startDataStream()`.

Use the command `NIrtd_logout` to end a session with the server.

Preprocessing and postprocessing functions

The RTDAPI uses the following pre-processing and post-processing functions:

1. `NIrtd_getNameCacheforDataColumn`
2. `NIrtd_getName`
3. `NIrtd_getFailedName`
4. `NIrtd_refreshNameCache`
5. `NIrtd_removeNameCacheforDataColumn`
6. `NIrtd_interpAgentState`
7. `NIrtd_setRecovery`

To translate agent telset login IDs, supervisor telset login IDs, skillset IDs, or application IDs into displayable names, the application must first initialize an internal name cache for each column ID by calling the function `NIrtd_getNameCacheforDataColumn`.

After the column name cache is initialized, individual ID values can be converted to a name by calling `NIrtd_getName`. If the `NIrtd_getName` routine returns `NIrtd_eNotFound`, then the application should indicate visually that the name is in a to be determined state, and then make an off-node blocking call to obtain the real name using the routine `NIrtd_getFailedName`.
Names that are changed on the server are not automatically propagated to the name cache of the third-party client. The name cache must be refreshed manually by calling `NIrtd_refreshNameCache`. `NIrtd_getName` can be called again to update each ID/name displayed.

When you finish with a name cache, make a call to `NIrtd_removeNameCacheforDataColumn` to free the memory used for the cache. Use the function `NIrtd_interpAgentState` to break down the multistate agent state value into multiple single state values.

Communication failure

Communication with the server can fail for many reasons. Some reasons can be detected by lower level communication software but others have no other symptom than the failure to propagate data to the application.

You can control a timer-based recovery mechanism. Use the command `NIrtd_setRecovery` to set the amount of time to wait before declaring a communication failure (the pullPlugTime plus the update rate) and the frequency at which the RTDAPI layer wakes up and checks the amount of time that has passed (wakeupGranularity).
Before a recovery attempt is made, the application's callback function is called with the return_code parameter of `NIrtd_eSTART_RECOVERY`.

After recovery, the application's callback function is called with a return_code parameter of either `NIrtd_eOK_RECOVERY` or `NIrtd_eBAD_RECOVERY`. If recovery fails, a delay of the pullPlugTime plus the update time occurs before another recovery attempt is made. The default pullPlugTime is 5 minutes with a wakeupGranularity of 1 minute.

Note: Communication with the server can fail due to the client's inability to accept data at the rate requested. The data propagation component on the server logs this kind of failed communication event. To remedy this situation, request a less frequent update rate or make performance enhancement changes to the client.

Debug functions

The RTDAPI uses the following debug functions:

1. `NIrtd_getFirstLowError`
2. `NIrtd_getCnt`

`NIrtd_getFirstLowError` should be called whenever an error code is returned by any API function. This function retrieves and resets the first lower level return code (that is, internal code value, which is useful in problem resolution).
`NIrtd_getCnt` retrieves the number of objects allocated. This is useful when trying to ensure the application is properly deallocating objects previously allocated.

Chapter 3: Installation

The RTDAPI Software Development Kit (RTDSDK) is required for developing and executing an RTDAPI application. The RTDSDK is available from Avaya DevConnect (www.devconnectprogram.com.

Development Environment

The RTDAPI application is developed, compiled and linked in the development environment. The development environment consistes of a C/C++ compiler/linker and the RTDAPI header and library files from the RTDSDK. The header files are in the `\include` folder while the libraries are in the `\lib` folder.

The RTDAPI has been tested using Microsoft Visual Studio 2015. While the API is not tested against all releases of Microsoft Visual Studio, it is expected to be compatible with all recent and future versions of Microsoft Visual Studio.

Execution Environment

The developed RTDAPI application requires an execution environment to connect to an operational AACC/ACCS. The execution environment consists of executables and Dynamic Lnk Libraries. The execution environment is contained in the `\bin` folder.

An application built using the RTDAPI is expected to run on any Windows platform that supports running 32-bit application. The RTDAPI has been tested on Windows 10. While the API is not tested against all Microsoft Windows releases, it is expected to be compatible with the latest versions of Microsoft Windows.

The execution environment has a dependency on the Microsoft Visual C++ 2015 Redistributable package. The RTDSDK installer detects if the redistributable is present. The installer can automatically install the redistributable if required.

Unicode / ANSI Flavours

The RTDAPI supports Unicode and ANSI build flavours. The application developer selects a build flavour dependent on the execution platform. UNICODE is the preferred flavour. ANSI is maintained for legacy support.

The library and execution environment are dependent on the build flavour. The default location for the library and run-time are:

	ANSI	**UNICODE**

Library	C:\Program Files (x86)\Avaya\RTDSDK\Ansi\lib	C:\Program Files (x86)\Avaya\RTDSDK\Unicode\lib
Run-time	C:\Program Files (x86)\Avaya\RTDSDK\Ansi\bin	C:\Program Files (x86)\Avaya\RTDSDK\Unicode\bin

Installing the RTDSDK

The RTD Software Development Kit (SDK) is available from Avaya DevConnect www.devconnectprogram.com.

How to install the RTDSDK

1. Remove any existing RTD SDK using the instructions in the Programmer's Guide for the version of the SDK.
2. Execute **RTDSDK.exe** and follow the installation wizard instructions.

How to uninstall the RTDSDK

From the **Control Panel**, select **Uninstall or Change a Program**. Then, select **Avaya RTD SDK** to un-install the SDK

Migrating Existing RTDAPI Applications

RTDAPI has been available since Symposium Call Center Server (SCCS).

An existing RTDAPI application designed for SCCS will require action to work with AACC/ACCS 7.

SCCS	Update Execution Environment	Rebuild
1.0	✓	✓
1.1 / **1.5** / **3.0** / **4.0** / **5.0** / **6.0**	✓	

An existing RTDAPI application designed for an earlier release of AACC/ACCS will work with AACC/ACCS 7. However, it is recommended that the execution environment is updated to take advantage of software quality improvements.

The environment is updated by removing the existing RTDSDK and installing the latest.

Chapter 4: Real-time statistics

Introduction

The tables in this section describe the statistics that belong to the Basic Status Reporting package. For each column in the tables, the data type is defined as Cumulative, State, or Admin. Statistics are available for multimedia contacts when the Open Queue feature is licensed and enabled. Telephony-specific statistics do not have meaning for multimedia contacts.

- **Cumulative—**The statistics are accumulated over a specified period of time (for example, the number of calls answered during an interval).
- **State**—The instantaneous state of the system (for example, the state of an agent at a given time).
- **Admin—**The value is entered by a data administrator and is not affected by call events (for example, a skillset ID).

For cumulative statistics, data can be collected in two different ways:

- **moving window**—The data is collected within the fixed size time window of 10 minutes that moves forward as time progresses. The fixed size time window is divided into a number of equal data sampling periods. As every sampling period expires, data collected in the current sampling period is added to the totals of the current time window while the values from the oldest sampling period within the current time window are subtracted from the totals. Therefore, the totals always represent the last 10 minutes of activity.
- **interval-to-date**—The data is collected on an interval basis. The interval is user-configurable in increments of 15 minutes up to a maximum of 24 hours. When the interval is complete, all data fields are reset to zero and collection starts for the next interval. The recommended minimum refresh rate (the rate at which the data is updated) for all statistics groups is 2 seconds.

Table definitions

The following tables contain the table definitions for interval-to-date and moving window statistics. Currently, you can configure the time interval used for interval-to-date statistics (15-minute increments, starting from 15 minutes to 24 hours), whereas the interval used for moving window calculations is set to 10 minutes.

Description	Statistic	Definition
Application statistics	Interval-to-date	NIrtd_INTRVL_APPL
Skillset statistics	Interval-to-date	NIrtd_INTRVL_SKLST
Agent statistics	Interval-to-date	NIrtd_INTRVL_AGENT
Nodal statistics	Interval-to-date	NIrtd_INTRVL_NODAL

Description	Statistic	Definition
IVR statistics	Interval-to-date	NIrtd_INTRVL_IVR
Route statistics	Interval-to-date	NIrtd_INTRVL_ROUTE
Application statistics	Moving window	NIrtd_MWIND_APPL
Skillset statistics	Moving window	NIrtd_MWIND_SKLST
Agent statistics	Moving window	NIrtd_MWIND_AGENT
Nodal statistics	Moving window	NIrtd_MWIND_NODAL
IVR statistics	Moving window	NIrtd_MWIND_IVR
Route statistics	Moving window	NIrtd_MWIND_ROUTE

Application statistics

Application statistics provide instantaneous state and cumulative performance measurement information on a per-application basis. An application corresponds to a single primary script (that provides call processing for a particular type of call) and all of its associated secondary scripts. For example, a department store's call center can have a catalog sales application and a credit card inquiry application.

Column	**Column ID**	**Data type**	**Description**	**Format**
Application ID	NIrtd_APPL_APPL_ID	Admin	A unique number to identify an application. (Key) (Translatable using NIrtd_getName and NIrtd_getValue)	ULONG
Calls Abandoned[a]	NIrtd_APPL_CALLS_ABAN	Cumulative	The number of local and incoming CDN calls abandoned.	ULONG
Calls Abandoned After Threshold[a]	NIrtd_APPL_CALLS_ABAN_AFT_THRESHOLD	Cumulative	The number of local and incoming network CDN calls abandoned after experiencing a delay greater than or equal to the service level threshold for the application. The delay is calculated from the time the call arrives (for local CDN calls) or from the time the call is logically queued (for incoming network CDN calls) to the time the call is abandoned.	ULONG
Calls Abandoned Delay[a]	NIrtd_APPL_CALLS_ABAN_DELAY	Cumulative	The total delay experienced by all abandoned local and incoming network CDN calls. The delay is calculated from the time the call arrives (for local CDN calls) or from the time the call is logically queued (for incoming network CDN calls) to the time the call is abandoned.	ULONG

Column	Column ID	Data type	Description	Format
Calls Answered[a]	NIrtd_APPL_CALLS_ANS	Cumulative	The number of local and incoming network CDN calls, ACD calls, and NACD calls answered. This also includes the number of local calls that are networked out and answered at the remote site.	ULONG
Calls Answered After Threshold[a]	NIrtd_APPL_CALLS_ANS_AFT_THRESHOLD	Cumulative	The number of local and incoming network CDN calls answered after experiencing a delay greater than or equal to the service level threshold for the application. The delay is calculated from the time the call arrives (for local CDN calls) or from the time the call is logically queued (for incoming network CDN calls) to the time the call is answered.	ULONG
Calls Answered Delay[a]	NIrtd_APPL_CALLS_ANS_DELAY	Cumulative	The total delay experienced by all answered local and incoming network CDN calls. The delay is calculated from the time the call arrives (for local CDN calls) or from the time the call is logically queued (for incoming network CDN calls) to the time the call is answered.	ULONG
Calls Waiting[a]	NIrtd_APPL_CALLS_WAITING	State	The number of local and incoming network CDN calls that are currently waiting. This also includes local calls that are logically queued at remote sites.	ULONG
Max. Waiting Time[a]	NIrtd_APPL_MAX_WAITING_TIME	State	The amount of time that the oldest unanswered local and incoming network CDN call has been in the system.	ULONG
Waiting Time[a]	NIrtd_APPL_WAITING_TIME	State	The total time waiting in the system of all local and incoming network CDN calls that are currently waiting.	ULONG
Calls Answered Delay At Skillset[a]	NIrtd_APPL_CALLS_ANS_DELAY_AT_SKILLSET	Cumulative	The delay experienced by all local and incoming network CDN calls from the time they are queued against the first skillset to the time they are answered.	ULONG

