



Transition of VoIP system from IPv4 to IPv6

Submitted by:
Varun Talwar
MacGregor
Mir

Project Readers:
Dr. M.H

Mr. Shahnawaz



Abstract

As we know, IPv4 is being transitioned to IPv6 slowly. VoIP systems would also have to undergo the necessary changes. The interesting point during this transition would be when some companies might have a mix of IPv4 and IPv6 network or an IPv4 VoIP system would be trying to connect to a VoIP system based on IPv6.

During the project, I wanted to create a Local Area network mixed with IPv4 and IPv6 and see their resulting effects on the VoIP system. This understandably would have an effect on the resources of the company as well, as they have to upgrade or use extra equipments. So I experimented with an open source telephony platform (Freeswitch) which had IPv6 capability.



Acknowledgment

I would like to express my gratitude to Dr. M.H. MacGregor for his help and guidance in order to complete the project.

A special thanks to Mr. Shahnawaz Mir for all the help extended by him in the MINT lab and the valuable technical guidance.



Table of Contents

Chapter 1 Introduction.....	Page 5
• IPv6 deployment issues	
• Project Objective	
Chapter 2 VoIP Protocols and Concepts.....	Page 8
• Real-Time Protocol(RTP)	
• Real-Time Control Protocol(RTCP)	
• H.323	
• Media Gateway Control Protocol(MGCP)	
• Session Initiation Protocol(SIP)	
• Skinny Client Control Protocol(SCCP)	
Chapter 3 FreeSWITCH.....	Page 22
Chapter 4 Network Design.....	Page 27
• SIP Communicator	
• SIP proxy server- FreeSwitch	
• Cisco 2821 router	
Chapter 5 Conclusion.....	Page 38
Appendix	
References	

Chapter 1 Introduction

In the past few years, the telecommunication industry has undergone the rapid changes in the way people and organizations communicate with each other. Many of these changes came from the explosive growth of the internet and its applications based on the Internet Protocol. The internet has become a universal means of communication.

Even though, IPv4 is a flexible and powerful mechanism. However, IPv4 is starting to exhibit limitations, not only with references to the need for raw increase of the IP address space, driven, for example, by new populations of user in large countries like China and India, along with new technologies with “always connected devices” (xDSL, Cable, PDAs, UMTS mobile telephone, etc), but also in reference of VoIP both in terms of the NAT issue as well as the QoS issue.

Third generation VoIP network constructs are just around the corner. These networks will be characterized by the following features:

- Full end to end IP based (specially IPv6-based).
- Fully accessible to any user in the world.
- SIP based for advanced signaling.
- Seamless integration with corporate enterprise networks from a protocol and security perspective.
- QoS enabled in the wireless LAN environment.
- Integrations with 3G cellular systems.
- Commercial grade service levels/reliability/security.
- Video conferencing support.
- End to end QoS enabled.

IPv6 deployment issues

Despite the initial expectations raised, IPv6 is clearly far from being extensively deployed and therefore it is too early to claim any complete success for it yet.

Various reasons for this are:

- The warnings regarding IPv4 address exhaustion have not yet materialized. By analyzing past data forecast that IPv4 addresses will hold out beyond 2030 unless new conditions arise that bring about a change in the current trend in address consumption such as a strong demand for addresses for mobile devices or the addition of a large number of users in China or India.
- Also, there have been technological obstacles to the success of IPv6. While basic IPv6 standards have been available for some time, the standardization process has not been smooth in some key areas like DHCPv6, Mobile IPv6.
- There have also been some changes made to the core specifications in recent years, such as the deprecation of sit-local addresses or updates to the programming interfaces.
- The biggest challenge is the requirement for applications using the socket interface to be ported to a new programming interface to be able to use IPv6 due to dependencies imposed by the socket interface on the specific protocol to be used.
- Other reasons have been inferior support for IPv6 compared to IPv4 in terms of both functionality and performance.

All these factors show the migration process will entail significant costs and complexities for networking organizations.

Project Objective

Since significant resources and number of issues are associated with migration to VoIP (IPv6 based). There is an excellent probability that not every organization



would be transferred to VoIPv6 at the same time. In fact, there would be a considerable transition time and thus, a mixed usage of VoIP systems based on IPv4 and IPv6, which might have different effects on the different VoIP systems depending upon the kind of communication hardware being used.

In this project, I tried to deploy a mix of VoIP system based on IPv4 and IPv6. The main hardware and software components involved were:

- Cisco 2821 router.
- Cisco 7941G IP phones.
- Freeswitch 1.0.5(Soft switch).
- Sip communicator (soft phone with IPv6).

