Asterisk integration

Getting Started

Once you have completed installing Voximal, you must configure it before attempting to place calls. The management and configuration procedures completely depend on the integration, as the reference VoiceXML browser components never directly access a configuration subsystem. Instead, all configuration parameters are passed to the initialization or resource creation functions for each VoiceXML browser component. The VoiceXML browser is configured through a text configuration file. Each line of this file defines a value for a property (parameter) applying to a given VoiceXML Engine; the possible property names are defined in the interface header files for the various components. The configuration file format is described in detail in a comment block at the top of the sample configuration file, /etc/voximald.conf.

NOTE:

Before any configuration update, make a backup copy of your voximald.cfg file.

Once you have made a backup copy, review the properties in the configuration file and modify their settings as appropriate. For an explanation of each property, refer to support. In most cases you can modify the following properties to get started (listed in priority order):

Configure the internet access to use http Configure the cache management Activate/ deactivate traces Configure the supported MIME types

Cache Management

In the Client configuration section, set client.inet.cacheDir to the directory where you want to store cached Internet files from URL fetches. Set the client.inet.cacheTotalSizeMB property to the desired size limit in megabytes for the Internet fetch cache. Processing VXI* produces standard traces under the /tmp/log.txt file. The trace file names are referenced in the /etc/vxmld.conf (old versions:/etc/openvxi/client.cfg) file. Their format can be found in the Log Files section of this document.

NOTE:

To disable the cache entry, set the maxage and maxstale properties to 0. This properties can be change in the default.xml file to globally disable the cache.

Webserver Configuration (optional)

The VXI* VoiceXML browser supports running against static content stored as local files as well as static or dynamic content served by a web server. The VoiceXML browser is generally used with a web server. Content can be VoiceXML documents, audio / images / video files (if supported by your version or license), and grammar files. When a web server delivers the content, it also provides the MIME

content type for the content and information about the desired caching behavior for the content. For the MIME content type for VoiceXML documents, a value of application/vxml+xml is recommended, but not mandatory, as the VoiceXML browser can use information within the XML document to verify it is a VoiceXML document. For prompt, recognize and record interfaces the MIME content type is more important. Support for the following audio formats is recommended for play and record operations, but not mandatory.

MIME Content	**Type Description**
audio/basic	RAW (headerless) 8kHz 8-bits mono mu-law [PCM] single channel. (G.711)
audio/x-alaw-basic	RAW (headerless) 8kHz 8-bits mono A-law [PCM] single channel. (G.711)
audio/x-wav	WAV (RIFF header) 8kHz 16-bits mono [PCM] single channel.
audio/X-WAV	WAV (RIFF header) codec GSM (6.10) 7-bits mono single channel.
audio/x-gsm	GSM (6.10) 7-bits mono single channel.
video/mp4	MP4 (Iso file) H.263 7 fps, mulaw/alaw/AMR-NB
video/3gp	3GP (Iso file) H.263 7fps, AMR-NB

The length of time the VoiceXML browser caches an item fetched from a web server is controlled by the caching values returned in the HTTP/1.1 header. The VoiceXML browser currently only supports the “Expires” header, which defines the date and time when the VoiceXML browser must discard the cached copy and re-fetch from the web server. Some web servers allow defining caching property defaults based on the MIME content type, useful for allowing caching of audio files but not dynamically generated VoiceXML documents, for example.

VoiceXML Asterisk module

The Vxi* VoiceXML browser installs the app_vxml Asterisk application module that uses the browser to execute VoiceXML pages.

Check the VXML application module For more information on applications, just type “core show applications” at the Asterisk CLI prompt. To show details of how you use that particular application in this file (the Asterisk Dial plan).

Type:

```
*CLI> core show applications
```

For example, the vxml application:

```
*CLI> core show application vxml
```

VoiceXML Asterisk Application

Application that launch a VoiceXML session in the asterisk channel and when complete, return control.

Syntax

```
Vxml([URL|Name|Number])
```

The URL can by set with different ways :

```
No parameter, the application will use the VoiceXML accounts (match called
number with accounts numbers) : Vxml()
Pass the URL as the appplication parameter. Example :
Vxml(file://tmp/test.vxml)
Pass the account name or the account number. Example: Vxml(test)
Pass the “@” to allocate a VoiceXML channel and pass the execution to this
dialplan extension. Example : Vxml(@600)
```

Set the VXML_URL variable before executing the vxml application. Don't pass any parameter, and use the configuration accounts section. If account number match with the called number, its URL(s) and parameters will be use.