Column	Column ID	Data type	Description	Format
Calls Given Termination Treatment[a]	NIrtd_APPL_CALLS_GIVEN_ TERMINATE	Cumulative	The number of local and incoming network CDN calls that were terminated with one of the following treatments: 1. given Force Busy, Force Overflow, Force Disconnect, Route Call, or Default. 2. reached a non-ISDN trunk while being routed to a remote site. (**Networking feature**) 3. transferred in an IVR session. (**IVR feature**) 4. networked out via an NACD queue. (**NACD feature**).	ULONG
Calls Offereda	NIrtd_APPL_CALLS_OFFE R	Cumulative	The number of local and incoming network CDN calls, ACD calls, and NACD calls that were offered.	ULONG
Time Before Interflow	NIrtd_APPL_DELAY_BEF_ INTERFLOW	Cumulative	The amount of time a call spent in the Master Application before interflowing to the Primary Application. For the Master Application, this value is the total delay before interflow to all Primary Applications. For each Primary Application, this provides a delay spent in the Master Application or calls answered at this application.	ULONG
Network Out Calls[b]	NIrtd_APPL_NETWRK_OU T_CALLS	Cumulative	**Networking feature** The number of local CDN calls that were networked out from this application.	ULONG
Network Out Calls Abandoned[b]	NIrtd_APPL_NETWRK_OU T_ABAN	Cumulative	**Networking feature** The number of outgoing network CDN calls that were networked out from this application and abandoned at destination sites.	ULONG
Network Out Calls Abandoned Delay[b]	NIrtd_APPL_NETWRK_OU T_ABAN_DELAY	Cumulative	**Networking feature** The total delay experienced by local CDN calls that were networked out from this application and abandoned at destination sites.	ULONG

Column	Column ID	Data type	Description	Format
Network Out Calls Answered[b]	NIrtd_APPL_NETWRK_OUT_ANS	Cumulative	**Networking feature** The number of local CDN calls that were networked out from this CCMS application and answered by an agent or by IVR, or received termination treatment, music, or RAN at destination sites.	ULONG
Network Out Calls Answered Delay[b]	NIrtd_APPL_NETWRK_OUT_ANS_DELAY	Cumulative	**Networking feature** The total delay experienced by all local CDN calls that were networked out from this application and answered by an agent or by IVR, or received termination treatment, music, or RAN treatment at destination sites.	ULONG
Network OutCalls Waiting[b]	NIrtd_APPL_NETWRK_OUT_CALLS_WAITING	State	**Networking feature** The number of local CDN call requests sent from this application that are currently waiting at destination sites.	ULONG
Network Out Calls Requested	NIrtd_APPL_NETWRK_OUT_CALLS_REQ	State	**Networking feature** The number of network calls that were sent to another site	ULONG

a) This statistic includes calls that originally entered AACC at this site and calls that were received at this site from the Contact Center network. Delays are calculated from the time the call enters this site if it is a local CDN call or from the time the call is logically queued to this site if it is a network call.
b) Network Out statistics refer to calls that originally entered the AACC at this site butt were sent to another site on the Contact Center network. Delays for Network Out statistics are calculated from the time the call arrives at the source site to the time the call is treated (either answered, abandoned, or terminated) at the destination site.

Skillset statistics

Skillset statistics provide instantaneous state and cumulative performance measurement information on a per-skillset basis. If the agent is not logged on, no statistical data is available for that particular skillset.

Column	Column ID	Data type	Description	Format
Skillset ID	NIrtd_SKLST_SKILLSET_	Admin	A unique number to identify a skillset. (Key) (Translatable using	ULONG

Column	Column ID	Data type	Description	Format
	ID		NIrtd_getName and NIrtd_getValue)	
Agents Available	NIrtd_SKLST_AGENT_AVAIL	State	The number of agents who are currently waiting for calls.	ULONG
Agents In Service	NIrtd_SKLST_AGENT_IN_SERVICE	State	The number of agents logged on for this skillset.	ULONG
Agents on Skillset Calls	NIrtd_SKLST_AGENT_ON_ICCM_CALL	State	The number of agents who are logged on for this skillset and are currently handling local and network CDN calls assigned to this skillset.	ULONG
Agents Not Ready	NIrtd_SKLST_AGENT_NOT_READY	State	The number of agents currently in the Not Ready State who are logged on for this skillset.	ULONG
Calls Waiting	NIrtd_SKLST_CALL_WAIT	State	The number of local and incoming network CDN calls currently waiting for an agent with this skillset.	ULONG
Longest Waiting Time Since Last Call	NIrtd_SKLST_LONGEST_WAIT_TIMES_SINCE_LAST_CALL	State	The longest waiting time of all idle agents who are currently waiting to answer calls for this skillset. The time is since last call.	ULONG
Max. Waiting Time	NIrtd_SKLST_MAX_WAIT_TIME	State	The maximum waiting time spent by all local and incoming network CDN calls that are currently waiting for an agent with this skillset.	ULONG
Waiting Time	NIrtd_SKLST_TOT_WAIT_TIME	State	The total waiting time spent by all local and incoming network CDN calls that are currently waiting for an agent assigned to this skillset.	ULONG
Expected Wait Time	NIrtd_SKLST_EXPECT_WAIT_TIME	State	The time that a new call is expected to wait before being answered by an agent with this skillset.	ULONG
Calls Answered After Threshold	NIrtd_SKLST_CALL_ANS_AFT_THRESHOLD	Cumulative	The number of local and incoming network CDN calls that were answered after experiencing a delay greater than or equal to the service level threshold for this skillset. This statistic is not applicable for ACD and NACD calls because answering delay	ULONG

Column	Column ID	Data type	Description	Format
			information is not available for these types of calls.	
Longest Waiting Time Since Login	NIrtd_SKLST_LONGEST_WAIT_TIMES_SINCE_LOGIN	State	The longest waiting time of all idle agents who are currently waiting to answer calls for this skillset. The time is calculated since logon.	ULONG
Agents on DN Calls	NIrtd_SKLST_AGENT_ON_DN_CALL	State	The number of agents who are logged on for this skillset but are currently handling DN calls. Note: CS1000 reports agent active on an outgoing DN call only after the called party answers the call.	ULONG
Skillset State	NIrtd_SKLST_SKILLSET_STATE	State	The state of the skillset (In Service or Out Of Service).	ULONG
Agents Unavailable	NIrtd_SKLST_AGENT_U NAVAILABLE	State	The number of agents who are currently unavailable to take calls. This value is calculated base on: (# Agents In Service) - (# Agents Available)	ULONG
Network Calls Waiting	NIrtd_SKLST_NETWRK_CALL_WAIT	State	**Networking feature** The number of incoming network CDN calls currently waiting at this skillset.	ULONG
Network Calls Answered	NIrtd_SKLST_NETWRK_CALL_ANS	State	**Networking feature** The number of incoming network CDN calls answered by an agent assigned to this skillset.	ULONG
Total Calls Answered Delay	NIrtd_SKLST_TOT_ANS_DELAY	Cumulative	The delay experienced by all local and incoming network CDN calls that were answered by an agent with this skillset from the time the calls were queued against the skillset until they were answered. This statistic is not	ULONG

Column	Column ID	Data type	Description	Format
			applicable for ACD and NACD calls because answer delay information is not available for these types of calls.	
Total Calls Answered	NIrtd_SKLST_TOT_CALL_ANS	Cumulative	The number of local and incoming network CDN calls, ACD calls, and NACD calls answered by an agent assigned to this skillset.	ULONG
Agent On Network Skillset Call	NIrtd_SKLST_AGENT_ON_NETWRK_ICCM_CALL	State	**Networking feature** The number of agents who are logged on for this skillset and are currently handling network CDN calls assigned to this skillset.	ULONG
Agent On Other Skillset Call	NIrtd_SKLST_AGENT_ON_OTHER_ICCM_CALL	State	The number of agents who are logged on for this skillset but are active on calls for other skillsets. The other skillset can be a local skillset, a network skillset, or an Agent Queue To skillset.	ULONG
Agent On ACD-DN Call	NIrtd_SKLST_AGENT_ON_ACD_CALL	State	The number of agents who are logged on for this skillset but are currently handling ACD-DN calls.	ULONG
Agent On NACD-DN Call	NIrtd_SKLST_AGENT_ON_NACD_CALL	State	The number of agents who are logged on for this skillset but are currently handling NACD-DN calls.	ULONG
Calls Offered	NIrtd_SKLST_CALL_OFFERED	Cumulative	The number of calls queued to this skillset; these calls might or might not be answered by this skillset. The count is not increased if a call is queued to this skillset more than once.	ULONG

Column	Column ID	Data type	Description	Format
Network Calls Offered	NIrtd_SKLST_NETWRK_CALL_OFFERED	Cumulative	The number of incoming network CDN calls queued to this skillset.	ULONG
SkillsetAbandon	NIrtd_SKLST_CALL_ABANDON	Cumulative	The number of calls that were abandoned by callers while being queued to this skillset.	ULONG
SkillsetAbandonDelay	NIrtd_SKLSET_CALL_ABANDONDELAY	Cumulative	The amount of delay experienced by calls that were abandoned by callers while being queued to this skillset; the delay value is calculated from the time the call was queued to this skillset to the time it was dequeued.	ULONG
SkillsetAbandonDelayAfterhreshold	NIlrtd_SKLSET_CALL_ABANDONDELAY_AFTERTHRESHOLD	Cumulative	The number of calls whose SkillsetAbandonDelay values were greater than or equal to the service level threshold.	ULONG
Queued CallAnswered	NIrtd_SKLSET_QUEUED_CALL_ANS	Cumulative	The number of queued calls that were answered for the skillset within the last interval-to-date or movingwindow.	ULONG

An agent can log on to more than one skillset at any time. Therefore, if an application sums Agents Available for each skillset, the value obtained is generally greater than the total number of agents in the contact center who are available to take calls. The same is true for Agents in Service and Agents Not Ready. This is not the case for Agents on Skillset Calls, that is, the sum of Agents on Skillset Calls for each skillset is equal to the total number of agents currently answering skillset calls in the contact center.

Agent statistics

Agent statistics provide instantaneous state information regarding an agent (call taker). These statistics provide a supervisor with a means to monitor what their agents are doing at any point in time. If the agent is not logged on, no statistical data is available for that particular agent.

