A MITEL
PRODUCT
GUIDE

Mitel OpenScape Contact Center V12

IVR API V12

IVR API Integration Guide

Integration Guide
10/2024

A31003-S22C0-N100-01-7620

Mitel

Contents

1 About this guide

This guide describes how to integrate an Interactive Voice Response (IVR) system with OpenScape Contact Center using the OpenScape Contact Center IVR Application Programming Interface (API).

1.1 Who should use this guide

This guide is intended for system integrators who want to integrate an interactive voice response (IVR) system with OpenScape Contact Center.

1.2 Formatting conventions

The following formatting conventions are used in this guide:

Bold
This font identifies OpenScape Contact Center components, window and dialog box titles, and item names.

Italic
This font identifies references to related documentation.

`Monospace Font`
This font distinguishes text that you should type, or that the computer displays in a message.

NOTE: Notes emphasize information that is useful but not essential, such as tips or alternative methods for performing a task.

IMPORTANT: Important notes draw special attention to actions that could adversely affect the operation of the application or result in a loss of data.

1.3 Documentation feedback

To report an issue with this document, call the Customer Support Center.

When you call, be sure to include the following information. This will help identify which document you are having issues with.

- **Title:** IVR API Integration Guide
- **Order Number:** A31003-S22C0-N100-01-7620

2 About the OpenScape Contact Center IVR API

This chapter provides an overview of the OpenScape Contact Center IVR Application Programming Interface (API). The OpenScape Contact Center IVR API is supported when connected to the following communication platforms:

- OpenScape Voice
- OpenScape 4000 or HiPath 4000
- OpenScape Business

NOTE: When connected to an OpenScape Voice communication platform, using a front-end monitored IVR with preprocessing IVR extensions is not supported.

2.1 Overview

An IVR system interacts with callers to obtain additional information about their requirements and provides information to the callers while they are waiting in queue for the appropriate user to handle their call.

You can create a customized IVR script that gathers information from your customers. For example, you can prompt callers to press 1 for Service or press 2 for Sales.

The IVR script can gather a customer's requirements from the following sources:

- **Network Information (public communication platform information)**
  - Automatic Number Identification (ANI)—information about the calling party
  - Dialed Number Identification Service (DNIS)—information about the called party
- **Call Prompting**
  - Dual-Tone Multi-Frequency (DTMF) signaling—push-button responses from the caller (for example, account or personal identification numbers)
  - Voice recognition—voice responses from the caller for more specific information (for example, the prompt could tell the customer to choose either Sales or Service)

- **Customer Database Retrieval**
  - Customer profile (for example, language, skill level, and location)
  - Customer history (for example, credit record, past purchases, and open problems)
  - Customer preferences (for example, a specific user)

The purpose of the IVR script is to identify a queue that describes the changing requirements of a call as it waits in queue.

You can design an IVR script to gather ANI and DNIS information from OpenScape Contact Center, prompt the caller for specific information, and then use this information to access data from a host or local database.

For example, you can prompt callers to enter their four-digit account number, then use this number to access database information and identify queues for the calls based on the DNIS information or the To and From information.

The text following the diagram describes the numbered steps.

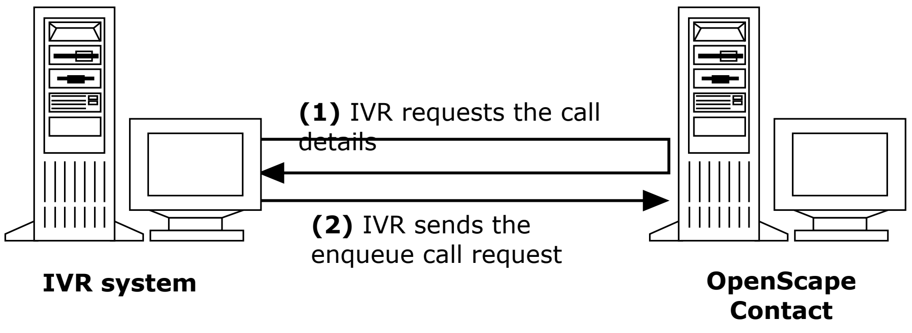
(1) IVR requests the call details
(2) IVR sends the enqueue call request
IVR system
OpenScape Contact

Figure 1 IVR system and OpenScape Contact Center interaction

1. When a call arrives at your site, it is answered by the IVR system. The IVR script then gathers the ANI and DNIS or To and From information of the transaction, determines the queue, and sends an enqueue call request to OpenScape Contact Center.

NOTE: The IVR system can also send an OpenScape Contact Center routing request to OpenScape Contact Center, supplying information collected by the IVR system. The Routing Server then assigns a queue and other routing information based on the configured system routing option, which can be used to send an enqueue call request.

2. The Routing Server then searches for the best available user to handle the call.
3. The IVR script can prompt the customers with a series of menus to determine the specific requirements of each customer. The more complex your menu system, the more detailed the information you can gather from your customers.

Each unique menu choice should correspond to a unique queue so that you can use the full power of OpenScape Contact Center skills-based routing capabilities.

NOTE: For more information on designing Interactive Voice Response (IVR) menu systems, contact your support representative.

NOTE: For more information on the available IVR functions, see Chapter 4, "Using the OpenScape Contact Center IVR API functions".

2.2 Installing the OpenScape Contact Center IVR API

To install the OpenScape Contact Center IVR Application Programming Interface (API), you must copy the **hppcivr.dll** file from the default installation folder on the OpenScape Contact Center main server machine to the IVR machine and reference the file in the IVR configuration. Additionally, you must copy the following files from the default installation folder to the application folder:

- **libssl.dll**
- **libcrypto.dll**

It is not necessary to make any reference to those files, they are internally used by the API.

2.3 IVR API functions for unmonitored IVR systems

OpenScape Contact Center uses IVR API functions to facilitate integration with unmonitored IVR resources.

NOTE: Unmonitored IVR systems are not supported when connected to an unsupported communication platform.

Since this type of IVR system is not monitored by OpenScape Contact Center, the following concepts must be taken into account:

- OpenScape Contact Center is notified, with ANI and DNIS numbers, when the call arrives at the IVR system.
- Coordination between OpenScape Contact Center and the IVR system is necessary to transfer a call from the IVR system to OpenScape Contact Center.
- Calls leaving the IVR system that are either disconnected or transferred out use an IVR API to notify OpenScape Contact Center.

The IVR system must provide the ANI and DNIS of the call to the OpenScape Contact Center system. This information can be obtained by the IVR system through the specific trunking protocol of the communication platform, for example, ISDN (Integrated Services Digital Network). The SetCallInfo API sets the ANI and DNIS locally before this information is sent to OpenScape Contact Center using the existing QueryCallInfo API.

QueryCallInfo API is used by the IVR system to notify OpenScape Contact Center when a call arrives at the IVR system. This API should be called as soon as possible after the IVR system begins processing the arriving call. The QueryCallInfo API has an optional parameter, TrunkConnectedIVRCall, which must be set to "1;" for unmonitored IVR systems.

Calls at an unmonitored IVR system that are enqueued to OpenScape Contact Center for IVR Hold or Queue Hold must be transferred to the OpenScape Contact Center route control group using a transit number. A transit number is a pilot number to the OpenScape Contact Center route control group. The transit number is necessary for tracking the unmonitored IVR call after it is transferred to OpenScape Contact Center.

OpenScape Contact Center maintains a pool of transit numbers, which are only used for transferring unmonitored IVR calls. The IVR system queries OpenScape Contact Center for a unique transit number, using the GetTransitNumber API. After receiving the transit number, the IVR system must immediately transfer the call. When the call reaches the OpenScape Contact Center RCG, OpenScape Contact Center identifies the call using the transit number and can then associate the call information previously received from the IVR system with the incoming call. The transit number then becomes available for use with another unmonitored IVR call.

NOTE: In the event of an error or time-out, the IVR system may choose to transfer the call to a non-OpenScape Contact Center device (such as voice mail or Xpressions), in which case it is not necessary to call GetTransitNumber.

You can use the GetTransitNumber API differently, depending on whether your communication platform uses Queue Hold or IVR Hold:

- **Queue Hold** — GetTransitNumber must be called immediately before the call is transferred to the OpenScape Contact Center RCG, which is typically soon after the call is enqueued. OpenScape Contact Center will then look for a user and divert the call to the assigned user.

- **IVR Hold** — GetTransitNumber must be called immediately before the call is transferred. The call is still transferred to the OpenScape Contact Center RCG using a transit number. For IVR Hold this is typically soon after the status of the call returned by the QueryCallStatus API is “Pending” or “Unanswered”. After transferring the call to the OpenScape Contact Center RCG, OpenScape Contact Center immediately diverts the call to the assigned user or designated time-out number.

If a transit number is requested and the IVR system cannot transfer the call immediately, then the ReleaseTransitNumber API must be called. This makes the transit number available for another call. Before attempting to transfer the call to OpenScape Contact Center again, the GetTransitNumber API must be called again to obtain a new transit number.

If a call at an unmonitored IVR system is disconnected (by the IVR system or by caller abandon) or transferred out to any number which is not a transit number, the CallerDisconnected API is used to notify OpenScape Contact Center.

NOTE: You should add the CallerDisconnected API to the hang-up branch of the IVR system and be integrated into other error handling procedures as necessary. If you do not add the API to the hang-up branch, your statistics will not be accurate, because OpenScape Contact Center will not know when the call ended.

Example: Unmonitored IVR system Step Sequence (IVR Hold)

The following is an example of an IVR system step sequence for an unmonitored IVR system:

1. Answer incoming IVR call.
2. Call Initialize(), followed by SetCallInfo(), followed by QueryCallInfo().
3. Play welcome message/main menu and process the call until it is ready to be enqueued to OpenScape Contact Center.
4. Call Enqueue().
5. Call QueryCallStatus() until call status is ‘Pending’ or ‘Unanswered’.
6. Call GetTransitNumber().
7. Immediately transfer the call using the number returned in GetTransitNumber().
8. OpenScape Contact Center will route the call to the assigned user or designated time-out extension.

The exact steps for any IVR system may differ from the steps above. The important things to note are:

- The order in which Initialize(), SetCallInfo(), and QueryCallInfo() are called.
- QueryCallInfo() is called as soon as possible so that OpenScape Contact Center can start tracking the call from the time it arrives at the IVR system.
- GetTransitNumber() is called only when the call is about to be transferred out of the IVR system to an OpenScape Contact Center resource (such as a user, RCG, or time-out extension), and the transfer must be completed immediately after GetTransitNumber() returns with a transit number.

3 Configuring the OpenScape Contact Center IVR API

This chapter describes the various IVR scenarios that you can use to enhance the gathering of your customer’s requirements. The scenarios are:

- IVR Hold
- Queue Hold
- Forget-me-not Queueing (FMNQ)

NOTE: Depending on the required caller experience, the IVR script can use one or more of these scenarios.

This chapter describes the various IVR scenarios that you can use to enhance the gathering of your customer’s requirements. The scenarios are:

- IVR Hold
- Queue Hold
- Forget-me-not Queueing (FMNQ)

NOTE: Depending on the required caller experience, the IVR script can use one or more of these scenarios.

3.1 IVR Hold configuration

You can use the IVR Hold configuration with an IVR system to hold an enqueued call until it is routed to a user.

NOTE: Since the IVR Hold configuration ties up the IVR extension and temporarily prevents the IVR system from handling incoming calls until the enqueued call has been routed to a user, you must alter the IVR Hold configuration to include more IVR extensions.

The following diagram shows the call flow for the IVR Hold configuration. Refer to the text following the call flow diagram for a description of the numbered steps.

