Oracle® Communications Interactive Session Recorder

API Reference Guide

Release 5.1

Formerly Net-Net Interactive Session Recorder

December 2015

ORACLE®

Copyright ©2015, 2004, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

About this Guide

Overview

The *Interactive Session Recorder API Reference Guide* provides information about:

- Methods for Recording
- Invoking API Commands
- REST and VoiceXML Commands
- Recording File Types/Formats Supported
- Return Codes
- Troubleshooting

Related Documentation

The following table lists related documents.

Document Name	Document Description
Interactive Session Recorder Release Notes	Contains information about new ISR features and fixed issues in each release of the ISR.
Interactive Session Recorder Installation Guide	Provides an overview of the ISR, hardware/software requirements and recommendations, storage considerations, pre-installation information, CIS and RSS installation procedures, post-install verification and configuration procedures, setting up and making a test call, and additional advanced topics about the ISR.
Interactive Session Recorder User's Guide	Contains information about using the ISR Dashboard for all levels of users. Provides information about viewing, playing, deleting recordings, running reports, managing user profiles (Super User, Account Administrator, and Tenant Administrator only).
Interactive Session Recorder Administrator Guide	Contains information about using the ISR Dashboard for the Administrator level user (Super User, Account Administrator, and Tenant Administrator). Provides information about creating and managing accounts, routes, and users. Also provides information about configuring the ISR, running reports, and viewing active calls.
Interactive Session Recorder Monitoring Guide	Contains information about installing and configuring the ISR Monitor. It also includes the Monitor database schema as well as the Monitor MIB.
Interactive Session Recorder Remote Archival Web Services Reference Guide	Contains information about the Remote Archival Web Service, its Control methods, WSDL definitions, DataType Definitions, sample responses to requests, and importing the Remote Archival Web Service's certificate into the client keystore.

Revision History

This section contains the revision history for this document.

Date	Revision Number	Description
July 31, 2012	Revision 1.00	• Initial release of the NN-ISR 5.1 software.
May 1, 2015	Revision 1.10	• Adds the **bookmark** REST API command.
December 2015	Revision 1.11	• Adds the Interactive Session Recorder Remote Archival Web Services Reference Guide to the list of Related Documentation.

1 About the NN-ISR

Introduction

This chapter introduces some basic concepts that apply to the key features and abilities of your NN-ISR. It is necessary that you understand the information included in this chapter to comprehend the ways to configure your NN-ISR. This chapter only provides a high level overview of some important NN-ISR concepts. Please refer to each chapter for more information about using the NN-ISR.

Methods for Recording

The NN-ISR sits on the Voice over Internet Protocol (VoIP) network waiting to accept connections for recording, via a Session Initiation Protocol (SIP) INVITE. The NN-ISR may act as a back-to-back User Agent (B2BUA) or a User Agent Server (UAS), depending on the call type.

Note: An INVITE is a SIP request message used to establish a media session between user agents. SIP INVITES are usually generated by the User Agent Client and bound for the User Agent Server.

The NN-ISR Offers two methods to invoke recording:

- Pass-through Mode
- Conference Mode

Also, to support the IETF draft of the SIP Session Recording (SIPREC) Protocol, the NN-ISR may act as a Session Recording Server (SRS) in a SIPREC environment.

Pass-through Mode

The NN-ISR operates between the incoming Public Service Telephone Network (PSTN)/SIP Gateway and destination SIP Gateway/PBX and handles incoming SIP INVITEs by forwarding them on to destination SIP Gateway/PBX for routing. After these connections are made, the NN-ISR "listens" to passing Real-Time Transport Protocol (RTP) traffic, and is able to record the audio going in both directions.

Features of the NN-ISR in pass-through mode include:

- Answering the call, applying customizable recording rules, and connecting to the SBC, SIP Gateway/PBX, or other SIP compatible device.
- Providing the most control over recording (start, stop, etc.) through three (3) primary means:
  – *Percentage Based Route Recording* - Recordings are done on a percentage (1-100%) basis configured in the route.
  – *API Controlled Recording* - Using the NN-ISR API and authorized application, controls recording as desired.
  – *Record & Save* - 100% recording on route; however, recording is discarded unless caller or callee press designated DTMF to mark recording to be saved.

Conference Mode

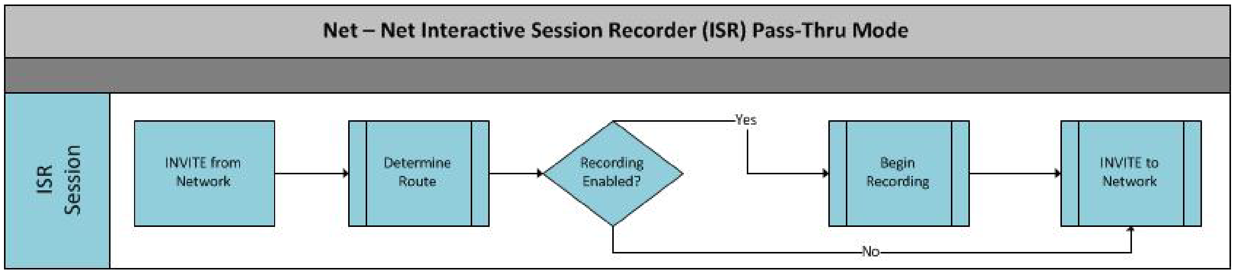
The NN-ISR operates as a SIP UAS. It DOES NOT reside between the incoming PTSN/SIP Gateway and the IVR software. Once invited in this mode, the NN-ISR "listens" to the call audio bridged to it, and is available for recording at any time.

Features of the NN-ISR in conference mode include:

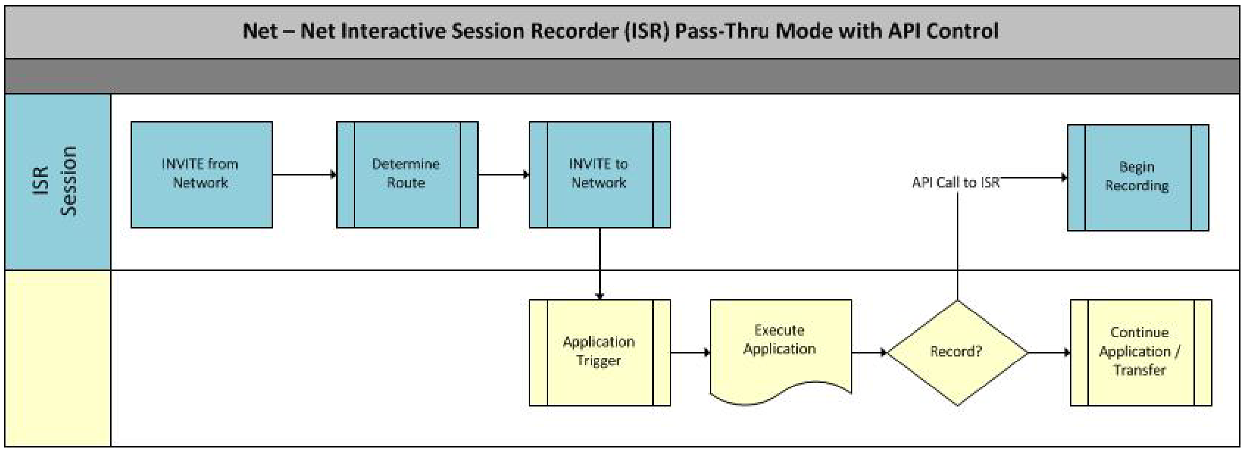
- May be invited to a call in conference for recording.
- Can be used in either SIP or TDM deployments:
  - *SIP* - SIP header information determines recording details.
  - *TDM* - Token-based recording with VXML API using SIP URIs
- Can be used for a conferenced IP extension off an IP Private Branch Exchange (PBX).
- Percentage Based Route Recording - Recording can be done on a percentage (1-100%) basis configured in the route.
- Recordings may be created through bridged transfer or route configuration in VoIP Gateway, SIP Proxy, or IP PBX.

