

Interactive Session Recorder

API Guide
Release 5.2

October 2016

Notices

Copyright© 2016, 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, then 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 about 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 unless otherwise set forth in an applicable agreement between you and Oracle. 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, except as set forth in an applicable agreement between you and Oracle.

Contents

1 About the ISR.....	9
2 ISR FACE.....	11
ISR FACE.....	11
Creating an API User.....	11
REST API Commands.....	12
URL-Encoding Special Characters.....	13
Logging In to FACE.....	13
audioRecording Commands.....	13
eventNotify.....	20
StartScreenCapture.....	21
StopScreenCapture.....	22
version.....	22
Return Codes.....	23
FACE Policy System.....	24
FACE Policy Configurations.....	24
Determining the UCID.....	25
External Event Notifications.....	25
Events and Notifications.....	25
Supported General Parameters.....	26
Supported Recording-Specific Parameters.....	26
Supported Special Parameters.....	27
External Event Notification and Parameters Example.....	27
A— Legacy RSS API Commands.....	29
Legacy RSS API Commands.....	29
REST Commands.....	29
API Commands.....	29
Recording File Types/Formats Supported.....	48
Return Codes.....	48
Troubleshooting.....	50
Common Problems.....	50
Logs.....	50
vmgConfig.xml.....	50
B— VXML API Commands.....	53
Determining ISR Channel and IP Address.....	53
QTime.jsp.....	53
Example VXML API Implementations.....	54
VXML Commands.....	56
GetToken.....	56
SaveRecording.....	57

Preface

About This Guide

The Interactive Session Recorder (ISR) Application Programming Interface (API) Guide provides information about:

- ISR FACE
- Invoking API Commands
- Legacy REST and VoiceXML Commands
- Recording File Types/Formats Supported
- Return Codes
- Troubleshooting

Related Documentation

The following table describes the documentation set for this release.

Document Name	Document Description
ISR Release Notes	Contains information about new ISR features, fixes, and known issues.
ISR Installation Guide	Provides an overview of the ISR, hardware/software requirements and recommendations, storage considerations, pre-installation information, installation procedures, post-install verification procedures, making the first call, and additional advanced topics about the ISR.
ISR User Guide	Contains information about using the ISR Dashboard for all levels of users. Provides information about viewing, playing, deleting recordings, running reports, and managing user profiles.
ISR Administrator Guide	Contains information about using the ISR Dashboard for the Administrator level user (Super User, Account Administrator, Tenant Administrator). Provides information about creating and managing accounts, routes, and users. Also provides information about configuring the ISR, running reports, viewing active calls, and securing the ISR deployment.
ISR API Reference Guide	Contains information about ISR FACE, VoiceXML Commands, legacy application programming interfaces (APIs), Recording File Types/Formats Supported, Return Codes, sendIPCRCommand.jsp Subdialog, Advanced Options, and Troubleshooting.
ISR Monitoring Guide	Contains information about installing and configuring the ISR Monitor, the Monitor database schema, and the Monitor MIB.

Document Name	Document Description
ISR Remote Archival Web Services Reference Guide	Contains information about the Remote Archival Web Service, its methods, WSDL definitions, DataType definitions, sample responses, and importing its certificates into the client keystore.
ISR Security Guide	Contains information about security considerations and best practices from a network and application security perspective for the ISR product.

Revision History

Date	Description
September 2016	<ul style="list-style-type: none"> Initial release of ISR 5.2 software.
October 2016	<ul style="list-style-type: none"> Fixes various typographical errors. Renames the following sections to more accurately fit the content: <ul style="list-style-type: none"> "FACE REST API" has been renamed to "ISR FACE". "FACE Authorization" has been renamed to "Login". "Recording Audio" has been renamed to "download". "Recording Metadata" has been renamed to "details". Updates the following sections of the guide for accuracy: <ul style="list-style-type: none"> ISR FACE REST API Commands FACE Policy System audioRecording Commands External Event Notifications Identifying a Session Without a UCID Legacy RSS API Commands Return Data Removes the following sections: <ul style="list-style-type: none"> audioRecording/pause audioRecording/resume FACE Custom Data (the remaining subsections within this section have been moved to the "audioRecording Commands" section of this guide.) Viewing and Editing the Custom Data Master List Viewing and Editing Custom Data Field Session Values Adds the following sections to the guide: <ul style="list-style-type: none"> "Input Parameters" has been added to "audioRecording" to describe supported HTTP POST, URL, and miscellaneous request parameters.

Date	Description
	<ul style="list-style-type: none"><li data-bbox="889 218 1170 247">• Return Data Examples

About the ISR

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

You can control the ISR using a policy defined on the ISR FACE 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.

ISR FACE

ISR FACE

The ISR supports FACE. FACE is a feature for the aggregation and control of events. It is a centralized component used to control both ISR and integrated third-party services. It can also retrieve audio and detailed metadata for recordings stored in the ISR. The initial third-party integration is with the ObserveIT Visual Session Recording solution. This provides the key activities necessary for desktop screen capture and for associating the captured screens to related audio recordings.

The controls currently available include full audio recording control (start, stop, pause, and resume, where pause and resume can aid in the omission of sensitive information from the recording), retrieval of recorded audio and metadata, as well as starting and stopping screen capture on an ObserveIT agent host. FACE also supports a policy system) allowing incoming events to trigger actions, with the current set of actions dedicated to screen capture control.