Column	Column ID	Data type	Description	Format
Agent ID	NIrtd_AGENT_AGENT_ID	Admin	A unique number to identify an agent. (Key) (Translatable using NIrtd_getName and NIrtd_getValue)	BYTE(17) STRING

Column	Column ID	Data type	Description	Format
State	NIrtd_AGENT_STATE	State	Indicates the state the agent is currently in. Note that this state can be one single state or a combination of two or more states. The following is a list of possible states: Undefined—the state of agent is unknown Busy Not Ready—Not Ready key activated Waiting for CDN call Reserved for a call **(NACD**/**Networking feature**) Skillset call active NACD call active **(NACD feature**) ACD call active DN In/Out call active CDN call on hold NACD call on hold **(NACD feature**) ACD call on hold DN In/Out call on hold DN In/Out call on hold and active CDN call active and DN In/Out call on hold NACD call active and DN In/Out call on hold (**NACD feature**) ACD call active and DN In/Out call on hold CDN call on hold and DN In/Out call active CDN call on hold and DN In/Out call on hold CDN call on hold and DN In/Out call active and on hold NACD call on hold and DN In/Out call active **(NACD feature**) NACD call on hold and DN In/Out call on hold **(NACD feature**) NACD call on hold and DN In/Out call active and on hold **(NACD feature**) ACD call on hold and DN In/Out call active ACD call on hold and DN In/Out call on hold ACD call on hold and DN In/Out call active and on hold Not Ready and DN In/Out call active Not Ready and DN In/Out call on hold Not Ready and DN In/Out call on hold and active Consultation with out caller CDN call presented Emergency Walkaway or Walkaway combination with other states	ULONG

Column	Column ID	Data type	Description	Format
Supervis or ID	NIrtd_AGENT_SUP ERVISOR_ID	Admin	Agent’s primary supervisor’s unique identifier.	BYTE(17) STRING
Time In State	NIrtd_AGENT_TIM E_IN_ST ATE	Cumulative	The length of time that the agent has been in this state. The only exception is when the agent is on a DN call, in which case the agent state is shown as BUSY.	ULONG
Answerin g Skillset	NIrtd_AGENT_ANS _SKILLSET	State	The ID of a skillset for which this agent is currently answering a skillset call. (Translatable using NIrtd_getName and NIrtd_getValue)	ULONG
DN In Time In State	NIrtd_AGENT_DN_ IN_TIME_IN_STAT E	Cumulative	The length of time an agent has been in the DN IN state; that is, answering incoming DN calls.	ULONG
DN Out Time In State	NIrtd_AGENT_DN_ OUT_TIME_IN_ST ATE	Cumulative	The length of time an agent has been in the DN OUT state; that is, making outgoing DN calls.	ULONG
Supervis or User ID	NIrtd_AGENT_SUP ERVISOR_USER_I D	Admin	Agent’s primary supervisor blue user ID. (Translatable using NIrtd_getName and NIrtd_getValue)	BYTE(16) BUFFER
Position ID	NIrtd_AGENT_POS ITION_I D	Admin	A unique identifier of the agent’s position ID.	ULONG
Not Ready Reason Code_Hi gh and Not Ready Reason Code_Lo w	NIrtd_AGENT_NOT _READY_REASON	State	The Not Ready reason code entered by the agent.	STRING
DN Out Call Number_ Hi gh and DN Out Call Number_ Low	NIrtd_AGENT_DN_ OUT_CALL_NUM	State	The DN number dialed by an agent.	STRING
Skillset Calls	NIrtd_AGENT_SKL ST_CALL_ANS	Cumulative	The number of local and incoming network CDN calls answered by an agent.	STRING

Column	Column ID	Data type	Description	Format
Answered				
DN InCall Answered	NIrtd_AGENT_DN_IN_CALL_ANS	Cumulative	The number of DN calls answered by an agent.	STRING
DN OutCall Made	NIrtd_AGENT_DN_OUT_CALL	State	The number of DN calls made by an agent.	STRING
Answering Application	NIrtd_AGENT_ANS_APP	State	A unique number to identify an application.	STRING
Answering CDN_Low And AnsweringCDN_High	NIrtd_AGENT_ANS_CDN	State	A special directory number that allows incoming calls to be queued at a CDN when they arrive at the switch.	STRING
Answering DNIS_High And Answering DNIS_Low	NIrtd_AGENT_ANS_DNIS	State	The phone number dialed by the incoming caller.	STRING

For CS1000 connectivity, an agent can be assigned multiple DN keys. Therefore, an agent can be in a state that they are answering a DN call as well as placing another DN call on hold.

Nodal statistics

Nodal statistics provide instantaneous state and cumulative accounting information for a next generation Call Center server. Usually, a call center has a single server and the nodal statistics are equal to the call center statistics. In the Basic Status Reporting package, only one nodal statistic is available.

Column	Column ID	Data type	Description	Format
Dummy Key	NIrtd_NODAL_DUMMY_KEY	Admin	An artificial key for use by the application. (This is provided to the application to make the interface	ULONG

Column	Column ID	Data type	Description	Format
			consistent, allowing for an easier application of delta, delete, and new table values.)	
Calls Offered[a]	NIrtd_NODAL_CALL_OFFER	Cumulative	The number of local CDN calls, incoming network CDN calls, ACD calls, and NACD calls that were offered to this site.	ULONG
Calls Answered[a]	NIrtd_NODAL_CALL_ANS	Cumulative	The number of local CDN calls, incoming network CDN calls, ACD calls, and NACD calls that were answered at this site.	ULONG
Calls Waiting[a]	NIrtd_NODAL_CALL_WAIT	State	The number of local CDN calls and incoming network CDN calls that are currently waiting to be answered.	ULONG
Network Calls Offered[b]	Nlrtd_NODAL_NETWRK_CALL_OFFER	Cumulative	**Networking feature** The number of incoming network CDN calls that were offered to this site.	ULONG
Network Calls Answered[b]	Nlrtd_NODAL_NETWRK_CALL_ANS	State	**Networking feature** The number of incoming network CDN calls that were answered at this site.	ULONG
Network Calls Waiting[b]	Nlrtd_NODAL_NETWRK_CALL_WAIT	State	**Networking feature** The number of incoming network CDN calls that are currently waiting to be answered.	ULONG

a. This statistic includes calls that originally entered the Contact Center Manager Server at this site and calls that were received at this site from the Contact Center network.
b. This statistic only includes calls that were received at this site from the Contact Center network.

IVR statistics

IVR statistics provide state and cumulative performance measurement information on a per-IVR queue basis. These statistics provide a means to monitor the usage of the port resources of an IVR queue from a real-time perspective.

Column	Column ID	Data type	Description	Format
IVR Queue ID	NIrtd_IVR_QUEUE_ID	Admin	A unique number to identify an IVR queue.	BYTE (8) STRING
Calls Waiting	NIrtd_IVR_CALL_WAIT	State	The number of local and incoming network CDN calls that are currently waiting at this IVR queue.	ULONG

Calls Answered	NIrtd_IVR_CALL_ANS	Cumulative	The number of local and incoming network CDN calls that were answered by this IVR queue.	ULONG
Calls Answered Delay	NIrtd_IVR_CALL_ANS_ DELAY	Cumulative	The total delay experienced by all local and incoming network CDN calls that were answered by this IVR queue. The delay begins when a call is queued against this IVR queue.	ULONG
Calls Answered After Threshold	NIrtd_IVR_CALL_ANS_ AFT_THRESHOLD	Cumulative	The number of local and incoming network CDN calls answered by this IVR queue that experienced a delay greater than or equal to the service level threshold for this IVR queue. The delay begins when a call is queued against this IVR queue.	ULONG
Calls Not Treated	NIrtd_IVR_CALL_NOT_ TREATED	Cumulative	The number of local and incoming network CDN calls that were abandoned or pulled back while waiting in this IVR queue.	ULONG
Calls Not Treated Delay	NIrtd_IVR_CALL_NOT_ TREATED_DELAY	Cumulative	The total delay experienced by all local and incoming network CDN calls that were abandoned or pulled back from this IVR queue. The delay begins when a call is queued against this IVR queue.	ULONG
Calls Not Treated After Threshold	NIrtd_IVR_CALL_NOT_ TREATED_AFT_THRE SHOLD	Cumulative	The number of local and incoming network CDN calls abandoned or pulled back while waiting in this IVR queue that experienced a delay greater than or equal to the service level threshold for this IVR queue. The delay begins when a call is queued against this IVR queue.	ULONG

Route statistics

Route statistics provide instantaneous and cumulative All Trunks Busy (ATB) information on a per-route basis.
Note: Route statistics are available for the CS1000 only.

Column	Column ID	Data type	Description	Format
Route Number	NIrtd_ROUTE_ROUTE_NO	Admin	A unique number to identify a route.	ULONG
All Trunks Busy	NIrtd_ROUTE_ATB_FLAG	State	Indicates whether all trunks in this route are currently busy.	BYTE(8) STRING
All Trunks Busy	NIrtd_ROUTE_ATB_TIME	Cumulative	The total time this route has been in the All Trunks Busy state.	ULONG

Chapter 5: Real-time API definition

Type definitions

RTDAPI includes internationalized (MBCS) support.

Structure	Arguments	Description
_NIrtd_enumType	NIrtd_eNumber NIrtd_eString NIrtd_eBuffer	St_value type: can be a number, string, or buffer
_NIrtd_stValue	NIrtd_enumType type ULONGnumber TCHARstring	Used to hold the contents of the columns in the statistic stream (string field size of NIrtd_cMaxString)
_NIrtd_stName	TCHAR*first_name TCHAR*last_name	Used to hold the first and last agent names (string field size of NIrtd_MaxName)
_NIrtd_stTable	ULONGnumberofrows ULONGnumberofcols NIrtd_tTabletable	Table of statistics
_NIrtd_stTableGroup	NIrtd_stTabledeletedValues NIrtd_stTablenewValues NIrtd_stTabledeltaValues	Group of statistic tables covering one data propagation
_NIrtd_cntType	QueryCnt /*# of queries */ ConjCnt /*# of conjunctions */ TGCnt /*# of Table Groups */ RowCnt /* # of Rows */ ValueCnt /* # of Values */ CacheCnt /* # of Name Caches */ NameCnt /* # of Names */	Counter types

Data element storage functions

Value functions

NIrtd_allocateValue()

This function allocates string space within a value structure.

```
NIrtd_allocateValue(NIrtd_stValue*value );
```

Parameter	Description
*value	A pointer to an NIrtd_stValue structure. The string pointer within the structure is allocated by this function call.

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.
NIrtd_eALLOC_FAILED	60052	The allocation of memory was not successful.

NIrtd_freeValue()

This function frees string space within a value structure.

```
VOID NIrtd_freeValue(NIrtd_stValue *value );
```

Parameter	Description
*value	A pointer to an NIrtd_stValue structure that has been fully allocated by a previous call to NIrtd_allocateValue. The string pointer within the structure is freed by this function call.

NIrtd_cpValue()

This function copies a value structure. The source and destination structures must already be fully allocated by previous calls to NIrtd_allocateValue.

```
ULONG NIrtd_cpValue(NIrtd_stValue *destvalue, NIrtd_stValue *srcvalue );
```

Parameter	Description
*destvalue	A pointer to the destination value structure.
*srcvalue	A pointer to the source value structure.

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.
NIrtd_eVALUE_INIT	60049	Either the source or destination structure is a null pointer or has a null string pointer.

Name functions

NIrtd_allocateName()

This function allocates string space within a name structure.

```
ULONG NIrtd_allocateName(NIrtd_stName *name );
```

Parameter	Description
*name	A pointer to an NIrtd_stName structure. The first_name and last_name string pointers within the structure are allocated by this function call.

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.
NIrtd_eALLOC_FAILED	60052	The allocation of memory was not successful.

NIrtd_freeName()

This function frees string space within a name structure.

```
VOID NIrtd_freeName(NIrtd_stName*name );
```

Parameter	Description
*name	A pointer to an NIrtd_stName structure that has been fully allocated by a previous call to NIrtd_allocateName. The first_name and last_name string pointers within the structure are freed by this function call.