Example Call Flows

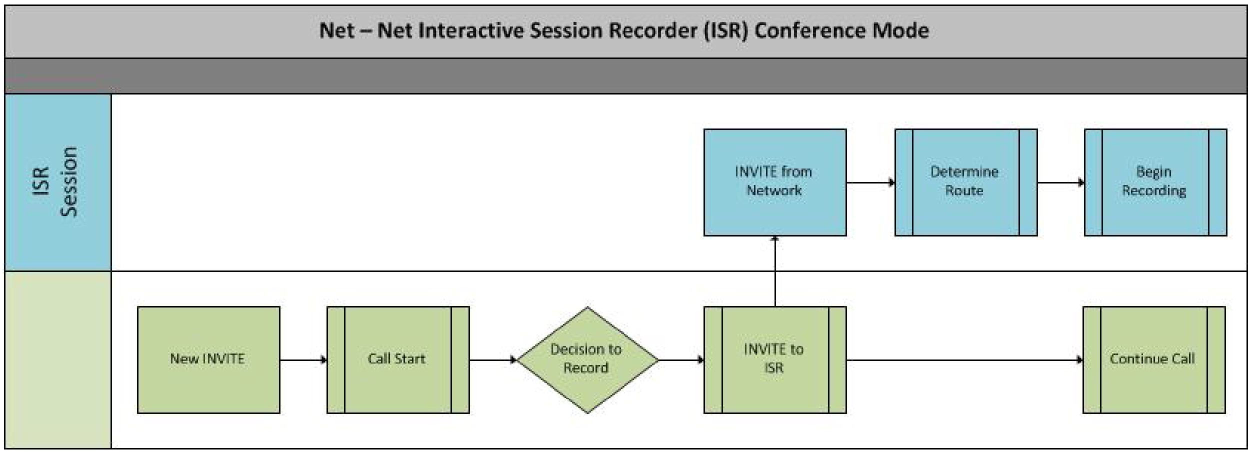
The following illustration shows the Pass-Through Mode with Percentage Recording.



The following illustration shows the Pass-Through Mode with API-Based Recording.



The following illustration shows the Conference Mode (100% Recording).



2 API Commands

Introduction

This chapter provides information about invoking and implementing NN-ISR API commands into your applications. You can control the NN-ISR using a policy defined on the NN-ISR or controlled by 3rd party applications with the ability to invoke a Representational State Transfer (REST) style Web Service. Each command is accompanied by sample code snippets.

REST is an architectural style for designing networked applications such as in large-scale software designs (i.e., World Wide Web). It uses HTTP to make calls between machines. REST emphasizes scalability of component interactions, generality of interfaces, independent deployment of components, and intermediary components to reduce interaction latency, enforce security, and encapsulate legacy systems.

This chapter includes information about:

- API Commands
- Recording File Types/Formats Supported
- Return Codes

REST Commands

The REST application program interface (API) provides an easy-to-use method to control session recording through the NN-ISR. It is a REST style service that works over HTTP. The commands are handled as resources.

Available Resources

Name	Value	Description
resource	<command to run>	recordStart, recordEnd, getFile, getFileInfo, updateFileInfo, deleteRecording, deleteFile, recordPause, recordUnpause, getRoutesForAccount, getRoutesForRealm

To use a REST command, send an HTTP request in the form:

```
http://<host:port>/IsrApi/rest/<resource>?<param1>=<value1>[&<paramN>=<va
lueN>]
```

API Commands

There are specific API commands you can implement to control recording on the NN-ISR. The API must reference sessions that are active on the NN-ISR.

The following table identifies each of these commands and provides a more in-depth description of each command in the following paragraphs.

You can implement the following commands to control recording on the NN-ISR:

API Command	Description
recordStart	Causes recording of both Real-time Transport Protocol (RTP) streams to start.
recordEnd	Causes recording of both RTP streams to end.
getFile	Downloads the specified recording file, which can then be saved or opened.
getFileInfo	Retrieves recorded file information including file_uri and recording duration from recorder database.
updateFileInfo	Updates recorded file information for the recording matching ANI or token, DNIS, and timeStamp.
deleteRecording	Deletes a recording and reduces the session statistics for count of recorded sessions and duration of recording. This is typically used in Record and Save Routes.
deleteFile	Deletes recording but does not alter the count of recorded calls or duration of recording in the session statistics.
recordPause	Pauses a recording in progress.
recordUnpause	Resumes a paused recording.
getRoutesForAccount	Returns a list of routes for the specified account.
getRoutesForRealm	Returns a list of routes for the specified realm.
bookmark	Inserts a cue point in a recording file (only supported if using WAV file recording types). For more information on recording types, see Recording File Types/Formats Supported.

Note: All commands are available in both REST and VXML implementations *except* getRoutesForAccount and getRoutesForRealm. These are available in REST only.

Determining the UCID

The Unique Call Identifier (UCID) can be one of several values associated with the session: the isr_ucid, ingress_callid, or egress_callid. For the isr_ucid, the RSS pulls the SIP Header matching the configured X-ISR-UCID header name from the SIP Headers, if it is present. For a SIPREC session, if the <*:ucid> field is present in the extension data, the parameter value will be used as the isr_ucid. In the case where both extension data and X-ISR-UCID SIP Headers are present, X-ISR-UCID SIP Header is used as the isr_ucid. The call-id from the SIP Headers of the initial INVITE are used for the ingress_callid. For a pass-through (or Record & Save) configuration, a second call leg (egress) is initiated by the RSS (acting as a B2BUA). The SIP call-id for this new call leg is the egress_callid.

The ucidSource parameter, which can optionally be used in conjunction with the ucid, denotes which field to match the ucid parameter against. The case-insensitive options are:

- Ingress_callid
- Egress_callid
- ISR_UCID

If the ucidSource parameter is not present, or has no value (""), all three fields will be matched against in the order: isr_ucid, ingress_callid, followed by egress_callid.

If none of the UCID values are known, the session can be determined using the TO/FROM values, session start time and RSS's SIP From URI, as long as it contains the RSS IP, channel ID, and call-id. Here is an example of the SIP From URI:

```
From: <sip:6003@172.30.58.111:5060>;channel=channel2;call-id=43-
0ea84426b88e01901b4de116d700;tag=3743477352
```

Alternatively, you can identify sessions by just the user portions of the TO/FROM values and session start time. This method is not normally recommended, as it utilizes a best match only to identify the intended session and it cannot be guaranteed which session will be selected in cases where multiple sessions match the provided criteria.

The following sections provide the parameters and values you can use with the API commands, including sample code.

recordStart

The **recordStart** command starts recording and creates an index entry for recording.

Required Input Parameters

Standard Implementation

Parameter	Value	Description
ucid	<UCID>	UCID of the session.
fileId	<Filename>	Name of the file to save the recording. NOTE: File name provided is appended to the Recording Directory specified in the NN-ISR Dashboard.

OR

Alternate Implementation

Parameter	Value	Description
sipFromUri	<SIP From URI>	The SIP FROM header, including RSS IP, channel ID, and call-id.
ani	<FROM>	Incoming FROM value or Token (see the command **getToken**).
dnis	<TO>	TO value associated with the call.

