

# Command Line Control of an Asterisk Confbridge

Darryn Anton Jordan  
jrddar001@myuct.ac.za

June 29, 2015

# Contents

0.1	Introduction . . . . .	2
0.1.1	Software and Hardware Used . . . . .	2
0.2	Preparation and Set-up . . . . .	3
0.2.1	Overview of system . . . . .	3
0.2.2	Asterisk Server . . . . .	3
0.2.3	Important CLI Commands . . . . .	3
0.2.4	Configuration File Descriptions . . . . .	4
0.2.5	SIP Clients . . . . .	6
0.3	C++ Confbridge Recorder . . . . .	7

## 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 a 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
```

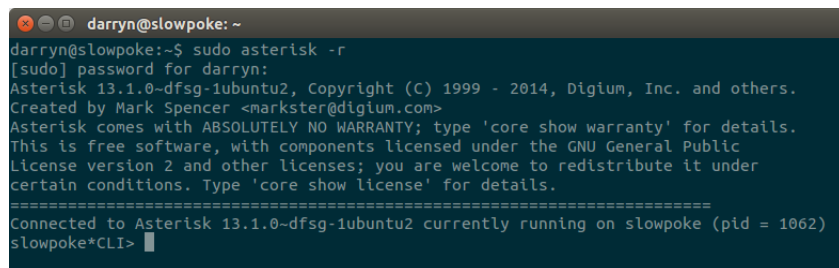
A screenshot of a terminal window titled 'darryn@slowpoke: ~'. The terminal shows the command 'sudo asterisk -r' being executed. It prompts for a password, then displays the Asterisk version (13.1.0-dfsg-1ubuntu2) and copyright information (1999-2014, Digium, Inc. and others). It also shows the warranty and license information. Finally, it indicates that the user is connected to Asterisk 13.1.0-dfsg-1ubuntu2 currently running on slowpoke (pid = 1062) and shows the prompt '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 receive 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 operator for name and announce arrival to others
exten => 100,2,Set(CONFBRIDGE(user,announce_join_leave)=yes)
exten => 100,3,ConfBridge(100,NeXtRad) ;link operator to confbridge

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

### manager.conf

```
[general]
enabled = yes                   ;enable AMI
port = 5038                     ;set port to listen on
bindaddr = 127.0.0.1           ;set IP

[admin]                         ;configure admin profile
secret = admin                 ;set admin password
;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

; --- 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.

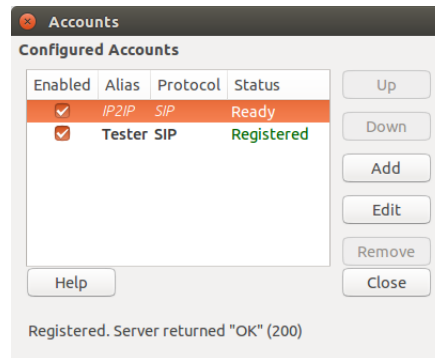


Figure 2: Account Settings

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

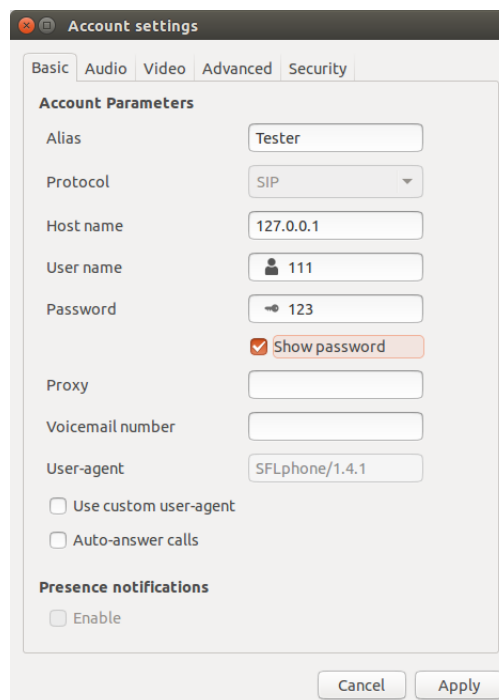


Figure 3: Basic Account Settings

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

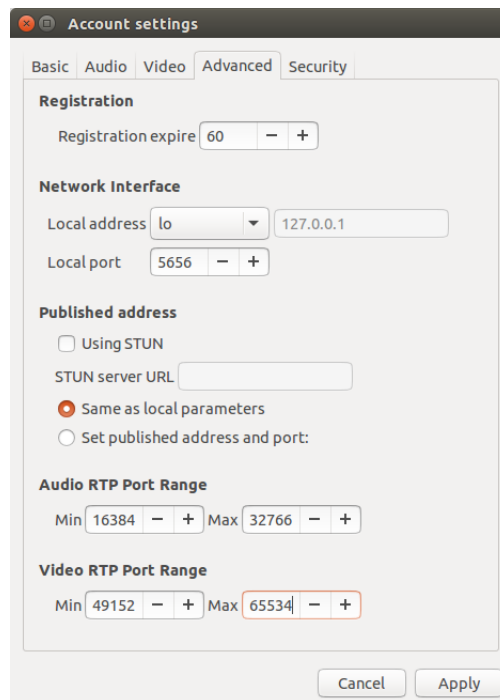


Figure 4: Advanced Account Settings

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:

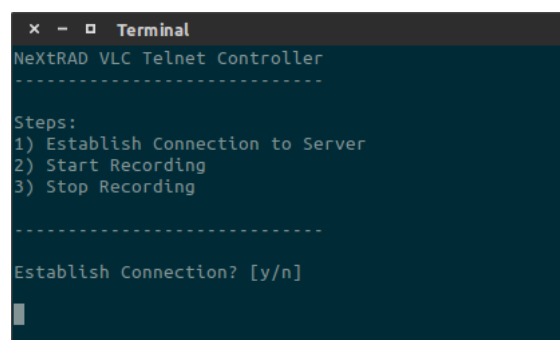
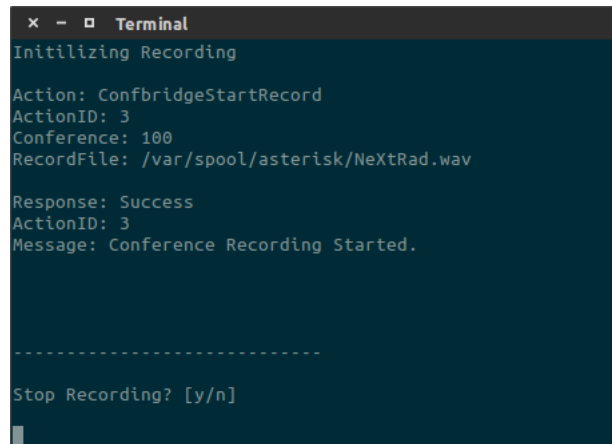


Figure 5: Welcome Screen



The user is then briefly shown the background commands, after which a prompt to start recording appears. If the program is successful in initiating recording, the output shown in Figure 6 will appear. During development, the only location which Asterisk had permission to store recorded files was: `/var/spool/asterisk`

A terminal window titled "Terminal" with a dark background. It displays the following text: "Initilizing Recording", "Action: ConfbridgeStartRecord", "ActionID: 3", "Conference: 100", "RecordFile: /var/spool/asterisk/NeXtRad.wav", "Response: Success", "ActionID: 3", "Message: Conference Recording Started.", a separator line of dashes, and "Stop Recording? [y/n]".

```
Terminal
Initilizing Recording

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

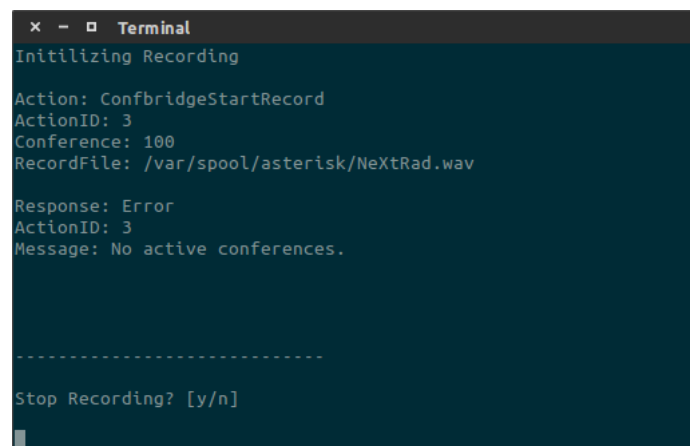
Response: Success
ActionID: 3
Message: Conference Recording Started.

-----

Stop Recording? [y/n]
```

Figure 6: Recording Success

**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:

A terminal window titled "Terminal" with a dark background. It displays the following text: "Initilizing Recording", "Action: ConfbridgeStartRecord", "ActionID: 3", "Conference: 100", "RecordFile: /var/spool/asterisk/NeXtRad.wav", "Response: Error", "ActionID: 3", "Message: No active conferences.", a separator line of dashes, and "Stop Recording? [y/n]".

```
Terminal
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();

//global variables
char buff[160];
char option;
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()
{
    system("clear\n"); //clear console
    cout << "Initilizing Recording" << endl << endl;

    write("Action: ConfbridgeStartRecord\n"); //start recoring
    write("ActionID: 3\n");
    write("Conference: 100\n");

    time_t rawtime; //
    struct tm * timeinfo; //
    char buffer[80]; //
    //get date & time as a a string
    time (&rawtime); //
    timeinfo = localtime(&rawtime); //

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

    string dateTime(buffer); //define date & time string

    write("RecordFile: /var/spool/asterisk/" + dateTime + ".wav\n"); //the
only location available for recording
    write("\n");
    sleep(2);
    read();

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

```

```

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

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(2);
    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(3);
    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(2);
}

```

```

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

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

    exit(0);
}

void welcome()
{
    cout << "NeXtRAD VLC Telnet 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;
}

```