CHAPTER 2

Cisco VoiceXML Troubleshooting

Revised: June 20, 2007, OL-11175-01

This chapter describes some of the troubleshooting techniques for the Cisco VoiceXML features. For a list of the latest troubleshooting FAQs, go to the developer support website at www.cisco.com/go/developersupport/.

Debugging Cisco VoiceXML Applications

To debug Cisco VoiceXML applications at the gateway level, refer to the *Cisco IOS Tcl IVR and VoiceXML Application Guide* for your Cisco IOS Release.

This section describes troubleshooting at the script level. To troubleshoot Cisco VoiceXML scripts, enable the **debug vxml error** and **debug vxml puts** commands on the gateway. **debug vxml error** displays all errors on the console, and **debug vxml puts** prints debugging statements used with the <log> element in the VoiceXML document.

<cisco-puts>

<cisco-puts> is a specific Cisco debugging element that is similar to the <log> element in the *VoiceXML 2.1* W3C Candidate Recommendation (June 13, 2005). The output from both elements is a log or debug message that may contain a combination of text and <value> elements.

Note In Cisco IOS Release 12.4(11)T and later releases, the <cisco-puts> and <cisco-putvar> elements are obsolete. Use the <log> and <value expr> elements for logging and debugging in these releases.

debug vxml puts

Note Use the **debug vxml** or **debug vxml puts** commands to debug Cisco VoiceXML scripts that use the <log> element.

Examples

1. This is an example of **debug vxml puts** to provide the output for <log>.

```
<log>    AccountInfo is  <value expr="session.c
```

```
om.cisco.proto_headers['AccountInfo']"/>
    </log>

<log>DEBUG: The ASR server matches the user voice input successfully
</log>
```

2. Here is a sample script for <cisco-data>:

```
<vxml version="2.0">

<form>
   <var name="output" expr="123"/>

   <block>
      <cisco-data src="http://server1/cgi-bin/ciscodata.tcl" name="MyData"
method="post" namelist="output"/>

      <log> The data sent is as follows:
         <value expr="MyData"/>
      </log>
   </block>
</form>
</vxml>
```

The output from **debug vxml puts** is a string that is specified by <log> in the script and the output is shown below:

```
Mar  1 03:17:22.551:   The data sent is as follows:

Hi Ray!
You just received data through cisco-data element.
Have a nice day.............
```

<cisco-debug>

Use <cisco-debug> to debug only a specific application. To disable debugging messages for all VoiceXML applications except the specific VoiceXML application you wish to debug, use the <cisco-debug> element in the VoiceXML document in conjunction with the **debug condition application voice** command.

See the *Cisco IOS cl IVR and VoiceXML Application Guide* for information on **debug** commands.

Note Add <cisco-debug> to the VoiceXML document for the application you want to debug, before you use Cisco IOS debug commands to debug that specific application.

For example:

Step 1 Turn on global debug for the areas you want to debug. For example:

```
debug vxml application
debug vxml trace
```

Note If you do not proceed with step 2 and end your task with step 1, you will see system messages for all the applications regardless of conditional debug being turned on or off.

Note The **debug condition application voice** command filters debugging output for only the **debug vxml** and **debug http client** commands. However, it does not filter ouptput for the **debug vxml error**, **debug vxml background**, **debug http client error**, or **debug http client background** commands.

Step 2 Add the <cisco-debug enabled = “true”/> and <cisco-debug enabled = “false”/> elements around the specific part of the VoiceXML document where you want to see debugging messages. For example:

```
<?xml version="1.0"?>
                <vxml version="1.0" application="root.vxml">
                    <form>
                        <block>
                            <cisco-debug enabled = "true"/>
                            <prompt>
                                <audio src="welcome.au" caching="fast"/>
                            </prompt>
                            <cisco-debug enabled = "false"/>
                            <goto next="getExtension.vxml?"/>
                        </block>
                    </form>
                </vxml>
```

Step 3 Add conditional debugging to the specific application you want to debug. For example:

Three applications named myapp1, myapp2, and myapp3, all of which can be loaded by using the **call application voice** command, are shown below:

```
call application voice myapp1 http://server1/vxml/test1.vxml
call application voice myapp2 http://server2/vxml/test2.vxml
call application voice myapp3 http://server3/vxml/test3.vxml
```

To debug only one of the applications, for example myapp1, use the **debug condition application voice** command to disable debug messages for the applications myapp2 and myapp3.

```
debug condition application voice myapp1
```

Note Debugging for myapp1 is performed for only those debug areas that have been enabled in step 1. Debugging for the specific session must be enabled through the <cisco-debug> tag as shown in step 2.

CallID and GUID in Debug Messages

The output debug messages are of the form //<callid>/<guid>/. If the information in these messages is not available at certain times, for example during call setup, the output debug messages are displayed as //-1/xxxxxxxxx/ or //-1//. Use the **voice call debug full-guid** Cisco IOS command to display guid in short or long format.

Note CallID and GUID values are still retained after the incoming call leg ceases.

Example

```
<log> callid is  <value expr="session.telephone.com.cisco.callid"/>
```

```
</log>

<if cond="session.telephone.com.cisco.callid == ''">
       <log> CALLID WAS NOT RECEIVED</log>
       <assign name="testResult" expr="'failed'"/>
</if>

<log> guid is  <value expr="session.telephone.com.cisco.guid"/>
</log>
```

The output is:

```
*Mar  1 04:42:07.558:   callid is 15

*Mar  1 04:42:07.558:   guid is C44404A9-151A-11CC-8066-B2BF937DE628
```

Error Events

Enabling the **debug vxml error** command displays a list of possible error events on the console. For a list of error events, see the Events and Errors section.

Some of the possible errors generated with the **debug vxml error** command enabled are:

error.badfetch

Possible Causes	Suggested Actions
• The VoiceXML interpreter throws this event when there is a failure in retrieving external components in the application. These external components can be VoiceXML documents, prerecorded files, or grammar files. • A badfetch error usually occurs when there is an error in fetching an external document.	• Verify that the external documents, audio prompts, or grammar files are available at the specified location mentioned in the URL. • If the external components are stored on a HTTP server, enable the **debug http client error** command. • If the external components are stored on a RTSP server, search for error.badfetch.rtsp.xxx, where xxx is a RTSP response code. For values of RTSP response codes, see RFC 2326 available on the IETF website.

error.semantic

Possible Causes	Suggested Actions
Logical errors such as referencing an undefined variable.	Verify that all variables referenced in the script are defined and valid.
Defining different grammar types in the same scope in the VoiceXML application.	Verify that only one grammar type is used at the time of recognizing user input.
Failure to define mandatory parameters in Cisco objects. For example, failure to define the account parameter in the authorize object will result in a semantic error.	Verify that all mandatory parameters are defined in Cisco objects used in the script.

error.unsupported.format

Possible Causes	Suggested Actions
A resource format is not supported by the platform.	Verify that all formats used in the script are supported by the specific platforms being used.

JavaScript/ECMA Script

When the <script> element or ECMA expression is used in a VoiceXML document, enable the **debug java** command for debugging.

```
debug java ?
apm2- JavaScript APM2 Utility Debugging
error- JavaScript Error Debugging
interpreter- JavaScript Interpreter Debugging
```