Parameter	Value	Description
timeStamp	<time>	Call Start time (in format YYYY-MM-DD HH:MM:SS)
fileId	<Filename>	Name of the file to save the recording. NOTE: File name provided is appended to the Recording Directory specified in the NN-ISR Dashboard.

OR

Additional Alternate Implementation (Use only if directed)

Parameter	Value	Description
ani	<FROM>	Incoming FROM value or Token (see the command **getToken**).
dnis	<TO>	TO value associated with the call.
timeStamp	<time>	Call Start time (in format YYYY-MM-DD HH:MM:SS)
fileId	<Filename>	Name of the file to save the recording. NOTE: File name provided is appended to the Recording Directory specified in the NN-ISR Dashboard.

Optional Input Parameters

Parameter	Value	Description
ucidSource	<source>	Indicates which ucid field to check: ISR_UCID, Ingress_callid, or Egress_callid.
waitTime	**Range**: 0 to 32000 **Default**: 0	Number of milliseconds to wait before executing this command.
fileType	**Range**: 2 - 14 **Default**: 8 (uLaw 8Bit /8Khz mono	Suggested recording format, codec negotiation may require recording to be in a different format. See Recording File Types/Formats Supported for the valid values for this parameter.
<customDataField1>	Value up to 255 characters	Data to be associated with the recording in the call recording database in custom field 1, identified by Route's Custom Data Field 1 Label*.
<customDataField2>	Value up to 255 characters	Data to be associated with the recording in the call recording database in custom field 2, identified by Route's Custom Data Field 2 Label*.
<customDataField3>	Value up to 255 characters	Data to be associated with the recording in the call recording database in custom field 3, identified by Route's Custom Data Field 3 Label*.

Parameter	Value	Description
<customDataField4>	Value up to 255 characters	Data to be associated with the recording in the call recording database in custom field 4, identified by Route's Custom Data Field 4 Label*.
agentId	<ID value of agent>	ID of the agent to which the call was transferred.
beep	**Range**: enabled disabled **Default**: disabled	Specify whether to play a beep tone during call recording. If beep is enabled, the following 3 parameters should be supplied.
beepFile	**Range**: <URL> **Default**: none (must be specified)	URL of the beep audio file to play.
beepDirection	**Range**: ToCaller ToThirdParty ToBoth **Default**: ToBoth (if beep=enabled)	Specify which leg of the call should hear the beep tone.
beepInterval	**Range**: <number of seconds> **Default**: 30 (if beep=enabled)	Frequency (in seconds) that the beep audio file should be played.

*The name of the parameter in the recordStart namelist should match the value of the Custom Data Field Label defined in the Admin Dashboard for the matching Route in order for the data to be properly associated.

Return Values

Parameter	Value	Description
code	-1 | 0 - 27	Numbered result code. Zero (0) indicates success. All other responses indicate command failure.
message	<Status Text>	See Return Codes for the valid values for this parameter.

recordStart Sample Implementation

The following is an example of how to invoke the **recordStart** command via the NN-ISR REST API.

Request:

```
http://172.30.58.130:9000/IsrApi/rest/recordStart?fileId=Test_Recordi
ng_2012-01-02_030405.pcm&ucidSource=isr_ucid&ucid=sdfIYASDs9d8f7IYfds
```

Response:

```
<response>
  <Code>
    0
  </Code>
  <message>
    RecordStart Command Returned Successfully
    </message>
  </response>
```

recordEnd

The **recordEnd** command stops recording and updates the index with any optional parameters that were passed in. You may use the **getFileInfo** command to retrieve the fileId of the recording if the recording was initiated automatically by the NN-ISR (based on route configuration.)

Required Input Parameters

Standard Implementation

Parameter	Value	Description
ucid	<UCID>	UCID of the session.
fileID	<Filename>	Name of the recording file.

OR

Alternate Implementation

Parameter	Value	Description
sipFromUri	<SIP From URI>	The SIP FROM header, including RSS IP, channel ID, and call-id.
ani	<FROM>	Incoming FROM value or Token (see the command **getToken**).
dnis	<TO>	TO value associated with the call.
timeStamp	<time>	Call Start time (in format YYYY-MM-DD HH:MM:SS)
fileId	<Filename>	Name of the recording file. NOTE: File name provided is appended to the Recording Directory specified in the NN-ISR Dashboard.

OR

Additional Alternate Implementation (Use only if directed)

Parameter	Value	Description
ani	<FROM>	Incoming FROM value or Token (see the command **getToken**).
dnis	<TO>	TO value associated with the call.
timeStamp	<time>	Call Start time (in format YYYY-MM-DD HH:MM:SS)
fileId	<Filename>	Name of the file to save the recording. NOTE: File name provided is appended to the Recording Directory specified in the NN-ISR Dashboard.

Optional Input Parameters

Parameter	Value	Description
ucidSource	<source>	Indicates which ucid field to check: ISR_UCID, Ingress_callid, or Egress_callid.
waitTime	**Range**: 0 to 32000 **Default**: 0	Number of milliseconds to wait before executing this command.
<customDataField1>	Value up to 255 characters	Data to be associated with the recording in the call recording database in custom field 1, identified by Route's Custom Data Field 1 Label*.
<customDataField2>	Value up to 255 characters	Data to be associated with the recording in the call recording database in custom field 2, identified by Route's Custom Data Field 2 Label*.
<customDataField3>	Value up to 255 characters	Data to be associated with the recording in the call recording database in custom field 3, identified by Route's Custom Data Field 3 Label*.
<customDataField4>	Value up to 255 characters	Data to be associated with the recording in the call recording database in custom field 4, identified by Route's Custom Data Field 4 Label*.
agentId	<ID value of agent>	ID of the agent to which the call was transferred.

*The name of the parameter in the recordEnd namelist should match the value of the Custom Data Field Label defined in the NN-ISR Admin Dashboard for the matching Route in order for the data to be properly associated.

Return Values

Parameter	Value	Description
code	-1 | 0 - 27	Numbered result code. Zero (0) indicates success. All other responses indicate command failure.
message	<Status Text>	See Return Codes for the valid values for this parameter.
duration	<Numeric character>	Length, in milliseconds, of recording.

recordEnd Sample Implementation

The following is an example of how to invoke the **recordEnd** command via the NN-ISR REST API.

Request:

```
http://172.30.58.130:9000/IsrApi/rest/recordEnd?fileId=Test_Recording
_2012-01-02_030405.pcm&ucidSource=isr_ucid&ucid=sdfIYASDs9d8f7IYfds
```

Response:

```
<response>
  <Code>
    0
  </Code>
  <message>
```

```
      RecordEnd Command Returned Successfully
    </message>
    <duration>
      23000
    </duration>
  </response>
```

getFile

The **GetFile** command downloads the specified recording file, which can then be saved or opened. You can use the **GetFileInfo** command to retrieve the fileId of the recording if the recording was initiated automatically by the NN-ISR (based on route configuration).

Required Input Parameters

Standard Implementation

Parameter	Value	Description
ucid	<UCID>	UCID of the session.

OR

Alternate Implementation

Parameter	Value	Description
fileId	<Filename>	Name of the recording file to be retrieved.

OR

Additional Alternate Implementation (Use only if directed)

Parameter	Value	Description
ani or token	<FROM>	Incoming FROM value or Token (see the command **getToken**).
dnis	<TO>	TO value associated with the call.
timeStamp	<time>	Call Start time (in format YYYY-MM-DD HH:MM:SS)

Optional Input Parameters

Parameter	Value	Description
ucidSource	<source>	Indicates which ucid field to check: ISR_UCID, Ingress_callid, or Egress_callid.
redirect	**Range**: <true | false> **Default:** false	When **true**, when the file is requested, the requester is redirected to the file URI using the HTTP 302 redirection status code, rather than having the file streamed through the API.