NIrtd_cpName()

This function copies a name structure. (The source and destination structures must already be fully allocated by previous calls to NIrtd_allocateName.)

```
ULONG NIrtd_cpName(NIrtd_stName *destname, NIrtd_stName *srcname );
```

Parameter	Description
*destname	A pointer to the destination name structure.
*srcname	A pointer to the source name structure.

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.
NIrtd_eNAME_INIT	60049	Either the source or destination structure is a null pointer or has a null first_name or last_name string pointer.

Query description functions

NIrtd_allocateQuery()

This function allocates a query structure and associates the query with the specified table.

```
ULONG NIrtd_allocateQuery(NIrtd_tQuery *query, NIrtd_tTableId table);
```

Parameter	Description	
*query	A pointer to an NIrtd_tQuery structure. NIrtd_tQuery is a private data structure. The space required by the query structure is allocated by this function call.	
table	The ID of the table to query.	
Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.
NIrtd_eQUERY_INITPARM	60012	Query parameter was not null on initialization.
NIrtd_eQUERY_INIT	60014	Failed to initialize query.
NIrtd_eTABLE	60003	Invalid table ID passed.

NIrtd_selectColumn()

This function associates a column ID with the query structure. Multiple columns can be associated with a query. However, the first selected column must be a table key.

```
ULONG NIrtd_selectColumn (NIrtd_tQuery *query, NIrtd_tColumnId column);
```

Parameter	Description
*query	A pointer to an NIrtd_tQuery structure. NIrtd_tQuery is a private data structure. The space required by the query structure should already be allocated by a previous call to NIrtd_allocateQuery.
column	The ID of a column to be retrieved by the query.

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.
NIrtd_eQUERY_NOINIT	60013	The query structure is null or not initialized properly. The space required by the query structure should already be allocated by a previous call to NIrtd_allocateQuery.
NIrtd_eTABLE	60003	The table associated with the query structure is invalid.
NIrtd_eCOLUMN	60004	Invalid column ID selected. (Either not valid for the table associated with the query or simply invalid.)
NIrtd_eKEY	60028	The first selected column must be a table key.

NIrtd_allocateConjunction()

This function allocates a conjunction structure, This provides a where clause for a query.

```
ULONG NIrtd_allocateConjunction (NIrtd_tConjunction *conj);
```

Parameter	Description

*conj	A pointer to an NIrtd_tConjunction structure. NIrtd_tConjunction is a private data structure. The space required by the conjunction structure is allocated by this function call.

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.
NIrtd_eCONJ_INIT	60015	Failed to initialize the conjunction or the passed pointer was invalid.

NIrtd_addCondition()

This function adds a condition to a conjunction.

```
ULONG NIrtd_addCondition(NIrtd_tConjunction *conj, NIrtd_tColumnId column,
NIrtd_tOperator oper, NIrtd_stValue *value);
```

Parameter	Description
*conj	A pointer to an NIrtd_tConjunction structure to which the condition is added. NIrtd_tConjunction is a private data structure. The space required by the conjunction structure should have already been allocated by a previous call to NIrtd_allocateConjunction.
column	The ID of the table column.
oper	The operator. The only operatror currently available is NIrtd_EQ.
***value**	A pointer to the operand value structure.

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.
NIrtd_eCOLUMN	60004	Invalid column ID selected.
NIrtd_eCONJ_NOINIT	60016	The conjunction structure is null or not initialized properly. The space required by the conjunction structure should already be allocated by a previous call to NIrtd_allocateConjunction.
NIrtd_eKEY	60028	A condition can only be applied to a key column. The passed column was not a key column.
NIrtd_eKEY_MISMATCH	60029	The table referenced by this column/condition does not match the table referenced by a previous condition added to this conjunction.
NIrtd_eDATA_INVALID	60023	An internal check of the NIrtd_stValue parameter indicates that the data type is invalid.
NIrtd_eVALUE_INIT	60049	The value parameter has not been initialized properly.

Conditions can be added together in a list to form a conjunction. Note that all of the conditions in a conjunction are joined by a logical and operation. Currently, only the equals operator is

supported in a condition. (A condition takes the form column equals value.)
The following code fragment creates a query that contains one conjunction. The conjunction contains one condition, which selects skillset statistics for the Sales skillset. Note that SALES contains the skillset ID for the Sales skillset.

```
NIrtd_tQuery query = NIrtd_NullQuery;
NIrtd_tConjunction conj = NIrtd_NullConjunction;
lRc = NIrtd_allocateQuery(&query, NIrtd_INTRVL_SKLST);
if (NIrtd_eOK != lRc)
goto quit;
lRc = NIrtd_selectColumn(&query,NIrtd_SKLST_SKILLSET_ID);
if (NIrtd_eOK != lRc)
goto quit;
lRc = NIrtd_selectColumn(&query,NIrtd_SKLST_CALL_WAIT);
if (NIrtd_eOK != lRc)
goto quit;
lRc = NIrtd_allocateConjunction(&conj);
if (NIrtd_eOK != lRc)
goto quit;
NIrtd_stValue value;
lRc = NIrtd_allocateValue(&value);
if (NIrtd_eOK != lRc)
goto quit;
value.type = NIrtd_eNumber;
value.number = SALES;
/* specify skillset with id = SALES */
lRc = NIrtd_addCondition(&conj,
NIrtd_SKLST_SKILLSET_ID,
NIrtd_EQ,
&value);
if (NIrtd_eOK != lRc)
goto quit;
/* add the conjunction to the query */
lRc = NIrtd_addConjunction(&query, &conj);
if (NIrtd_eOK != lRc)
goto quit;
```

The column selected in a condition must be a table key. Valid table keys are:

- NIrtd_APPL_APPL_ID for the Application table
- NIrtd_SKLST_SKILLSET_ID for the Skillset table
- NIrtd_AGENT_AGENT_ID for the Agent table
- NIrtd_NODAL_DUMMY_KEY for the Nodal table. There are no real keys for the Nodal table; rather, this key is used to make the interface consistent for the application when dealing with the application of new, deleted, and delta tabl updates.
- NIrtd_IVR_QUEUE_ID for the IVR table
- NIrtd_ROUTE_ROUTE_NO for the Route table column IDs

NIrtd_addConjunction()

This function adds a conjunction to a query.

```
ULONG NIrtd_addConjunction(NIrtd_tQuery *query, NIrtd_tConjunction *conj);
```

Parameter	Description
*query	A pointer to an NIrtd_tQuery structure. NIrtd_tQuery is a private data structure. The space required by the query structure

	should already be allocated by a previous call to NIrtd_allocateQuery.
*conj	A pointer to an NIrtd_tConjunction structure to add to the query. NIrtd_tConjunction is a private data structure. The space required by the conjunction structure should have already been allocated by a previous call to NIrtd_allocateConjunction.

Return code	**Error no.**	**Description**
NIrtd_eOK	0	Operation successful.
NIrtd_eCOLUMN	60004	Invalid column ID selected.
NIrtd_eQUERY_NOINIT	60013	The query structure is null or not initialized properly. The space required by the query structure should already be allocated by a previous call to NIrtd_allocateQuery.
NIrtd_eCONJ_NOINIT	60016	Conjunction found to be null. The space required by the conjunction structure should already be allocated and initialized in a previous call to NIrtd_allocateConjunction.
NIrtd_eTABLE	60003	The table associated with the query structure is invalid.
NIrtd_eKEY_MISMATCH	60029	The table reference associated with the query does not match the table referenced by the conjunction.

All of the conjunctions in a query are joined by a logical *or* operation.
The following code fragment creates a query that contains one conjunction. The conjunction contains one condition, which selects skillset statistics for the Sales skillset.

```
NIrtd_tQuery query = NIrtd_NullQuery;
NIrtd_tConjunction conj = NIrtd_NullConjunction;
lRc = NIrtd_allocateQuery(&query, NIrtd_INTRVL_SKLST);
if (NIrtd_eOK != lRc)
goto quit;
lRc = NIrtd_selectColumn(&query,NIrtd_SKLST_SKILLSET_ID);
if (NIrtd_eOK != lRc)
goto quit;
lRc = NIrtd_selectColumn(&query,NIrtd_SKLST_CALL_WAIT);
if (NIrtd_eOK != lRc)
goto quit;
lRc = NIrtd_allocateConjunction(&conj);
NIrtd_eOK != lRc;
if (NIrtd_eOK != lRc)
goto quit;
NIrtd_stValue value;
lRc = NIrtd_allocateValue(&value);
if (NIrtd_eOK != lRc)
goto quit;
value.type = NIrtd_eNumber;
value.number = SALES;
/* specify skillset with id = SALES */
lRc = NIrtd_addCondition(&conj,
NIrtd_SKLST_SKILLSET_ID,
NIrtd_EQ,
&value);
if (NIrtd_eOK != lRc)
goto quit;
```

```
/* add the conjunction to the query */
lRc = NIrtd_addConjunction(&query, &conj);
if (NIrtd_eOK != lRc)
goto quit;
```

The following code fragment creates a query that contains two conjunctions. Each conjunction contains one condition, which selects a particular skillset based on skillset ID.

```
NIrtd_tQuery query = NIrtd_NullQuery;
NIrtd_tConjunction conj1 = NIrtd_NullConjunction;
NIrtd_tConjunction conj2 = NIrtd_NullConjunction;
NIrtd_stValue value1;
NIrtd_stValue value2;
/* You can with initialize the value's unioned number/string
pointer or ensure that NIrtd_allocateValue is call in all cases
where even if it fails, the pointer will be set to null */
lRc = NIrtd_allocateValue(&value1);
careabit = NIrtd_allocateValue(&value2);
if (NIrtd_eOK != lRc)
goto quit;
lRc = careabit;
if (NIrtd_eOK != lRc)
goto quit;
lRc = NIrtd_allocateQuery(&query, NIrtd_INTRVL_SKLST);
if (NIrtd_eOK != lRc)
goto quit;
lRc = NIrtd_selectColumn(&query,NIrtd_SKLST_SKILLSET_ID);
if (NIrtd_eOK != lRc)
goto quit;
lRc = NIrtd_selectColumn(&query,NIrtd_SKLST_CALL_WAIT);
if (NIrtd_eOK != lRc)
goto quit;
lRc = NIrtd_allocateConjunction(&conj1);
if (NIrtd_eOK != lRc)
goto quit;
lRc = NIrtd_allocateConjunction(&conj2);
if (NIrtd_eOK != lRc)
goto quit;
value1.type = NIrtd_eNumber;
value1.number = SALES;
value2.type = NIrtd_eNumber;
value2.number = SUPPORT;
/* specify skillset with id = SALES */
lRc = NIrtd_addCondition(&conj1,
NIrtd_SKLST_SKILLSET_ID,
NIrtd_EQ,
&value1);
if (NIrtd_eOK != lRc)
goto quit;
/* specify skillset with id = SUPPORT */
lRc = NIrtd_addCondition(&conj2,
NIrtd_SKLST_SKILLSET_ID,
NIrtd_EQ,
&value2);
if (NIrtd_eOK != lRc)
goto quit;
/* add the conjunction to the query */
lRc = NIrtd_addConjunction(&query, &conj1);
if (NIrtd_eOK != lRc)
goto quit;
/* add the conjunction to the query */
lRc = NIrtd_addConjunction(&query, &conj2);
```

```
        if (NIrtd_eOK != lRc)
        goto quit;
```

NIrtd_getValue()

This function takes string name values (agent names, application names, and skillset names) and looks up the corresponding ID values (agent telset login IDs, application IDs, and skillset IDs) so that they can then be passed to the NIrtd_addCondition function.

```
ULONG NIrtd_getValue(NIrtd_tAPIauth *authorization, NIrtd_stName *name, NIrtd_tColumnId
column, NIrtd_stValue *value);
```

Parameter	Description
*authorization	A pointer to an NIrtd_tAPIauth structure. NIrtd_tAPIauth is a semi-private data structure holding authorization information. The authorization structure is obtained in the call to NIrtd_login.
*name	A pointer to a structure that contains the first_name and last_name of the value requested. For applications and skillsets, the first_name should be set to a null string.
column	The ID of the table column.
*value	A pointer to a structure to contain the string or numeric value required.

