ASTERISK CONFIGURATION

DOCUMENTATION REPORT

**Topic: Sip.conf**

Sip.conf is the configuration file for initializing user settings.



If you look through the *sip.conf.sample* file in the *./configs* subdirectory of your Asterisk source you will notice a wealth of options available. Fortunately, the default options are normally all you need, and therefore you can create a very simple configuration file that will allow most standard SIP telephones to connect with Asterisk.

Sample Sip.conf file which configures two users 1000 and 1001

[1000]

type = friend

host = dynamic

secret = 1000

[1001]

type = friend

host = dynamic

secret = 1001

Type Description:

type = Description

peer Match incoming requests to a configuration entry using the source IP address and port number.

user Match incoming requests to a configuration entry using the username in the From header of the SIP

request. This name is matched to a section in *sip.conf* with the same name in square brackets.

friend This enables matching rules for both peer and user. This is the setting most commonly used for SIP phones.

Host Description:

The host option is used when we need to send a request to the telephone (such as when we want to call someone). Asterisk needs to know where the device is on the network. By defining the value as dynamic, we let Asterisk know that the telephone will tell us where it is on the network instead of having its location defined statically.

Command: sip show peers, displays registered user.

**Topic: Asterisk CLI Commands**

Start asterisk server:

service asterisk start, asterisk -r

Stop Asterisk Service:

core stop now

**Topic: Extension.conf**

When Asterisk receives a call connection, whether an incoming call from outside, or from an internal extension, that call belongs to a *context*. **Which context** the call belongs to depends on what [**channel**](http://www.voip-info.org/wiki/view/Asterisk+channels) the call came in on.

**Topic: Configuration Files**

**Example 1: Dial Application**

This application dials a phone number with the destination channel specified

Syntax : Dial(Channel/Exten,Time)

Channel : The Channel of the destination

Time: Maximum time to wait

**Step 1: Configure Sip.conf**

Here two users of extension 1000 and 1001 are configured

[1000]

type = friend

host = dynamic

secret = 1000

[1001]

type = friend

host = dynamic

secret = 1000

**Step 2: Configure Extension.conf**

[Default]

exten=>1001,1,Dial(SIP/1001,10)

same => 1001,2,Hangup()

Above in the extensions.conf when any user dials the number 1001, he is routed to the extension.

**Step 3: Xlite, Express Talk**

Configure the Sip account settings with corresponding username, password fields, domain(IP of asterisk server) as configured in sip.conf file.

**Example 2: Parse an URL**

**Description**

CURL(url[|post-data])  
url - URL to retrieve  
post-data - Optional data to send as a POST (GET is default action)

**Return value**

Returns the resulting string/page.

**Configuration**

Make sure curl is on your system (on "Ubuntu 8.04 server" users can run apt-get install libcurl4-dev). Then in the asterisk sources re-run the ./configure script and check if it detects curl.  
Run make menuselect and look under the Dialplan Functions for the func\_curl entry, making sure there are no XXX's there before compiling and installing asterisk.

**Example**

exten => s,1,Set(foo=${CURL(http://somewhere.com/somepage.html?x=5&y=4)})exten=>s,2,Verbose(foo)

Above example retrieves the response of the url page and displays it on the command prompt.

Example 3: Voice Mail Sample

The **VoiceMail()** applications takes two parameters:

1. **Mailbox**
   * This parameter specifies the mailbox in which the voice mail message should be left. It should be a mailbox number and a voice mail context concatenated with an at-sign (**@**), like **6001@default**. (Voice mail boxes are divided out into various voice mail context, similar to the way that extensions are broken up into dialplan contexts.) If the voice mail context is omitted, it will default to the **default** voice mail context.
2. **Options**
   * One or more options for controlling the mailbox greetings. The most popular options include the u option to play the unavailable message, the **b** option to play the busy message, and the **s** option to skip the system-generated instructions.

Syntax: Template to configure mailbox

|  |
| --- |
| mailbox=>pin,full name,email address, short email address, mailbox options |

Example:

|  |
| --- |
| [vm-demo]  6001 => 8762,Alice Jones,alice@example.com,alice2@example.com,attach=no|tz=central|maxmsg=10  6002 => 9271,Bob Smith,bob@example.com,bob2@example.com,attach=yes|tz=eastern |

Difference Between VoiceMail and VoiceMailMain :

Command : VoiceMail( VoiceMailNumber@Context,u)

Here we register a voice mail on the voice mail box of the user registered in voicemail.conf

Command : VoiceMailMain(Context)

This command is used to retrieve the voice mail of the user when he enters the correct password.

The following program is written in extension.conf

|  |
| --- |
| [users]  exten => 6000,1,Answer(500)  exten => 6000,n,Playback(hello-world)  exten => 6000,n,Hangup()    exten => 6001,1,Dial(SIP/demo-alice,20)  exten => 6001,n,VoiceMail(6001@vm-demo,u)    exten => 6002,1,Dial(SIP/demo-bob,20)  exten => 6002,n,VoiceMail(6002@vm-demo,u)    exten => 6500,1,Answer(500)  exten => 6500,n,VoiceMailMain(@vm-demo) |

6001: If Alice doesn’t answer the call, drop a voice mail

6002 : If Bob doesn’t answer the call, drop a voice mail

6003: Alice and Bob gets their voice mail using this extension by dialing their password.

**Example: Simple Auto – Attendant IVR**

**Extension.conf**

[demo-menu]

exten => s,1,Answer(500)

   same => n(loop),Background(press-1&or&press-2)

   same => n,WaitExten()

exten => 1,1,Playback(you-entered)

   same => n,SayNumber(1)

   same => n,Goto(s,loop)

exten => 2,1,Playback(you-entered)

   same => n,SayNumber(2)

   same => n,Goto(s,loop)

exten => 6598,1,Goto(demo-menu,s,1)

Goto: Statement

Goto statements switch the execution of the program to the desired context.

Syntax :Goto(Context, Extension, Priority)

In the above context we access background application infinitely within the same context.

Example-5:Control Statements