Return Values

Parameter	Value	Description
code	-1 | 0 - 27	Numbered result code. Zero (0) indicates success. All other responses indicate command failure.
message	<Status Text>	See Return Codes for the valid values for this parameter.

GetFile sample Implementation - VXML

The following is an example of how to invoke the command.

The following is an example of how to invoke the **GetFile** command via the NN-ISR REST API.

Request:

```
http://172.30.58.130:9000/IsrApi/rest/getFile?ucidSource=isr_ucid&uci
d=sdfIYASDs9d8f7IYfds
```

Response:

The file.

getFileInfo

Query the NN-ISR database for information about a recording.

Required Input Parameters

Standard Implementation

Parameter	Value	Description
ucid	<UCID>	UCID of the session.

OR

Alternate Implementation

Parameter	Value	Description
fileId	<Filename>	Name of the recording file for which the associated information is requested.
ani	<FROM>	Incoming FROM value or Token (see the command **getToken**).
dnis	<TO>	TO value associated with the call.
timeStamp	<time>	Call Start time (in format YYYY-MM-DD HH:MM:SS)

OR

Additional Alternate Implementation (Use only if directed)

Parameter	Value	Description
ani	<FROM>	Incoming FROM value or Token (see the command **getToken**).
dnis	<TO>	TO value associated with the call.
timeStamp	<time>	Call Start time (in format YYYY-MM-DD HH:MM:SS)

Optional Input Parameters

Parameter	Value	Description
ucidSource	<source>	Indicates which ucid field to check: ISR_UCID, Ingress_callid, or Egress_callid.
inProgress	<true | false>	When true, a search is performed for currently recording files only and when false, a search is performed for completed recordings only. By default this value is false.

Return Values

Parameter	Value	Description
code	-1 | 0 - 27	Numbered result code. Zero (0) indicates success. All other responses indicate command failure.
message	<Status Text>	See Return Codes for the valid values for this parameter.
file_uri	<URI>	URI of the recording file, which can be called for playback or parsed to get the fileId to be used in other API commands.
duration	<Numeric character>	Length, in milliseconds, or recording.

getFileInfo Sample Implementation

The following is an example of using the **getFileInfo** REST API commands.

Request:

```
http://172.30.58.130:9000/IsrApi/rest/getFileInfo?ucidSource=isr_ucid
&ucid=sdfIYASDs9d8f7IYfds
```

Response:

```
<response>
  <Code>
    0
  </Code>
  <message>
    GetFileInfo Command Returned Successfully
  </message>
  <fileUri>
    http://172.30.58.130:9000/Recordings/Test_Recording_2012-01-
02_030405.pcm
  </fileUri>
  <duration>
    23000
  </duration>
</response>
```

updateFileInfo

Attach custom data to a recording file. Use this command to change the From or To associated with a recording, or to update the agent ID or custom data. The filename can also be changed, but not while the recording is in process.

Required Input Parameters

Standard Implementation

Parameter	Value	Description
ucid	<UCID>	The UCID of the session.

OR

Alternate Implementation

Parameter	Value	Description
oldFileId	<Filename>	Name of the recording file for which the associated information is to be updated.
ani	<FROM>	Incoming FROM value or Token (see the command **getToken**).
dnis	<TO>	TO value associated with the call.
timeStamp	<time>	Call Start time (in format YYYY-MM-DD HH:MM:SS)

OR

Additional Alternate Implementation

Parameter	Value	Description
oldFileId	<Filename>	Name of the recording file for which the associated information is to be updated.

OR

Additional Alternate Implementation (Use only if directed)

Parameter	Value	Description
ani	<FROM>	Incoming FROM value or Token (see the command **getToken**).
dnis	<TO>	TO value associated with the call.
timeStamp	<time>	Call Start time (in format YYYY-MM-DD HH:MM:SS)

Optional Input Parameters

Parameter	Value	Description
ucidSource	<source>	Indicates which ucid field to check: ISR_UCID, Ingress_callid, or Egress_callid.
newfileId	<Filename>	New filename to be associated with the recording.
<customDataField1>	Value up to 255 characters	Data to be associated with the recording in the call recording database in custom field 1, identified by Route's Custom Data Field 1 Label*.
<customDataField2>	Value up to 255 characters	Data to be associated with the recording in the call recording database in custom field 2, identified by Route's Custom Data Field 2 Label*.
<customDataField3>	Value up to 255 characters	Data to be associated with the recording in the call recording database in custom field 3, identified by Route's Custom Data Field 3 Label*.
<customDataField4>	Value up to 255 characters	Data to be associated with the recording in the call recording database in custom field 4, identified by Route's Custom Data Field 4 Label*.
agentId	<ID value of agent>	ID of the agent to which the call was transferred.
newAni	<FROM>	FROM value to be associated with the recording in the call recording database.
newDnis	<TO>	TO value to be associated with the recording in the call recording database.
liveCall	**Range**: true false **Default**: false	When the recording you are updating is currently in progress, set this value to **true**. When **false**, both in progress and completed recordings are checked.

*The name of the parameter in the updateFileInfo namelist should match the value of the Custom Data Field Label defined in the Admin Dashboard for the matching Route in order for the data to be properly associated.

Return Values

Parameter	Value	Description
code	-1 | 0 - 27	Numbered result code. Zero (0) indicates success. All other responses indicate command failure.
message	<Status Text>	See Return Codes for the valid values for this parameter.

updateFileInfo Sample Implementation

The following is an example of using the **updateFileInfo** REST API commands.

Request:

```
http://172.30.58.130:9000/IsrApi/rest/updateFileInfo?newFileId=Rename
d_Recording_7347.pcm&agentId=11&ucidSource=isr_ucid&ucid=sdfIYASDs9d8
f7IYfds
```

Response:

```
<response>
<Code>
0
</Code>
<message>
UpdateFileInfo Command Returned Successfully
</message>
</response>
```

deleteFile

This is the standard command used to delete a recording. You may use the getFileInfo command to retrieve the fileId of the recording if the recording was initiated automatically by the NN-ISR (based on route configuration).

Required Input Parameters

Standard Implementation

Parameter	Value	Description
ucid	<UCID>	The UCID of the session.

OR

Alternate Implementation

Parameter	Value	Description
fileId	<Filename>	Name of the recording file to be deleted.
ani	<FROM>	Incoming FROM value or Token (see the command **getToken**).
dnis	<TO>	TO value associated with the call.
timeStamp	<time>	Call Start time (in format YYYY-MM-DD HH:MM:SS)

OR

Additional Alternate Implementation

Parameter	Value	Description
fileId	<Filename>	Name of the recording file to be deleted.

OR

Additional Alternate Implementation (Use only if directed)

Parameter	Value	Description
ani	<FROM>	Incoming FROM value or Token (see the command **getToken**).
dnis	<TO>	TO value associated with the call.
timeStamp	<time>	Call Start time (in format YYYY-MM-DD HH:MM:SS)

Optional Input Parameters

Parameter	Value	Description
ucidSource	<source>	Indicates which ucid field to check: ISR_UCID, Ingress_callid, or Egress_callid.

Return Values

Parameter	Value	Description
code	-1 | 0 - 27	Numbered result code. Zero (0) indicates success. All other responses indicate command failure.
message	<Status Text>	See Return Codes for the valid values for this parameter.

deleteFile Sample Implementation

The following is an example of using the **deleteFile** REST API command.