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.
NIrtd_eINVALID_AUTH	60019	Failed to validate preauthorization. Ensure NIrtd_login has been called.
NIrtd_eCOLUMN	60004	Invalid column ID selected.
NIrtd_eVALUE_INIT	60049	The passed value parameter was null.
NIrtd_eTABLE	60036	The passed name parameter was null.
NIrtd_eCOL_NOT_FOUND	60050	The passed column ID value was valid but not stored in the name cache.

NIrtd_freeConjunction()

This function frees memory associated with a conjunction.

```
ULONG NIrtd_freeConjunction (NIrtd_tConjunction *conj);
```

Parameter	Description
*conj	A pointer to a conjunction structure that has been fully allocated by a previous call to NIrtd_allocateConjunction. The conjunction structure is freed by this function call.

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.

NIrtd_freeQuery()

This function frees memory associated with a query.

```
ULONG NIrtd_freeQuery (NIrtd_tQuery *query);
```

Parameter	Description
*query	A pointer to a query structure that has been fully allocated by a previous call to NIrtd_allocateQuery. The query structure is freed by this function call.

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.

Data access functions

NIrtd_allocateRow()

This function allocates a row structure and then retrieves the column data from row <index> of table <table> and copies the data into the allocated row.

```
ULONG NIrtd_allocateRow( NIrtd_stTable *table, NIrtd_tRow *row, ULONG index);
```

Parameter	Description
*row	A pointer to an NIrtd_tRow structure. NIrtd_tRow is a private data structure. The space required by the row structure is allocated by this function call.
*table	A pointer to an NIrtd_stTable structure. NIrtd_stTable is a semi-private data structure holding statistical data information returned from the server.
index	The row in the table to be copied into the allocated row structure.

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.
NIrtd_eROW_INIT	60002	Allocation and initialization of row failed.
NIrtd_eROW_INVALID	60021	The row indicated does not exist.
NIrtd_eTABLE	60003	The passed table was found to be null.

NIrtd_getCol()

This function retrieves one column of data from the passed row structure and returns it in the value structure.

```
ULONG NIrtd_getCol (NIrtd_stValue *value, ULONG index);
```

Parameter	Description
*value	A pointer to an NIrtd_tValue structure in which to return the retrieved value. The space required by the value structure should already be allocated by a previous call to NIrtd_allocateValue.
*row	A pointer to an NIrtd_tRow structure from which to retrieve data. The space required by the NIrtd_tRow structure should already be allocated by a previous call to NIrtd_allocateRow.
index	The column in the row to retrieve. (Range = 0 .. number of columns selected)

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.
NIrtd_eCOL_INVALID	60022	The column selected does not exist.
NIrtd_eDATA_INVALID	60023	The data, from the row at the index indicated, is an invalid value type.
NIrtd_eVALUE_INIT	60049	The passed value parameter is invalid. The space required by the value structure should already be allocated by a previous call to NIrtd_allocateValue.
NIrtd_eROW_INIT	60002	The passed row is invalid. The space required by the NIrtd_tRow structure should already be allocated by a previous call to NIrtd_allocateRow.

NIrtd_freeRow()

This function frees the memory associated with a row structure.

```
ULONG NIrtd_freeRow (NIrtd_tRow *row);
```

Parameter	Description
*row	A pointer to a row structure that has been fully allocated by a previous call to NIrtd_allocateRow. The row structure is freed by this function call.

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.

NIrtd_freeTableGroup()

This function frees the storage for the group of table values returned by NIrtd_singleDataRequest() or NIrtd_StartDataStream(). Data received from the server is stored in a table group by the API.

Each time data is received, the API allocates memory for a new table group. The application must free the table group memory.

```
ULONG NIrtd_freeTableGroup (NIrtd_stTableGroup *table);
```

Parameter	Description
*table	A pointer to an NIrtd_stTableGroup structure. NIrtd_stTableGroup is a semi-private data structure holding statistical data information returned from the server. The subtables and overall structure will be deallocated by this function call.

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.

Data request functions

NIrtd_login()

This function is used to log on to a server and to obtain authorization from the AACC/ACCS security server. The NIrtd_login user is set up using Contact Center Server Utility. The newly created user must have Real-time Display privileges to receive the RTD data. The new user's details are used for the registration process when logging on.

```
ULONG NIrtd_login(NIrtd_tAPIauth *authorization, TCHAR *userID, TCHAR *passWord);
```

Parameter	Description
*authorization	A pointer to an NIrtd_tAPIauth structure. NIrtd_tAPIauth is a semi-private data structure holding authorization information. Memory for the private structure is allocated at logon and deallocated at logoff.
*servname	A string containing the IP address of the server.
*userId	A user ID set up on the server to receive NIrtd requests.
*password	The password for the user ID.

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.
NIrtd_eSERVER	60001	Invalid server name.
NIrtd_eUSERID	60006	Invalid user ID.
NIrtd_ePASSWORD	60007	Invalid password.
NIrtd_eUSERS	60008	Too many users logged on.
NIrtd_eAUTH_INIT	60017	Failed to initialize authorization structure.
NIrtd_eINVALID_AUTH	60019	The authorization parameter was found to be null.
NIrtd_eLISTENER_INIT	60026	Failed to initialize a listener for data propagation.
NIrtd_eLOGIN_FAIL	60009	Communication failure during logon attempt.

NIrtd_eSERV_INIT	60018	Unable to create the indicated server entry. Either a memory limit was reached or a limit on the number of server connections was reached.
NIrtd_eLOGIN	60031	PC user logon notification.
NIrtd_eLOGIN_ERR	60032	PC user logon error.
NIrtd_eLOGIN_NO	60033	No PC user logged on yet.
NIrtd_eLOGIN_ALREADY	60034	PC user logged on already.
NIrtd_eDIDNOTBUY	60042	Failed to log on to the server because the real-time access feature was not purchased.
NIrtd_eREMOTE_SYSREC_FAIL	60043	Failed to log on due to a failure to read the system record at the remote site.

NIrtd_singleDataRequest()

This function obtains real-time data from the server and puts it into a table group structure. Before calling NIrtd_singleDataRequest, the application must log on to a server using NIrtd_login().

```
ULONG NIrtd_singleDataRequest( NIrtd_stTableGroup **tableGroup, NIrtd_tAPIauth
*authorization, NIrtd_tQuery *query);
```

Parameter	Description
**tableGroup	A pointer to an NIrtd_stTableGroup structure. NIrtd_stTableGroup is a semi-private data structure holding statistical data information returned from the server.
*authorization	A pointer to an NIrtd_tAPIauth structure. NIrtd_tAPIauth is a semi-private data structure holding authorization information. The authorization structure is obtained in the call to NIrtd_login.
*query	A pointer to an NIrtd_tQuery structure. NIrtd_tQuery is a private data structure. The space required by the query structure should already be allocated and initialized by previous API function calls.

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.
NIrtd_eINVALID_AUTH	60019	Failed to validate preauthorization. Ensure NIrtd_login has been called.
NIrtd_eREG_INIT	60020	Failed to allocate registration.
NIrtd_eNOT_AVAIL	60005	The query asks for a statistic that is not currently being collected by the server.
NIrtd_eQUERY_NOINIT	60013	The query parameter was found to be invalid.
NIrtd_eINVALID_TABLE	60025	The passed table group variable was null.
NIrtd_eALLOC_FAILED	60052	Storage for an internal component failed..

NIrtd_eCOMM	60055	Communication with the server failed.
NIrtd_eINVALID_REG	60044	The single request registration ID is invalid with the server.

NIrtd_startDataStream()

This function is used to request a stream of regular data updates from a server.

```
ULONG NIrtd_startDataStream( NIrtd_tAPIauth *authorization, NIrtd_tQuery *query, ULONG
updateRate, NIrtd_funCallback callback, NIrtd_tRequestId *requestId );
```

Parameter	Description
*authorization	A pointer to an NIrtd_tAPIauth structure. NIrtd_tAPIauth is a semi-private data structure holding authorization information. The authorization structure is obtained in the call to NIrtd_login.
*query	A pointer to an NIrtd_tQuery structure. NIrtd_tQuery is a private data structure. The space required by the query structure should already be allocated and initialized by previous API function calls.
updateRate	The minimum frequency (in milliseconds) with which data is updated.
callback	The function to be executed when data is retrieved. (*NIrtd_funCallback) (ULONG return_code, NIrtd_tRequestIdrequestid, NIrtd_stTableGroup *tableGroup, void * yourpointer)
*yourpointer	An application pointer that is passed back to the application when the callback function is called.
*requestId	A value returned by the API so that this request can be canceled by NIrtd_stopDataStream().

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.
NIrtd_eINVALID_AUTH	60019	Failed to validate preauthorization. Ensure NIrtd_login has been called.
NIrtd_eREG_INIT	60020	Failed to allocate registration.
NIrtd_eUPDATE	60011	Invalid update rate.
NIrtd_eNOT_AVAIL	60005	The query asks for a statistic that is not currently being collectedby the server.
NIrtd_eNOT_FOUND	60030	The query was not successfully registered with the data collection part of the server. See the server event logs for further details.
NIrtd_eQUERY_NOINIT	60013	The query parameter was found to be invalid.
NIrtd_eINVALID_REG	60044	The requestId parameter was found to be null.
NIrtd_eLIMIT_REACHED	60051	The update rate indicated is past the limit defined for this interface.
NIrtd_eNULL_CALLBACK	60053	The callback function parameter was found to be null.

Return code	Error no.	Description
NIrtd_eALLOC_FAILED	60052	Storage for an internal component failed.
NIrtd_eCOMM	60055	Communication with the server failed.

The server is informed that the requested data should be provided at regular time intervals as specified by the updateRate (in milliseconds). The minimum update rate is 2000 milliseconds. The exception to this is a request for Agent statistics. The update rate is a minimum of 1000 ms. The server does not send updates more frequently than updateRate; however, updates can take longer than the specified rate depending on the load on the server.

The requestId returned by this function is required by NIrtd_stopDataStream() to cancel this request.

The callback function is invoked whenever new data arrives from the server.