Chapter 2 VoIP Protocols and Concepts

In a VoIP network, just like in a traditional telephone network requires two paths: a protocol stack that includes Real-time transport protocol (RTP) for audio path and one or more call control models that provide the signaling path.

Due to the time-sensitive nature of voice traffic, UDP/IP was the logical choice to carry voice. More information is needed on a packet by packet basis than UDP offered, however. So, for real time or delay sensitive traffic the Internet Engineering Task Force (IETF) adopted the RTP. VoIP rides on top of RTP, which rides on top UDP. Therefore, VoIP is carried with an RTP/UDP/IP packet header. We will look at some detail of the Real-Time protocol (RTP) which is used to carry the audio traffic in a VoIP network.

The most common call control models used are H.323, MGCP and SIP. For IPv6 SIP is the most important call control protocol. Also, we would be using a mix of call controls in our project which would be SCCP and SIP.

Real-Time Protocol

RTP is the standard for transmitting delay-sensitive traffic across packet based networks. RTP rides on top of UDP and IP. RTP gives receiving stations information that it is not in the connectionless UDP/IP streams. As shown in figure, two important bits of information are sequence information and time stamping. RTP uses the sequence information to determine whether the packets are arriving in order or not and it uses the time stamping information to determine the inter arrival packet time (Jitter).

Version	IHL	Type of Service	Total Length			
Identification			Flags	Fragment Offset		
Time To Live		Protocol	Header Checksum			
Source Address						
Destination Address						
Options				Padding		
Source Port			Destination Port			
Length			Checksum			
V=2	P	X	CC	M	PT	Sequence Number
Timestamp						
Synchronization Source (SSRC) Identifier						

Real-time transport header

RTP can be for media on demand as well as for interactive services such as Internet telephony. RTP consists of a data part and a control part, the latter called RTP control protocol (RTCP).

The data part of RTP is a thin protocol that provides support for applications with real-time properties such as continuous media (for example audio and video) including timing reconstruction, loss detection and content identification.

RTCP provides support for real-time conferencing of groups of any size within an internet. This support includes source identification and support for gateways such as audio and video bridges as well as multicast to unicast translators. It also offers QoS feedback from receivers to the multicast group, as well as, support for the synchronization of different media streams.

Using RTP is important for real-time traffic but a few drawbacks exist. The IP/RTP/UDP headers are 20, 8 and 12 bytes respectively. This adds up to a 40 byte header which is twice as big as the payload when using G.729 with two speech samples (20ms). We can compress this large header to 2 or 4 bytes by using RTP header compression (CRTP).

Real-time control protocol (RTCP)

The RTP control (RTCP) supplements RTP by handling the administrative and reporting aspects of an RTP multicast conference. RTCP is specified in RFC 1889 as part of RTP. Even though RTCP is designed to scale for large conferences, it is useful in a simple point to point VoIP call to provide QoS feedback from the receiver to the sender in each direction.

For large multicast conferences the bandwidth of RTP media tends to remain constant because only a few people can speak at a time even if thousands of people are listening. RTCP control information is sent from each participant to every other participant, so scalability is a big issue. If each participant sends a 100 byte packet every second, then a conference with 10,000 people causes each participant to receive 1 Mbps of control information. RTCP addresses this problem by transmitting packet less often as the number of detected conference participants increases. The RTCP algorithm limits control bandwidth to approximately 5 percent of the bandwidth in the media stream by default, although applications can adjust this amount as necessary.