Request:

```
http://172.30.58.130:9000/IsrApi/rest/deleteFile?ucidSource=isr_ucid&
ucid=sdfIYASDs9d8f7IYfds
```

Response:

```
http://172.30.58.130:9000/IsrApi/rest/deleteFile?ucidSource=isr_ucid&
ucid=sdfIYASDs9d8f7IYfds
```

deleteRecording

This is a specialized method for deleting a recording. It removes the recording from the call statistics associated with the session, such as whether the session was recorded and the duration of the recording. It is primarily used for the Record Message implementation to allow users to listen to and rerecord messages as needed. You may use the getFileInfo command to retrieve the fileId of the recording if recording was initiated automatically by the NN-ISR (based on route configuration).

Required Input Parameters

Standard Implementation

Parameter	Value	Description
ucid	<UCID>	The UCID of the session.

OR

Alternate Implementation

Parameter	Value	Description
fileId	<Filename>	Name of the recording file to be deleted.
ani	<FROM>	Incoming FROM value or Token (see the command GetToken).

Parameter	Value	Description
dnis	<TO>	TO value associated with the call.
timeStamp	<time>	Call Start time (in format YYYY-MM-DD HH:MM:SS)

OR

Additional Alternate Implementation

Parameter	Value	Description
fileId	<Filename>	Name of the recording file to be deleted.

OR

Additional Alternate Implementation (Use only if directed)

Parameter	Value	Description
ani	<FROM>	Incoming FROM value or Token (see the command GetToken).
dnis	<TO>	TO value associated with the call.
timeStamp	<time>	Call Start time (in format YYYY-MM-DD HH:MM:SS)

Optional Input Parameters

Parameter	Value	Description
ucidSource	<source>	Indicates which ucid field to check: ISR_UCID, Ingress_callid, or Egress_callid.

Return Values

Parameter	Value	Description
code	-1 | 0 - 27	Numbered result code. Zero (0) indicates success. All other responses indicate command failure.
message	<Status Text>	See Return Codes for the valid values for this parameter.

deleteRecording Sample Implementation

The following is an example of using the **deleteRecording** command.

Request:

```
http://172.30.58.130:9000/IsrApi/rest/deleteRecording?ucidSource=isr_
ucid&ucid=sdfIYASDs9d8f7IYfds
```

Response:

```
<response>
  <Code>
```

```
          0
        </Code>
        <message>
          DeleteRecording Command Returned Successfully
        </message>
      </response>
```

recordPause

Pauses a recording in progress. Recordings which are paused can be resumed using the **recordUnpause** command. Please note, no silence is inserted into the recording to indicate the pause and resuming of the recording.

You may use the **getFileInfo** command to retrieve the fileId of the recording if the recording was initiated automatically by the NN-ISR (based on route configuration).

Required Input Parameters

Standard Implementation

Parameter	Value	Description
ucid	<UCID>	The UCID of the session.
fileId	<Filename>	Name of the recording file.

OR

Alternate Implementation

Parameter	Value	Description
sipFromUri	<SIP From URI>	The SIP FROM header, including RSS IP, channel ID, and call-id.
ani	<FROM>	Incoming FROM value or Token (see the command **getToken**).
dnis	<TO>	TO value associated with the call.
timeStamp	<time>	Call Start time (in format YYYY-MM-DD HH:MM:SS)
fileId	<Filename>	Name of the recording file.

OR

Additional Alternate Implementation (Use only if directed)

Parameter	Value	Description
ani	<FROM>	Incoming FROM value or Token (see the command **getToken**).
dnis	<TO>	TO value associated with the call.
timeStamp	<time>	Call Start time (in format YYYY-MM-DD HH:MM:SS)
fileId	<Filename>	Name of the recording file.

Optional Input Parameters*)*

Parameter	Value	Description
ucidSource	<source>	Indicates which ucid field to check: ISR_UCID, Ingress_callid, or Egress_callid.

Return Values

Parameter	Value	Description
code	-1 | 0 - 27	Numbered result code. Zero (0) indicates success. All other responses indicate command failure.
message	<Status Text>	See Return Codes for the valid values for this parameter.

recordPause Sample Implementation

The following is an example of using the **recordPause** REST API command.

Request:

```
http://172.30.58.130:9000/IsrApi/rest/recordPause?fileId=Test_Recordi
ng_2012-01-02_030405.pcm&ucidSource=isr_ucid&ucid=sdfIYASDs9d8f7IYfds
```

Response:

```
<response>
  <Code>
    0
  </Code>
  <message>
    RecordPause Command Returned Successfully
  </message>
</response>
```

recordUnpause

Resumes a paused recording. Recordings which may be paused using the **recordPause** command and resumed using the **recordUnpause** command. Please note, no silence is inserted into the recording to indicate the pause and resuming of the recording.

You may use the **getFileInfo** command to retrieve the fileId of the recording if the recording was initiated automatically by the NN-ISR (based on route configuration).

Required Input Parameters

Standard Implementation

Parameter	Value	Description
ucid	<UCID>	The UCID of the session.
fileId	<Filename>	Name of the recording file.

OR

Alternate Implementation

Parameter	Value	Description
sipFromUri	<SIP From URI>	The SIP FROM header, including RSS IP, channel ID and call-id.
ani	<FROM>	Incoming FROM value or Token (see the command **GetToken**).
dnis	<TO>	TO value associated with the call.
timeStamp	<time>	Call Start time (in format YYYY-MM-DD HH:MM:SS)
fileId	<Filename>	Name of the recording file.

OR

Additional Alternate Implementation (Use only if directed)

Parameter	Value	Description
ani	<FROM>	Incoming FROM value or Token (see the command **GetToken**).
dnis	<TO>	TO value associated with the call.
timeStamp	<time>	Call Start time (in format YYYY-MM-DD HH:MM:SS)
fileId	<Filename>	Name of the recording file.

Optional Input Parameters*)*

Parameter	Value	Description
ucidSource	<source>	Indicates which ucid field to check: ISR_UCID, Ingress_callid, or Egress_callid.

Return Values

Parameter	Value	Description
code	-1 | 0 - 27	Numbered result code. Zero (0) indicates success. All other responses indicate command failure.
message	<Status Text>	See Return Codes for the valid values for this parameter.

recordUnpause Sample Implementation

The following is an example of using the **recordUnpause** REST API command.

Request:

```
http://172.30.58.130:9000/IsrApi/rest/recordUnpause?fileId=Test_Recor
ding_2012-
0102_030405.pcm&ucidSource=isr_ucid&ucid=sdfIYASDs9d8f7IYfds
```

Response:

```
<response>
  <Code>
    0
  </Code>
  <message>
    RecordUnPause Command Returned Successfully
  </message>
</response>
```

getRoutesForAccount

Retrieves a list of all routes belonging to the specified account.

Required Input Parameters

Standard Implementation

Parameter	Value	Description
accountName	<accountName>	The Name of the account in which to search for routes.

Return Values

Parameter	Value	Description
code	-1 | 0 - 27	Numbered result code. Zero (0) indicates success. All other responses indicate command failure.
message	<Status Text>	See Return Codes for the valid values for this parameter.
routes	<array of routes>	An array of routes corresponding to the specified account. Each route is comprised of the following elements, in this order: • account_name • realm_label • route_pattern • priority • is_recording_enabled • percent_to_record If a route belongs to more than one realm, a separate element is returned for each.

getRoutesForAccount Sample Implementation

The following is an example of using the **getRoutesForAccount** REST API command.