The ISR FACE contains the following components:

- FACE Policy System—Handles automatic actions for specific ISR events.
- External Event Notification Listener—Receives events from ISR components and passes them to Policy System.
- FACE REST API—API for communication and control of ISR components.

For information on deploying and configuring FACE, see the *Oracle Communications Interactive Session Recorder Installation Guide*.

Creating an API User

In order to use the **audioRecording** commands of the FACE REST API, you must create an API User via the Dashboard.


To Add an API User:

1. After logging into the ISR Dashboard, click **Admin** in the main menu (or **Edit System Configurations** on the Home page).
2. Click **Users**.
The Users page displays.
3. Click **New User**

The New User page displays.


4. Enter the relevant **Account**, **Name**, **Email**, **Password**, and **Preferred Time Zone Locale** for the User.
5. **Type of User**—Select API User from the dropdown menu.
6. Click **Create**.

For more information on configuring Users, see "Managing Users" in the *Oracle Communications Interactive Session Controller Administrator Guide*.


 **Note:** Your API User password will expire based on the value you configure in the Dashboard Security Settings **Users Password Expires in** parameter. When the API User password expires, the FACE API client application/s will no longer be authorized for requests until the password has been changed and reflected accurately in the client user's password configuration. For more information on configuring Dashboard Security Settings, see "Managing User Dashboard Security Settings" in the *Oracle Communications Interactive Session Controller Administrator Guide*.

REST API Commands

The following table shows and describes the specific API commands you can implement using the ISR FACE REST API.


 **Note:** Use GET as the HTTP method unless otherwise directed in the command description.

API Command	Description
audioRecording/ <command>	Retrieve recording metadata or audio, control ISR recording, or update certain metadata. The <command> argument can be one of the following values: <ul style="list-style-type: none"> • start • stop • pause • resume • details • download • redirect

API Command	Description
	<ul style="list-style-type: none"> delete <p> Note: These commands require an authentication token.</p>
login	Authenticate a user/client to receive a token.
eventNotify	Notify the FACE of the specified event, possibly causing Policy Actions to fire.
StartScreenCapture	Begin ObserveIT Screen Recording.
StopScreenCapture	End ObserveIT Screen Recording.
version	Return the version number of the FACE software.

URL-Encoding Special Characters

Special characters should be URL encoded (also known as percent-encoded) in URL parameters. For example, the plus (+) and at (@) signs are commonly used as SIP URI parameters (%2B and %3A, respectively).

 **Note:** Special characters should only be encoded within parameter values; you must leave them unencoded when they have meaning as part of the URL, for example, ampersand (&) when it is used to separate URL parameters.

Logging In to FACE

To generate an authorization token, an HTTP POST, in which the request body contains the "userEmail", "password", and optionally the "expirationSeconds" (the number of seconds before expiration of the token), must be sent to the following URL:

```
<scheme>://<host>:<port>/Face/login
```

If the user's credentials are correct but the password has expired, FACE returns an error message and does not create or return a token. Similarly, FACE returns a generic error message if the credentials are invalid or if a user has been locked out due to too many failed login attempts.

If the credentials are accurate, a new token is generated, added to the data structure, and returned. The expiration time is calculated based on the user's "expirationSeconds" value; if the user has not provided an "expirationSeconds" parameter, or that value is larger than the ISR's default maximum, the ISR's default maximum (1 day) is used.

The "userId" parameter limits which recordings the user may access and/or control, based on the user's account. A new user type, the "API User" must be created, and only that user type is allowed to use the FACE API. For more information, see "Creating an API User in this guide".

audioRecording Commands

This section describes FACE recording access and control.

The FACE audioRecording commands use a subset of standard HTTP methods to help determine the type of action to take. In any POST, the FACE API checks the request body for custom data (as well as other customer-definable ISR data related to the method, (for example, "AgentId") and if present, updates the appropriate values. Any data defined by, or unique to the system (for example, SIPREC metadata and extension data, call start time, to, from, and duration) cannot be updated. If the request is a GET or DELETE, the FACE API ignores the request body. In all audioRecording commands, URL request parameters are used solely to determine which item the request acts upon. No parameters in the request URL are used directly for updating any ISR data.

download

The following URL returns audio, similar to the RSS API's getFile method.

```
<scheme>://<host>:<port>/Face/audioRecording/download
```

redirect

The following URL offers a redirect response containing the direct URL serving the audio recording.

```
<scheme>://<host>:<port>/Face/audioRecording/redirect
```

details

The following URL returns the recording metadata.

```
<scheme>://<host>:<port>/Face/audioRecording/details
```

start | pause | resume | stop

The following URL controls audio with the HTTP POST request method, where the command parameter can be start, pause, resume, or stop.

```
<scheme>://<host>:<port>/Face/audioRecording/<command>
```

When the body of the request is empty, it causes the action only. When the body of the request contains content, it causes the action as well as updates the recording in the ISR.

delete


The following URL deletes recordings using the HTTP POST request method.

```
<scheme>://<host>:<port>/Face/audioRecording/delete
```

Alternatively, the following request URL deletes the recording using the HTTP DELETE method.

```
<scheme>://<host>:<port>/Face/audioRecording/details
```

Both the POST and DELETE methods delete both the audio file and all of its associated metadata.

 **Note:** Depending on your Archiver configuration, the length of time it may take for the file and metadata to be removed from the system can vary.