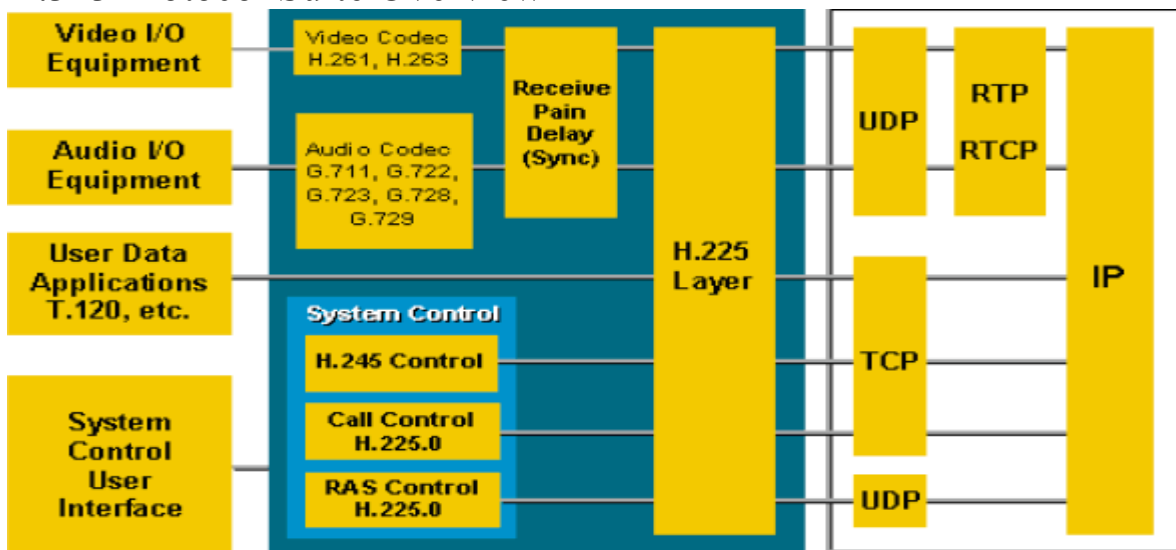
H.323

H.323 is an ITU Telecommunication standardization sector (ITU-T) specification for transmitting audio, video and data across an Internet Protocol network including the internet. When compliant with H.323, vendor's products and applications can communicate and interoperate with each other. The H.323 standard addresses call signaling and control, multimedia transport and control and bandwidth control for point to point and multipoint conferences. The H series of recommendations also specifies H.320 for Integrated Services Digital Network (ISDN) and H.324 for Plain Old Telephone Service (POTS) as transport mechanisms.

The H.323 standard consists of the following components and protocols:

Feature	Protocol
Call Signaling	H.225
Media Control	H.245
Audio Codecs	G.711, G.722, G.723, G.728, G.729
Video Codecs	H.261, H.263
Data Sharing	T.120
Media Transport	RTP/RTCP

H.323 Protocol Suite Overview



H.323 defines the following protocols:

- H.245 for capabilities exchange.
- H.225.0 for call setup.
- H.225.0 for registration, admission and status (RAS) control.

The real-time aspects of H.323 rely on User Datagram Protocol (UDP) while both the session-oriented control procedures and the data media type of H.323 use TCP.

In addition to the above protocols the following functional components make up the H.323 environment:

- **H.323 terminals** – This is an endpoint that provides real-time voice communications with other endpoints. It must be capable of transmitting and receiving G.711 (a-law and u-law) 64 kbps pulse coded modulation (PCM) encoded voice and may also support other encoded voice formats such as G.729 and G.723.1.
- **H.323 Gateway** – This is an optional type of endpoints that provides interoperability H.323 endpoints and endpoints located on a switched circuit network (SCN), such as the PSTN or an enterprise voice network. Ideally, the gateway is transparent to both the H.323 endpoint and the SCN based endpoint. The H.323 gateway performs the translation between audio, video and data formats, it performs conversion between call setup signals and procedures. Also, H.323 gateway is responsible for conversion between communication control signals and procedures.
- **H.323 Gatekeeper** – This is an optional component that provides call control support and services to H.323 endpoints. Although a gatekeeper is considered a distinct component it can be collocated with any other H.323 component. The scope of endpoints over which a gatekeeper exercises its authority is called a Zone. H.323 defines a one to one relationship between a zone and a gatekeeper. When a gatekeeper is included, it must perform the following functions:
 - Address Translation – Converts and alias address to an IP address.
 - Admission Control – Limits access to network resources based on call bandwidth restrictions.
 - Bandwidth Control – Responds to bandwidth requests and modifications.
 - Zone Management – provides services to registered endpoints.
 - Call Control Signaling – Performs call signaling on behalf of the

- endpoints (Gatekeeper-routed call signaling).
- Call Authorization – Rejects calls based on authorization failure.
- Bandwidth Management – Limits the number of concurrent accesses to IP network resources (Call admission control).
- Call Management – Maintains a record of ongoing calls.

Media Gateway Control Protocol (MGCP)

The second popular VoIP call signaling protocol is the Media Gateway Control Protocol (MGCP). MGCP is an IETF standard and is described as a centralized device control protocol with simple end points. It allows a central control component or call agent for remotely control various devices. It is referred to as a stimulus protocol because the endpoints and gateways can not function alone. MGCP incorporated the IETF Session Description Protocol (SDP) to describe the type of session to initiate.

MGCP defines a number of components and concepts.