Request:

```
http://172.30.58.130:9000/IsrApi/rest/getRoutesForAccount?accountName
=test
```

Response:

```
<result>
     <code>0</code>
     <message>ACK</message>
     <routes>
          <route>
               <account_name>test</account_name>
```

```
                                 <realm_label>splan</realm_label>
                                 <route_pattern>%</route_pattern>
                                 <priority>1.0</priority>
                                 <is_recording_enabled>1</is_recording_enabled>
                                 <percent_to_record>100</percent_to_record>
                              </route>
                              <route>
                                 <account_name>test</account_name>
                                 <realm_label>test</realm_label>
                                 <route_pattern>%</route_pattern>
                                 <priority>1.0</priority>
                                 <is_recording_enabled>1</is_recording_enabled>
                                 <percent_to_record>100</percent_to_record>
                              </route>
                              <route>
                                  <account_name>test</account_name>
                                  <realm_label>splan</realm_label>
                                  <route_pattern>blah</route_pattern>
                                  <priority>5.0</priority>
                                  <is_recording_enabled>1</is_recording_enabled>
                                  <percent_to_record>0</percent_to_record>
                              </route>
                              <route>
                                  <account_name>test</account_name>
                                  <realm_label>test</realm_label>
                                  <route_pattern>blah</route_pattern>
                                  <priority>5.0</priority>
                                  <is_recording_enabled>1</is_recording_enabled>
                                  <percent_to_record>0</percent_to_record>
                              </route>
                        </routes>
                   </result>
```

getRoutesForRealm

Retrieves a list of all routes belonging to the specified realm.

Required Input Parameters

Standard Implementation

Parameter	Value	Description
realmLabel	<realm label>	The label of the realm in which to search for routes.

Return Values

Parameter	Value	Description
code	-1 | 0 - 27	Numbered result code. Zero (0) indicates success. All other responses indicate command failure.

Parameter	Value	Description
message	<Status Text>	See Return Codes for the valid values for this parameter.
routes	<array of routes>	An array of routes corresponding to the specified realm. Each route is comprised of the following elements, in this order: • account_name • realm_label • route_pattern • priority • is_recording_enabled • percent_to_record If a route belongs to more than one realm, a separate element is returned for each.

getRoutesForRealm Sample Implementation

The following is an example of using the **getRoutesForRealm** REST API command.

Request:

```
http://172.30.58.130:9000/IsrApi/rest/getRoutesForRealm?realmLabel=sp
lan
```

Response:

```
<result>
      <code>0</code>
      <message>ACK</message>
      <routes>
            <route>
                <account_name>test</account_name>
                <realm_label>splan</realm_label>
                <route_pattern>%</route_pattern>
                <priority>1.0</priority>
                <is_recording_enabled>1</is_recording_enabled>
                <percent_to_record>100</percent_to_record>
             </route>
             <route>
                <account_name>test</account_name>
                <realm_label>splan</realm_label>
                <route_pattern>blah</route_pattern>
                <priority>5.0</priority>
                <is_recording_enabled>1</is_recording_enabled>
                <percent_to_record>0</percent_to_record>
             </route>
      </routes>
</result>
```

bookmark

Inserts a cue point in a recording file (only supported if using WAV file recording types). For more information on recording types, see Recording File Types/Formats Supported.

Required Input Parameters

Standard Implementation

Parameter	Value	Description
ucid	<UCID>	The UCID of this session.

OR

Alternate Implementation

Parameter	Value	Description
fileId	<Filename>	Name of the recording file.
ani	<FROM>	Incoming FROM value or Token (see the command **GetToken**).
dnis	<TO>	TO value associated with the call.
timeStamp	<time>	Call Start time (in format YYY-MM-DD HH:MM:SS)

OR

Additional Alternate Implementation (Use only if directed)

Parameter	Value	Description
ani	<FROM>	Incoming FROM value or Token (see the command **GetToken**).
dnis	<TO>	TO value associated with the call.
timeStamp	<time>	Call Start time (in format YYY-MM-DD HH:MM:SS)

Optional Input Parameters*)*

Parameter	Value	Description
ucidSource	<source>	Indicates which ucid field to check: ISR_UCID, Ingress_callid, or Egress_callid.
bookmarkLabel	<Text of label name>	Label to use for the Bookmark (for example, “Agent Transfer” or “GetPinTry1”)
bookmarkTime	Range: to 3200 Default: 0	Offset, in milliseconds, to set the bookmark, calculated from the current time.

Return Values

Parameter	Value	Description
result_code	-1 | 0 - 27	Numbered result code. Zero (0) indicates success. All other responses indicate command failure.
result_text	<Status Text>	See Return Codes for the valid values for this parameter.

Bookmark Sample Implementation

The following is an example of using the **bookmark** REST API command.

Request:

```
http://172.30.58.130:9000/IsrApi/rest/bookmark?bookmarkLabel=SecondCo
lorChoice&ucidSource =isr_ucid&ucid=sdfiYASDs9d8f7IYfds
```

Response:

```
<Result>
<code>0</code>
<message>ACK/message>
</result>
```

Recording File Types/Formats Supported

The NN-ISR supports the following recording file types and formats:

File Type	Header	Format
2	WAVE	Linear/PCM 8bit/8KHz stereo
3	WAVE	Linear/PCM 16bit/8KHz stereo
4	WAVE	Linear/PCM 8bit/8KHz mono
5	WAVE	Linear/PCM 16bit/8KHz mono
6	WAVE	uLaw 8bit/8Khz stereo
7	WAVE	aLaw 8bit/8Khz stereo
8	WAVE	uLaw 8bit/8Khz mono
9	WAVE	aLaw 8bit/8Khz mono
10	N/A	uLaw 8bit/8Khz mono (raw)
11	N/A	aLaw 8bit/8Khz mono (raw)
12	N/A	PCM 8bit/8KHz mono (raw)
13	WAVE	ADPCM 4bit/8KHz mono
14	WAVE	ADPCM 4bit/8KHz stereo

Return Codes

The NN-ISR supports the following return codes:

Return Code	Status Text
0	<success>: [ACK %command% Command Returned 'Successfully', where <command> equals one of the supported commands. This file has been renamed by the recorder, please use the following new filename for future commands: %filename%]
-1	NACK

Return Code	Status Text
1	Feature Not Licensed
2	Host Not Authorized
3	Parameter Missing
4	Missing Command - command parameter is required
5	Invalid Command
6	Missing Parameter - ANI parameter is required.
7	Missing Parameter - DNIS parameter is required.
8	Missing Parameter - channelId parameter is required.
9	Missing Parameter - fileId parameter is required.
10	Error establishing database connection
11	<file error>: [Error retrieving filename for ANI %ANI% DNIS %DNIS% at timestamp of %timeStamp% Error retrieving file info for ANI %ANI% DNIS %DNIS% at timestamp of %timeStamp% Error retrieving file for ANI %ANI% DNIS %DNIS% at timestamp of %timeStamp%]
12	<record state transition error>: [Recording has already started on this channel. Recording is not paused, cannot unpause recording. Recording is started by another session or not started, cannot pause recording. Recording has already been paused, cannot pause recording. Recording is started by another session or not started, cannot unpause recording.]
13	Recording is started by another session or not started, cannot end recording.
14	Invalid Parameter for RecordEnd - fileId not found from recording index.
15	Error Indexing for RecordEnd - %SQLException%
16	Error Generating Token
17	<RecordAndSave route error>: [Couldn't find route information for ANI %ANI% DNIS %DNIS% or route is not RecordAndSave application type Route is not RecordAndSave application type Couldn't find route information]
18	ANI and DNIS are required for command **SaveRecording**.
19	No recording found with supplied criteria.
20	RSS could not successfully perform the operation.
21	Cannot change name of file while recording is in progress.
22	Cannot rename a file that is set to be deleted.
23	Could not convert file to playable format.

Return Code	Status Text
24	Timed out waiting for file conversion.
25	Conversion already initiated by another user, please try again later.
26	No session or file was found matching the provided UCID.
27	Missing Parameter - %parameter% is required.