Input Parameters

The following section describes all the HTTP POST request parameters, URL request parameters, and miscellaneous parameters.

HTTP POST Request Parameters

The following parameters can be included in the request body in an HTTP POST request:

- filename
- agentId
- agentTerminal
- sensitive (flag)
- setNeverExpire (flag)
- any defined Custom Data Field names (For more information, see "Viewing and Editing the Custom Data Master List" in the *Oracle Communications Interactive Session Recorder Administrator Guide*.)

URL Request Parameters

The URL request parameters are used to select the recording on which to act. Only recordings for which the user has permission to access are included in the results.

The following are options for selection criteria:

General:

- isrUcid
- ingressCallId
- egressCallId


- `thirdpartyId` (for example, `ObserveIT` screen capture ID)
- `ani/from`
- `dnis/to`
- `filename`
- `start`
- `earliestStart` (only recordings starting at this time or later are matched, can be used with `latestStart` to refine the time range)
- `latestStart` (only recordings starting at this time or earlier are matched, can be used with `earliestStart` to refine the time range)
- `aor`
- any SIPREC extension data defined as "searchable" (`metadata_types` table)

Recording control only:

- `recordingId/tmpRecordingId`
- any defined Custom Data Field names (for more information, see "Viewing and Editing the Custom Data Master List" in the *Oracle Communications Interactive Session Recorder Administrator Guide*.)


Non-recording control only:

- `recordingId/tmpRecordingId`
- any defined Custom Data Field names (for more information, see "Viewing and Editing the Custom Data Master List" in the *Oracle Communications Interactive Session Recorder Administrator Guide*.)

 **Note:** Parameter values are case-sensitive.

The following miscellaneous parameters may also be included in some cases:


- `maxListLength`: Limits the maximum number of items returned that match the supplied Selection Criteria. Valid values are integers between 1 and 1000.
- `fillWithSilence`: By default, silence is not inserted into the recording to indicate the pause. Set this parameter to true to fill the paused section of audio with padded silence. If omitted or set to false, the paused section of audio is truncated.

 **Note:** This parameter is only applicable for the pause command.

- `codecProfile`: Overrides the Codec Profile during ad hoc recording (for more information, see *Managing Recording Format Profiles* in the *Oracle Communications Interactive Session Recorder Administrator Guide*).

The following are valid values:

- 1 (Default)
- 2 (Smallest)
- 3 (Small)
- 4 (Best Quality)
- 5 (Firefox Compatible)

 **Note:** This parameter is only applicable for the start command.

Return Data

All recording metadata, including custom data and SIPREC metadata and extension data, is incorporated into the response when recording details are requested and a single entry is matched. This same data is included as part of the return for all `audioRecording` commands (with minor exceptions, such as retrieving the audio).

In the event that the selection criteria don't uniquely determine a recording to act upon, a limited list is returned containing suggested selection criteria and metadata for each entry, to help the user or client application make the decision. The default maximum list length of 1000 items can be overridden by including a "maxListLength"

parameter in the request URL. If the number of matches has exceeded the maximum list length, the ISR provides the following status message:

```
Selection criteria insufficient to determine recording. There were more matches than maxListLength, please refine your search if the entry you require is not in this list."
```

If the number of matches is less than `maxListLength`, the status message is:

```
"Selection criteria insufficient to determine recording. Returning all matches.
```

Currently, both XML and JSON response formats are supported. By default, XML is returned, but the "Accept" header of the request can be set to specify which option the client prefers (`application/json` or `application/xml`).

The following examples show a single match return in XML format, then the same return data in JSON format, and lastly a multiple match return in XML format which contains less than `maxListLength` entries.



Note: These examples are not meant to reflect all possibilities and the data returned may differ in your system.

Single Match XML Example

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
  <code>0</code>
  <message>ACK</message>
  <recording>
    <recordingId>2342</recordingId>
    <filename>8675309-dferi8dsmrt0gdk.wav</filename>
    <fileStatus>0</fileStatus>
    <ani>1234</ani>
    <dnis>8675309</dnis>
    <account>1</account>
    <duration>30037</duration>
    <startTime>2016-07-19 16:21:45.356</startTime>
    <rss>1</rss>
    <archived>0</archived>
    <route>13</route>
    <archivalFailCount>0</archivalFailCount>
    <agentId>007</agentId>
    <archiverMode>primary</archiverMode>
    <sensitive>0</sensitive>
    <pauseLength>3102</pauseLength>
    <deleteFlag>>false</deleteFlag>
    <location>1</location>
    <archiverAction>0</archiverAction>
    <conversionStatus>0</conversionStatus>
    <isrUcid>dferi8dsmrt0gdk</isrUcid>
    <ingressCallId>asdlkfulsadf123987</ingressCallId>
    <pausedWithSilence>>false</pausedWithSilence>
    <customDataSets>
      <customData>
        <name>cdName1</name>
        <value>some value</value>
      </customData>
      <customData>
        <name>otherName</name>
        <value>other value</value>
      </customData>
    </customDataSets>
    <siprecDataSets>
      <siprecData>
        <session>
```