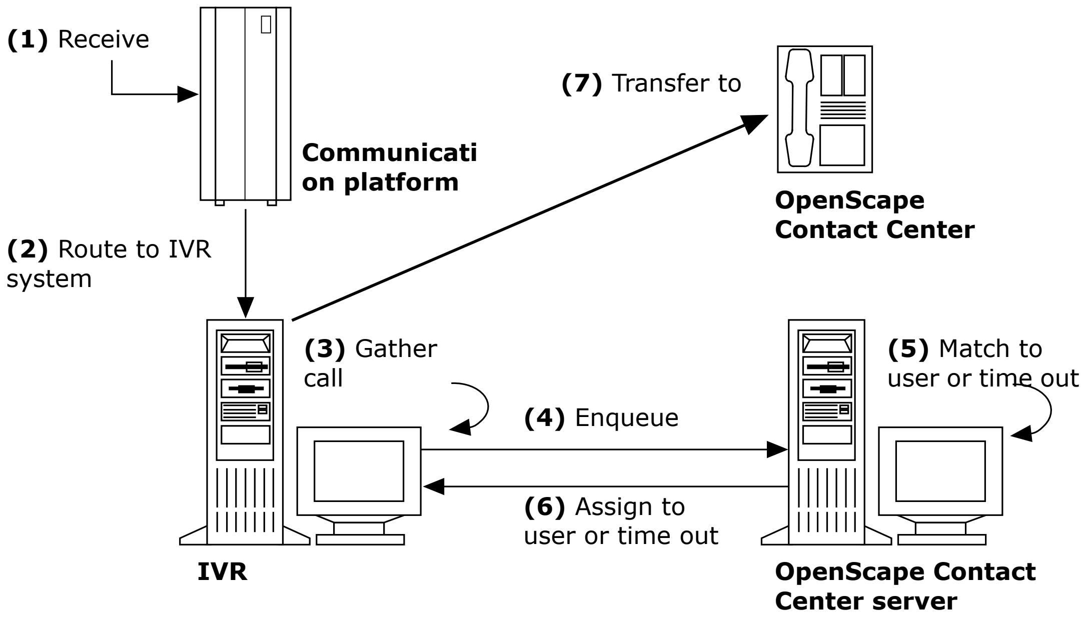
(1) Receive
Communicati on platform
(7) Transfer to
OpenScape Contact Center
(2) Route to IVR system
(3) Gather call
(4) Enqueue
(5) Match to user or time out
(6) Assign to user or time out
IVR
OpenScape Contact Center server

Figure 2 Call flow for the IVR Hold configuration

1. A new call arrives.
2. The call is routed to the IVR system.
3. The IVR system gathers information from the caller as to the purpose of the call, and then uses this information to identify the routing parameters for the call, including the queue.
4. The IVR system sends an enqueue call request to OpenScape Contact Center. The Routing Server then searches for the best available user to handle the call.
5. OpenScape Contact Center then does one of the following:
   - Assigns the call to the best available user
   - Reserves the call for a specific user (optional)

   If the call cannot be assigned to a user by the end of the last call step in the queue, it is routed to a time-out extension.

6. OpenScape Contact Center notifies the IVR system of the extension of the assigned user or the time-out target. If the call times out, the IVR system does one of the following:
   - Transfers the call to a default number specified by the IVR script or transfers the call to the time-out extension provided by the Routing Server. The time-out extension can either be a specific number defined by the queue of the call or a global default number used by all queues that do not have a specific number defined.
   - Prompts the customer for additional information and uses the routing parameters to enqueue the call again. At this time, the process of matching the call to a user is repeated.
7. The IVR system transfers the call to the assigned user.

NOTE: When connected to an OpenScape Voice communication platform, if the transfer fails and the call is reconnected to the IVR system, the IVR application must process the call as a new call.

3.2 Queue Hold configuration

You can use the Queue Hold configuration with an IVR system to transfer calls to an OpenScape Contact Center ACD/UCD/MLHG group in the communication platform, where calls wait until eligible users become available to handle the calls.

When a user becomes available, OpenScape Contact Center diverts the call from the OpenScape Contact Center ACD/UCD/MLHG group to the user. The Queue Hold configuration frees the IVR extensions for incoming calls by transferring calls to the communication platform while the Routing Server searches for available users.

This configuration reduces IVR extension requirements so that your IVR system can handle a greater number of incoming calls.

The following diagram shows the call flow for the Queue Hold configuration. Refer to the text following the call flow diagram for a description of the numbered steps.

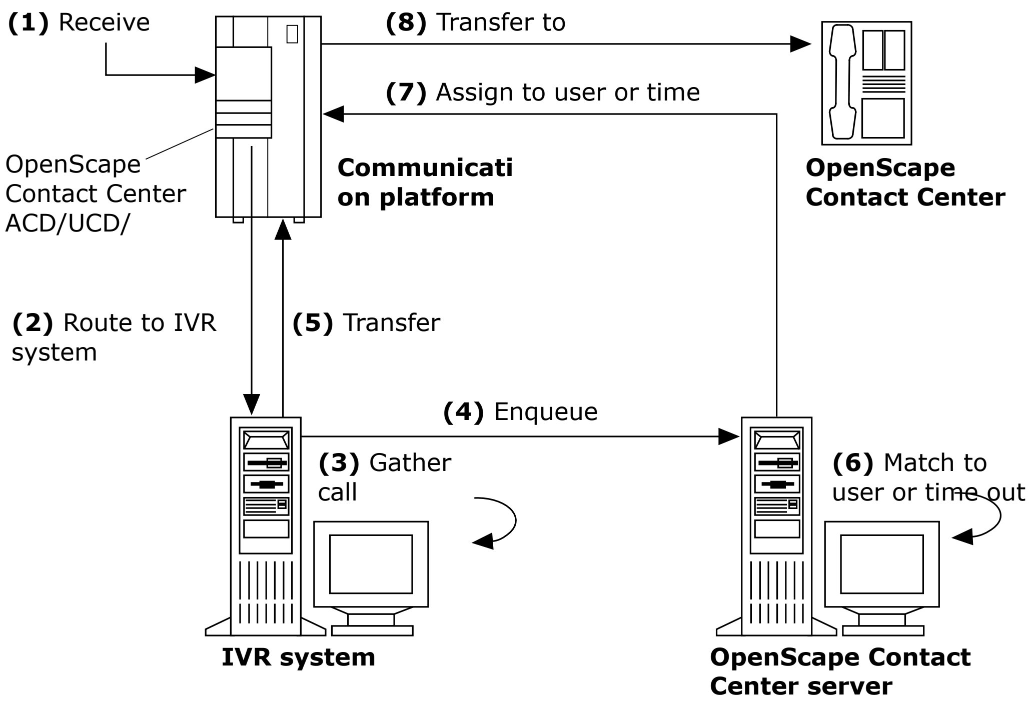
(1) Receive
(8) Transfer to
(7) Assign to user or time
OpenScape Contact Center ACD/UCD/
Communication platform
OpenScape Contact Center
(2) Route to IVR system
(5) Transfer
(4) Enqueue
(3) Gather call
(6) Match to user or time out
IVR system
OpenScape Contact Center server

Figure 3 Call flow of the Queue Hold configuration

1. A new call arrives.
2. The call is immediately routed to an IVR system.
3. The IVR system gathers information from the caller as to the purpose of the call, and then uses this information to identify the routing parameters for the call, including the queue.
4. The IVR system sends an enqueue call request to OpenScape Contact Center. The Routing Server then searches for the best available user to handle the call.
5. At this time, the IVR system transfers the call to an OpenScape Contact Center ACD/UCD/MLHG group.
6. OpenScape Contact Center then does one of the following:
    - Assigns the call to the best available user
    - Reserves the call for a specific user (optional)

    If the call cannot be assigned to a user by the end of the last call step in the queue, it is routed to a time-out extension.

7. The call is assigned to a specific user or time-out target.
8. The call is transferred to the assigned user.

NOTE: When connected to an OpenScape Voice communication platform, if the transfer fails and the call is reconnected to the IVR system, the IVR application must process the call as a new call.

3.3 FMNQ Configuration

You can set an FMNQ call so that it is either "interruptible" or "uninterruptible". If the call is set to "interruptible", the caller can be moved from an IVR system to the best available user extension, regardless of whether the caller is entering information into the IVR system. If the call is "uninterruptible", the routing of the call is temporarily suspended until the caller either:

- Enters critical information without the risk of that information being lost
- Listens to critical information without interruption

NOTE: Unmonitored IVR resources do not support FMNQ

3.4 Writing an IVR script

A customized IVR script gathers information from your customers, provides call management functions, and calls the IVR API functions provided with OpenScape Contact Center.

The IVR API functions are provided in a module file called **hppcivr.dll**, which you must include when running the IVR script. This module file is located in the default OpenScape Contact Center folder on the main server machine.

3.4.1 Writing an IVR script for the IVR Hold configuration

The following is the suggested flow for an IVR script for the IVR Hold configuration:

1. Initialize the connection between the IVR system and OpenScape Contact Center.
2. Check the system status to ensure that the Routing Server is available.
3. Gather the ANI and DNIS information for the call.
4. Identify the queue using the IVR script. For example, you could assign a queue to the call based on a combination of the call's ANI information and the customer's selection.
5. Enqueue the call using the Enqueue function. Ensure that the parameters specify that the IVR system will hold the call until assigned to a user, using the IVR Hold configuration. This notifies the Routing Server that the IVR system will transfer the call when it is assigned.
6. Check the state of the enqueued call regularly (for example, after every action or IVR function call) and take the appropriate action depending on the results:
   - If the call is assigned, transfer the call to the user's extension.
   - If the call has timed out, transfer the call to the default number defined in the IVR script or to the extension returned by the Routing Server.
   - If there is a Routing Server error, transfer the call to the default number defined in the IVR script.

The following diagram shows a sample IVR script for the IVR Hold configuration.

NOTE: When a call is successfully enqueued, the IVR script must regularly check the state of the call using the QueryCallStatus function, as well as handle conditions, such as enqueue errors and timed-out calls. In these cases, the IVR script should either transfer the call to a non-OpenScape Contact Center extension or enqueue the call with different parameters.

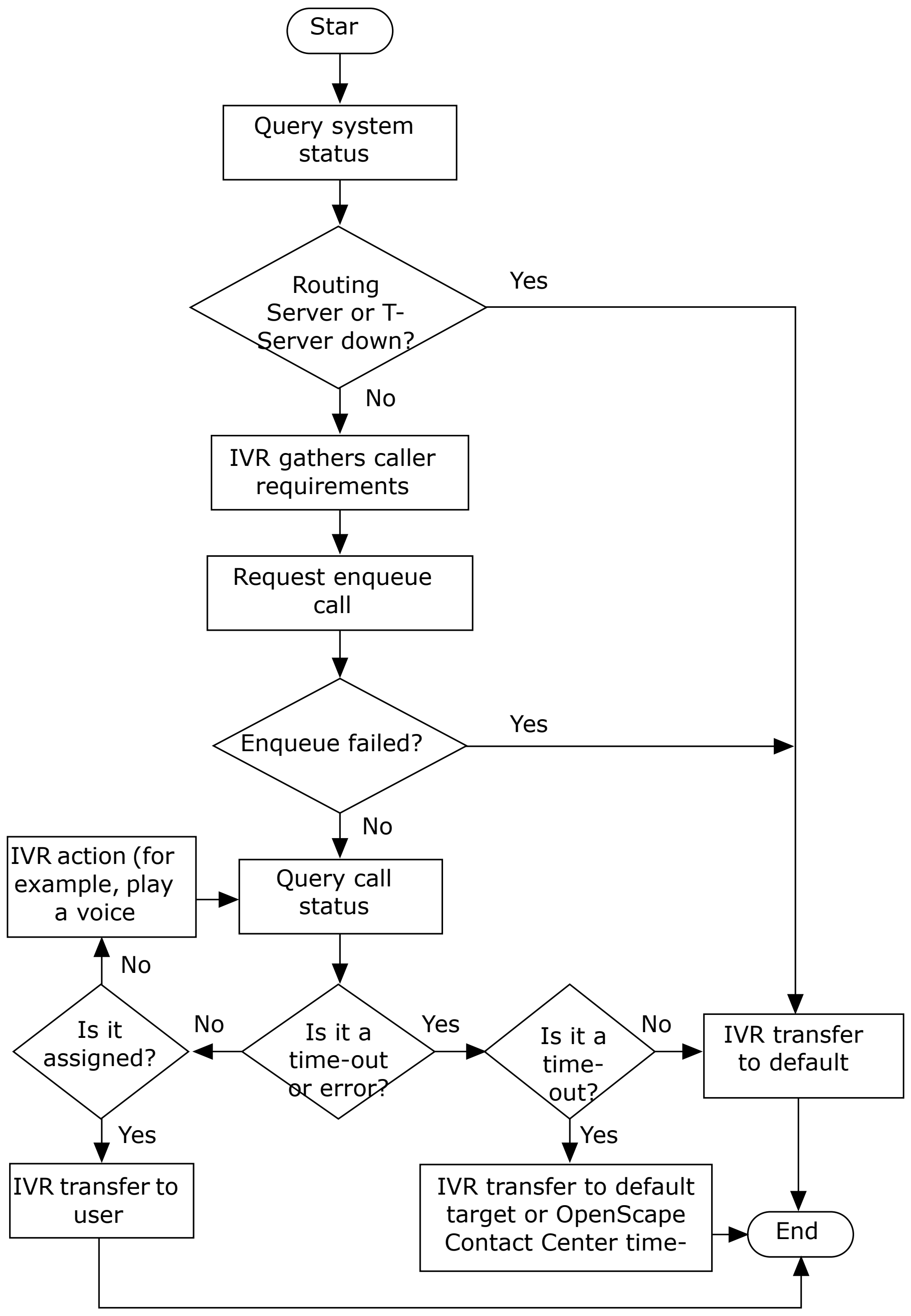
Star
Query system status
Routing Server or T-Server down?
Yes
No
IVR gathers caller requirements
Request enqueue call
Enqueue failed?
Yes
No
IVR action (for example, play a voice
Query call status
No
Is it assigned?
No
Is it a time-out or error?
Yes
Is it a time-out?
No
IVR transfer to default
Yes
Yes
IVR transfer to user
IVR transfer to default target or OpenScape Contact Center time-
End