3 Troubleshooting

Introduction

This section provides the information required to troubleshoot your NN-ISR if required, after installing and using it in your network.

Common Problems

The following identifies some answers to issues you may encounter after installing and using the NN-ISR.

Issue	Resolution
I get a busy message.	1. Make sure your NN-ISR is on and ready to accept calls. 2. Ensure that you are not over the port capacity limit for your route.
I can't get it to answer.	1. Double check your NN-ISR settings in the vmgConfg.xml file. Is the IP address correct? Is the SIP Port correct? 2. Ensure that your 800 number is configured on the network.

Logs

Within the NN-ISR home directory, are the directories:

/cxc_common/ISR/ApiLog/IsrApi.log

These directories contain all of the logs associated with the NN-ISR operation. You can access these logs as required to view operational information about the NN-ISR that can be used for troubleshooting purposes.

vmgConfig.xml

The directory that contains the installation files on the NN-ISR is located at:

/cxc/vmgConfig.xml

This directory is the default directory for all installation files. It also contains the default vmgConfig.xml file, which includes all current configuration settings. If you change parameters in this file, the NN-ISR service requires a restart for the changes to take affect.

A VXML API Commands

In addition to the REST API described above, there is a VXML API which can be used to control recording and other API functions from a VoiceXML dialog. The VXML API is controlled via the sendIPCRCommand.jsp subdialog. It has the same commands as the REST API, but with slightly different parameters, and it also has some additional commands outlined below. The sendIPCRCommand.jsp subdialog can be accessed at:

```
http://<host:port>/IsrApi/sendIPCRCommand.jsp
```

Determining NN-ISR Channel and IP Address

The main difference between the REST commands and VXML commands is that instead of using the fromSipUri parameter, the VXML API has two parameters, channelId and mixerIp. For SIP INVITES initiated by the NN-ISR, these parameters can usually be found in the FROM header. For example, in the following FROM header, the channelId is 2 and the mixerIp is 172.30.58.111.

```
From: <sip:6003@172.30.58.111:5060>;channel=channel2;call-id=43-
0ea84426b88e01901b4de116d700;tag=3743477352
```

Required Input Parameters

Name	Value	Description
command	<command to run>	RecordStart, RecordEnd, GetToken, Bookmark, GetFile, GetFileInfo, UpdateFileInfo, DeleteFile, DeleteRecording, SaveRecording, RecordPause, RecordUnPause

QTime.jsp

QTime.jsp is a subdialog utility that returns a timestamp. In Time Division Multiplex (TDM) implementations, recording is initiated by passing Automatic Number Identification (ANI), Direct Number Identification Service (DNIS), and time of call to the recorder to perform a lookup in the NN-ISR index to find the call that most closely matches those values. The time of call expected is the time the recorder receives the call. It is important that an API developer get that time immediately at the start of the call in order for the correct call to be found when the StartRecord command is given. The NN-ISR installation includes a subdialog call to supply the current time on the recorder.

To access, use a subdialog call to:

```
http://<host:port>/IsrApi/Qtime.jsp
```

Example VXML API Implementations

This section provides VXML examples for API commands that are supported for both REST and VXML. For more information on the following examples, see Chapter 2, API Commands.

RecordStart and RecordEnd Sample Implementation - VXML

The following is an example of how to invoke the **recordStart** and **recordEnd** commands via the NN-ISR VXML API.

```
<?xml version="1.0" ?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
  <var name="command" expr="'RecordStart'"/>
  <var name="ucid" expr="'<ingress_callid of the call>'"/>
  <var name="ucidSource" expr="'ingress_callid'"/>
  <var name="fileId" expr="'Test_Recording_2012-01-
02_031023.pcm'"/>
  <!-- This will start recording immediately -->
  <!-- and send the output to the specified file. -->
  <form id="start">
    <subdialog name="record_start"
        src="http://10.8.172.150:9000/IsrApi/sendIPCRCommand.jsp"
        namelist="command ucid ucidSource fileId"/>
    <field name="color">
     <prompt> Please say the name of your favorite color.</prompt>
      <grammar mode="voice" root="color_choice" version="1.0">
        <rule id="color_choice">
          <one-of>
            <item>red</item>
            <item>blue</item>
            <item>green</item>
            <item>yellow</item>
            <item>orange</item>
            <item>purple</item>
          </one-of>
        </rule>
      </grammar>
      <filled>
        <prompt> You have chosen <value expr="color"/>. </prompt>
      </filled>
    </field>
    <!-- This will stop recording immediately -->
    <block>
      <assign name="command" expr="'RecordEnd'"/>
    </block>
    <subdialog name="record_end"
        src="http://10.8.172.150:9000/IsrApi/sendIPCRCommand.jsp"
        namelist="command ucid ucidSource fileId"/>
  </form>
</vxml>
```

GetFileInfo Sample Implementation - VXML

The following is an example of how to invoke the **getFileInfo** command via the NN-ISR VXML API.

```
<var name="command"/>
<var name="ucid"/>
<var name="ucidSource"/>
<var name="duration"/>
<var name="fileUri"/>
<var name="resultCode"/>
<var name="resultText"/>
<form id="getFileInfo">
  <block>
    <assign name="command" expr="'GetFileInfo'"/>
    <assign name="ucid" expr="'s87dfKDSAFd8f3esaneb'"/>
    <assign name="ucidSource" expr="'ingress_callid'"/>
  </block>
  <subdialog name="get_file_info"
      src="http://10.8.172.150:9000/IsrApi/sendIPCRCommand.jsp"
      namelist="command ucid ucidSource">
    <filled>
     <assign name="resultCode" expr="get_file_info.result_code"/>
     <assign name="resultText" expr="get_file_info.result_text"/>
      <assign name="fileUri" expr="get_file_info.file_uri"/>
      <assign name="duration" expr="get_file_info.duration"/>
    </filled>
  </subdialog>
</form>
```

UpdateFileInfo Sample Implementation - VXML

The following is an example of how to invoke the **updateFileInfo** command via the NN-ISR VXML API.

```
<var name="command"/>
<var name="ucid" expr="'s87dfKDSAFd8f3esaneb'"/>
<var name="ucidSource" expr="'ingress_callid'"/>
<var name="resultCode"/>
<var name="resultText"/>
<!-- here are the parameters and values that will be changed -->
<var name="newFileId" expr="'renamed_8345.wav'"/>
<var name="agentId" expr="'3359'"/>
<!-- this is the custom_data we are passing in, it required a custom
data
name "test" to be set up for the route. Supports up to 4 fields -->
<var name="test" expr="'testdata'"/>
<form id="updateFileInfo">
  <block>
    <assign name="command" expr="'UpdateFileInfo'"/>
  </block>
  <subdialog name="update_agent_id"
      src="http://10.8.172.150:9000/IsrApi/sendIPCRCommand.jsp"
      namelist="command ucid ucidSource newFileId agentId test">
    <filled>
      <assign name="resultCode" expr="get_token.result_code"/>
      <assign name="resultText" expr="get_token.result_text"/>
    </filled>
  </subdialog>
</form>
```

DeleteFile Sample Implementation - VXML

The following is an example of how to invoke the **deleteFile** command via the NN-ISR VXML API.

```
<var name="ucid" expr="'s87dfKDSAFd8f3esaneb'"/>
<var name="ucidSource" expr="'ingress_callid'"/>
<var name="command" expr="'DeleteFile'"/>
<subdialog name="delRecording"
    src="http://10.8.172.150:9000/IsrApi/sendIPCRCommand.jsp"
    namelist="command ucid ucidSource "/>
</subdialog>
```

deleteRecording Sample Implementation - VXML

The following is an example of how to invoke the **deleteRecording** command via the NN-ISR VXML API.

```
<var name="ucid" expr="'s87dfKDSAFd8f3esaneb'"/>
<var name="ucidSource" expr="'ingress_callid'"/>
<var name="command" expr="'DeleteRecording'"/>
<subdialog name="delRecording"
    src="http://10.8.172.150:9000/IsrApi/sendIPCRCommand.jsp"
    namelist="command ucid ucidSource"/>
</subdialog>
```