```
<startTime>2016-07-19 08:07:06</startTime>
<siprecSessionId>azsxdc654</siprecSessionId>
<extensionDataSets>
  <extensionData>
    <name>apkt:ucid</name>
    <values>
      <value>dferi8dsmrt0gdk</value>
    </values>
  </extensionData>
</extensionDataSets>
</session>
<participants>
  <participant>
    <siprecParticipantId>fvgbhn34</siprecParticipantId>
    <aor>8675309@sh.net</aor>
    <name>8675309</name>
    <extensionDataSets>
      <extensionData>
        <name>apkt:callingParty</name>
        <values>
          <value>>false</value>
        </values>
      </extensionData>
      <extensionData>
        <name>apkt:realm</name>
        <values>
          <value>splan</value>
        </values>
      </extensionData>
    </extensionDataSets>
  </participant>
  <participant>
    <siprecParticipantId>vfbgnh43</siprecParticipantId>
    <aor>1234@soh.net</aor>
    <name>1234</name>
    <extensionDataSets>
      <extensionData>
        <name>apkt:callingParty</name>
        <values>
          <value>>true</value>
        </values>
      </extensionData>
    </extensionDataSets>
  </participant>
</participants>
<streams>
  <stream>
    <mode>a</mode>
    <participantId>234</participantId>
    <siprecStreamId>rtfgvb59</siprecStreamId>
    <label>someLabel</label>
    <extensionDataSets/>
  </stream>
  <stream>
    <mode>b</mode>
    <participantId>235</participantId>
    <siprecStreamId>bvgftr95</siprecStreamId>
    <label>otherLabel</label>
    <extensionDataSets/>
  </stream>
</streams>
</siprecData>
</siprecDataSets>
<dtmfDigits>
```

```
<dtmfDigit>
  <key>3</key>
  <offsetMs>1038</offsetMs>
  <origin>caller</origin>
</dtmfDigit>
<dtmfDigit>
  <key>#</key>
  <offsetMs>13337</offsetMs>
  <origin>caller</origin>
</dtmfDigit>
</dtmfDigits>
</recording>
</result>
```

Single Match JSON Example

```
{
  "result": {
    "code": 0,
    "message": "ACK",
    "recording": {
      "recordingId": 2342,
      "filename": "8675309-dferi8dsmrt0gdk.wav",
      "fileStatus": 0,
      "ani": "1234",
      "dnis": "8675309",
      "account": "1",
      "duration": 30037,
      "startTime": "2016-07-19 16:21:45.356",
      "rss": 1,
      "archived": 0,
      "route": 13,
      "archivalFailCount": 0,
      "agentId": "007",
      "archiverMode": "primary",
      "sensitive": 0,
      "pauseLength": 3102,
      "deleteFlag": false,
      "location": 1,
      "archiverAction": 0,
      "conversionStatus": 0,
      "isrUcid": "dferi8dsmrt0gdk",
      "ingressCallId": "asdlkfulsadf123987",
      "pausedWithSilence": false,
      "customDataSets": [
        {
          "customData": {
            "name": "cdName",
            "value": "some value"
          }
        },
        {
          "customData": {
            "name": "otherName",
            "value": "other value"
          }
        }
      ],
      "siprecDataSets": [
        {
          "siprecData": {
            "session": {
              "startTime": "2016-07-19 08:07:06",
              "siprecSessionId": "azsxdc654",
              "extensionDataSets": [
                {
                  "extensionData": {
                    "name": "apkt:ucid",
                    "values": [
                      {
                        "value": "dferi8dsmrt0gdk"
                      }
                    ]
                  }
                }
              ]
            }
          }
        }
      ]
    }
  }
}
```

```

    ]
  }}
]
},
"participants": [
  {"participant": {
    "siprecParticipantId": "fvgbhn34",
    "aor": "8675309@sh.net",
    "name": "8675309",
    "extensionDataSets": [
      {"extensionData": {
        "name": "apkt:callingParty",
        "values": [
          {"value": "false"}
        ]
      }},
      {"extensionData": {
        "name": "apkt:realm",
        "values": [
          {"value": "splan"}
        ]
      }}
    ]
  }},
  {"participant": {
    "siprecParticipantId": "vfbgnh43",
    "aor": "1234@soh.net",
    "name": "1234",
    "extensionDataSets": [
      {"extensionData": {
        "name": "apkt:callingParty",
        "values": [
          {"value": "true"}
        ]
      }}
    ]
  }}
],
"streams": [
  {"stream": {
    "mode": "a",
    "participantId": "234",
    "siprecStreamId": "rtfgvb59",
    "startTime": "2016-07-19 08:07:06",
    "label": "someLabel",
    "extensionDataSets": []
  }},
  {"stream": {
    "mode": "b",
    "participantId": "235",
    "siprecStreamId": "bvqftr95",
    "startTime": "2016-07-19 08:07:06",
    "label": "otherLabel",
    "extensionDataSets": []
  }}
]
}}
],
"dtmfDigits": [
  {"dtmfDigit": {
    "key": "3",
    "offsetMs": 1038,
    "origin": "caller"
  }},

```

```

    {"dtmfDigit":{
      "key":"#",
      "offsetMs":13337,
      "origin":"caller"
    }}
  ]
}
}}
```

Multiple Match XML Example


