Command Line Control of an Asterisk Confbridge

Darryn Anton Jordan
jrddar001@myuct.ac.za

July 15, 2015

Contents

0.1 Introduction

This document details the method used to configure an Asterisk server and describes a C++ program used to record calls made in a ConfBridge using the Asterisk Manager Interface (AMI). The aim is to create a platform where node operators can group call through an Asterisk ConfBridge.

0.1.1 Software and Hardware Used

- Ubuntu 15.04
- Asterisk 13.1.0
- Code::Blocks 13.12
- Boost Asio Library
- SFLphone 1.4.1

0.2 Preparation and Set-up

0.2.1 Overview of system

Each node is assigned a SIP account and an extension number. This number can be used for one-on-one calls. Furthermore, a ConfBridge is created. This is essentially a conference call with a specific extension. Recording of the ConfBridge is achieved using the developed C++ program.

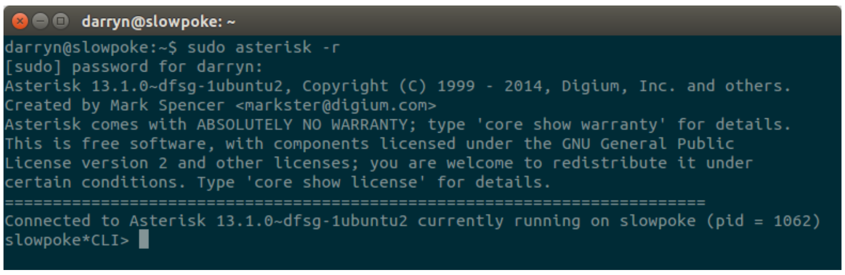
0.2.2 Asterisk Server

The server is required to have Asterisk installed:

```
sudo apt−get install asterisk
```

Once installed, the command line interface can be accessed as follows:

```
sudo asterisk −r
```



```
darryn@slowpoke: ~
darryn@slowpoke:~$ sudo asterisk -r
[sudo] password for darryn:
Asterisk 13.1.0~dfsg-1ubuntu2, Copyright (C) 1999 - 2014, Digium, Inc. and others.
Created by Mark Spencer <markster@digium.com>
Asterisk comes with ABSOLUTELY NO WARRANTY; type 'core show warranty' for details.
This is free software, with components licensed under the GNU General Public
License version 2 and other licenses; you are welcome to redistribute it under
certain conditions. Type 'core show license' for details.
=========================================================================
Connected to Asterisk 13.1.0~dfsg-1ubuntu2 currently running on slowpoke (pid = 1062)
slowpoke*CLI>
```

Figure 1: Asterisk CLI

The Asterisk server is configured by editing four *.conf* files:

- sip.conf - manage SIP accounts and set the server IP.
- extensions.conf - control what happens when extensions are dialled.
- confbridge.conf - configure a confbridge.
- manager.conf - allows control through the AMI.

These files are located at:

```
/etc/asterisk
```

These files must be replaced with the provided ones, in order to allow recording. It is recommended to make a copy of the original contents before replacing.

0.2.3 Important CLI Commands

- Whenever a *.conf* file is altered, the *reload* command must be used to refresh the server.
- To restart the server, use *core restart now.*
- View users present in a confbridge, *confbridge list.*
- Commands can be made from terminal, without entering the CLI:

  ```
    sudo asterisk −rx "command"

  For example:
    sudo asterisk −rx "confbridge list"
  ```