NIrdt_stopDataStream()

This function is used to cancel the flow of data updates previously initiated by NIrtd_startDataStream().

```
ULONG NIrtd_stopDataStream( NIrtd_tAPIauth *authorization,  NIrtd_tRequestId requestId);
```

Parameter	Description
*authorization	A pointer to an NIrtd_tAPIauth structure. NIrtd_tAPIauth is a semi-private data structure holding authorization information. The authorization structure is obtained in the call to NIrtd_login.
*requestId	The requestId that was obtained from NIrtd_startDataStream().

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.
NIrtd_eINVALID_AUTH	60019	Failed to validate preauthorization. Ensure NIrtd_login has been called.
NIrtd_eINVALID_REG	60044	The requestId parameter was found to be null.
NIrtd_eALLOC_FAILED	60052	Storage for an internal component failed.
NIrtd_eCOMM	60055	Communication with the server failed.

NIrtd_logout()

This function is used to log off from the server.

```
ULONG NIrtd_logout(NIrtd_tAPIauth *authorization)
```

Parameter	Description
*authorization	A pointer to an NIrtd_tAPIauth structure. NIrtd_tAPIauth is a semi-private data structure holding authorization information.

	The authorization structure is obtained in the call to NIrtd_login.

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.
NIrtd_eINVALID_AUTH	60019	Failed to validate preauthorization. Ensure NIrtd_login has been called.
NIrtd_eMUST_DEREG_FIRST	60024	Attempt to log off before deregistration failed.
NIrtd_eLOGOUT_FAIL	60010	Failed to log off from the security server.

Preprocessing and postprocessing functions

NIrtd_getNameCacheforDataColumn()

This function obtains the current list of names and IDs for the given column and places them into a cache structure for quick access using the NIrtd_getName function.

```
ULONG NIrtd_getNameCacheforDataColumn(NIrtd_tAPIauth* authorization, NIrtd_tColumnId
column );
```

Parameter	Description
*authorization	A pointer to an NIrtd_tAPIauth structure. NIrtd_tAPIauth is a semi-private data structure holding authorization information. The authorization structure is obtained in the call to NIrtd_login.
column	The ID of the table column.

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.
NIrtd_eINVALID_AUTH	60019	Failed to validate preauthorization. Ensure NIrtd_login has been called.
NIrtd_eCOLUMN	60004	Invalid column ID selected.
NIrtd_eALLOC_FAILED	60052	An internal memory allocation failed.
NIrtd_eCOLUMN_INIT	60035	Allocation failed for a column database object.
NIrtd_eNAME_INIT	60036	Allocation failure for a name database object or name list structure.
NIrtd_eAGENTLIST_GET	60039	Failed to obtain the list of agent names from the server.
NIrtd_eSSLIST_GET	60037	Failed to obtain the list of skillset names from the server.
NIrtd_eAPPLIST_GET	60038	Failed to obtain the list of application names from the server.

NIrtd_getName()

This function translates a column ID and value into a name. The name structure is allocated by the calling application and is updated to contain the first and last name of the value. Where only one name is appropriate (such as application and skillset names), the last name contains the name and the first name is set to a null string.

```
ULONG NIrtd_getName(NIrtd_tAPIauth *authorization, NIrtd_tColumnId column, NIrtd_stValue
*value, NIrtd_stName *name);
```

Parameter	Description
*authorization	A pointer to an NIrtd_tAPIauth structure. NIrtd_tAPIauth is a semi-private data structure holding authorization information. The authorization structure is obtained in the call to NIrtd_login.
column	The ID of the table column.
*value	A pointer to an NIrtd_tValue structure that contains the value to retrieve the name for. (For example, the value of the column as retrieved from NIrtd_getCol.)
*name	A pointer to a structure to contain the name retrieved.

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.
NIrtd_eINVALID_AUTH	60019	Failed to validate preauthorization. Ensure NIrtd_login has been called.
NIrtd_eNOT_FOUND	60030	The indicated column ID value was not found in the cache.
NIrtd_eCOLUMN	60004	Invalid column ID selection passed.
NIrtd_eCOL_NOT_FOUND	60035	The indicated column was not found to be loaded in the cache.
NIrtd_eCOLUMN_INIT	60035	The indicated column has not been loaded into cache.
NIrtd_eVALUE_INIT	60049	The passed value parameter was found to be null.
NIrtd_eNAME_INIT	60036	The passed name parameter was found to be null.
NIrtd_eDATA_INVALID	60023	The type of the passed value parameter was found to be invalid.

NIrtd_getFailedName()

When getting the name from the name cache has failed in the call to NIrtd_getName(), this function translates a column ID and value into a name. The name structure is allocated by the calling application and is updated to contain the first and last name of the value. Where only one name is appropriate (such as application and skillset names), the last name contains the name and the first name is set to a null string.

```
ULONG NIrtd_getFailedName(NIrtd_tAPIauth *authorization, NIrtd_tColumnId column,
NIrtd_stValue *value, NIrtd_stName *name);
```

Parameter	Description
*authorization	A pointer to an NIrtd_tAPIauth structure.

	NIrtd_tAPIauth is a semi-private data structure holding authorization information. The authorization structure is obtained in the call to NIrtd_login.
column	The ID of the table column.
*value	A pointer to an NIrtd_tValue structure that contains the value to retrieve the name for. (For example, the value of the column as retrieved from NIrtd_getCol.)
*name	A pointer to a structure to contain the name retrieved.

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.
NIrtd_eINVALID_AUTH	60019	Failed to validate preauthorization. Ensure NIrtd_login has been called.
NIrtd_eNAME_INIT	60036	Allocation failure for individual name or list structure; or the = passed name parameter was found to be null.
NIrtd_eAGENTLIST_GET	60039	Failed to obtain the list of agent names from the server.
NIrtd_eSSLIST_GET	60037	Failed to obtain the list of skillset names from the server.
NIrtd_eAPPLIST_GET	60038	Failed to obtain the list of application names from the server.
NIrtd_eCOLUMN	60004	Invalid column ID selection passed.
NIrtd_eVALUE_INIT	60049	The passed value parameter was found to be null.
NIrtd_eDATA_INVALID	60023	The type of the passed value parameter was found to be invalid.
NIrtd_eID_NAME_MISMATCH	60041	Value content mismatch as compared to the indicated column.
NIrtd_eNOT_FOUND	60030	The indicated column ID value was not found in the cache.
NIrtd_eCOL_NOT_FOUND	60050	The indicated column was not found to be loaded in the cache.

NIrtd_refreshNameCache()

This function refreshes the name cache. An image of the current name values is retrieved from the server and is available for all future calls to NIrtd_getName(). This routine is useful when a number of name changes occurred on the server but are not reflected in the current cache of names on the API client. An application can make the call to NIrtd_refreshNameCache and thenusing a background thread, go through all the IDs and names currently displayed and update them if necessary.

```
ULONG NIrtd_refreshNameCache(NIrtd_tAPIauth *authorization );
```

Parameter	Description
*authorization	A pointer to an NIrtd_tAPIauth structure. NIrtd_tAPIauth is a semi-private data structure holding authorization information. The authorization structure is obtained in the call to NIrtd_login.

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.
NIrtd_eINVALID_AUTH	60019	Failed to validate preauthorization. Ensure NIrtd_login has been called.
NIrtd_eNAME_INIT	60036	Allocation failure for individual name or list structure; or the = passed name parameter was found to be null.
NIrtd_eAGENTLIST_GET	60039	Failed to obtain the list of agent names from the server.
NIrtd_eSSLIST_GET	60037	Failed to obtain the list of skillset names from the server.
NIrtd_eAPPLIST_GET	60038	Failed to obtain the list of application names from the server.
NIrtd_eCOLUMN	60004	Invalid column ID selection passed.
NIrtd_eVALUE_INIT	60049	The passed value parameter was found to be null.
NIrtd_eDATA_INVALID	60023	The type of the passed value parameter was found to be invalid.

NIrtd_removeNameCacheforDataColumn()

This function removes a name cache from memory that is no longer going to be used. When NIrtd_logout is called, it removes all name caches associated with the server being logged off. The NIrtd_removeNameCacheforDataColumn routine should be used when a cache is no longer needed but the application will remain logged on to the server.

```
ULONG NIrtd_removeNameCacheforDataColumn( NIrtd_tAPIauth *authorization, NIrtd_tColumnId
column);
```

Parameter	Description
*authorization	A pointer to an NIrtd_tAPIauth structure. NIrtd_tAPIauth is a semi-private data structure holding authorization information. The authorization structure is obtained in the call to NIrtd_login.
column	The ID of the table column.

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.
NIrtd_eINVALID_AUTH	60019	Failed to validate preauthorization. Ensure NIrtd_login has been called.
NIrtd_eCOLUMN	60004	Invalid column ID selected..
NIrtd_eCOL_NOT_FOUND	60050	The indicated column was not found to be loaded in the cache.
NIrtd_eCACHE_REMOVAL	60040	A failure occurred when trying to remove a column of data from the cache database.

NIrtd_interpAgentState()

This function breaks the multistate agent state value into multiple single state values. The AgtState.h file contains constants for comparison against multistate values and for single state values. Use this routine and the returned single state values for comparison with the AgtState.h single state constants.

```
void NIrtd_interpAgentState(ULONG returnedAgentState, ULONG *ngccCallState, ULONG
*dnOutCallState, ULONG *dnInCallState, ULONG *nacdCallState, ULONG *acdCallState, ULONG
*walkawayState );
```

Parameter	**Description**
returnedAgentState	The multi-state agent state value returned by the RTDAPI.

*ngccCallState	The Contact Center call state:
	eNGCC_ACTIVE = 0X00000100,
	eNGCC_ONHOLD= 0X00000080,
	eNGCC_NOTRDY= 0X00000040,
	eNGCC_BRK= 0X00000020,
	eNGCC_IDLE= 0X00000010,
	eNGCC_RESERVE= 0X00000008,
	eNGCC_CALL_PRESENT= 0X00000004,
	eNGCC_CONSULTATION= 0X00000002,
	eNGCC_EMERGENCY= 0X00000001

*dnOutCallState	The DN Out call state:
	eDN_OUT_ACTIVE= 0X00000400,
	eDN_OUT_ONHOLD= 0X00000200,
	eDN_OUT_ACTIVE_ONHOLD= 0X00000600

*dnInCallState	The DN In call state:
	eDN_IN_ACTIVE= 0X00001000,
	eDN_IN_ONHOLD= 0X00000800,

*nacdCallState	The NACD call state:
	eNACD_ACTIVE = 0X00004000,
	eNACD_ONHOLD = 0X00002000

*acdCallState	he ACD call state:
	eACD_ACTIVE = 0X00010000,
	eACD_ONHOLD = 0X00008000

*walkawayState	The Walkaway call state: eNGCC_WALKAWAY= 0x10000000

NIrtd_setRecovery()

This function changes the amount of time to wait before declaring communication failure. The full pull plug time is equal to the pullPlugTime plus the update rate. The wakeupGranularity parameter is the frequency at which the RTDAPI layer wakes up and checks the amount of time that has passed. The default is a pullPlugTime of 5 minutes with a wakeupGranularity of 1 minute.

```
ULONG NIrtd_setRecovery( ULONG pullPlugTime, ULONG wakeupGranularity);
```

Parameter	**Description**
ullPlugTime	This time plus the update rate to wait before declaring communication failure with the server and starting recovery actions.

wakeupGranularity	The amount of time to wait before waking up to check pull plug timers on the system.