Figure 4 *Sample flowchart for an IVR script in the IVR Hold configuration*

3.4.2 Writing an IVR script for the Queue Hold configuration

The following is a suggested flow for a customized IVR script for the Queue Hold configuration:

1. Initialize the connection between the IVR system and OpenScape Contact Center.
2. Check the system status to ensure that the Routing Server is available.
3. Gather the ANI and DNIS information for the call.
4. Identify the queue using the IVR script. For example, you could assign a queue to the call based on a combination of the call’s ANI information and the selections the customer makes.
5. Enqueue the call with the Enqueue function. Ensure that the parameters specify that the IVR system will transfer the call to an OpenScape Contact Center ACD/UCD/MLHG group, using the Queue Hold configuration. This notifies the Routing Server that OpenScape Contact Center must transfer the call when it is assigned.
6. Transfer the call to an OpenScape Contact Center ACD/UCD/MLHG group. OpenScape Contact Center automatically holds the call in the OpenScape Contact Center ACD/UCD/MLHG group until the call is assigned to a user.

The following diagram shows a sample IVR script for the Queue Hold configuration.

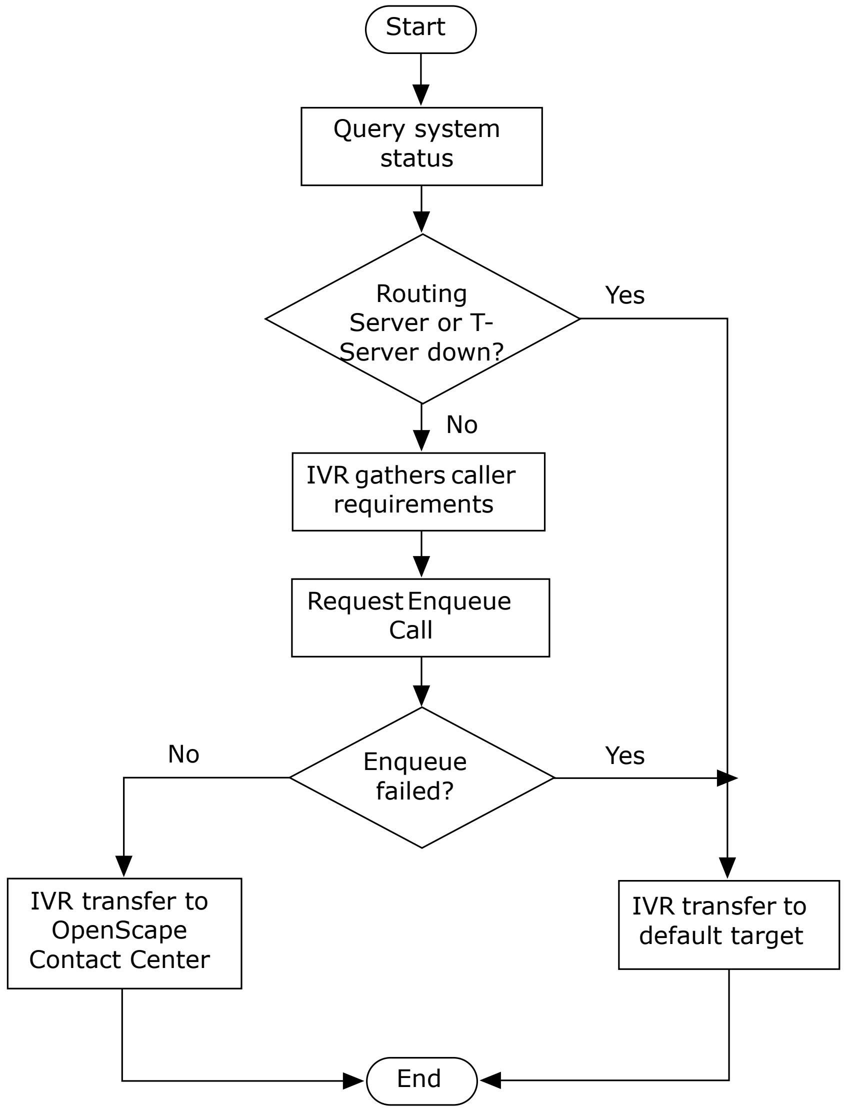
Start
Query system status
Routing Server or T-Server down?
Yes
No
IVR gathers caller requirements
Request Enqueue Call
Enqueue failed?
No
Yes
IVR transfer to OpenScape Contact Center
IVR transfer to default target
End

Figure 5 Sample flowchart of an IVR script for the Queue Hold configuration

3.4.3 Writing an IVR script for the FMNQ configuration

The following is a suggested flow for a customized IVR script for the FMNQ configuration:

1. Initialize the connection between the IVR system and OpenScape Contact Center.
2. Use the QueryCallInfo function to gather the ANI and DNIS information for the call and to obtain the value of the FMNQ flag.
3. If the flag is set to 1, start the FMNQ call flow and determine whether the caller wants to enter an "interruptible" or "uninterruptible" session.
4. If the caller wants to enter an "uninterruptible" session, set the call to non-transferable (0).
5. Continue to gather caller requirements inside the "uninterruptible" session.
6. When the "uninterruptible" session has ended, set the call to transferable (1).

NOTE: When connected to an OpenScape Voice communication platform, you must ensure that the call is transferred to the Music On Hold Hunt Group.

NOTE: Before the IVR script disconnects or transfers an FMNQ call, you must issue the SetCallTransferable (0) and receive a successful return code. If you not receive a successful return code, this indicates that the call is in the process of being transferred and you must wait for the process to complete.

The following diagram shows a sample IVR script for the FMNQ configuration.

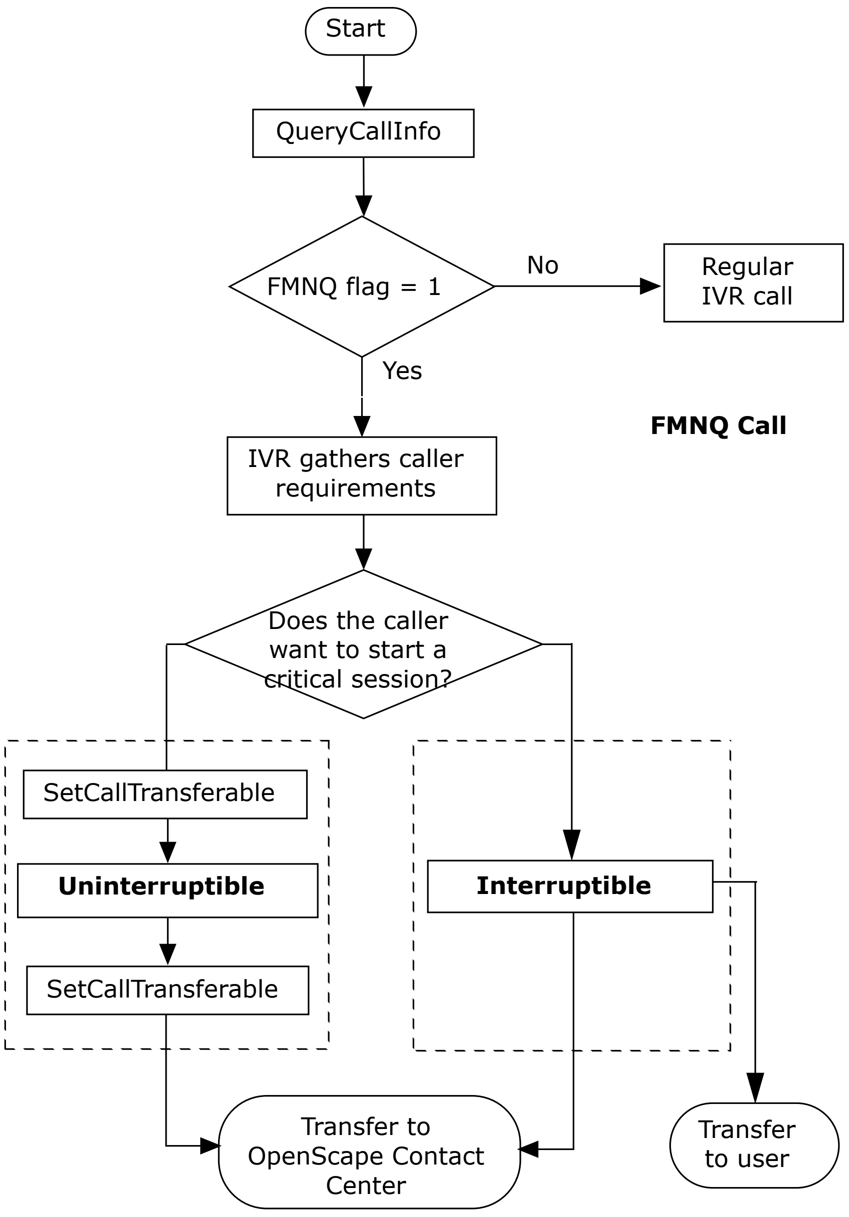

Figure 6 Sample flowchart of an IVR script for the FMNQ configuration

3.5 Using an IVR system in a multitenant environment

In a multitenant environment, IVR resources are system-level resources that are shared across multiple business units.

If you are deploying a front-end IVR application in a multitenant environment, the IVR system must contain the logic necessary for it to be aware of business units.

For example:

- The IVR system needs to determine which business unit an IVR call will be routed to.
- When an IVR system enqueues a call into the OpenScape Contact Center system, the application must enqueue the call against the correct business unit.
- In the Queue Hold scenario, an IVR system needs to know the correct pilot number for each business unit, and needs to transfer the call to the correct business unit.

Two API functions are provided for use in a multitenant environment:

- GetBusinessUnit function – retrieves the business unit for a specified call. For more information, see Section 4.9, "GetBusinessUnit", on page 41.
- SetBusinessUnit function – sets the business unit for a specified call. The SetBusinessUnit function can be called only once for each IVR call. For more information, see Section 4.21, "SetBusinessUnit", on page 61.

NOTE: In a multitenant environment, the SetBusinessUnit function must be called before the following functions are invoked: CreateCallback, Enqueue, EnqueueForAgent, GetBusinessUnit.