0.2.4 Configuration File Descriptions

sip.conf

```
[general]
bindaddr=0.0.0.0:5060            ;listen on IPv4 wildcard, UDP default port
localnet=127.0.0.1/255.255.255.0 ;server IP, must be changed accordingly

[111]                            ;account username/extension number
type=friend                      ;account can make and recieve calls
host=dynamic                     ;dynamic IP address
secret=123                       ;account password

[222]                            ;create as many accounts as needed
type=friend
host=dynamic
secret=234

[333]
type=friend
host=dynamic
secret=345

[444]
type=friend
host=dynamic
secret=456
```

extensions.conf

```
[default]

exten => 100,1,Answer()                 ;if 100 is dialed, server answers
;ask node opperator for name and announce arrival to others
exten => 100,2,Set(CONFBRIDGE(user,announce_join_leave)=yes)
exten => 100,3,ConfBridge(100,NeXtRad)  ;link opperator to confbridge

exten => 101,1,ConfBridge(100,NeXtRad)  ;link opperator to confbridge

;if other extension called, dial that number (one-on-one calls).
exten => _XXX,1,Dial(SIP/${EXTEN})
```

manager.conf

```
; By default asterisk will listen on localhost only.
[general]
enabled = yes
port = 5038
bindaddr = 127.0.0.1

[admin]
secret = admin
;deny = 0.0.0.0/0.0.0.0
;permit = 137.158.131.201/255.255.255.255
write = all,system,call,log,command,agent,user,config
```

confbridge.conf

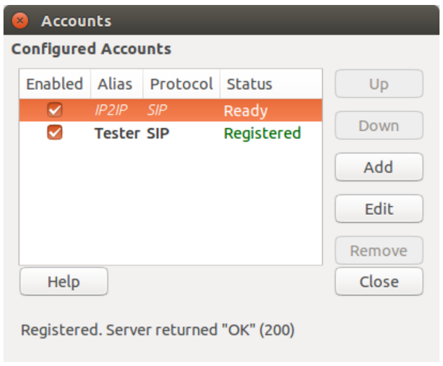
```
[general]

; --- ConfBridge User Profile Options ---
[default_user]
type=user
music_on_hold_when_empty=yes
```

```
; --- ConfBridge Bridge Profile Options ---
[default_bridge]
type=bridge

[NeXtRad]                                       ;confbridge name
type=bridge
record_file=/var/spool/asterisk/NeXtRad.wav     ;save location
record_conference=no                            ;no record from start
sound_leave=default                             ;turn leave sound off
sound_join=default                              ;turn join sound off

; --- ConfBridge Menu Options ---
[sample_user_menu]
type=menu
*=playback_and_continue(conf-usermenu)
*1=toggle_mute
1=toggle_mute
*4=decrease_listening_volume
4=decrease_listening_volume
*6=increase_listening_volume
6=increase_listening_volume
*7=decrease_talking_volume
7=decrease_talking_volume
*8=leave_conference
8=leave_conference
*9=increase_talking_volume
9=increase_talking_volume
```

0.2.5 SIP Clients

SFLphone 1.4.1

This client was used during testing of the asterisk server. Note that the status must be registered in order to work.



Accounts

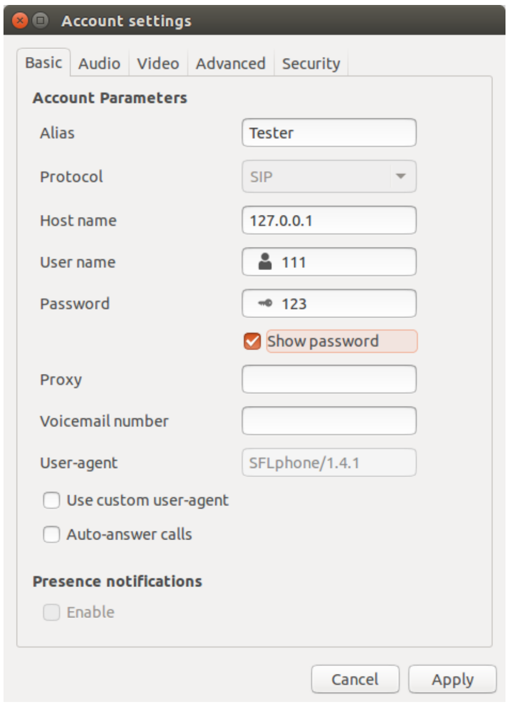
Configured Accounts

Enabled	Alias	Protocol	Status
✓	IP2IP	SIP	Ready
✓	Tester	SIP	Registered

Registered. Server returned "OK" (200)

Figure 2: Account Settings

Figures 3 and 4 display the account settings of a functioning account.