The following describes how to execute a VoiceXML session.

Variables

Variables to set before or filled after the Vxml() execution :

VXML_URL

If the variable VXML_URL has been set when vxml runs, the value of that variable will be used for the URL unless the parameter is not set to the application.

VXML_LOCAL

Force the called number (variable in VoiceXML context session.connection.local.uri).

VXML_REMOTE

Force the caller number (variable in VoiceXML context, session.connection.remote.uri).

VXML_MARK

Allow to add his value/mark in the VoiceXML browser logs associated to this call/VoiceXML session.

VXML_ID

If the variable VXML_ID has been set when vxml runs, the VoiceXML session ID variable called “telephone.id” is set with this value (in the VoiceXML execution session context).

VXML_PARAM

If the variable VXML_PARAM (VXML_AAI is an alias) has been set when vxml runs, the value of that variable will be used as “telephone.param” (in the VoiceXML execution session context) and session.connection.aai.

VXML_RESULT

After execution, the VoiceXML result of <exit> tag and the property 'expr' are accessible by the variable VXML_RESULT.

VXML_ERROR

After execution, the VoiceXML application notify the error cause, if the VoiceXML session cannot be launched.

- VXML_ERROR=(empty) ; No error occurs.
- VXML_ERROR=INITALISATION ; Session refused, Asterisk module not connected to the VoiceXML browser.
- VXML_ERROR=LICENSE ; Session refused, license locking
- VXML_ERROR=ACCOUNT_LIMIT ; Session refused, max session for the account reached.
- VXML_ERROR=SPEECH ; Session refused, cannot allocate the Speech (ASR) ressource.
- VXML_ERROR=TTY/TDD ; Session refused, no TTD availabe.
- VXML_ERROR=BILLING ; Session refused, billing interface refuse the session.
- VXML_ERROR=INTERPRETER ; Session refused, critical error with the VoiceXML browser connection.

Examples

Example use with an URL parameter:

```
[incoming]
exten => s,1,Answer
exten => s,n,Wait(3)
exten => s,n,Vxml(http://links.i6net.com/index.vxml)
exten => s,n,Hangup
```

Example to catch the Vxml() errors :

```
exten => _X.,1,Vxml()
exten => _X.,n,NoOp(${VXML_ERROR})
exten => _X.,n,GotoIf($["${VXML_ERROR}" == ""]?hangup)
exten => _X.,n,Busy()
exten => _X.,n(hangup),Hangup()
```

Asterisk Online Help

Online help can be accessed by typing the following command at the CLI prompt: *CLI> help vxml

CLI Management Commands

Now your VXI* VoiceXML browser and Asterisk PBX are running, you can manage the VXML service using the Asterisk prompt *CLI>:

```
*CLI> vxml debug
```

Enable VoiceXML debugging for the Asterisk application.

```
*CLI> vxml debug interpreter all
```

Enable VoiceXML debugging for the interpreter application ().

```
*CLI> vxml no debug
```

This command disables VoiceXML debugging for the Asterisk application.

```
*CLI> vxml no interpreter debug
```

This command disables VoiceXML debugging for the interpreter application.

```
*CLI> vxml show cache / vxml cache show
```

Print the files in cache (if debug mode is enabled) and/or the number of file in the cache.

```
*CLI> vxml cache clear
```

Delete all the files in the cache.

```
*CLI> vxml cache purge
```

Delete all the files in the cache older than the maxage parameter set in the configuration.

```
*CLI> vxml show license
```

Use this command to show the license information.

```
*CLI> vxml reload
```

Reload the configuration.

```
*CLI> vxml show configuration
```

Use this command to show the configuration summary of VoiceXML interpreter.

```
*CLI> vxml show accounts
```

Show the accounts configured.

```
*CLI> vxml show account <number>
```

Show the accounts details of the account ID specified.

```
*CLI> vxml show statitiscs
```

Provides a dump statistics on VoiceXML interpreter

```
*CLI> vxml show dates
```

Provides a dump dates on VoiceXML interpreter

*CLI> vxml show sessions </code> Provides a dump sessions on VoiceXML interpreter.

```
*CLI> vxml show session
```

Provides a full dump session on VoiceXML interpreter.

```
*CLI> vxml originate chantype/number=application(parameters)
```

Originate an outgoing call, you can request to use a VoiceXML session.