An optional BusinessUnitName parameter can be specified in the QueryQueueStatistics function to indicate the business unit that a queue belongs to. In a non-multitenant environment, the BusinessUnitName parameter is ignored.

4 Using the OpenScape Contact Center IVR API functions

This chapter describes how to use the OpenScape Contact Center IVR API functions and provides examples of how to create IVR scripts.

NOTE: An unmonitored IVR system is an IVR system that is not monitored by OpenScape Contact Center.

4.1 Passing parameters

When calling OpenScape Contact Center IVR API functions, you can pass information by value or by reference. The following sections describe these two methods.

Passing parameters by value

When you pass parameters by value, you are passing information from the IVR script to OpenScape Contact Center. Each of the IVR API functions requires that different information be sent to OpenScape Contact Center, so that the application can carry out the requested action. For example, the IVR script must pass the queue name to OpenScape Contact Center when invoking the Enqueue function, so that the application can enqueue the call.

Passing parameters by reference

When you pass parameters by reference, you are passing information from OpenScape Contact Center to the IVR script. When calling an IVR API function, the IVR script specifies a variable name for variables passed by reference. OpenScape Contact Center returns the required information in the specified variable. For example, OpenScape Contact Center returns the ANI and DNIS information for a call when the IVR script invokes the QueryCallInfo function.

NOTE: Parameters that are passed by reference must be preceded by the “greater than” (>) symbol.

4.2 CallerDisconnected

NOTE: This function is used by IVR scripts for unmonitored IVR systems only.

The CallerDisconnected function allows IVR systems to notify OpenScape Contact Center about a disconnected call. The T-Server then deletes the call and releases the transit number (if applicable) associated with the call. The T-Server then sends a notification to other OpenScape Contact Center components.

Syntax

```
CallerDisconnected (CallID, TermType)
```

Parameters

Name	Type	Range	Description
CallID	String passed by value	18 characters	The CallID for the current call.
TermType	Integer passed by value	0, 1, or 2	0 – Indicates the call was terminated as a result of an internal issue. 1 – Indicates the call was transferred out of scope. 2 – Indicates the call was abandoned.

Table 1 Parameters for the CallerDisconnected function

Return Codes

The most common codes returned by this function are as follows:

0	Successful.
–308	QueryCallInfo was not called prior to invoking this function. You must call QueryCallInfo to obtain the CallID.
–310	This function can only be used for unmonitored IVR calls.
–910	The version of the IVR API DLL does not correspond to the version of OpenScape Contact Center.
–931	There is an inconsistency monitoring the type of call between the IVR API and the T-Server.
Other	Any other code indicates failure. For more information about a specific code, see Chapter 5, “Return codes”.

Example

```
CallerDisconnected (‘1341023126268002’, 2)
```

4.3 CreateCallback

The CreateCallback function attempts to create a callback in the Callback Server. The IVR system can indicate that a customer should be called back when a user becomes available. This function supports a maximum of 1000 bytes of contact data.

NOTE: You must call the Initialize and QueryCallInfo functions before invoking this function. In a multitenant environment, you must also call the SetBusinessUnit function before invoking this function. For more information, see Section 4.13, "Initialize", on page 46, Section 4.15, "QueryCallInfo", on page 49, and Section 4.21, "SetBusinessUnit", on page 61.

Syntax

```
CreateCallback (CallID, QueueName, Description, TimeZoneOffset,
ContactName, >CallbackID, Schedule1, Schedule2, Schedule3)
```

Parameters

You must enter the parameters in the order that they appear in the following table.

Name	Type	Range	Description
CallID	String passed by reference	18 characters	The CallID returned from QueryCallInfo should be provided as input.
QueueName	String passed by value	32 characters	The callback queue to use for the callback.
Description (optional)	String passed by reference	100 characters	A string containing a brief description of the call, which is displayed in the OpenScape Contact Center Client Desktop application so that users can identify the call.
TimeZoneOffset	Integer passed by value	0 to +60*12 or 0 to -60*12	The difference in minutes between the caller's local time and UTC (Coordinated Universal Time).
ContactName	String passed by value	80 characters	The contact name.
CallbackID	String passed by reference	17 characters	A unique ID for the callback created.

Table 2 Parameters for the CreateCallback function

Name	Type	Range	Description
Schedule1	String passed by value	47 characters for the telephone number. 81 characters for the entire string.	A string indicating the telephone number where the caller can be reached and the period of time (in the caller's local time) during which the caller wants to receive callbacks. The format is: `telephone number;mm/dd/yyyy;hh:mm;mm/dd/yyyy;hh:mm` where the first date and time (using a 24-hour clock) is the start time, and the second is the end time.
Schedule2 (optional)	String passed by value	47 characters for the telephone number. 81 characters for the entire string.	A string indicating the telephone number where the caller can be reached and the period of time (in the caller's local time) during which the caller wants to receive callbacks. The format is: `telephone number;mm/dd/yyyy;hh:mm;mm/dd/yyyy;hh:mm` where the first date and time (using a 24-hour clock) is the start time, and the second is the end time.
Schedule3 (optional)	String passed by value	47 characters for the telephone number. 81 characters for the entire string.	A string indicating the telephone number where the caller can be reached and the period of time (in the caller's local time) during which the caller wants to receive callbacks. The format is: `telephone number;mm/dd/yyyy;hh:mm;mm/dd/yyyy;hh:mm` where the first date and time (using a 24-hour clock) is the start time, and the second is the end time.

Table 2 Parameters for the CreateCallback function

Return Codes

The most common codes returned by this function are as follows:

0	Successful.
–308	QueryCallInfo was not called prior to invoking this function. You must call QueryCallInfo to obtain the CallID.
–314	The specified business unit name was not set or it did not match the name of any business unit in the database. (This error is applicable only when OpenScape Contact Center is configured as a multitenant system.)
–1006	You are attempting to create a duplicate callback.
–1028	A schedule time is invalid.

–1029	The callback schedules have no mutual time with the contact center open hours.
–1031	A telephone number in the request is in the excluded numbers list.
–1033	A callback cannot accept contact data longer than 1000 bytes.
–1040	All schedules have already expired.
–1045	The callback queue name is invalid.
–1047	The schedule contains a date that is too far in the future.
Other	Any other code indicates failure. For more information about a specific code, see Chapter 5, “Return codes”.

Example

The following example requests that OpenScape Contact Center create a callback for the current call (identified by the CallID) using “CBQueue” as the callback queue.

```
CreateCallback(CallID, 'CBQueue', 'Callback', 0, 'Alan Smith',
>CBCallID,'5555550199;06/10/2010;11:00;06/11/2010;16:00',
'5555550199;06/11/2010;12:00;06/11/2010;14:00')
```

The Callback ID is returned after the callback is created.

Telephone number format

The telephone number parameter must be entered in canonical format as follows:

+[CountryCode] Space [(AreaCode) Space] SubscriberNumber [--Extension]

The telephone number can also be a dialable address obtained from a communication platform or as a result of calling a TAPI function. The following strings are correct telephone numbers:

- +1 (555) 555-0199
- (555) 555-0199
- +1 555-0199
- +1 (555) 555-0199--1212
- 5555550199

The extension part of a telephone number is truncated by the Callback Server before sending it to the T-Server. It is used only when displaying the callback in the Client Desktop and Manager applications, so that a user can manually dial the extension, if required.

4.4 DeleteCallback

The DeleteCallback function is used to delete an existing callback that was created using the CreateCallback function. This function takes as its parameter the CallbackID that was returned from the CreateCallback function.

NOTE: You must call the Initialize and QueryCallInfo functions before invoking this function. For more information, see Section 4.13, “Initialize”, on page 46 and Section 4.15, “QueryCallInfo”, on page 49.

Syntax

```
DeleteCallback (CallbackID)
```

Parameters

Name	Type	Range	Description
CallbackID	String passed by value	17 characters	The CallbackID returned from a successful CreateCallback request that is to be deleted.

Table 3 Parameters for the DeleteCallback function

Return Codes

The most common codes returned by this function are as follows:

0	Successful.
–1021	Invalid CallbackID.
Other	Any other code indicates failure. For more information about a specific code, see Chapter 5, “Return codes”.

Example

The following example asks for the callback with the CallbackID of `CBCallID` to be deleted.

```
DeleteCallback(CBCallID)
```

4.5 Dequeue

The Dequeue function sends a request to the Routing Server to dequeue a specified call. If you are using the IVR Hold configuration, you can use the Dequeue function to dequeue a call so that the IVR script can transfer the call to an extension or perform some other action with the call.

If the IVR script permits the caller to continue to input choices after the call has been enqueued, you can use further caller input to change the routing of the call. For example, if callers wait too long in queue, they might decide to leave a voice message; the IVR script can then dequeue the call and transfer it to a voice mail extension.

NOTE: You must call the Initialize and QueryCallInfo functions before invoking this function. For more information, see Section 4.13, “Initialize”, on page 46 and Section 4.15, “QueryCallInfo”, on page 49.

Syntax

```
Dequeue (CallID)
```

Parameters

You must enter the parameters in the order that they appear in the following table.

Name	Type	Range	Description
CallID	String passed by value	18 characters	The CallID for the call you want to dequeue.

Table 4 Parameters for the Dequeue function

Return Codes

The most common codes returned by this function are as follows:

0	Successful.
–304	Initialize was not called before invoking this function. You should call Initialize at the start of your script.
–306	One or more of the parameters is either incorrect or is of the wrong type.
–308	QueryCallInfo was not called prior to invoking this function. You must call QueryCallInfo to obtain the CallID.
–701	The request timed out before the function could be completed. It may be an indication of network problems or a busy server.
–801	Unable to connect to one of the OpenScape Contact Center servers. Either the server name is invalid or the server is not operational.

Other	Any other code indicates failure. For more information about a specific code, see Chapter 5, “Return codes”.

Example

The following example sends a request to dequeue a call.

```
Dequeue('1341023126268002')
```

4.6 Disconnect

The Disconnect function disconnects the specified call on the IVR extension.

NOTE: This function does not support unmonitored IVR calls.

NOTE: You must call the Initialize and QueryCallInfo functions before invoking this function. For more information, see Section 4.13, “Initialize”, on page 46 and Section 4.15, “QueryCallInfo”, on page 49.

Syntax

```
Disconnect (CallID)
```

Parameters

Name	Type	Range	Description
CallID	String passed by value	18 characters	The CallID for the call you want to disconnect.

Table 5 Parameters for the Disconnect function

Return codes

The most common codes returned by this function are as follows:

0	Successful.
–304	Initialize was not called before invoking this function. You should call Initialize at the start of your script.
–306	One or more of the parameters is either incorrect or is of the wrong type.
–311	The function does not support unmonitored IVR calls.
–701	The function request timed out before the function could be completed. This will normally occur only when querying the status of a call. This may also be an indication of network problems or a busy server.

–801	Unable to connect to one of the OpenScape Contact Center servers. Either the server name is invalid or the server is not operational.
–903	There was no call associated with the specified CallID. Ensure that all IVR extensions are represented in the database.
–914	The operation you are attempting has failed.
Other	Any other code indicates failure. For more information about a specific code, see Chapter 5, "Return codes".

Example

```
Disconnect('1341023126268002')
```

4.7 Enqueue

The Enqueue function requests that the Routing Server enqueue a call and find the best available user to handle the call. This function passes the routing information of a call to the Routing Server, including the queue and the call's initial priority.

NOTE: If you want to associate the contact data with the call, you must set these parameters before enqueuing the call.

Use this function to enqueue calls in either the IVR Hold or the Queue Hold configuration. After a call is enqueued in the IVR Hold configuration, the IVR script must check if the call has been assigned using the QueryCallStatus function, and then transfer the call to the user's extension. After a call is enqueued in the Queue Hold configuration, the call should be immediately transferred to an OpenScape Contact Center ACD/UCD/MLHG group.