Account settings

Account Parameters

Alias	Tester
Protocol	SIP
Host name	127.0.0.1
User name	111
Password	123
	[✓] Show password
Proxy	
Voicemail number	
User-agent	SFLphone/1.4.1

[ ] Use custom user-agent

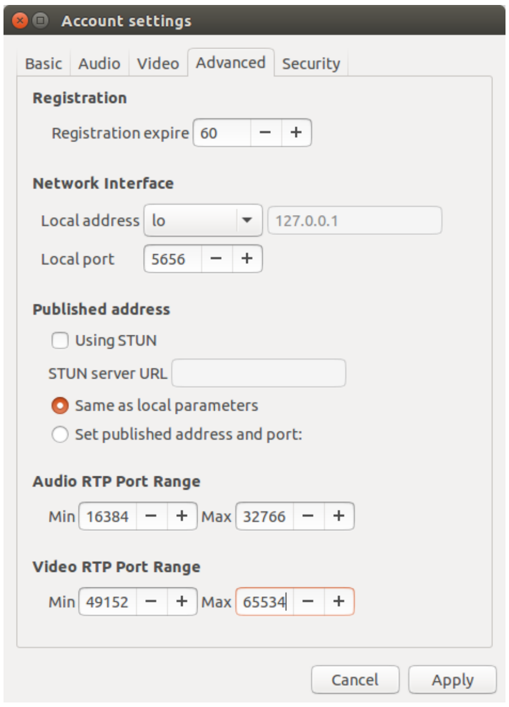
[ ] Auto-answer calls

Presence notifications

[ ] Enable

Figure 3: Basic Account Settings

NB Note that the port number is different to the one specified in the *sip.conf* file.


Account settings

Basic | Audio | Video | Advanced | Security

Registration

Registration expire: 60

Network Interface

Local address: lo — 127.0.0.1

Local port: 5656

Published address

[ ] Using STUN

STUN server URL:

[✓ Same as local parameters]

Set published address and port:

Audio RTP Port Range

Min 16384 Max 32766

Video RTP Port Range

Min 49152 Max 65534

Figure 4: Advanced Account Settings

To test the server confbridge, dial 100 using SFLphone. The user is prompted to speak his/her name and then press the hash key. The user is announced to any other users present in the confbridge and the conference call commences.

Linphone 3.6.1

Linphone is a cross-platform SIP client (iOS, Android, Windows, OS X and Linux). It has been tested to work perfectly with this system on iOS and Linux. The set-up is extremely similar to the method described above. Friendly reminder: port number is 5656. (NOT 5060)

0.3 C++ Confbridge Recorder

Upon compiling the code in Code::Blocks, the following console application will appear:

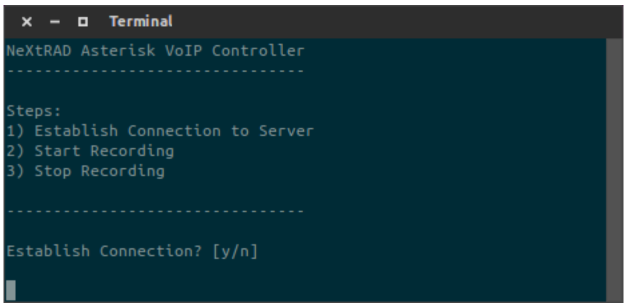
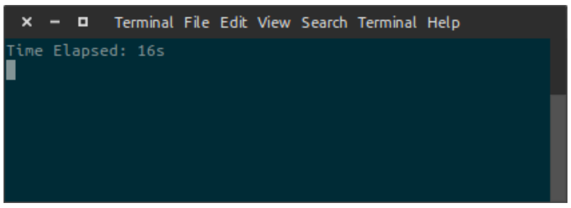

```
NeXtRAD Asterisk VoIP Controller
--------------------------------

Steps:
1) Establish Connection to Server
2) Start Recording
3) Stop Recording

--------------------------------

Establish Connection? [y/n]
```

Figure 5: Welcome Screen

The user is then breifly shown the background commands, afterwhich a prompt to start recording appears. If the program is successful in initiating recording, the output shown in Figure 6 will appear.