Return code	Error no.	Description
NIrtd_eOK	0	Operation successful.
NIrtd_eSET_ONLY_ON_INIT	60045	The operation can only occur prior to calling NIrtd_login.

Debug functions

NIrtd_getFirstLowError()

This function retrieves and resets the first lower level return code. This should be called whenever an error code is returned by any API function. This internal lower level return code value is useful for Avaya Technology in problem resolution. The third-party application should make this code value available in some form. Note that this value is subject to being reset or updated in parallel when dealing with a multithreaded, third-party application that makes calls with multiple threads to the NIrtd API.

```
ULONG NIrtd_getFirstLowError ();
```

NIrtd_getCnt()

This function retrieves the number of objects allocated. This is primarily of use when trying to ensure the RTD client is properly deallocating objects previously allocated.

```
int NIrtd_getCnt (NIrtd_cntType i);
```

Parameter	Description
i	The type of counter to be retrieved

```
/* Counter types */
typedef enum _NIrtd_cntType
{
QueryCnt, /* Number of queries */
ConjCnt, /* Number of conjunctions */
TGCnt, /* Number of Table Groups */
RowCnt, /* Number of Rows */
ValueCnt, /* Number of Values */
CacheCnt, /* Number of Name Caches */
NameCnt /* Number of Names */
} NIrtd_cntType;
```

Error codes

The following table lists the error numbers and their corresponding events (for debugging purposes):

Error no.	Definition	**Description**
0	NIrtd_eOK	Operation successful.
60001	NIrtd_eSERVER	Invalid server ID passed.
60002	NIrtd_eROW_INIT	Allocation and initialization of row failed.
60003	NIrtd_eTABLE	Invalid table ID (query) passed
60004	NIrtd_eCOLUMN	Invalid column selection (query) passed
60005	NIrtd_eNOT_AVAIL	The query asked for a statistic that is not currently being collected by the server
60006	NIrtd_eUSERID	Invalid user ID passed
60007	NIrtd_ePASSWORD	Invalid password passed
60008	NIrtd_eUSERS	Too many users logged on to the server
60009	NIrtd_eLOGIN_FAIL	General logon failure
60010	NIrtd_eLOGOUT_FAIL	General logoff failure
60011	NIrtd_eUPDATE	Invalid update rate passed
60012	NIrtd_eQUERY_INITPARM	Query parm found not to be null on initialization
60013	NIrtd_eQUERY_NOINIT	Query found to be null. Should be initialized in call to NIrtd_allocateQuery
60014	NIrtd_eQUERY_INIT	Failed to initialize query
60015	NIrtd_eCONJ_INIT	Failed to initialize conjunction
60016	NIrtd_eCONJ_NOINIT	Conjunction found to be null. Should be initialized in call to NIrtd_allocateConjunction
60017	NIrtd_eAUTH_INIT	Failed to initialize authorization structure
60018	NIrtd_eSERV_INIT	Failed to initialize the server structure
60019	NIrtd_eINVALID_AUTH	Failed to validate preauthorization. Ensure NIrtd_Login has been called.
60020	NIrtd_eREG_INIT	Failed to allocate registration
60021	NIrtd_eROW_INVALID	The row indicated does not exist
60022	NIrtd_eCOL_INVALID	The column indicated does not exist
60023	NIrtd_eDATA_INVALID	The data returned from the remote server was invalid
60024	NIrtd_eMUST_DEREG_FIRST	Attempt to log off before deregistration failed
60025	NIrtd_eINVALID_TABLE	Attempt to free an unallocated group table
60026	NIrtd_eLISTENER_INIT	Failed to initialize a listener for data propagation
60027	NIrtd_eTABLE_GROUP_INIT	Failed to allocate and initialize the group table being returned
60028	NIrtd_eKEY	Passed value is not a key or is not a key for this table

Error no.	Definition	**Description**
60029	NIrtd_eKEY_MISMATCH	Keys in this conjunction are from different tables
60030	NIrtd_eNOT_FOUND	The requested name was not found
60031	NIrtd_eLOGIN	PC user logon notification
60032	NIrtd_eLOGIN_ERR	PC user logon error
60033	NIrtd_eLOGIN_NO	No PC user logon yet
60034	NIrtd_eLOGIN_ALREADY	PC user logged on already
60035	NIrtd_eCOLUMN_INIT	Need to initialize the name/column cache
60036	NIrtd_eNAME_INIT	Failed to allocate name
60037	NIrtd_eSSLIST_GET	Failed to get skillset name cache
60038	NIrtd_eAPPLIST_GET	Failed to get application name cache
60039	NIrtd_eAGENTLIST_GET	Failed to get agent name cache
60040	NIrtd_eCACHE_REMOVAL	Failed to remove a name a cache
60041	NIrtd_eID_NAME_MISMATCH	Value content mismatch as compared to the indicated column
60042	NIrtd_eDIDNOTBUY	Failed to log on to the server because the real-time access feature was not purchased
60043	NIrtd_eREMOTE_SYSREC_FAIL	Failed to log on due to a failure to read the system record at the remote site
60044	NIrtd_eINVALID_REG	Invalid registration identifier passed
60045	NIrtd_eSET_ONLY_ON_INIT	The setRecovery routine can only be called prior to calls for real-time data propagation
60046	NIrtd_eSTART_RECOVERY	A deregistration / reregistration attempt is being made
60047	NIrtd_eOK_RECOVERY	The deregistration / reregistration attempt was successful
60048	NIrtd_eBAD_RECOVERY	The deregistration / reregistration attempt was not successful and a retry has been scheduled
60049	NIrtd_eVALUE_INIT	The passed value is invalid
60050	NIrtd_eCOL_NOT_FOUND	Column not loaded into name cache
60051	NIrtd_eLIMIT_REACHED	A passed parameter exceeds the defined limit
60052	NIrtd_eALLOC_FAILED	A required memory allocation failed
60053	NIrtd_eNULL_CALLBACK	A null callback function was passed
60054	NIrtd_eWORKER_INIT	Failed to initialize a worker for registration recovery
60055	NIrtd_eCOMM	A communications failure occurred

Error no.	Definition	**Description**
60056	NIrtd_eRDC_FAILURE	Failure in DP communication with RDC or a failure of the defined query when being processed by RDC. See the server logs.
60057	NIrtd_eIVRLIST_GET	Failed to get IVR name cache
60058	NIrtd_eROUTELIST_GET	Failed to get Route name cache
60059	NIrtd_eDP_FAILURE	Failed in DP
60060	NIrtd_eLOGIN_NO_EULA	No End-User License Agreement (that is, no user selected the “yes” box after the agreement is displayed)
60061	NIrtd_eLOGIN_NO_SERVER_VE RSION	Failure to obtain the server version
60062	NIrtd_eLOGIN_FAILED_SERVER _VERSI ON	Wrong version number

Chapter 6: Sample Application

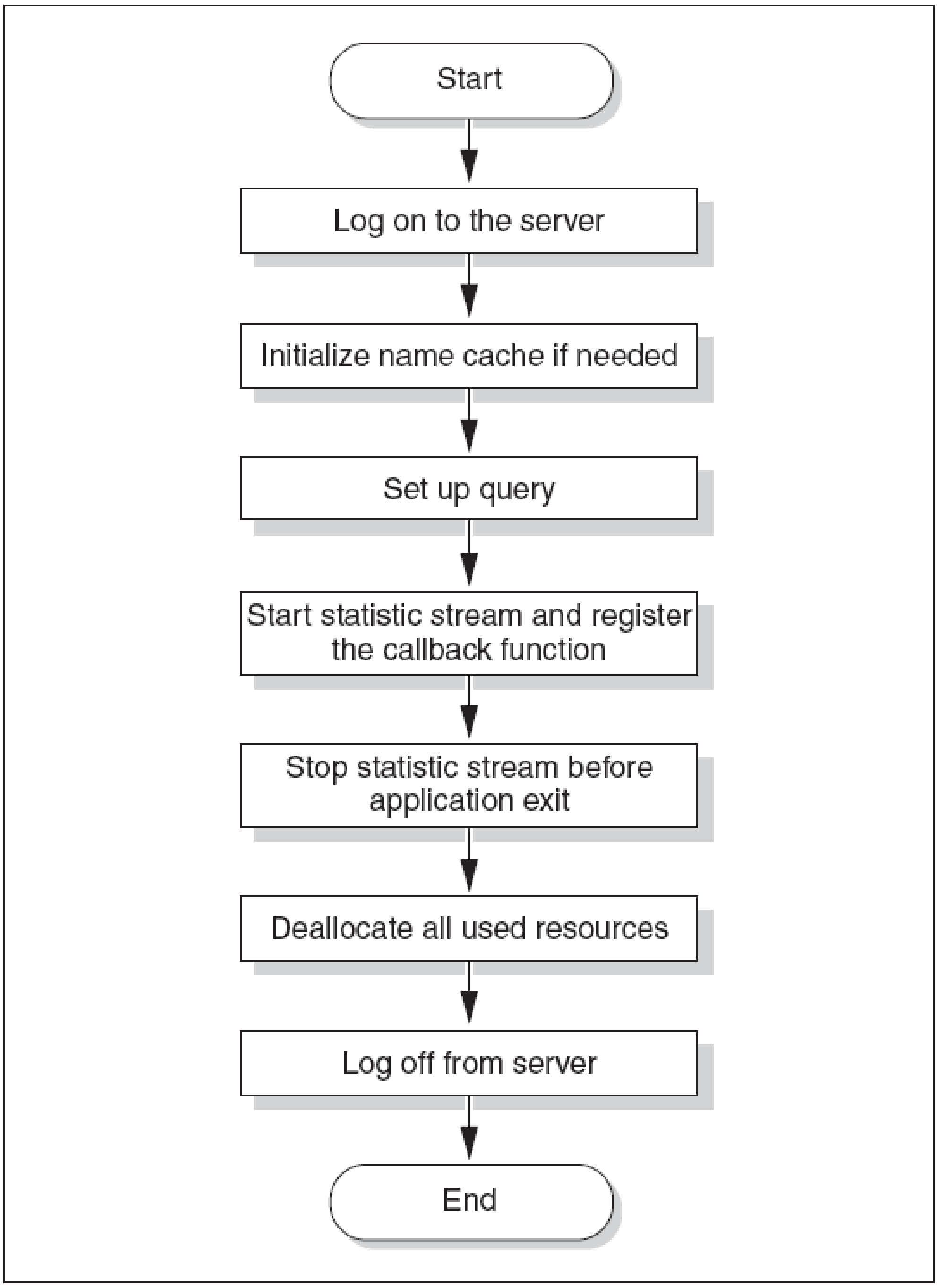
Introduction

This chapter goes through the basic skeleton of a RTDAPI application. It presents diagrams showing the primary process flows. The source code for the sample is available from the SDK.