```

<?xml version="1.0" encoding="UTF-8"?>
<result>
  <code>-1</code>
  <message>Selection criteria insufficient to determine recording.
Returning all matches.</message>
  <matches>
    <match>
      <recordingId>2486</recordingId>
      <ingressCallId>1-2505@1.2.3.4</ingressCallId>
      <filename>5678-1-2505@1.2.3.4.wav</filename>
      <from>8675309</from>
      <to>5678</to>
      <start>2016-10-10 11:24:34</start>
    </match>
    <match>
      <recordingId>2519</recordingId>
      <ingressCallId>181-1739@1.2.3.4</ingressCallId>
      <filename>9012-181-1739@1.2.3.4.wav</filename>
      <from>8675309</from>
      <to>9012</to>
      <start>2016-10-10 12:04:49</start>
    </match>
  </matches>
</result>
```

eventNotify

The **eventNotify** command allows the ISR and external systems to notify FACE that an event of interest has occurred. The RSS' ISR API is typically used to notify FACE of events such as audio recording starting or stopping, or that a recording was paused, but third-party systems can use **eventNotify**, as well, to notify FACE of interesting Events such as a screen recording started or ended.

Currently FACE supports **AudioRecordingStarted** and **AudioRecordingEnded** Events, although custom event types can be configured. See *FACE Policy System* for more information.

 **Note:** The Optional Input Parameters listed below are used in the currently supported Events. Other Events may require different parameters.


Required Input Parameters

Standard Implementation

Parameter	Value	Description
event	<event type>	The type of event

Optional Input Parameters

Parameter	Value	Description
calledAors	<AORS List>	The list of Addresses of Record (AOR) for the called parties of the session
ucid	<ISR UCID>	The ISR UCID identifying the session

 **Note:** For more information about using ISR UCIDs, see "Determining the ISR UCID".

Example

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

Request:

```
https://1.2.3.4:8443/Face/eventNotify?
event=AudioRecordingStarted&ucid=fbd5715afe5aca679d1c9230dce37e73@
1.2.3.5&calledAors=sip:agentX@1.2.3.4.11,sip:agentY@1.2.3.4.13
```

Response:

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

StartScreenCapture

The **StartScreenCapture** command causes FACE to issue an ObserveIT **AgentRemoteControl** command to start a screen capture for a specified in-progress recording. To use this command, the AgentId must be registered with the ISR in the `thirdparty_service_params` database table. IsrId is also required and should be the ISR_UCID for the session. If successful, the **StartScreenCapture** command also creates a correlation entry between the IsrId and the returned ObserveIT session ID for looking up recordings for playback.

Required Input Parameters

Standard Implementation

Parameter	Value	Description
IsrId	<ISR UCID>	UCID of the session
AgentId	<agent_ID>	The registered DNIS or ANI of the Agent Screen to capture

 **Note:** For more information about using ISR UCIDs, see "Determining the ISR UCID".

Example

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

Request:

```
https://1.2.3.4:8443/Face/StartScreenCapture?IsrId=
fbd5715afe5aca679d1c9230dce37e73@1.2.3.5&AgentId=6789
```

Response:

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

StopScreenCapture

The **StopScreenCapture** command causes FACE to issue an ObserveIT **AgentRemoteControl** command to terminate a specified in-progress screen recording. To use this command, the AgentId must be registered with the ISR in the `thirdparty_service_params` database table.

Required Input Parameters

Standard Implementation

Parameter	Value	Description
AgentId	<agent_ID>	The registered DNIS or ANI of the Agent Screen to capture

Example

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

Request:

```
https://1.2.3.4:8443/Face/StopScreenCapture?AgentId=6789
```

Response:

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

version

The **version** command returns the version and build information for the current FACE API application.

Required Input Parameters

None

Example

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

Request:

```
https://1.2.3.4:8443/Face/version
```

Response:

```
<result>  
  <version>5.2.0M0P0 build <build #></version>  
</result>
```

Return Codes

The FACE REST API follows the ISR API convention for return codes.

Return Code	Status Text
0	ACK
0	%command% command returned successfully (where %command% equals one of the supported commands)
0	This file has been renamed by the recorder, please use the following new filename for future commands: %filename%
-1	NACK
-1	%miscellaneous error message%
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	Error retrieving filename for ANI %ANI% DNIS %DNIS% at timestamp of %timeStamp%
11	Error retrieving file info for ANI %ANI% DNIS %DNIS% at timestamp of %timeStamp%
11	Error retrieving file for ANI %ANI% DNIS %DNIS% at timestamp of %timeStamp%
11	Error retrieving file with supplied criteria
12	Recording has already started on this channel.
12	Recording is not paused, cannot unpaue recording.
12	Recording is started by another session or not started, cannot pause recording.
12	Recording has already been paused, cannot pause recording.
12	Recording is started by another session or not started, cannot unpaue 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	Couldn't find route information

Return Code	Status Text
17	Route is not RecordAndSave application type
17	Couldn't find route information for ANI %ANI% DNIS %DNIS% or route is not RecordAndSave application type
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
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

FACE Policy System

ISR FACE is able to create and handle Event Policies for the controlling and indexing of audio recordings, screen recordings, and their corresponding metadata. You can create policies to match specific events (for example, AudioRecordingStarted) which can generate a Policy Action (for example, InitiateScreenRecording). This allows FACE to have the power and flexibility to control both audio and screen recording based on your network and application setup.

For more information, see *ISR External Event Notifications* in this guide to see which ISR Events are currently supported and could be used to trigger FACE Policies.

FACE Policy Configurations

While the ISR does not come with any existing Events, ISR and FACE do follow some conventions and come with some default Event types and supported Policy Actions already configured. You can implement new Events to be used to initiate FACE Policy Actions.

Default Event Types


Event Name	Description
AudioRecordingStarted	Event fired by an external event notification when an RSS begins a new audio recording
AudioRecordingEnded	Event fired by an external event notification when an RSS finishes recording an audio file

When an Event with an associated Policy is matched, FACE can trigger an Action to complete a task, such as starting or stopping audio or screen recording, or notifying a third-party component of the Event.

Default Policy Actions

Action Name	Description
InitiateRecordScreen	Action initiated by FACE to begin ObserveIT Screen Recording

Action Name	Description
TerminateRecordScreen	Action initiated by FACE to end ObserveIT Screen Recording

 **Note:** When FACE is configured, it creates a Policy that, when audio recording starts or stops, triggers start or stop, respectively, of ObserveIT Screen Recording on the IP address contained in the first called participant's AOR.

Determining the UCID

The Unique Call Identifier (UCID) accurately identifies the specific session related to a request and is the recommended parameter when using any ISR API. It can be one of two values associated with the session: the `isr_ucid` or `ingress_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 is 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 is used for the `ingress_callid`.

There has been a lot of interest (both internal and field-wide) focused on populating SIPREC metadata with an X-ISR-UCID header or a SIPREC extension data field, mainly for purpose of API recording access and control. Currently, there are 2 SPL implementations to address populating the X-ISR-UCID field, one that properly sets the field with a provided UCID/GUID, and another that adds the value as a field in the SIPREC extension data.

The current ISR release has been enhanced to now populate the primary recording X-ISR-UCID field regardless of SPL implementation. However, this has led to confusion with existing integrations. Please refer to the *Oracle Communications Interactive Session Recorder 5.2 Release Notes* and future Maintenance Guides for updated and detailed information.

External Event Notifications

The External Event Notification feature provides a way to notify RESTful services of events occurring in the ISR. It can be used in two ways: to notify third-party applications of new recordings, new sessions, and updates to existing sessions and recordings, or to notify new ISR components, acting as integrations to third-party platforms, of critical ISR events.

When communicating with third-party applications, the external event notification system propagates session and recording events to a separate server that interoperates with the ObserveIT Visual Session Recording solution (the ISR FACE feature). The data provided in these notifications allows the service to coordinate ISR recordings with ObserveIT screen capture recordings in order to provide simultaneous audio and video playback on applications such as the ISR Dashboard.

External event notifications may be configured with a series of default settings for interoperation with ISR FACE and the ObserveIT Visual Session Recording solution. For more information on the interaction between ISR FACE and ObserveIT Visual Recording solution, see the "Face Policy System" section of this guide. If you require more extensive external event notification configuration, contact your Oracle representative.

Events and Notifications

The majority of the external event notifications handling Events and queueing and sending Notifications is implemented in the ISR APIs.

Supported Events

The following Events are implemented in the current version of the external event notifications:

Event	Event Description
RECORDING_STARTED_EVENT	Recording has started
RECORDING_PAUSED_EVENT	Recording paused
RECORDING_RESUMED_EVENT	Recording has resumed
RECORDING_ENDED_EVENT	Recording has ended

Notifications

When external event notification is configured, upon successful completion of a command, the API queues a Notification for a successful result for the Event. If a command is unsuccessful, the API queues a Notification for an unsuccessful result for the Event.

The API supports configuration of external event notification on a per Route, per Account, and per Realm basis. Global Events, which are delivered regardless of the Route, Account, or Realm of the Event, are also possible. The API supports multiple Notification destinations for an Event such that an Event that matches each Route, Account, and Realm configured generates Notifications for each of these matching criteria. Additionally, multiple destinations for the same Route, Account, and Realm can be configured for each supported event type.

For example:

Route => destination A, and Account => destination B (Both Route and Account for the Event have destinations configured)

Account => destination A, and Account => destination B (Redundant Event sinks for Account)

Supported General Parameters

The following General Parameters may be included in an Event:

Parameter Name	Description
SESSION_ID	SIPREC Session ID
ANI	The caller
DNIS	The callee
ISR_UCID	ISR UCID
INGRESS_CALLID	Ingress Call ID
EGRESS_CALLID	Egress Call ID
UNKNOWN_UCID	Unknown UCID
CALL_ID	Call ID
RESULT	Result (success or fail)

Supported Recording-Specific Parameters

The following Recording-Specific Parameters may be included in a recording Event (all currently supported Events are recording Events, but that may not be true of future Events).

Parameter Name	Description
FILENAME	Filename
START_TIME	Start time
END_TIME	End time

Parameter Name	Description
DURATION	Duration
PAUSE_LENGTH	Pause length
PAUSE_SILENCE	Whether the pause contains silence
CALLING_AORS	Calling party AORs
CALLED_AORS	Called party AORs

Supported Special Parameters

In some circumstances, it is necessary to include parameters not normally determined during the event. However, it is desirable not to have to look up or process these parameters for every Event if they are not being requested. To accommodate this need, special parameter handlers are allowed during the construction of the Notification. There are two supported special parameters.

Parameter Name	Description
CALLING_AORS	Calling party AORs
CALLED_AORS	Called party AORs

External Event Notification and Parameters Example

The following REST command shows an example of a **RECORDING_STARTED_EVENT** Notification Destination, along with the Event type, called party AORs, and ingress call ID parameters for the Notification:

```
https://1.2.3.4:8443/Face/eventNotify?
event=AudioRecordingStarted&calledAors=sip:
7654321@1.2.3.1:5060&ucid=13-23132@1.2.3.4
```

Legacy RSS API Commands

Legacy RSS API Commands

This chapter provides information about invoking and implementing ISR API commands into your applications and 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 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>=<valueN>]`

API Commands

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

API Command	Description
recordStart	Causes recording of both Real-time Transport Protocol (RTP) streams to start.

Legacy RSS API Commands

API Command	Description
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.



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 is 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`.

ucidSource

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 are matched against in the order: `isr_ucid`, `ingress_callid`, and `egress_callid`.

Setting and Using the UCID with SIPREC SPL

The ISR provides two SPL implementations to address populating the X-ISR-UCID field for the purpose of API recording access and control. One method properly sets the field with a provided UCID/GUID known as "AvayaCiscoUCID64". The other method adds the value as a field in the SIPREC extension data known as "SipHeaderExtensionMetadata.1.2".

Upgrading From ISR Version 5.1M5 or Earlier

The ISR populates the primary recording X-ISR-UCID field regardless of SPL implementation. This may impact expected behaviors when the SRC/SBC is configured to use both the "AvayaCiscoUCID64" and the "SipHeaderExtensionMetadata.1.2" SPL.

If the expectation exists, when upgrading from ISR Version 5.1M5 or earlier, that a third-party UCID value populates the ISR UCID using the above two SPL configurations to pull Communication Session (CS) UCID and Call ID values into the Recording Session (RS) extension data, note the following order of precedence.

When the SRC/SBC is configured for both the "AvayaCiscoUCID64" and the "SipHeaderExtensionMetadata.1.2.spl" SPL the ISR's UCID is populated in the following order of precedence:

- X-ISR-UCID (or any other field name configured in the "<Sip><IsrUcidHeaderField>" tag of the RSS configuration file /cxc/vmgConfig.xml), for example:

```
<participant id="hq18GJs3TtJdhjPsfPNV8A=="
session="BYiC7uSZQGN3VQdzWI1HWw=="
  <nameID aor="sip:sipp@192.168.10.1">
    ...
    <extensiondata xmlns:apkt="http://acmepacket.com/siprec/
extensiondata">
    ...
      <apkt:header label="X-ISR-UCID">
        <value>X-ISR-
UCID-1-150323161627-1979582260@0a0af9f9</value>
      </apkt:header>
    ...
  </extensiondata>
```

- Call ID populated in the RS extension data, for example:

```
<participant id="hq18GJs3TtJdhjPsfPNV8A=="
session="BYiC7uSZQGN3VQdzWI1HWw=="
  <nameID aor="sip:sipp@192.168.10.1">
    ...
    <extensiondata xmlns:apkt="http://acmepacket.com/siprec/
extensiondata">
    ...
      <apkt:header label="Call-ID">
        <value>1-150323161627-1979582260@0a0af9f9</
value>
      </apkt:header>
    </extensiondata>
```

- Third-party UCID, for example:

```
<session id="/Bo3JDljRnZluCz1VhPHeg=="
  <associate-time>2016-03-15T01:33:46</associate-time>
  <extensiondata xmlns:apkt="http://acmepacket.com/siprec/
extensiondata">
    <apkt:ucid>00FA080200000F56E7667A;encoding=hex</apkt:ucid>
    <apkt:callerOrig>true</apkt:callerOrig>
    ...
```

To revert to the behavior prior to 5.2 in this very specific instance, "AvayaCiscoUCID64" SPL should be changed to no longer pull the "Call-ID" and "X-ISR-UCID" into the RS or change the header names if still required. To pursue other options, contact your Oracle account representative for more information.

Identifying a Session without a UCID

In a situation where none of the UCID values are known, ANI, DNIS, and the timeStamp of the start of the session can be used. 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.

recordStart

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

Required Input Parameters

Legacy RSS API Commands

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 ISR Dashboard.

OR

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 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.
<custom field name>	Value up to 255 characters	Custom Data Field value to be associated with the recording for a given Custom Data Field name.
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.

Parameter	Value	Description
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.

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 ISR REST API.

Request:

```
http://172.30.58.130:8080/IsrApi/rest/recordStart?
fileId=Test_Recording_2012-01-02_030405.pcm&ucidSource=isr_ucid&ucid=sdfIYASDs
9d8f7IYfds
```

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 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 (Use only if directed)

Legacy RSS API Commands

Parameter	Value	Description
ani	<FROM>	Incoming FROM value or Token (see the command <code>getToken</code>).
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 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.
<custom field name>	Value up to 255 characters	Custom Data Field value to be associated with the recording for a given Custom Data Field name.
agentId	<ID value of agent>	ID of the agent to which the call was transferred.

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 ISR REST API.

Request:

```
http://172.30.58.130:8080/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 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.

Legacy RSS API Commands

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

Request:

```
http://172.30.58.130:8080/IsrApi/rest/getFile?  
ucidSource=isr_ucid&ucid=sdfIYASDs9d8f7IYfds
```

Response:

The file.

getFileInfo

Query the 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.
fileUri	<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, of recording.

getFileInfo Sample Implementation

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

Request:

```
http://172.30.58.130:8080/IsrApi/rest/getFileInfo?
ucidSource=isr_ucid&ucid=sdfIYASDs9d8f7IYfds
```

Response:

```
<response>
  <Code>
    0
  </Code>
  <message>
    GetFileInfo Command Returned Successfully
  </message>
  <fileUri>
    http://172.30.58.130:8080/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.

Legacy RSS API Commands

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)

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.
<custom field name>	Value up to 255 characters	Custom Data Field value to be associated with the recording for a given Custom Data Field name.
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.
setNeverExpire	Range: true false	When true, ensures this recording is never deleted due to expiration.

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://172.30.58.130:8080/IsrApi/rest/updateFileInfo?
newFileId=Renamed_Recording_7347.pcm&agentId=11&ucidSource=isr_ucid&ucid=sdfIY
ASDs9d8f7IYfds
```

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 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.