NOTE: You must call the Initialize and QueryCallInfo functions before invoking this function. In a multitenant environment, you must also call the SetBusinessUnit function before invoking this function. For more information, see Section 4.13, "Initialize", on page 46, Section 4.15, "QueryCallInfo", on page 49, and Section 4.21, "SetBusinessUnit", on page 61.

Syntax

```
Enqueue (CallID, IVRHold, QueueName, InitialPriority,
Description,>EstimatedWait, >CallsinQueue, ANIDNIS)
```

Parameters

You must enter the parameters in the order that they appear in the following table.

Name	Type	Range	Description
CallID	String passed by value	18 characters	The CallID for the call you want to enqueue.
IVRHold	Integer passed by value	0 or 1	0 - Indicates the call will be transferred to an ACD/UCD/MLHG group for the Queue Hold configuration. 1- Indicates the call will wait at the IVR extension for the IVR Hold configuration.
QueueName	String passed by value	32 characters	The name of the queue for the call.
InitialPriority	Integer passed by value	1 to 100	Specifies the priority of a call, where 1 is the lowest and 100 is the highest priority.
Description (optional)	String passed by value	100 characters	A string containing a brief description of the call, which is displayed in the OpenScape Contact Center Client Desktop application so that users can identify the call.
EstimatedWait	Integer passed by reference	Greater than or equal to 0	The estimated wait time in seconds for this queue.
CallsInQueue	Integer passed by reference	Greater than or equal to 0	The number of calls in queue for the specified queue.
ANIDNIS (optional)	String passed by value	161	The ANI and DNIS numbers of the call. The Statistics Server stores these numbers in the OpenScape Contact Center database. The ANI and DNIS must be separated by a semicolon.

Table 6 Parameters for the Enqueue function

Return Codes

The most common codes returned by this function are as follows:

0 Successful.

–304 Initialize was not called before invoking this function. You should call Initialize at the start of your script.

–305	This code indicates an unknown error (for example, user error or the system is unstable).
–306	One or more of the parameters is either incorrect or is of the wrong type.
–314	The specified business unit name was not set or it did not match the name of any business unit in the database. (This error is applicable only when OpenScape Contact Center is configured as a multitenant system.)
–701	The request timed out before the function could be completed. It may be an indication of network problems or a busy server.
–801	Unable to connect to one of the OpenScape Contact Center servers. Either the server name is invalid or the server is not operational.
–903	There was no call associated with the specified CallID. Ensure that all IVR extensions are represented in the database.
–905	The specified queue did not match the name of any queue in the database.
–907	The T-Server is not available. This error means the Routing Server could not enqueue the call because the T-Server was not available.
–**908**	The T-Server is not available.
Other	Any other code indicates failure. For more information about a specific code, see Chapter 5, “Return codes”.

Example

The following example asks to enqueue the call with the specified CallID.

```
Enqueue(‘1341023126268002’, 1, ‘Sales’, 12, ‘Call enqueued from
IVR’, >EstWaitTime, >CallInQueue, ‘9055557900;69030’)
```

4.8 EnqueueForAgent

The EnqueueForAgent function asks the Routing Server to enqueue a call for a specified user and time. If the AgentWaitTime expires, the call is no longer reserved for a specified user and is enqueued to the queue specified by the QueueName. This function passes the routing information of a call to the Routing Server, including the queue and the call’s initial priority. The function also passes to the Routing Server information that appears in the OpenScape Contact Center Client Desktop application, such as the description.

Use this function to enqueue calls in either the IVR Hold or the Queue Hold configuration. After a call is enqueued in the IVR Hold configuration, the IVR script must check if the call has been assigned using the QueryCallStatus function, and then transfer the call to the user's extension. After a call is enqueued in the Queue Hold configuration, the call should be immediately transferred to an OpenScape Contact Center ACD/UCD/MLHG group.

NOTE: You must call the Initialize and QueryCallInfo functions before invoking this function. In a multitenant environment, you must also call the SetBusinessUnit function before invoking this function. For more information, see Section 4.13, "Initialize", on page 46, Section 4.15, "QueryCallInfo", on page 49, and Section 4.21, "SetBusinessUnit", on page 61.

Syntax

```
EnqueueForAgent (CallID, IVRHold, QueueName, InitialPriority,
Description, AgentID, AgentWaitTime, >EstimatedWait,
>CallsinQueue, ANIDNIS)
```

Parameters

You must enter the parameters in the order that they appear in the following table.

Name	Type	Range	Description
CallID	String passed by value	18 characters	The CallID for the call you want to enqueue.
IVRHold	Integer passed by value	0 or 1	0 - Indicates the call will be transferred to an ACD/UCD/MLHG group for the Queue Hold configuration. 1 - Indicates the call will wait at the IVR extension for the IVR Hold configuration.
QueueName	String passed by value	32 characters	The name of the queue for the call.
InitialPriority	Integer passed by value	1 to 100	Specifies the priority of a call, where 1 is the lowest and 100 is the highest priority.
Description (optional)	String passed by value	100 characters	A string containing a brief description of the call, which is displayed in the OpenScape Contact Center Client Desktop application so that users can identify the call.
AgentID	String passed by value	8 characters	The user for whom you want to reserve the call.

Table 7 Parameters for the EnqueueForAgent function

Name	**Type**	**Range**	**Description**
AgentWaitTime	Integer passed by value	Greater than 0	The maximum time in seconds the call can wait for a reserved user. If this time expires, the call is enqueued to the queue specified by the QueueName.
EstimatedWait	Integer passed by value	Greater than or equal to 0	The estimated wait time in seconds for this queue.
CallsInQueue	Integer passed by value	Greater than or equal to 0	The number of calls in queue for the specified queue.
ANIDNIS (optional)	String passed by value	161 characters	The ANI and DNIS numbers of the call. The Statistics Server stores these numbers in the OpenScape Contact Center database. The ANI and DNIS must be separated by a semicolon.

Table 7 Parameters for the EnqueueForAgent function

Return codes

The most common codes returned by this function are as follows:

0	Successful.
–304	Initialize was not called before invoking this function. You should call Initialize at the start of your script.
–305	This code indicates an unknown error (for example, user error or the system is unstable).
–306	One or more of the parameters is either incorrect or is of the wrong type.
–314	The specified business unit name was not set or it did not match the name of any business unit in the database. (This error is applicable only when OpenScape Contact Center is configured as a multitenant system.)
–701	The request timed out before the function could be completed. It may be an indication of network problems or a busy server.
–801	Unable to connect to one of the OpenScape Contact Center servers. Either the server name is invalid or the server is not operational.
–903	There was no call associated with the specified CallID. Ensure that all IVR extensions are represented in the database.
–904	The specified agent ID did not match the ID of any user in the database.
–905	The specified queue did not match the name of any queue in the database.
–907	The T-Server is not available. This error means the Routing Server could not enqueue the call because the T-Server was not available.
–**908**	The T-Server is not available.
Other	Any other code indicates failure. For more information about a specific code, see Chapter 5, “Return codes”.

Example

The following example asks to enqueue the call for user 790006 for 120 seconds, after which it will be enqueued to “CT_Pilot1”.

```
EnqueueForAgent (‘75510243484780002’, 0, ‘Support’, 8, ‘Call
reserved by IVR for user’, ‘790006’, 120, >EstWaitTime,
>CallInQueue)
```

4.9 GetBusinessUnit

NOTE: The GetBusinessUnit function is supported only in a multitenant environment.

The GetBusinessUnit function retrieves the business unit for the specified call.

NOTE: You must call the Initialize, QueryCallInfo, and SetBusinessUnit functions before invoking this function. For more information, see Section 4.13, "Initialize", on page 46, Section 4.15, "QueryCallInfo", on page 49, and Section 4.21, "SetBusinessUnit", on page 61.

Syntax

```
GetBusinessUnit (CallID, >BusinessUnitName)
```

Parameters

You must enter the parameters in the order that they appear in the following table.

Name	Type	Range	Description
CallID	String passed by value	18 characters	The CallID of the call that you want to retrieve the business unit for.
BusinessUnit Name	String passed by reference	32 characters	The name of the business unit for the specified call.

Table 8 Parameters for the GetBusinessUnit function

Return Codes

The most common codes returned by this function are as follows:

0	Successful.
–306	One or more of the parameters is either incorrect or is of the wrong type.
–400	The system has low system resources.
–914	The operation you are attempting has failed.
Other	Any other code indicates failure. For more information about a specific code, see Chapter 5, "Return codes".

Example

The following example retrieves the business unit name for the specified CallID.

```
GetBusinessUnit (‘827102321745900 2’, >BusinessUnitName)
```

4.10 GetCallTransferable

The GetCallTransferable function returns the value of the flag set by the SetCallTransferable function and determines whether the call can be transferred by OpenScape Contact Center to an assigned user.

NOTE: This function does not support unmonitored IVR calls.

NOTE: You must call the Initialize and QueryCallInfo functions before invoking this function. For more information, see Section 4.13, “Initialize”, on page 46 and Section 4.15, “QueryCallInfo”, on page 49.

Syntax

```
GetCallTransferable (CallID, >FlagValue)
```

Parameters

You must enter the parameters in the order that they appear in the following table.

Name	Type	Range	Description
CallID	String passed by value	18 characters	The CallID for the call you want to transfer.
FlagValue	Integer passed by reference	0 or 1	Indicates whether a call can be transferred to an assigned user. If set to false (0), the system suspends matching a user to the call until the flag is set to true (1). If the call is transferred back to the queue by the IVR, the flag is automatically reset to true.

Table 9 Parameters for the GetCallTransferable function

Return codes

The most common codes returned by this function are as follows:

0 Successful.

–304	Initialize was not called before invoking this function. You should call Initialize at the start of your script.
–306	One or more of the parameters is either incorrect or is of the wrong type.
–311	The function does not support unmonitored IVR calls.
–701	The request timed out before the function could be completed. It may be an indication of network problems or a busy server.
–801	Unable to connect to one of the OpenScape Contact Center servers. Either the server name is invalid or the server is not operational.
Other	Any other code indicates failure. For more information about a specific code, see Chapter 5, “Return codes”.

Example

The following example returns whether the call can be transferred by OpenScape Contact Center to an assigned user.

```
GetCallTransferable (‘0831023896468002’, >FlagValue)
```

4.11 GetContactData

The GetContactData function retrieves the contact data for a specified CallID. If you want to associate the contact data with the call, you must set the contact data before enqueuing the call.

NOTE: You must call the Initialize and QueryCallInfo functions before invoking this function. For more information, see Section 4.13, “Initialize”, on page 46 and Section 4.15, “QueryCallInfo”, on page 49.

Syntax

```
GetContactData (CallID, Key, >Value)
```

Parameters

You must enter the parameters in the order that they appear in the following table.

Name	Type	Range	Description
CallID	String passed by value	18 characters	The CallID for the call you want to get the contact data for.
Key	String passed by value	32 characters	The key name for the contact data value.

Table 10 Parameters for the GetContactData function

Name	Type	Range	Description
Value	String passed by reference	128 characters	The value returned for the key.

Table 10 Parameters for the GetContactData function

Return Codes

The most common codes returned by this function are as follows:

0	Successful.
–304	Initialize was not called before invoking this function. You should call Initialize at the start of your script.
–306	One or more of the parameters is either incorrect or is of the wrong type.
–307	The key name for the contact data value was not set prior to invoking this function.
–701	The request timed out before the function could be completed. It may be an indication of network problems or a busy server.
–801	Unable to connect to one of the OpenScape Contact Center servers. Either the server name is invalid or the server is not operational.
Other	Any other code indicates failure. For more information about a specific code, see Chapter 5, “Return codes”.

Example

The following example retrieves the contact data for the key “Name” for the specified CallID.

```
GetContactData (‘8271023217459002’, 'Name', >Value)
```

4.12 GetTransitNumber

NOTE: This function is used by IVR scripts for unmonitored IVR systems only.

The GetTransitNumber function queries OpenScape Contact Center for a transit number. The IVR system must transfer the call to the transit number received in this request. The transit number expires if the call is not transferred within 30 seconds. The transit number is then available for use with another call.