RecordPause Sample Implementation - VXML

The following is an example of how to invoke the **recordPause** command via the NN-ISR VXML API.

```
<var name="ucid" expr="'s87dfKDSAFd8f3esaneb'"/>
<var name="ucidSource" expr="'ingress_callid'"/>
<var name="fileId" expr="'Test_Recording_2012-01-
02_031023.pcm'"/>
<var name="command" expr="'RecordPause'"/>
<subdialog name="pauseRecording"
    src="http://10.8.172.150:9000/IsrApi/sendIPCRCommand.jsp"
    namelist="command ucid ucidSource fileId"/>
</subdialog>
```

RecordUnPause Sample Implementation - VXML

The following is an example of how to invoke the **recordUnPause** command via the NN-ISR VXML API.

```
<var name="ucid" expr="'s87dfKDSAFd8f3esaneb'"/>
<var name="ucidSource" expr="'ingress_callid'"/>
<var name="fileId" expr="'Test_Recording_2012-01-
02_031023.pcm'"/>
<var name="command" expr="'RecordUnPause'"/>
<subdialog name="pauseRecording"
    src="http://10.8.172.150:9000/IsrApi/sendIPCRCommand.jsp"
    namelist="command ucid ucidSource fileId"/>
</subdialog>
```

VXML Commands

This section describes API commands that are supported for VXML only.

GetToken

The GetToken command retrieves a unique identifier to enable recording.

Required Input Parameters

The GetToken command has no required input parameters.

Optional Input Parameters

The GetToken command has no optional input parameters.

Return Values

Parameter	Value	Description
result_code	-1 | 0 - 27	Numbered result code. Zero (0) indicates success. All other responses indicate command failure.
result_text	<Status Text>	See Return Codes for the valid values for this parameter.
token	<10-digit number>	Unique 10-digit identifier to be used to reference the call being recorded - in place of ANI.

GetToken sample Implementation - VXML

The following is an example of using the **GetToken** command.

```
<var name="command"/>
<var name="resultCode"/>
<var name="resultText"/>
<var name="token"/>
<form id="getTokenId">
  <block>
    <assign name="command" expr="'GetToken'"/>
  </block>
  <subdialog name="get_token"
```

```
      src="http://10.8.172.150:9000/IsrApi/sendIPCRCommand.jsp"
      namelist="command">
    <filled>
      <assign name="resultCode" expr="get_token.result_code"/>
      <assign name="resultText" expr="get_token.result_text"/>
      <assign name="token" expr="get_token.token"/>
    </filled>
  </subdialog>
</form>
```

Bookmark

The **Bookmark** command inserts a cue point in a recording file (only supported if using WAVE file recording types; for information about recording types, see Recording File Types/Formats Supported).

Required Input Parameters

Standard Implementation

Parameter	Value	Description
ucid	<UCID>	UCID of the session.

OR

Alternate Implementation

Parameter	Value	Description
channelId	<channel ID>	Number of the channel servicing the session for which this command should be executed. For information on how to determine the value, see Determining NN-ISR Channel and IP Address.
mixerip	<IP of ISR>	IP Address of the NN-ISR which has the call.
ani or token	<FROM>	Incoming FROM value or Token (see the command **getToken**).
dnis	<TO>	TO value associated with the call.
timeStamp	<time>	Call Start time (in format YYYY-MM-DD HH:MM:SS)

OR

Additional Alternate Implementation (Use only if directed)

Parameter	Value	Description
ani or token	<FROM>	Incoming FROM value or Token (see the command **getToken**).
dnis	<TO>	TO value associated with the call.
timeStamp	<time>	Call Start time (in format YYYY-MM-DD HH:MM:SS)

Optional Input Parameters

Parameter	Value	Description
ucidSource	<source>	Indicates which ucid field to check: ISR_UCID, Ingress_callid, or Egress_callid.
waitTime	**Range**: 0 to 32000 **Default**: 0	Number of milliseconds to wait before executing this command.
bookmarkLabel	<Text of label name> **Default:** “my bookmark”	Label to use for the Bookmark (i.e., “Agent Transfer” or “GetPinTry1”)
bookmarkTime	**Range:** 0 to 3200- **Default:** 0	Offset, in milliseconds, to set the bookmark, calculated from the current time.

Return Values

Parameter	Value	Description
result_code	-1 | 0 - 27	Numbered result code. Zero (0) indicates success. All other responses indicate command failure.
result_text	<Status Text>	See Return Codes for the valid values for this parameter.

Bookmark sample Implementation - VXML

The following is an example of using the **Bookmark** command.

```
<var name="bookmarkLabel" expr="'SecondColorChoice'"/>
<var name="ucid" expr="'s87dfKDSAFd8f3esaneb'"/>
<var name="ucidSource" expr="'ingress_callid'"/>
<block>
  <assign name="command" expr="'Bookmark'"/>
</block>
<subdialog name="bookmark"
    src="http://10.8.172.150:9000/IsrApi/sendIPCRCommand.jsp"
    namelist="command ucid ucidSource bookmarkLabel"/>
</subdialog>
```

SaveRecording

The **SaveRecording** command saves the recording for Record and Save routes. This command also bookmarks the recording when it is called. You may use the **GetFileInfo** command to retrieve the fileId of the recording if the recording was initiated automatically by the NN-ISR (based on route configuration.)

Required Input Parameters

Standard Implementation

Parameter	Value	Description
ucid	<UCID>	UCID of the session.

OR

Alternate Implementation

Parameter	Value	Description
channelId	<Channel ID>	Number of the channel servicing the session for which this command should be executed. For information on how to determine this value, see Determining NN-ISR Channel and IP Address.
mixerip	<IP of ISR>	IP Address of the NN-ISR which has the call.
ani or token	<FROM>	Incoming FROM value or Token (see the command **getToken**).
dnis	<TO>	TO value associated with the call.
timeStamp	<time>	Call Start time (in format YYYY-MM-DD HH:MM:SS)

OR

Additional Alternate Implementation (Use only if directed)

Parameter	Value	Description
ani or token	<FROM>	Incoming FROM value or Token (see the command **getToken**).
dnis	<TO>	TO value associated with the call.
timeStamp	<time>	Call Start time (in format YYYY-MM-DD HH:MM:SS)

Optional Input Parameters

Parameter	Value	Description
ucidSource	<source>	Indicates which ucid field to check: ISR_UCID, Ingress_callid, or Egress_callid.
fileId	<Filename>	Name of the recording file associated with the call.

Return Values

Parameter	Value	Description
result_code	-1 | 0 - 27	Numbered result code. Zero (0) indicates success. All other responses indicate command failure.
result_text	<Status Text>	See Return Codes for the valid values for this parameter.

SaveRecording sample Implementation - VXML

The following is an example of using the **SaveRecording** command.

```
<var name="command" expr="'SaveRecording'"/>
<var name="ucid" expr="'DSfd87KSDFis60MNf8d'"/>
```

```
<var name="ucidSource" expr="'egress_callid'"/>
<subdialog name="saveRecording"
    src="http://10.8.172.150:9000/IsrApi/sendIPCRCommand.jsp"
    namelist="command ucid ucidSource"/>
</subdialog>
```