Legacy RSS API Commands

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://172.30.58.130:8080/IsrApi/rest/deleteFile?ucidSource=isr_ucid&ucid=sdfIYASDs9d8f7IYfds
```

Response:

```
http://172.30.58.130:8080/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 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.

deleteRecording Sample Implementation

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

Request:

```
http://172.30.58.130:8080/IsrApi/rest/deleteRecording?ucidSource=isr_ucid&ucid=sdfIYASDs9d8f7IYfds
```

Response:

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

Legacy RSS API Commands

```
<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 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
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 ISR Dashboard.

Optional Input Parameters

Parameter	Value	Description
ucidSource	<source>	Indicates which ucid field to check: ISR_UCID, Ingress_callid, or Egress_callid.
fillWithSilence	<true false>	Whether or not recorded audio is padded with silence during a pause to preserve the accuracy of the duration of the call.

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://172.30.58.130:8080/IsrApi/rest/recordPause?
fileId=Test_Recording_2012-01-02_030405.pcm&ucidSource=isr_ucid&ucid=sdfIYASDs
9d8f7IYfds
```

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 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 (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 ISR Dashboard.

Optional Input Parameters

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

Return Values

Legacy RSS API Commands

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://172.30.58.130:8080/IsrApi/rest/recordUnpause?fileId=Test_Recording_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: <ul style="list-style-type: none"> account_name realm_label route_pattern priority is_recording_enabled percent_to_record

Parameter	Value	Description
		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://172.30.58.130:8080/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

Legacy RSS API Commands

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.
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://172.30.58.130:8080/IsrApi/rest/getRoutesForRealm?realmLabel=splan
```

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

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 ISR Channel and IP Address.
mixerip	<IP of ISR>	IP Address of theISR 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.

Legacy RSS API Commands

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://10.8.172.150:8080/IsrApi/sendIPCRCommand.jsp"
  namelist="command ucid ucidSource bookmarkLabel"/>
</subdialog>
```

Recording File Types/Formats Supported

The 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 ISR supports the following return codes:

Return Code	Status Text
0	<success>: [ACK

Return Code	Status Text
	<p>%command% Command Returned 'Successfully', where <command> equals one of the supported commands.</p> <p>This file has been renamed by the recorder, please use the following new filename for future commands: %filename%</p> <p>]</p>
-1	NACK
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	<p><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%]</p>
12	<p><record state transition error>: [Recording has already started on this channel. Recording is not paused, cannot unpaue 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 unpaue recording.]</p>
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	<p><RecordAndSave route error>: [Couldn't find route information for ANI %ANI% DNIS %DNIS% or route is not RecordAndSave application type</p>

Legacy RSS API Commands

Return Code	Status Text
	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.
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.

Troubleshooting

This section provides the information required to troubleshoot your 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 ISR.

Issue	Resolution
I get a busy message.	<ol style="list-style-type: none">1. Make sure your 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.	<ol style="list-style-type: none">1. Double check your ISR settings in the vmgConfig.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 ISR home directory, are the files:

`/exc_common/ISR/ApiLog/`

`/exc_common/ISR/ISRLogs/`

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

vmgConfig.xml

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

`/exc/`

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 ISR service requires a restart for the changes to take affect.

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 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 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 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 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.

RecordStart and RecordEnd Sample Implementation - VXML

The following is an example of how to invoke the **recordStart** and **recordEnd** commands via the 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://10.8.172.150:8080/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://10.8.172.150:8080/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 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://10.8.172.150:8080/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 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://10.8.172.150:8080/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 ISR VXML API.

```

<var name="ucid" expr="'s87dfKDSAFd8f3esaneb'"/>
<var name="ucidSource" expr="'ingress_callid'"/>
<var name="command" expr="'DeleteFile'"/>
<subdialog name="delRecording"
  src="http://10.8.172.150:8080/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 ISR VXML API.

```

<var name="ucid" expr="'s87dfKDSAFd8f3esaneb'"/>
<var name="ucidSource" expr="'ingress_callid'"/>
<var name="command" expr="'DeleteRecording'"/>
<subdialog name="delRecording"
  src="http://10.8.172.150:8080/IsrApi/sendIPCRCommand.jsp"
  namelist="command ucid ucidSource"/>
</subdialog>

```

RecordPause Sample Implementation - VXML

VXML API Commands

The following is an example of how to invoke the **recordPause** command via the 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://10.8.172.150:8080/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 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://10.8.172.150:8080/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://10.8.172.150:8080/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>

```

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 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 ISR Channel and IP Address.
mixerip	<IP of ISR>	IP Address of the 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.

VXML API Commands

Parameter	Value	Description
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://10.8.172.150:8080/IsrApi/sendIPCRCommand.jsp"
  namelist="command ucid ucidSource"/>
</subdialog>
```