NOTE: For Queue Hold, this function is called after calling the Enqueue function. For IVR Hold, it is called after receiving Pending or Unanswered status in the QueryCallStatus function.

Syntax

```
GetTransitNumber(CallID, >TransitNumber)
```

Parameters

You must enter the parameters in the order that they appear in the following table.

Name	Type	Range	Description
CallID	String passed by value	18 characters	The CallID for the call you want to enqueue.
TransitNumbe r	String passed by reference	80 characters	A pilot number that the IVR will use to transfer the call.

Table 11 Parameters for the GetTransitNumber function

Return Codes

The most common codes returned by this function are as follows:

0	Successful.
–308	QueryCallInfo was not called prior to invoking this function. You must call QueryCallInfo to obtain the CallID.
–310	This function can only be used for unmonitored IVR calls.
–910	The version of the IVR API DLL does not correspond to the version of OpenScape Contact Center.
–927	All registered transit numbers are busy.

–929	No transit numbers have been configured.
–931	There is an inconsistency monitoring the type of call between the IVR API and the T-Server.
Othe r	Any other code indicates failure. For more information about a specific code, see Chapter 5, “Return codes”.

Example

```
GetTransitNumber (‘1341023126268002’, >TransitNumber)
```

4.13 Initialize

The Initialize function initializes the connection to the OpenScape Contact Center servers and must be called before any of the other OpenScape Contact Center IVR API functions are invoked. This function clears the ANI and DNIS that were set up during the previous call on this extension. Therefore, invoke this function at the beginning of your IVR script and ensure that OpenScape Contact Center is operational before routing calls. Do not invoke the Initialize function multiple times for the same call.

Syntax

```
Initialize (LocalPort, AdminServerName, RpcTimeout)
```

Parameters

You must enter the parameters in the order that they appear in the following table.

Name	Type	Range	Description
LocalPort	String passed by value	1 – 32767	The TCP/IP port number to be used by the IVR API. This must be a number that is not already in use.
AdminServer Name	String passed by value	128 characters	The network name of the OpenScape Contact Center Administration Server at the site on which you are working, for example, 6000@servername. If the system is configured for high availability (warm standby), you must specify the cluster name.
RpcTimeout	Integer passed by value	Greater than 0	The length of time in milliseconds between when the function makes a call and returns the call. When a user calls the function, the system expects a result within the time-out period or the call fails. Use a value of 10000 or greater.

Table 12 Parameters for the Initialize function

Return Codes

The most common codes returned by this function are as follows:

0	Successful.
–306	One or more of the parameters is either incorrect or is of the wrong type.
–910	The version of the IVR API DLL does not correspond to the version of OpenScape Contact Center.
Othe r	Any other code indicates failure. For more information about a specific code, see Chapter 5, "Return codes".

Example

The following example initializes a connection between the OpenScape Contact Center server called "deerhurst" and the IVR system. The IVR's extension number is `6060`. All the parameters are passed by value.

```
Initialize('6060', '6000@deerhurst', 10000)
```

4.14 QueryAgentStatus

The QueryAgentStatus function polls for the status of a user associated with a given device. This function will return the status of the user as an integer value depending on whether the user is logged on or not. You must specify either the agent ID or the device. If the agent ID is specified and the user is logged on, the device will be filled in and the status returned. If the device is specified and there is a user logged on to the specified device, the agent ID will be filled in and the status returned.

NOTE: You must call the Initialize function before invoking this function. For more information, see Section 4.13, "Initialize", on page 46.

Syntax

```
QueryAgentStatus (>AgentID, >Device, >AgentStatus)
```

Parameters

You must enter the parameters in the order that they appear in the following table.

Name	Type	Range	Description
AgentID	String passed by reference	8 characters	A string identifying the user; empty if the Device is provided.
Device	String passed by reference	16 characters	The user extension you want to query; empty if AgentID is provided.
AgentStatus	Integer passed by reference	Greater than 0	Indicates the status of a user. 1 - Dialing 2 - Line busy 3 - Ringing 4 - Talking 5 - Line queued 6 - Holding 7 - Consulting 8 - Out of service 9 - Available 10 - Unavailable 11 - Work 12 - Logged off 13 - Unknown 14 - Pending 15 - Processing 16 - Post-processing

Table 13 Parameters for the QueryAgentStatus function

Return Codes

The most common codes returned by this function are as follows:

0	Successful.
–904	The specified agent ID did not match the ID of any user in the database.
–909	The specified device did not match any device in the database.
Other	Any other code indicates failure. For more information about a specific code, see Chapter 5, “Return codes”.

Example

The following example polls for the status of a user associated with a given device and returns the user status associated with the specified device.

```
QueryAgentStatus(‘188001’, >Device, >AgentStatus)
```

4.15 QueryCallInfo

The QueryCallInfo function queries the T-Server for the CallID and the ANI and DNIS numbers associated with the current IVR extension and the call being processed. Both the ANI and DNIS numbers are returned as strings. Invoke this function when a call arrives at an IVR extension to obtain the ANI and DNIS information for the call.

NOTE: You must call the Initialize function before invoking this function. For more information, see Section 4.13, "Initialize", on page 46.

Syntax

```
QueryCallInfo (Device, >CallID, >originalANI, >originalDNIS,
>UUID, >Visits, >FMNQ, >InternalANI, >InternalDNIS,
TrunkConnectedIVRCall)
```

Parameters

You must enter the parameters in the order that they appear in the following table.

Name	Type	Range	Description
Device	String passed by value	16 characters	The IVR extension on which the IVR script is running.
CallID	String passed by reference	18 characters	The CallID for the current call.
originalANI	String passed by reference	80 characters	The ANI number associated with the call when it first enters the system.
originalDNIS	String passed by reference	80 characters	The DNIS number associated with the call when it first enters the system.
UUID (optional)	String passed by reference	32 characters	The UUID associated with the current call.
Visits (optional)	Integer passed by reference	Greater than or equal to 0	The number of times that this call has been answered by an OpenScape Contact Center monitored IVR.
FMNQ (optional)	Integer passed by reference	0 or 1	Indicates whether it is an FMNQ scenario. 0 - No 1 - Yes

Table 14 Parameters for the QueryCallInfo function

Name	Type	Range	Description
InternalANI (optional)	String passed by reference	80 characters	The number from which the call placed to the IVR ACD/UCD/MLHG group originated.
Internal DNIS (optional)	String passed by reference	80 characters	The pilot number associated with the IVR ACD/UCD/MLHG group.
TrunkConnect edIVRCall	String passed by value	Unlimited.	Indicates if the call is monitored or unmonitored. This parameter must be "1;" for an unmonitored IVR call. Any other value indicates a monitored IVR call.

Table 14 Parameters for the QueryCallInfo function

Return codes

The most common codes returned by this function are as follows:

0	Successful.
–304	Initialize was not called before invoking this function. You should call Initialize at the start of your script.
–306	One or more of the parameters is either incorrect or is of the wrong type.
–701	The request timed out before the function could be completed. It may be an indication of network problems or a busy server.
–801	Unable to connect to one of the OpenScape Contact Center servers. Either the server name is invalid or the server is not operational.

Example

The following example queries for the CallID and the ANI and DNIS numbers of the call currently connected to the IVR extension.

```
QueryCallInfo('36200', >CallID, >originalANI, >originalDNIS)
```

4.16 QueryCallStatus

The QueryCallStatus function polls for the status of the call associated with a specified CallID. The target to which the call should be transferred is returned by the Extension parameter based on the status of the call (Pending or Unanswered). If the status of the call is Disconnected, then the call should be disconnected by the IVR system (in the IVR Hold scenario).

After a call has been enqueued, regularly invoke the QueryCallStatus function to determine if the call has been assigned.

NOTE: If the IVR system is unmonitored, you can call the GetTransitNumber function to query OpenScape Contact Center for a transit number to which the call can be transferred. For more information, see Section 4.12, “GetTransitNumber”, on page 45.

NOTE: You must call the Initialize and QueryCallInfo functions before invoking this function. For more information, see Section 4.13, “Initialize”, on page 46 and Section 4.15, “QueryCallInfo”, on page 49.

Syntax

```
QueryCallStatus (CallID, Timeout, >Extension, >PositionInQueue)
```

Parameters

You must enter the parameters in the order that they appear in the following table.

Name	Type	Range	Description
CallID	String passed by value	18 characters	The CallID returned from QueryCallInfo.
Timeout	Integer passed by value	Greater than or equal to 0	The length of time (in milliseconds) spent waiting for a change in call status on the given device. Specifying zero (0) indicates an immediate poll of the current status.
Extension	String passed by reference	80 characters	The target to which the call should be transferred.
Position InQueue	Integer passed by reference	Greater than 0	The position of the call in the queue, as returned by the Routing Server.

Table 15 Parameters for the QueryCallStatus function

Return Codes

The most common codes returned by this function are as follows:

0	The call is idle.
1	The call is in Queued state. In this case, continue checking the call status.
2	The call is in Pending state. In this case, transfer the call to the returned extension.
3	The call is in Unanswered state. In this case, transfer the call to the returned extension or to another time-out extension.
4	An error has occurred. In this case, transfer the call to a non-OpenScape Contact Center extension.
5	The call must be disconnected.
6	The call must be transferred to the returned extension.
–304	Initialize was not called before invoking this function. You should call Initialize at the start of your script.
–306	One or more of the parameters is either incorrect or is of the wrong type.
–308	QueryCallInfo was not called prior to invoking this function. You must call QueryCallInfo to obtain the CallID.
–701	The request timed out before the function could be completed. It may be an indication of network problems or a busy server.
–801	Unable to connect to one of the OpenScape Contact Center servers. Either the server name is invalid or the server is not operational.
–903	There was no call associated with the specified CallID. Ensure that all IVR extensions are represented in the database.
–908	The T-Server is not available.
Other	Any other code indicates failure. For more information about a specific code, see Chapter 5, "Return codes".

NOTE: You can receive the –701 return code while calling the QueryCallStatus function for the following two reasons:
• If you specify a non-zero Timeout, the QueryCallStatus waits for the events to arrive. In this case, the –701 return code only indicates that you have waited the specified time without anything happening.
• If you specify a Timeout of 0 and the call is in an queued state, then the position in queue will be queried from the Routing Server. In this case, the –701 return code indicates that there are network problems or a busy server.

Example

The following example waits for a maximum of 60,000 milliseconds to monitor a change in the status. If the call changes to the pending state within that time period, the function returns the user extension to which the IVR system should transfer the call, using the TRANSFER_EXT variable.

```
QueryCallStatus('1341023126268002', 60000, >TRANSFER_EXT)
```

4.17 QueryQueueStatistics

The QueryQueueStatistics function determines the number of calls in queue, the estimated wait time for a call, the average wait time for the queue, the time that the oldest call has been in queue, and the service level for the specified queue.

These calculations assume the call will be enqueued to the Routing Server immediately, regardless of whether the call has been enqueued. You should invoke QueryQueueStatistics before invoking Enqueue to ensure an accurate response from the Routing Server.

NOTE: You must call the Initialize function before invoking this function. For more information, see Section 4.13, "Initialize", on page 46.

Syntax

```
QueryQueueStatistics (QueueName, >CallsInQueue, >EstimatedWait,
>AverageWait, >OldestCallinQueue, >ServiceLevel,
BusinessUnitName)
```

Parameters

You must enter the parameters in the order that they appear in the following table.

Name	Type	Range	Description
QueueName	String passed by value	32 characters	The queue for which the query is performed.
CallsInQueue	Integer passed by reference	Greater than or equal to 0	The number of calls in queue for the specified queue.
EstimatedWait	Integer passed by reference	Greater than or equal to 0	The estimated wait time in seconds for the specified queue.

Table 16 Parameters for the QueryQueueStatistics function

Name	Type	Range	Description
AverageWait	Integer passed by reference	Greater than or equal to 0	The average wait time in seconds for the specified queue.
OldestCall InQueue	Integer passed by reference	Greater than or equal to 0	The time in seconds that the oldest call has been in the queue.
ServiceLevel	Integer passed by reference	0 – 100	The service level for the specified queue.
BusinessUnit Name (optional)	String passed by value	32 characters	The name of the business unit for the specified queue. This parameter is required in a multitenant environment and ignored in a non-multitenant environment.