- **Endpoints** – Endpoints represent the point of interconnection between the packet network and the traditional telephone network. Endpoints can be physical, representing a Foreign Exchange Station (FXS) port a channel in a T1 or they can be logical representing an attachment point to an announcement server for an example. To manage an endpoint the call agent must recognize the characteristics of an endpoint. For this reason the endpoints are categorized into several types, the being that the call agent will be configured to manage a type of endpoint rather than to manage each endpoint individually. An endpoint is identified by an identifier that consists of two parts: a local name of the endpoints in the context of the gateway by an @ sign. the RFC defines eight types of endpoints, they are as follows:
 - Digital Service Zero (DS0) – Represents a single channel (DS0) in the digital hierarchy. A Digital channel endpoint supports more than one connection.
 - Analog Line – Represents a client side interface such as FXS or switch



side interface such as Foreign Exchange Office (FXO) to the traditional telephone network. An analog line endpoint supports more than one connection.

- Announcement server access point – Represents access to an announcement server for example to play recorded messages. An announcement server endpoint may have only one connection.
 - Interactive voice response (IVR) access point – Represents access to an IVR service. An IVR endpoint has one connection.
 - Conference bridge access point – Represents access to a specific conference. Each conference is modeled as a distinct endpoint. A conference bridge endpoint supports more than one connection.
 - Packet Relay – Represents access that bridges two connections for interconnecting incompatible gateways or relaying them through a firewall environment. A packet relay endpoint has two connections.
 - Wiretap Access Point – Represents access for recording or playing back a connection. A wiretap access point endpoint has one connection.
 - ATM Trunk Side Interface – Represents a single instance of an Audio channel in the context of an ATM network and supports more than one connection.
- **MGCP gateways** – Gateways are clustering points for endpoints. Gateways handle the translation of audio between the switched circuit network and the packet network. A gateway interacts with one call agent only, therefore, it associated with one call agent at a time. The RFC defines seven types of gateways:
 - Trunk Gateway ISDN User part (ISUP) – supports digital circuit endpoints subject to ISDN signaling.
 - Trunk Gateway Multi frequency (MF) – Typically supports digital or analog circuit endpoints that are connected to a service provider and subject to MF signaling.
 - Network Access Server (NAS) – Supports connection to endpoints over which data (Modem) applications are provided.
 - Combined NAS/VoIP gateway – Supports connection to endpoints over



which a combination of voice and data access is provided.

- Access Gateway – Supports analog and digital endpoints connected to a PBX.
 - Residential Gateway – Supports endpoints connected to traditional analog interfaces.
 - Announcement Server- Supports endpoints that represent access to announcement services.
- **MGCP Call Agent** – A call agent of Media Gateway Controller (MGC) represents the central controller in an MGCP environment. It exercises control over the operation of a gateway and its associated endpoints by requesting that a gateway observe and report events. In response to the events, the call agent instructs the endpoints what signal, if any, the endpoint should send to the attached telephone environment. This requires a call agent to recognize each endpoint type that it supports. In addition to recognizing the signaling characteristics of each physical and logical interface that is attached to a gateway. A call agent uses its directory of endpoints and the relationships that each endpoint has with the dial plan to determine call routing. Call agents initiate all VoIP call legs.

Now that we understand the different components that make up the MGCP environment we need to look at some of the basic MGCP concepts. The basic MGCP concepts can be classified into three groups:

- **Calls and Connections** – Allow end to end calls to be established by connecting two or more endpoints. To establish a call, the call agent instructs the gateway that is associated with each endpoint to make a connection with a specific endpoint or an endpoint of a particular type. The gateway returns the session parameters of its connection to the call agent, which in turn sends these to the other gateway. With this method each gateway acquires the necessary parameters to establish RTP session between the endpoints. All connections that are associated with the same call will share a common call ID and the same media stream. At the conclusion of a call, the call agent sends a



delete connection request to each gateway. To create a multipoint call, the call agent instructs an endpoint to create multiple connections. The endpoint is responsible for mixing the audio signals.

- **Event and Signals** – Events and signals help the call agent to instruct the gateway of the basic as well as complicated call control and signaling procedures. The call agent informs the gateway of what events to monitor on an endpoint, what to do if an event occurs and when to generate a notification to the call agent. For example, an event is an analog line in an off-hook condition. Using signals, the call agent requests that the gateway to provide dial tone upon observing the off-hook event. Events and signals are assigned simple case insensitive codes. For example, the code for an off-hook transition event is “hd” and the code dial tone signal is “dl”.
 - Examples of events include – Continuity detection, Dual tone multi frequency (DTMF) digits, Fax tones, Hookflash, Modem tones, off-hook transition and on-hook transition.
 - Examples of Signals include – Answer tone, busy tone, call waiting tone, confirm tone, continuity tone, dial tone, ringing, ringback tone, DTMF tone, Network congestion tone, off-hook warning tone and so on.
- **Packages and Digit Maps** – Packages are collections of commonly occurring events and signal that are relevant to specific type of endpoint. For example, “offhook” an event and “dial tone” a signal are unique to managing subscriber lines. Consequently, they are associated with the “line” package. All events and signals are placed in exactly one package. The RFC defines ten packages as follows:



<i>Reference</i>	<i>Package Name</i>
G	Generic
M	Multifrequency
D	DTMF
T	Trunk
N	Network Access Server
L	Line
A	Announcement server
R	RTP
H	Handset
S	Script

Packages cluster events and signals by their relevance to various types of endpoints. Conceptually, gateways also cluster endpoints of different types. It is appropriate then to associate packages with gateways. The table below lists the gateways and identified the packages that are associated with them:

<i>Gateway Type</i>	<i>Package</i>
Trunk Gateway (ISUP)	G,D,T,R
Trunk Gateway (MF)	G,M,D,T,R
Network Access Server	G,M,T,N
Combined NAS/VoIP	G,M,D,T,N,R
Access Gateway (VoIP)	G,D,M,R
Access Gateway (VoIP & NAS)	G,D,M,N,R
Residential Gateway	G,D,L,R
Announcement Server	A,R

A digit map is a specification of the dial plan. When you download a digit map to a gateway for use on an endpoint, it allows the gateway to collect digits until the gateway either finds a match or concludes that the digit dialed could not possibly match a specification. When either condition occurs, the gateway notifies the call agent. Without a digit map, a gateway must notify the call agent on each digit dialed, which places a heavy burden on the call agent and the network connecting



the gateway and call agent.

This completes the discussion on the MGCP components and concepts. One other thing to know is the MGCP control commands. A call agent uses control commands or messages to direct its gateways and their operational behavior. Gateways use the following commands in response to requests from a call agent and for notifying the call agent of events and abnormal behavior:

- Endpoint Configuration (EPCF) – it identifies the coding characteristics of the endpoint interface on the line side of the gateway. The call agent issues the command.
- Notification Request (RQNT) – it instructs the gateway to watch for events on an endpoint and the action to take when they occur. The call agent issues the command.
- Notify (NTFY) – informs the call agent of an event for which notifications is requested. The gateway issues the command.
- Create connection (CRCX) – instructs the gateway to establish a connection with an endpoint. The call agent issues the command.
- Modify connection (MDCX) – instructs the gateway to update its connection parameters for a previously established connection. The call agent issues the command.
- Delete connection (DLCX) – informs the recipient to delete a connection. The call agent or gateway can issue the command and it is issued to advise that there are no longer resources to sustain the call.
- Audit endpoint (AUEP) – requests the status of an endpoint. The call agent issues the command.
- Audit connection (AUCX) – requests the status of a connection. The call agent issues the command.
- Restart in progress (RSIP) – notifies the call agent that the gateway and its endpoints are removed from service or are being placed back in service. The gateway issues the command.

Session Initiation Protocol (SIP)

SIP provides another framework for establishing, maintaining and terminating multimedia sessions (VoIP calls) with one or more participants. Sessions may be based on unicast, multicast or both communications. SIP operates on the principle of session invitations; SIP initiates sessions or invites participants into established sessions. Descriptions of these sessions are advertised via the Session and the Session Definition Protocol (SDP).

Multimedia sessions are established and terminated by the following services:

- User location Services.
- User capabilities services – Select the media type and parameters.
- User availability services.
- Call setup services – Establish a session between parties and manage call progress.
- Call handling services – Transfer and terminate calls.

The primary motivation behind SIP was to support next generation communications that use the internet, however, IETF has made great progress in allowing SIP to work with legacy voice networks.

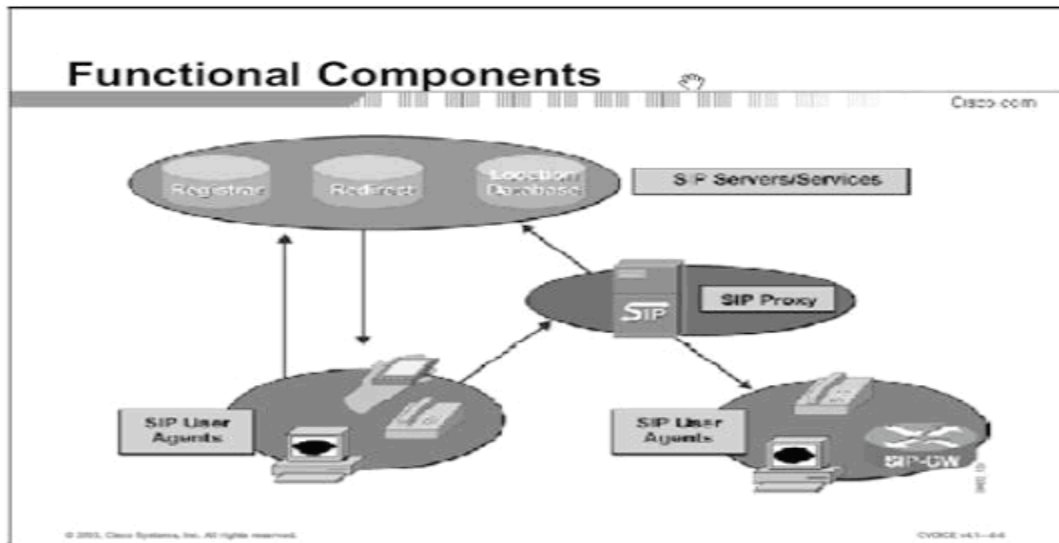
SIP is a peer to peer protocol; the peers in a session are called User Agents (UA). A UA consists of two components:

- User Agent Client (UAC) – a client application that initiates a SIP request.
- User Agent Server (UAS) – a server application that contacts the user when a SIP invitation is received and then returns a response on behalf of the user to the originator.

A SIP UA can function as a UAC or UAS during a session but not as both in the

same session. The initiating UA uses a UAC and the terminating UA uses a UAS.

The physical components of a SIP network are grouped into following two categories, user agents and SIP servers.



User agent includes the following devices:

- IP telephone – acts as a UAC or UAS on a session by session basis.
- Gateway – acts as a UAC or UAS and provides call control support. Gateways provide many services, the most common being a translation function between SIP user agents and other terminal types by translating transmission formats and communication procedures. The gateway provides call setup and termination on both the IP side and the switched circuit network (SCN) side.

SIP servers include the following devices:

- **Proxy server** – Intermediate component that receives SIP requests from a client and then forwards the request on behalf of the client to the next SIP



server in the network. The next server can be another proxy or UAS. Proxy servers provide function such as authentication, authorization, network access control, routing and security.

- **Redirect server** – Provides a UA with information on the next server the UA should contact. The server can be another network server or UA, the UA then redirects the invitation to the server identified by the redirect server.
- **Registrar server** – Requests from UACs for registration of their current location.
- **Location server** – A service providing address resolution services to SIP proxy or redirect servers. A location server can include a database of registrations or access to commonly used tools such as Lightweight directory access Protocol (LDAP).

If a UAC recognizes the destination UAS the client communicate directly with the server. In situations where the client is unable to establish a direct relationship, the client solicits the assistance of a network server. The following describes setting up a call using a proxy server.

Skinny Client Control Protocol (SCCP)

Skinny Client Control Protocol (SCCP) is a Cisco propriety protocol. It is a simple and easy to use protocol that is used between the Cisco Call Manager and the Cisco IP phones. Since this is a very light wait protocol it can be used with the phones which do not have much intelligence and processing power. Although skinny is the default protocol used by the Cisco IP phones it is possible to download SIP program which can loaded in to the IP phones which will enable them to use SIP to communicate with the call manager.

Chapter 3 Freeswitch

Freeswitch is an open source telephony platform designed to facilitate the creation of voice and chat driven products scaling from a soft-phone up to a soft-switch. Its core library, libfreeswitch, is capable of being embedded into other projects, as well as being used as a stand-alone application.

Following are the software packages among its primary dependencies:

- Apache Portable Runtime.
- SQLite – a lightweight implantation of a SQL engine.
- PCRE – perl Compatible Regular Expressions.
- Sofia-SIP – an open-source SIP user agent library.
- libspeex – speex DSP library (replaced libresample as of version 1.0.3).
- Spandsp
- libSRTP – an open-source implementation of the Secure Real-time Transport Protocol.

Freeswitch is a modular application, where modules can extend the functionality of Freeswitch but the abstraction layer prevents inter-module dependency. The goal is to ensure that one module is not required to load another.

Freeswitch is neither a pure switch which simply route calls like GnuGK and SER, nor a PABX or IVR like Asterisk and its derivatives; In fact it occupies a space between them. Freeswitch provides building blocks from which applications can be built such a PABX, a voicemail system or a conferencing system. Freeswitch supports various technologies such as SIP and H.323, which makes it easy to interface with other open source PBX systems such as sipXecs, call weaver or Asterisk.

Freeswitch supports both wide and narrow band codecs making it an ideal solution to bridge legacy devices to the future. The voice channels and the conference bridge module all can operate at 8, 12, 16, 24, 32 or 48 kHz and can bridge channels of



different rates. Freeswitch build natively and runs standalone on several operating systems including Windows, Mac OS X, Linux, BSD and Solaris on both 32 and 64 bit platforms.