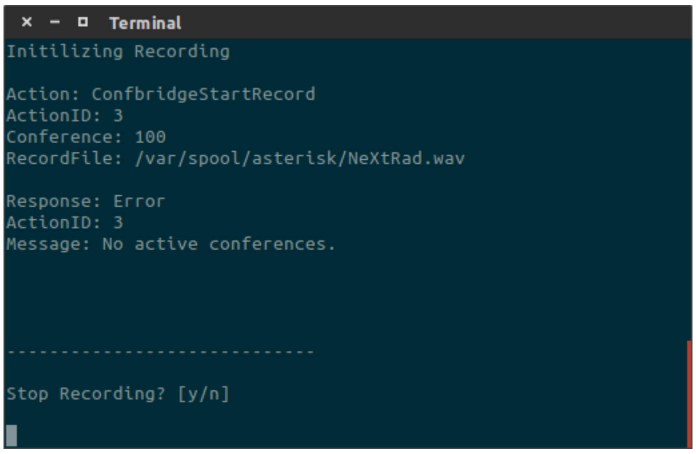


```
Time Elapsed: 16s
```

Figure 6: Recording Success

During development, the only location which Asterisk had permission to store recorded files was: /var/spool/asterisk
It is possible to store the recordings in any folder, if the folder is given full permissions using terminal: *chmod 777 /path/to/folder*

NB Note that a confbridge only exists once a user is present. Thus, a user must be in the confbridge before recording can take place. The following error occurs if no users are present:



```
Initilizing Recording

Action: ConfbridgeStartRecord
ActionID: 3
Conference: 100
RecordFile: /var/spool/asterisk/NeXtRad.wav

Response: Error
ActionID: 3
Message: No active conferences.




----------------------------

Stop Recording? [y/n]
```