The following entries are the Asterisk CLI commands for the VoiceXML browser.

VoiceXML example

Add extensions to the Asterisk dial plan /etc/asterisk/extensions.conf:

```
exten => 888,1,Answer
exten => 888,n,Wait(3)
exten => 888,n,Vxml(file:///root/example.vxml)
exten => 888,n,Hangup
```

You can create and edit the file /root/example.vxml with the GNU text editor, VI, for example.

```
# vi /root/example.vxml
```

NOTE:

This example will work if you have text-to-speech configured. If not, use a pre-recorded wav or gsm file to replace the “Hello world!” text by an <audio> tag. For more information, see the format extensions supported by Asterisk.

```
<?xml version="1.0"?>
<vxml version = "2.0" xmlns="http://www.w3.org/2001/vxml">
 <form>
  <block><audio src=”hello.wav”/></block>
 </form>
</vxml>
```

Save the file in the same directory as the VoiceXML script (relative reference in this example).

Reload the extensions configuration with:

```
CLI> extensions reload
```

Call the service by calling:

```
SIP:888@<your server address>
```

Troubleshooting (for Support)

This chapter covers troubleshooting procedures for the VoiceXML Browser, describing some basic techniques that can be used when working with Vxi.

Collecting Information for Technical Support As part of the process of reporting problems, download log files and core files from the VoiceXML Browser and send them to I6NET, together with the current configuration files.

```
/tmp/log.txt (default configuration)
```

Log Files

The log files contain information about the operation of the VoiceXML Browser. The file is /tmp/log.txt, which details the VoiceXML processing on the VoiceXML Browser. If a failure occurs and you need to contact I6NET support (at support@i6net.com), they may ask you to activate traces to allow analysis of the system functions and make a complete appraisal of the problem. To do so you must follow these

procedures:

Edit the configuration file client.cfg in /etc/openvxi/. The levels are defined by these lines which are the# API/general log traces for each component:

```
client.log.diagTag.2000                    VXIInteger  0
client.log.diagTag.2001                    VXIInteger  0
client.log.diagTag.3000                    VXIInteger  0
client.log.diagTag.3001                    VXIInteger  0
client.log.diagTag.3002                    VXIInteger  0
```

The ranges are associated to different interfaces:

```
200x:  Cache interface
300x:  Internet HTTP interface
400x:  ECMAscript interface
500x:  Prompt interface
600x:  Recognize interface
700x:  Telephony/Session interface
800x:  XML Interpreter
```

```
900x:  Object interface
1000x: Main/Client application
```

To enable a level, set the 1 value and 0 to disable. To validate the modifications, the VoiceXML interpreter must be restarted. A simple way to do this is to stop and restart this application with the Asterisk and VoiceXML script:

```
# /etc/init.d/asterisk stop
# /etc/init.d/vxml stop
# /etc/init.d/vxml start
# /etc/init.d/asterisk start
```

The log file is generated in the temporary directory, and is named log.txt. The location and filename are configurable. To purge the file type:

```
# > /tmp/log.txt
```

NOTE:

1. Never delete the /tmp/log.txt directly, otherwise you should restart the VoiceXML browser to generate a new one.
2. An Apache/PHP script exists generate the traces from a standard Internet browser (Internet Explorer, Mozilla/firefox…). Get it from our web site
3. At the end of the trace record, don’t forget to stop it to recover optimal real time function. With the V4.x release, you can dynamically enable/disable the interpreter traces with an Asterisk *CLI> command.

Full traces:

```
*CLI> vxml debug interpreter
```

Disable the interpreter traces:

```
*CLI> vxml no debug interpreter
```

Current Sessions To determine how many VoiceXML sessions are currently active on the system, use the statistics dump on the CLI. This command displays the current number of sessions on the VoiceXML Browser:

```
*CLI> vxml show statistics
```

File descriptors To find out how many file descriptors are being used follow this:

Find out program PID:

```
# ps -ef | grep asterisk
```

Find out program PID (other command):

```
# pidof asterisk
```

List of files opened by PID (details):

```
# ls -l /proc/[PID]/fd
```

List of files opened by PID (counter):

```
# ls -l /proc/[PID]/fd | wc -l
```

More information about file descriptors:

```
# lsof | grep "[PID]"
```