Following are the features of Freeswitch:

- Call recording.
- Centralized user/ Domain directory.
- High performance multi-threaded core engine.
- Configuration via cURL to you HTTP server.
- XML configuration files for easy parsing.
- Configurable RFC2833 payload type.
- Wideband conferencing.
- Proper ENUM/ISN dialing built in
- Subscription server
- Basic IP/PBX features
- Many supported codecs
 - G.722.1
 - G.722.1C
 - G.722
 - G.711
 - CELT (32kHz and 48kHz)
 - G.726 AAL2 and RFC3551
 - G.723.1
 - G.729
 - AMR
 - iLBC
 - Speex
 - lpc10
 - DVI4

Even though the configuration of Freeswitch is discussed in Design and Implementation, however some of the important default configuration files in Freeswitch are discussed below:

- **freeswitch.xml** – This is the main configuration file. When the Freeswitch is started it will locate this file and process it. The default freeswitch.xml file includes other files to allow for easier maintenance of configuration data.
- **vars.xml** – It is used to define variable that are used system wide.

autoload_configs: It is a directory where a lot of the freeswitch configuration resides. This directory is located in /conf/autoload_configs. The default freeswitch.xml preprocesses xml files matched by the glob conf/autoload_configs/*.xml.

- **modules.conf.xml** – It tells freeswitch which module to load. There are certain modules required for operation so it should not be edited unless a specific module is required to be removed or added.
- **sofia.conf.xml** – it is used to create SIP endpoints. This file contains “X-PRE-PROCESS” directive that includes other XML files that define one or more “SIP profiles”.

SIP Profiles: The default profiles are “internal”, “internal-ipv6” and “external”.

- **internal.xml and internal_ipv6.xml:** Both of the files are located in /conf/sip_profiles. This profile generally refers to devices that reside in the internal network.
- **external.xml** – This file is also located in /conf/sip_profiles. This profile generally refers to devices or gateways that reside outside of the network.

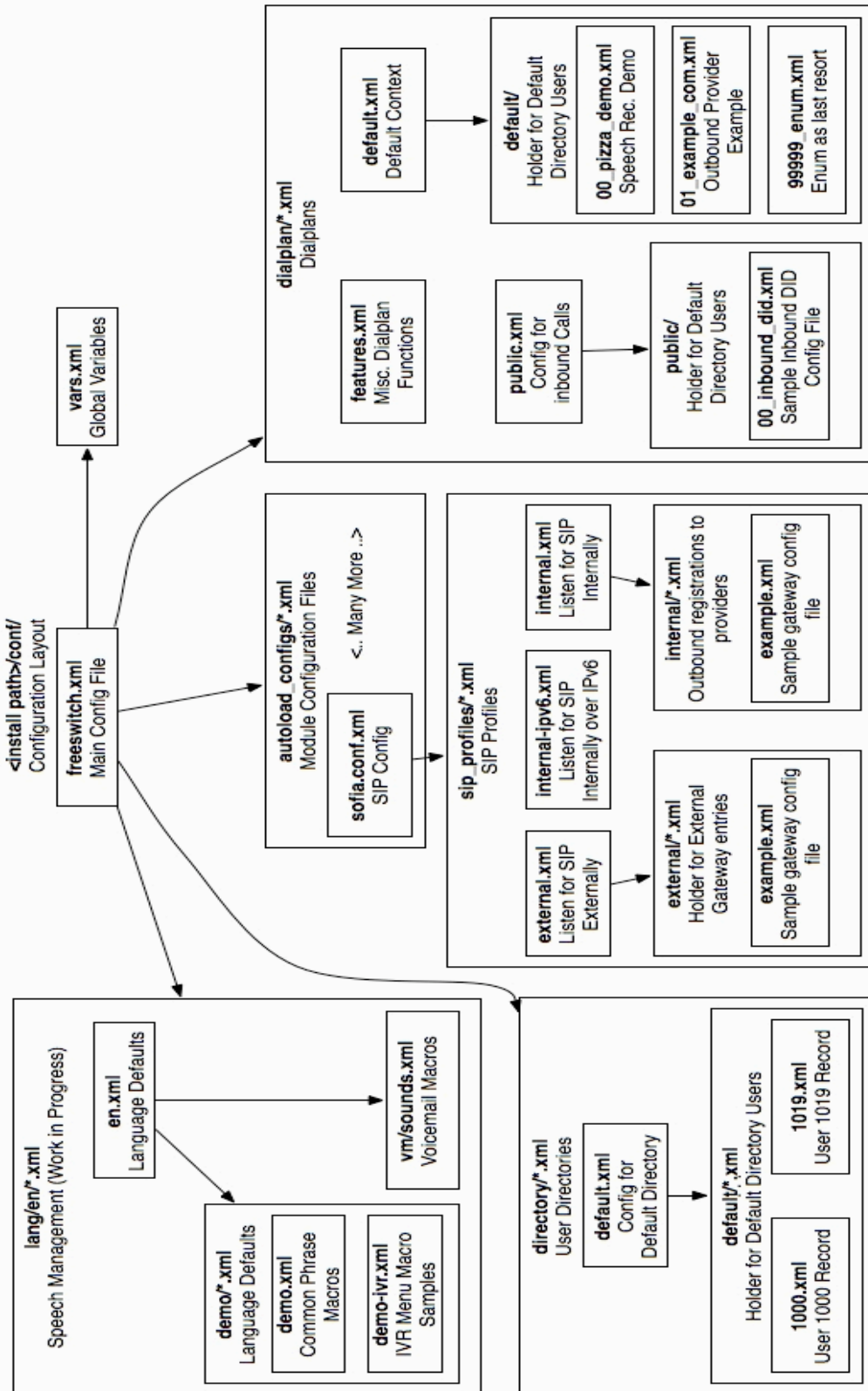
Dialplan: The Freeswitch dialplan is a full featured, XML-based call routing mechanism. XML files can be created or edited and they are included in **default.xml** file.