Table 16 Parameters for the QueryQueueStatistics function

Return Codes

The most common codes returned by this function are as follows:

0	Successful.
–304	Initialize was not called before invoking this function. You should call Initialize at the start of your script.
–306	One or more of the parameters is either incorrect or is of the wrong type.
–314	The specified business unit name was not set or it did not match the name of any business unit in the database. (This error is applicable only when OpenScape Contact Center is configured as a multitenant system.)
–701	The request timed out before the function could be completed. It may be an indication of network problems or a busy server.
–801	Unable to connect to one of the OpenScape Contact Center servers. Either the server name is invalid or the server is not operational.
–905	The specified queue did not match the name of any queue in the database.
Other	Any other code indicates failure. For more information about a specific code, see Chapter 5, “Return codes”.

Example

The following example queries for the statistics for the queue “Sales”.

```
QueryQueueStatistics(‘Sales’, >CallsInQueue, >EstimatedWait,
>AverageWait, >OldestCallinQueue, >ServiceLevel, ‘Toronto’)
```

4.18 QueryRoutingInfo

The QueryRoutingInfo function sends a request to determine routing information for a call. The query is based on information collected and submitted by the IVR system as part of an OpenScape Contact Center routing request.

You should enqueue, disconnect, or transfer the call based on the query result.

If a Transfer component is encountered in a workflow, then the code returned is 2. The transfer destination that is returned may be in canonical form, depending on how the Transfer component has been configured. In this case, if you want to transfer the call to this destination using only a hook-flash transfer, or other internal IVR transfer mechanism, then you must translate the number appropriately. If you want to transfer the call using the IVR API, then the parsing will be handled directly by OpenScape Contact Center.

NOTE: You must call the Initialize and QueryCallInfo functions before invoking this function. For more information, see Section 4.13, "Initialize", on page 46 and Section 4.15, "QueryCallInfo", on page 49.

NOTE: If you want the contact data values to be considered in the routing decision, you must set these values, using the SetContactData function, before invoking this function. To retain the original contact data values, you must copy and reset the values before invoking the Enqueue or EnqueueForAgent functions. For more information, see Section 4.24, "SetContactData", on page 65.

Syntax

```
QueryRoutingInfo (CallID, >Destination, >AgentID,
>AgentWaitTime, >Description, >InitialPriority, >EstimatedWait,
>CallsinQueue, ANIDNIS)
```

Parameters

You must enter the parameters in the order that they appear in the following table.

Name	Type	Range	Description
CallID	String passed by value	18 characters	The CallID of the call for which you want to obtain the appropriate routing information.

Table 17 Parameters for the QueryRoutingInfo function

Name	Type	Range	Description
Destination	String passed by reference	80 characters	Return code 0 – The queue to which the call should be enqueued. Return code 1 – The call should be disconnected. Return code 2 – The target to which the call should be transferred.
AgentID	String passed by reference	8 characters	Returns the ID of the user who should be reserved to handle the call. **Note:** Valid only if the return code is 0.
AgentWaitTime	Integer passed by reference	Greater than 0	Returns the maximum time in seconds the caller must wait for a reserved user before the call is released and goes to the specified queue. This is only valid if an AgentID is returned. **Note:** Valid only if the return code is 0.
Description	String passed by reference	100 characters	Provides a description of the call, which is displayed in the OpenScape Contact Center Client Desktop application so that users can identify the call. **Note:** Valid only if the return code is 0.
InitialPriority	Integer passed by reference	1 to 100	Returns the priority of the call relative to all the other calls queued in the Routing Server, including calls associated with the same queue as the current call. The default is 1. **Note:** Valid only if the return code is 0.
EstimatedWait	Integer passed by reference	Greater than or equal to 0	Returns the estimated wait time in seconds for the specified queue. **Note:** Valid only if the return code is 0.
CallsInQueue	Integer passed by reference	Greater than or equal to 0	Returns the number of calls for the specified queue. **Note:** Valid only if the return code is 0.

Table 17 Parameters for the QueryRoutingInfo function

Name	Type	Range	Description
ANIDNIS (optional)	String passed by value	161 characters	The ANI and DNIS numbers to be used in the routing decision. The ANI and DNIS must be separated by a semicolon. If contact data has already been associated with this call and contains ANI and DNIS values, these values take precedence over the values you specify here.

Table 17 Parameters for the QueryRoutingInfo function

Return Codes

The most common codes returned by this function are as follows:

0	Successful. Indicates that the call must be enqueued with the returned information.
1	Successful. Indicates that the call must be disconnected.
2	Successful. Indicates that the call must be transferred to the returned target.
–304	Initialize was not called before invoking this function. You should call Initialize at the start of your script.
–306	One or more of the parameters is either incorrect or is of the wrong type.
–400	The system has low system resources.
–801	Unable to connect to one of the OpenScape Contact Center servers. Either the server name is invalid or the server is not operational.
–915	Unable to determine the queue and other routing information.
–916	The Routing Server encountered an error in the workflow execution.
Other	Any other code indicates failure. For more information about a specific code, see Chapter 5, “Return codes”.

Example

The following example queries the Routing Server for routing information.

```
QueryRoutingInfo(‘1341023126268002’, >Destination, >AgentID,
>AgentWaitTime, >Description, >InitialPriority, >EstimatedWait,
>CallsInQueue, ‘9055557900;69030’)
```

4.19 QuerySystemStatus

The QuerySystemStatus function determines the status of the system. This function should be called to verify the system status at the beginning of the IVR script and before enqueuing a call.

NOTE: You must call the Initialize function before invoking this function. For more information, see Section 4.13, "Initialize", on page 46.

Syntax

```
QuerySystemStatus (>OverallStatus, >IndividualServerStates,
>Entries)
```

Parameters

You must enter the parameters in the order that they appear in the following table.

Name	Type	Range	Description
OverallStatus	Integer passed by reference	0 or 1	0 - Indicates the system is not operational 1 - Indicates the system is operational.

IndividualServerStates	String passed by reference	250 characters	A string containing each server name and its state. For example:
			server-name1=1;
			server-name2=2

Entries	Integer passed by reference	Greater than or equal to 0	The number of servers in the status string.

Table 18 Parameters for the QuerySystemStatus function

The IVR script should parse the individual server states to find the status of the Routing Server and T-Server. If the Routing Server state is not 1, or the T-Server state is not 8, the call should be transferred to an ACD/UCD/MLHG number that routes calls to a backup ACD/UCD/MLHG group.

Status values

The following codes indicate the status of the various servers:

0	The server is not operational.
1	The server is operational.
2	The server is inactive. This means that the Administration Server is operational, but it has not received an indication of the status for the specified server.

Only the T-Server uses the following codes:

3	The T-Server is not operational.
4	The T-Server is in the process of shutting down.
5	The T-Server is out of service because it cannot connect to the CSTA provider.
6	The T-Server is in the process of starting.
7	The T-Server is in the process of initializing.
8	The T-Server is operational.
9	The communication platform is overloaded and is having problems carrying out the T-Server's requests.
10	The T-Server is in the process of recovering.

Return Codes

The most common codes returned by this function are as follows:

0	Successful.
–304	Initialize was not called before invoking this function. You should call Initialize at the start of your script.
–306	One or more of the parameters is either incorrect or is of the wrong type.
–701	The request timed out before the function could be completed. It may be an indication of network problems or a busy server.
–801	Unable to connect to one of the OpenScape Contact Center servers. Either the server name is invalid or the server is not operational.
–901	There was no status information available. The Administration Server is not operational but the Routing Server is still available. In this case, the status of the other servers is unknown. This error is returned only by the QuerySystemStatus function.
Other	Any other code indicates failure. For more information about a specific code, see Chapter 5, "Return codes".

Example

The following example queries for the status of each of the servers. The function passes the S_OVERALLSTATUS and S_SERVER_STATES by reference to return the status of each of the servers. Also, I_ENTRIES returns the number of servers in the S_SERVER_STATES string.

```
QuerySystemStatus(>S_OVERALLSTATUS, >S_SERVER_STATES,
>I_ENTRIES)
```

4.20 ReleaseTransitNumber

NOTE: This function is used by IVR scripts for unmonitored IVR systems only.

The ReleaseTransitNumber function allows applications to request that OpenScape Contact Center release a transit number for use by another call. This function must be called before the transit number expires. This means that the transit number will become available for another OpenScape Contact Center call.

The IVR system may use the ReleaseTransitNumber function to inform the T-Server that the transit number is no longer needed. The application must call the GetTransitNumber function again to request a new transit number before attempting to transfer the call to OpenScape Contact Center.

Syntax

```
ReleaseTransitNumber (CallID)
```

Parameters

Name	Type	Range	Description
CallID	String passed by value	18 characters	The CallID for the current call.

Table 19 Parameters for the ReleaseTransitNumber function

Return Codes

The most common codes returned by this function are as follows:

0	Successful.
–308	QueryCallInfo was not called prior to invoking this function. You must call QueryCallInfo to obtain the CallID.
–310	This function can only be used for unmonitored IVR calls.
–910	The version of the IVR API DLL does not correspond to the version of OpenScape Contact Center.
–928	No transit number has been allocated for this CallID.
–931	There is an inconsistency monitoring the type of call between the IVR API and the T-Server.
Othe r	Any other code indicates failure. For more information about a specific code, see Chapter 5, “Return codes”.

Example

```
ReleaseTransitNumber (‘1341023126268002’)
```

4.21 SetBusinessUnit

NOTE: The SetBusinessUnit function is supported only in a multitenant environment.

The SetBusinessUnit function sets the business unit for the specified call.

NOTE: The SetBusinessUnit function can be called only once for each IVR call.

NOTE: You must call the Initialize and QueryCallInfo functions before invoking this function. For more information, see Section 4.13, “Initialize”, on page 46, and Section 4.15, “QueryCallInfo”, on page 49.

Syntax

```
SetBusinessUnit (CallID, BusinessUnitName)
```

Parameters

You must enter the parameters in the order that they appear in the following table.

Name	Type	Range	Description
CallID	String passed by value	18 characters	The CallID for the call that you want to set the business unit for.
BusinessUnit Name	String passed by value	32 characters	The name of the business unit for the specified call.

Table 20 Parameters for the SetBusinessUnit function

Return Codes

The most common codes returned by this function are as follows:

0	Successful.
–304	Initialize was not called before invoking this function. You should call Initialize at the start of your script.
–306	One or more of the parameters is either incorrect or is of the wrong type.
–308	QueryCallInfo was not called prior to invoking this function. You must call QueryCallInfo to obtain the CallID.

–313	The SetBusinessUnit function was already called for this call. (This error is applicable only when OpenScape Contact Center is configured as a multitenant system.)
–314	The specified business unit name was not set or it did not match the name of any business unit in the database. (This error is applicable only when OpenScape Contact Center is configured as a multitenant system.)
–400	The system has low system resources.
–801	Unable to connect to one of the OpenScape Contact Center servers. Either the server name is invalid or the server is not operational.
–910	The version of the IVR API DLL does not correspond to the version of OpenScape Contact Center.
–955	The multitenancy feature is not licensed.
Other	Any other code indicates failure. For more information about a specific code, see Chapter 5, “Return codes”.

Example

The following example sets the name of the business unit for the specified call.

```
SetBusinessUnit (‘8271023217459002’, ‘WirelessDept’)
```

4.22 SetCallInfo

NOTE: This function is used by IVR scripts for unmonitored IVR systems only.

The SetCallInfo function stores ANI and DNIS information for unmonitored IVR calls for transferring to OpenScape Contact Center. This API can only be called before QueryCallInfo and after Initialize.

Syntax

```
SetCallInfo(ANI, DNIS)
```

Parameters

You must enter the parameters in the order that they appear in the following table.