Application design

Basic skeleton

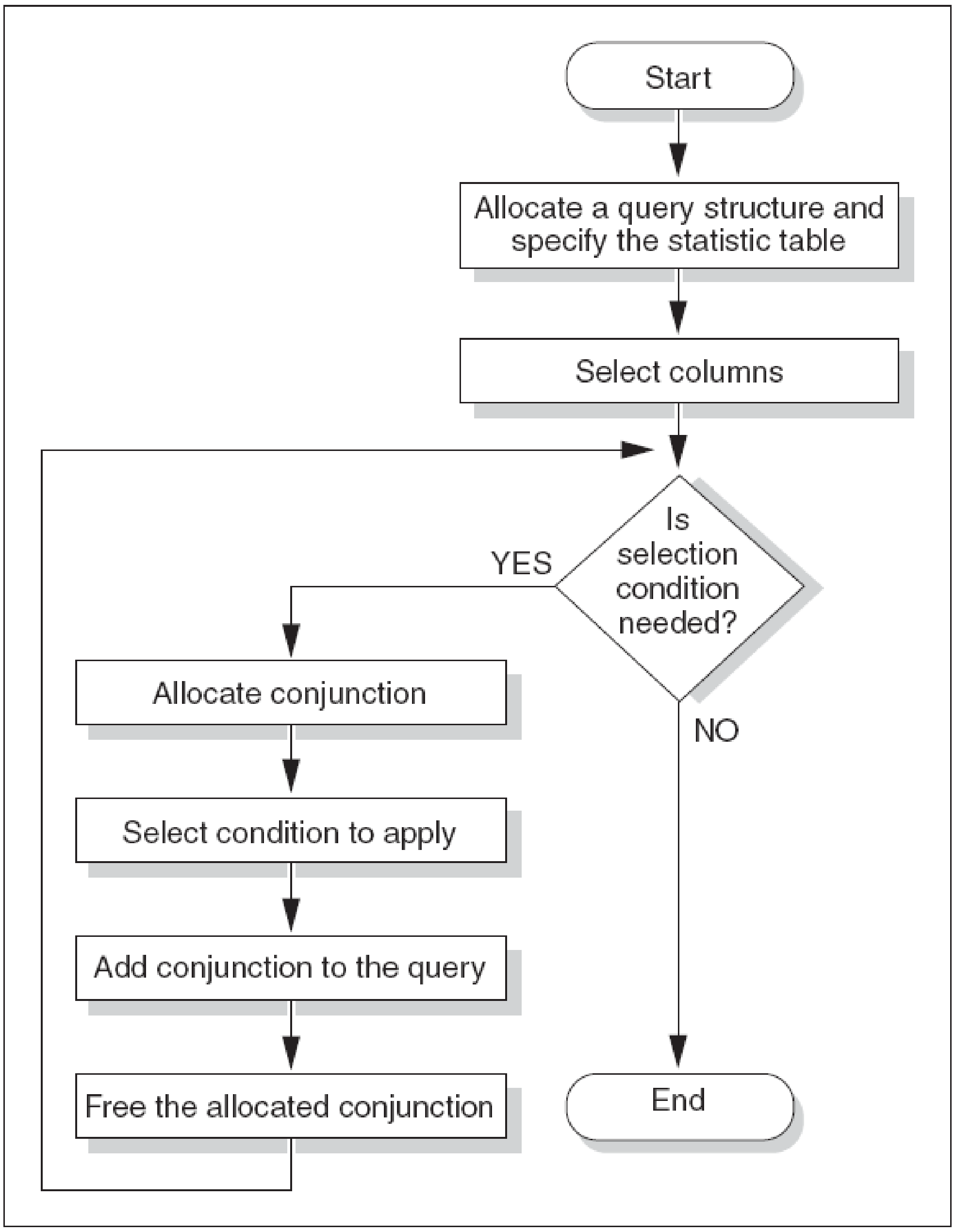
The following chart illustrates the basic skeleton of an RTDAPI application. The followingchart illustrates the basic skeleton of an RTDAPI application.


Start
Log on to the server
Initialize name cache if needed
Set up query
Start statistic stream and register the callback function
Stop statistic stream before application exit
Deallocate all used resources
Log off from server
End

The following sections illustrate how to set up the queries and the callback function.

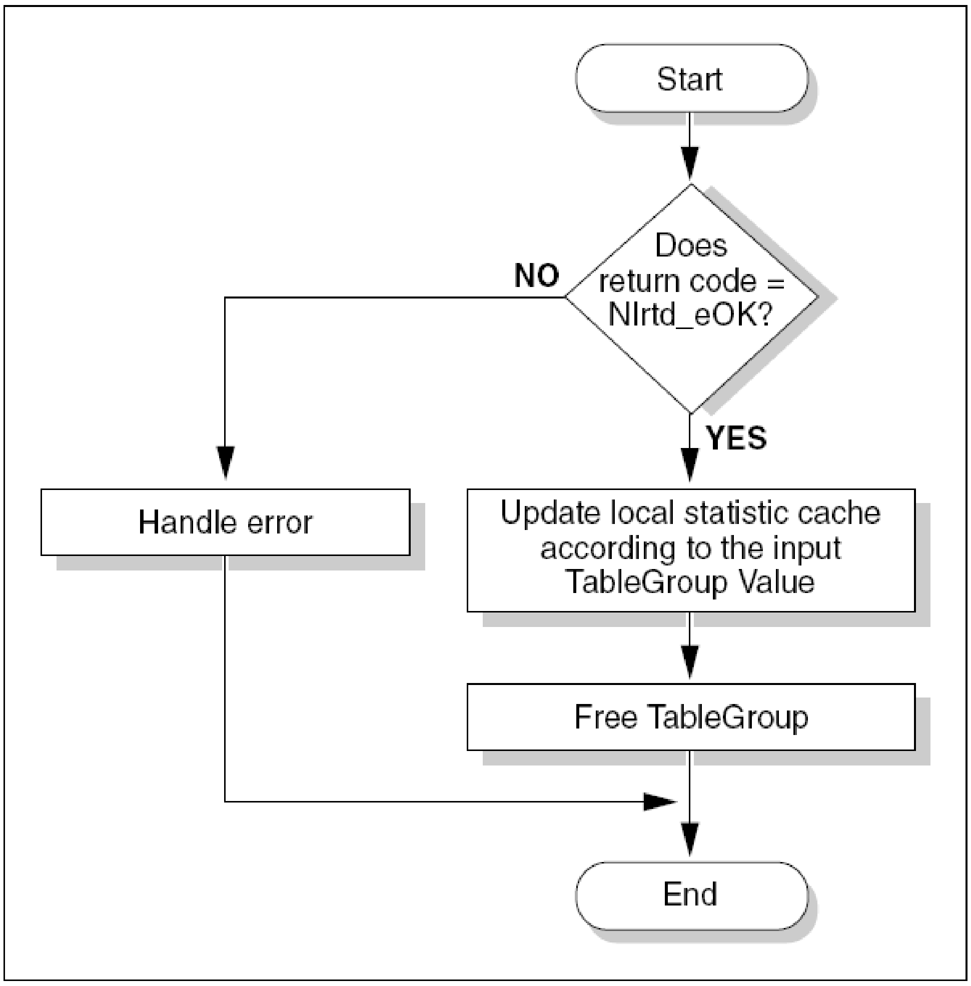
Setup query

A query is a selection criterion consisting of a statistic table, columns to select, and the selection condition.


Start
Allocate a query structure and specify the statistic table
Select columns
Is selection condition needed?
YES
NO
Allocate conjunction
Select condition to apply
Add conjunction to the query
Free the allocated conjunction
End

Callback function

The registered callback function is invoked every time a new statistic data stream comes in. The TableGroup input parameter contains delta information about the changes to the statistics. A return code is also passed to the application for error handling.


Start
Does return code = NIrtd_eOK?
NO
YES
Handle error
Update local statistic cache according to the input TableGroup Value
Free TableGroup
End

Source Code Description

This basic RTDAPI application is a console program that displays skillset statistics on-screen. This application asks for the skillset ID and the number of agents available for that particular skillset from the server.

A global skillset statistic cache (stat_list) is maintained within the application. The cache is updated (add rows, delete rows, and update rows) based on the delta data stream from the server. The cache content is sent to the console every time a data stream comes in from the server. This program runs until the sleep_time (input argument) expires.

Other than the main function and the data stream callback function, four utility functions (newRows, deleteRows, updateRows, and displayCache) are implemented. The following sections describe each of the application functions, including the list of API functions used.

	Purpose	**RTDAPI methods used**

Main	▪ perform logon to the server ▪ setup name cache ▪ setup query ▪ register the callback function and start the data stream operation ▪ wait for the sleep timer to expire ▪ clean up all resources before exit	▪ NIrtd_login ▪ NIrtd_allocateQuery ▪ NIrtd_allocateDataColumnforNameCache ▪ NIrtd_selectColumn ▪ NIrtd_startDataStream ▪ NIrtd_stopDataStream ▪ NIrtd_removeNameCacheforDataColumn ▪ NIrtd_freeQuery ▪ NIrtd_logout
Callback_function	▪ check the passed in return code ▪ if return code is OK, update the global statistic cache. Then deallocate the passed in TableGroup pointer. ▪ if not OK, print out the error code	▪ NIrtd_freeTableGroup
NewRows	▪ new statistic records are added into the global cache. ▪ For each new statistic record: – find the last record in the global cache list – append the record	▪ NIrtd_allocateRow ▪ NIrtd_allocateValue ▪ NIrtd_getCol ▪ NIrtd_freeRow ▪ NIrtd_freeValue
DeleteRows	▪ Delete records from the global statistic cache. ▪ For each record that is removed: o find the corresponding record o free all allocated resources o join the broken link list	▪ NIrtd_allocateRow ▪ NIrtd_allocateValue ▪ NIrtd_getCol ▪ NIrtd_freeRow ▪ NIrtd_freeValue
UpdateRows	▪ update the global statistic cache content ▪ For each updated record:	▪ NIrtd_allocateRow ▪ NIrtd_allocateValue ▪ NIrtd_getCol ▪ NIrtd_cpValue

	○ find the record from the global cache ○ copy the updated data record into the cache	▪ NIrtd_freeValue ▪ NIrtd_freeRow
DisplayCache	▪ display the content of the global cache on the console ▪ For each item in the global skillset statistic cache: ○ translate the skillset ID to skillset name ○ display the skillset name and the number of agents available on-screen	▪ NIrtd_allocateName ▪ NIrtd_getName ▪ NIrtd_getFailedName ▪ NIrtd_freeName

Programming tips

The RTDAPI is internally one thread that listens for data and calls the application's callback function. Two critical sections are used in the API code: one for the name database and one for the database or real-time data registrations. This means that:

- Only one thread can add, delete, change, or lookup real-time data registrations at any point in time. This also means that the registration database is locked during the execution of the application callback function.
- Only one thread can add, delete, change, or lookup names at any point in time.
  The following is the development advice, based on the preceding information:
- Take the TableGroup data passed to the application callback function, place it in a queue, and signal another thread to handle the displaying of the update.
- You can refresh the name cache database using the same display thread as the one that normally displays the data updates. Once refreshed, you can use a background thread to re-display all the names on the screen. This means that the update thread and background re-display thread alternate in being able to access the name database.
- Be aware of how you use your own pointer, which you can pass and have passed back to your application in the callback function. Think of the concurrence of its use. This means ensuring that the data associated is read only so that when these API functions are later upgraded to have multiple listening and callback threads, the callbacks can be executed concurrently.
- When the TableGroup information is passed to the application, most likely yourapplication will want to maintain a list of items for display and will want to apply the new, deleted, and delta items to this list. Keep in mind that the way to do this is to use the keys (agent telset

login ID, application ID, and skillset ID) to index into the display list that you are maintaining. Note that the agent telset login ID is really a string where as the others are ULONGs.

Last Page