Figure 7: Confbridge Error

Appendix A: Code Listing

```
//includes
#include <boost/asio.hpp>
#include <iostream>
#include <string>
#include <stdio.h>
#include <ctime>

//namespaces
using namespace boost::asio;
using namespace std;

//function declarations
void login();
void start();
void stop();
void connect();
void logout();
void beep();
string getTimeDateStamp();
int getElapsedSeconds(int startTime);

//global variables
char buff[160];
char option;

time_t rawtime;
struct tm *timeinfo;

io_service service;
ip::tcp::socket sock(service);

//functions
void clearBuffer()
{
    for (int i = 0; i < 161; i++)
    {
        buff[i] = ' '; //clear all elements of the array
    }
}

void connect()
{
    system("clear\n"); //clear console
    cout << "Attempting to establish connection..." << endl << endl;
    ip::tcp::endpoint ep(ip::address::from_string("127.0.0.1"), 5038); //
        define the endpoint at the known server address & port

    try
        {
            sock.connect(ep); //attempt to connect to the endpoint, if no
                exception is thrown the connection is successful
            login(); //login to AMI as admin

            system("clear\n");
            cout << "Start Recoding? [y/n]" << endl << endl;
            while(true) //wait for response
            {
                cin >> option;
                if (option == 'y')
                    {start();}
                else if (option == 'n')
```

```
                    {logout();} //close program
                else
                    {cout << "Invalid Response. Please use 'y' or 'n'" <<
                        endl;}
            }

        }
    catch (boost::system::system_error const& e) //exception was thrown,
        connection failed
        {
            cout << "Warning: could not " << e.what() << endl << endl;

            cout << "Retry Connection? [y/n]" << endl << endl;
            while(true)
            {
                cin >> option;
                if (option == 'y')
                    {connect();} //restart connection
                else if (option == 'n')
                    {logout();}
                else
                    {cout << "Invalid Response. Please use 'y' or 'n'" <<
                        endl;}
            }
        }
}

void write(string text)
{
    sock.write_some(buffer(text)); //write to the terminal
    cout << text;                  //echo to console
}

void read()
{
    clearBuffer();
    sock.read_some(buffer(buff)); //read terminal response to the buffer
        array
    cout << buff << endl;         //echo to console
}

void start()
{
    int oldTime, newTime = 0;
    time_t rawStartTime;

    system("clear\n");            //clear console
    cout << "Initilizing Recording" << endl << endl;

    write("Action: ConfbridgeStartRecord\n"); //start recoring
    write("ActionID: 3\n");
    write("Conference: 100\n");
    write("RecordFile: /var/spool/asterisk/" + getTimeDateStamp() + ".wav\n"
        ); //the only location available for recording
    write("\n");

    int startTime = time(&(rawStartTime));

    while(newTime < 31)
    {
        newTime = getElapsedSeconds(startTime);
        if (newTime > oldTime)
        {
            beep();
            system("clear\n");             //clear console
            cout << "Time Elapsed: " << newTime << "s" << endl;
            oldTime = newTime;
        }
    }
    stop();
}

void stop()
{
    system("clear\n");             //clear console
    cout << "Terminating Recording" << endl << endl;

    write("Action: ConfbridgeStopRecord\n"); //start recoring
    write("ActionID: 4\n");
    write("Conference: 100\n");
    write("\n");
    sleep(1);
    read();

    cout << endl << "--------------------------------" << endl << endl;

    cout << "Begin New Recording? [y/n]" << endl << endl;
    while(true)
    {
        cin >> option;
        if (option == 'y')
            {start();}
        else if (option == 'n')
            {logout();}
        else
            {cout << "Invalid Response. Please use 'y' or 'n'" << endl;}
    }
}

void login()
{
    write("Action: login\n"); //Login
    write("ActionID: 1\n");
    write("Username: admin\n");
    write("Secret: admin\n");
    write("\n");
    sleep(1);
    read();

    sleep(1);
    system("clear\n");

    write("Action: Events\n"); //Turn Event Logging Off
    write("ActionID: 2\n");
    write("EventMask: off\n");
    write("\n");
    sleep(1);
    read();
    sleep(1);
}

void logout()
{
    system("clear\n");

    write("Action: Logoff\n");
    write("ActionID: 5\n");
    write("\n");
    sleep(1);
    read();

    exit(0);
}

string getTimeDateStamp()
{
    char stringBuffer[80];
                                    //get date & time as a a string
    time(&rawtime);
    timeinfo = localtime(&rawtime);

    strftime(stringBuffer,80,"%d.%m.%Y-%I:%M:%S",timeinfo); //set date &
        time format

    string dateTime(stringBuffer);         //define date & time string
    return(dateTime);
}

int getElapsedSeconds(int startTime)
{
    time_t currentTime;
    int elapsed = time(&currentTime) - startTime;
    return elapsed;
}

void beep()
{
    write("Action: Originate\n"); //start recoring
    write("ActionID: 6\n");
    write("Channel: LOCAL/101\n");
    write("Application: Playback\n");
    write("Data: beep\n");
    write("\n");
}

void welcome()
{
    cout << "NeXtRAD Asterisk VoIP Controller" << endl;
    cout << "--------------------------------" << endl << endl;
    cout << "Steps:" << endl;
    cout << "1) Establish Connection to Server" << endl;
    cout << "2) Start Recording" << endl;
    cout << "3) Stop Recording" << endl << endl;
    cout << "--------------------------------" << endl << endl;
}

int main()
{
    welcome(); //display welcome screen

    cout << "Establish Connection? [y/n]" << endl << endl;
    while(true) //wait for response
    {
        cin >> option;
        if (option == 'y')
            {connect();} //chose yes - attempt connection
        else if (option == 'n')
            {break;} //chose no - close program
        else
```

```
            {cout << "Invalid Response. Please use 'y' or 'n'" << endl;} //
                request new response
    }
    return 0;
}
```