Name	Type	Range	Description
ANI	String passed by value	80 characters	The ANI of the call.

Table 21 Parameters for the SetCallInfo Function

Name	Type	Range	Description
DNIS	String passed by value	80 characters	The DNIS of the call.

Table 21 Parameters for the SetCallInfo Function

Return Codes

The most common codes returned by this function are as follows:

0	Successful.
–304	Initialize was not called before invoking this function. You should call Initialize at the start of your script.
–306	One or more of the parameters is either incorrect or is of the wrong type.
–312	Wrong sequence of operations (for example, SetCallInfo called after QueryCallInfo).
Other	Any other code indicates failure. For more information about a specific code, see Chapter 5, “Return codes”.

Example

```
SetCallInfo('9055551234','12345')
```

4.23 SetCallTransferable

NOTE: Before the IVR script disconnects or transfers an FMNQ call, you must invoke the SetCallTransferable (0) function and receive a successful return code. If you do not receive a successful return code, this indicates that the call is in the process of being transferred and you must wait for the process to complete and then invoke the SetCallTransferable (0) function again.

The SetCallTransferable function specifies whether the call can be transferred by OpenScape Contact Center to an assigned user.

NOTE: This function does not support unmonitored IVR calls.

NOTE: You must call the Initialize and QueryCallInfo functions before invoking this function. For more information, see Section 4.13, “Initialize”, on page 46 and Section 4.15, “QueryCallInfo”, on page 49.

Syntax

```
SetCallTransferable (CallID, FlagValue)
```

Parameters

You must enter the parameters in the order that they appear in the following table.

Name	Type	Range	Description
CallID	String passed by value	18 characters	The CallID of the call you want to transfer or disconnect.
FlagValue	Integer passed by value	0 or 1	Indicates whether a call can be transferred to an assigned user. If set to false (0), the system suspends matching a user to the call until the flag is set to true (1). If the call is transferred back to the queue by the IVR system, the flag is automatically reset to true (1).

Table 22 Parameters for the SetCallTransferable function

Return codes

The most common codes returned by this function are as follows:

0	Successful.

–304	Initialize was not called before invoking this function. You should call Initialize at the start of your script.
–306	One or more of the parameters is either incorrect or is of the wrong type.
–311	The function does not support unmonitored IVR calls.
–701	The request timed out before the function could be completed. It may be an indication of network problems or a busy server.
–801	Unable to connect to one of the OpenScape Contact Center servers. Either the server name is invalid or the server is not operational.
Other	Any other code indicates failure. For more information about a specific code, see Chapter 5, “Return codes”.

Example

The following example specifies that the call may be transferred by OpenScape Contact Center to an assigned user.

```
SetCallTransferable(‘0741023896297002’, 1)
```

4.24 SetContactData

The SetContactData function sets the contact data for a specified CallID and adds the key-value pair if it does not exist. If you want the contact data to be available for the call, you must set the contact data before enqueuing the call.

NOTE: You must call the Initialize and QueryCallInfo functions before invoking this function. For more information, see Section 4.13, “Initialize”, on page 46 and Section 4.15, “QueryCallInfo”, on page 49.

Syntax

```
SetContactData (CallID, Key, Value)
```

Parameters

You must enter the parameters in the order that they appear in the following table.

Name	Type	Range	Description
CallID	String passed by value	18 characters	The CallID for the call you want to set the contact data for.
Key	String passed by value	32 characters	The key name for the contact data value.
Value	String passed by value	128 characters	The value to be set.

Table 23 Parameters for the SetContactData function

Return Codes

The most common codes returned by this function are as follows:

0	Successful.
–304	Initialize was not called before invoking this function. You should call Initialize at the start of your script.
–306	One or more of the parameters is either incorrect or is of the wrong type.
–308	QueryCallInfo was not called prior to invoking this function. You must call QueryCallInfo to obtain the CallID.
–701	The request timed out before the function could be completed. It may be an indication of network problems or a busy server.
–801	Unable to connect to one of the OpenScape Contact Center servers. Either the server name is invalid or the server is not operational.
Other	Any other code indicates failure. For more information about a specific code, see Chapter 5, “Return codes”.

Example

The following example sets the contact data for a specified CallID and adds the key-value pair if it does not exist.

```
SetContactData(‘8271023217459002’, ‘Name’, ‘John Doe’, 0)
```

4.25 SetDisplay

The SetDisplay function sets the telephone display for the first answering user of the specified call.

NOTE: This function is only available on an OpenScape 4000 or HiPath 4000 communication platform. You must call the Initialize and QueryCallInfo functions before invoking this function. For more information, see Section 4.13, "Initialize", on page 46 and Section 4.15, "QueryCallInfo", on page 49.

Syntax

```
SetDisplay (CallID, DisplayInfo)
```

Parameters

You must enter the parameters in the order that they appear in the following table.

Name	Type	Range	Description
CallID	String passed by value	18 characters	The CallID of the call for which you want to set the display.
DisplayInfo	String passed by value	240 characters	The string that is displayed on the answering user device.

Table 24 Parameters for the SetDisplay function

Return Codes

The most common codes returned by this function are as follows:

0	Successful.
–304	Initialize was not called before invoking this function. You should call Initialize at the start of your script.
–306	One or more of the parameters is either incorrect or is of the wrong type.
–701	The request timed out before the function could be completed. It may be an indication of network problems or a busy server.
–801	Unable to connect to one of the OpenScape Contact Center servers. Either the server name is invalid or the server is not operational.
Othe r	Any other code indicates failure. For more information about a specific code, see Chapter 5, "Return codes".

Example

The following example sets the telephone display for the first answering user of the specified call to "IVR Call".

```
SetDisplay('1341023126268002', 'IVR Call')
```

4.26 Transfer

The Transfer function performs a transfer and indicates the success or failure of the transfer. This API will not wait for the target party to answer.

NOTE: This function does not support unmonitored IVR calls.

NOTE: You might want to use this method instead of the IVR hook flash transfer because this function works more quickly. You must call the Initialize and QueryCallInfo functions before invoking this function. For more information, see Section 4.13, "Initialize", on page 46 and Section 4.15, "QueryCallInfo", on page 49.

Syntax

```
Transfer (CallID, ToDevice)
```

Parameters

You must enter the parameters in the order that they appear in the following table.

Name	Type	Range	Description
CallID	String passed by value	18 characters	The CallID for the call you want to transfer.
ToDevice	String passed by value	80 characters	A string identifying the device to which the call is being transferred.

Table 25 Parameters for the Transfer function

Return codes

The most common codes returned by this function are as follows:

0	Successful.
–304	Initialize was not called before invoking this function. You should call Initialize at the start of your script.
–306	One or more of the parameters is either incorrect or is of the wrong type.
–311	The function does not support unmonitored IVR calls.

–701	The function request timed out before the function could be completed. This will normally occur only when querying the status of a call. This may also be an indication of network problems or a busy server.
–801	Unable to connect to one of the OpenScape Contact Center servers. Either the server name is invalid or the server is not operational.
–903	There was no call associated with the specified CallID. Ensure that all IVR extensions are represented in the database.
–909	The specified device did not match any device in the database.
–913	The function passed a “To Device” that was not valid.
–914	The operation you are attempting has failed.
–917	The telephone number cannot be translated correctly by TAPI.
Other	Any other code indicates failure. For more information about a specific code, see Chapter 5, “Return codes”.

Example

The following example transfers the call with the specified CallID to extension 5678.

```
Transfer(‘1341023126268002','5678’);
```

Telephone number format

The telephone number parameter must be entered in canonical format as follows:

+[CountryCode] Space [(AreaCode) Space] SubscriberNumber [--Extension]

The telephone number can also be a dialable address obtained from a communication platform or as a result of calling a TAPI function. The following strings are correct telephone numbers:

- +1 (555) 555-0199
- (555) 555-0199
- +1 555-0199
- 5555550199

5 Return codes

This chapter describes all the codes returned by the IVR functions. For more information on how these codes relate to the specific functions, see Chapter 4, “Using the OpenScape Contact Center IVR API functions”.

Code	Description
0	Successful. QueryRoutingInfo — Indicates that the call must be enqueued with the returned information.
1	QueryCallStatus — Indicates that the call is in Queued state. In this case, continue checking the call status. QueryRoutingInfo — Indicates that the call must be disconnected.
2	QueryCallStatus — Indicates that the call is in Pending state. In this case, transfer the call to the returned extension. QueryRoutingInfo — Indicates that the call must be transferred to the returned target.
3	QueryCallStatus — Indicates that the call is in Unanswered state. In this case, transfer the call to the returned extension or to another time-out extension.
4	QueryCallStatus — Indicates that an error has occurred. In this case, transfer the call to a non-OpenScape Contact Center extension.
5	QueryCallStatus — Indicates that the call must be disconnected.
6	QueryCallStatus — The call must be transferred to the returned extension.
–304	Initialize was not called before invoking this function. You should call Initialize at the start of your script.
–305	This code indicates an unknown error (for example, user error or the system is unstable).
–306	One or more of the parameters is either incorrect or is of the wrong type.
–307	The key name for the contact data value was not set prior to invoking this function.
–308	QueryCallInfo was not called prior to invoking this function. You must call QueryCallInfo to obtain the CallID.
–310	This function can only be used for unmonitored IVR calls.
–311	The function does not support unmonitored IVR calls.
–312	Wrong sequence of operations (for example, SetCallInfo called after QueryCallInfo).

Table 26 Return codes

Code	Description
–313	The SetBusinessUnit function was already called for this call. (This error is applicable only when OpenScape Contact Center is configured as a multitenant system.)
–314	The specified business unit name was not set or it did not match the name of any business unit in the database. (This error is applicable only when OpenScape Contact Center is configured as a multitenant system.)
–400	The system has low system resources.
–701	The request timed out before the function could be completed. It may be an indication of network problems or a busy server.
–801	Unable to connect to one of the OpenScape Contact Center servers. Either the server name is invalid or the server is not operational.
–901	There was no status information available. The Administration Server is not operational but the Routing Server is still available. In this case, the status of the other servers is unknown. This error is returned only by the QuerySystemStatus function.
–903	There was no call associated with the specified CallID. Ensure that all IVR extensions are represented in the database.
–904	The specified agent ID did not match the ID of any user in the database.
–905	The specified queue did not match the name of any queue in the database.
–907	The T-Server is not available. This error means the Routing Server could not enqueue the call because the T-Server was not available.
–**908**	The T-Server is not available.
–**909**	The specified device did not match any device in the database.
–**910**	The version of the IVR API DLL does not correspond to the version of OpenScape Contact Center.
–**911**	The function contained data that became corrupted for an unknown reason. The function was therefore unable to connect with the Routing Server. The function was unsuccessful.
–913	The function passed a “To Device” that was not valid.
–914	The operation you are attempting has failed.
–915	Unable to determine the queue and other routing information.
–916	The Routing Server encountered an error in the flow execution.
–917	The telephone number cannot be translated correctly by TAPI.
–927	All registered transit numbers are busy.
–928	No transit number has been allocated for this CallID.
–929	No transit numbers have been configured.
–931	There is an inconsistency monitoring the type of call between the IVR API and the T-Server.
–955	The multitenancy feature is not licensed.

Table 26 Return codes

Code	**Description**
–1006	You are attempting to create a duplicate callback.
–1021	Invalid CallbackID.
–1028	A schedule time is invalid.
–1029	The callback schedules have no mutual time with the contact center open hours.
–1031	A telephone number in the request is in the excluded numbers list.
–1033	A callback cannot accept contact data longer than 1000 bytes.
–1040	All schedules have already expired.
–1045	The callback queue name is invalid.
–1047	The schedule contains a date that is too far in the future.
Other	Any other negative code indicates failure. If another code is returned: ● Ensure that all site, queue, and user names (or ID numbers) in the IVR script match the names (or ID numbers) in the production copy of the OpenScape Contact Center database. ● Ensure that all communication platform resources are properly entered into the production copy of the OpenScape Contact Center database. ● Contact your next level of support.

Table 26 Return codes

Index

mitel.com