Directory: The directory holds authentication credentials for other sip endpoints



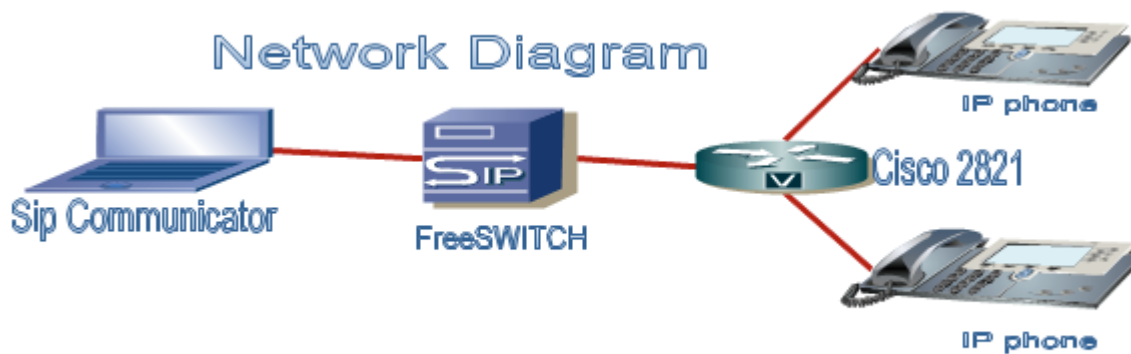
that will register to freeswitch. The directory configuration default is configured to process the `/conf/directory/default/*.xml` by the configuration included in `freeswitch.xml`.

Freeswitch has a large number of other configuration files as well. The figure on the next page shows the various configuration files and their default contents along with the standard modules.



Chapter 4 Network Design

As we have already discussed VoIP protocols and Freeswitch. We can take in depth look at the actual design of my VoIP network setup.



Now we are going to discuss all the network components one by one as shown in the figure:

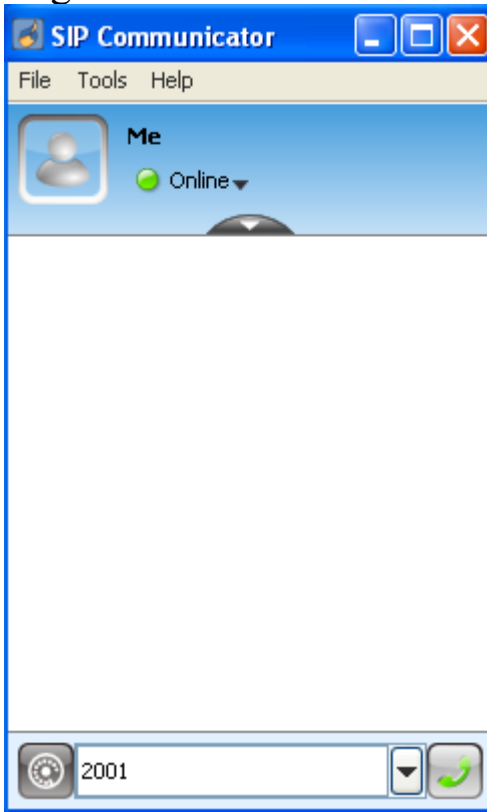
SIP Communicator:

SIP communicator is an audio/video Internet phone and instant messenger that supports some of the most popular instant messaging and telephony protocols such as SIP, Jabber, AIM/ICQ, MSN, IRC, etc.

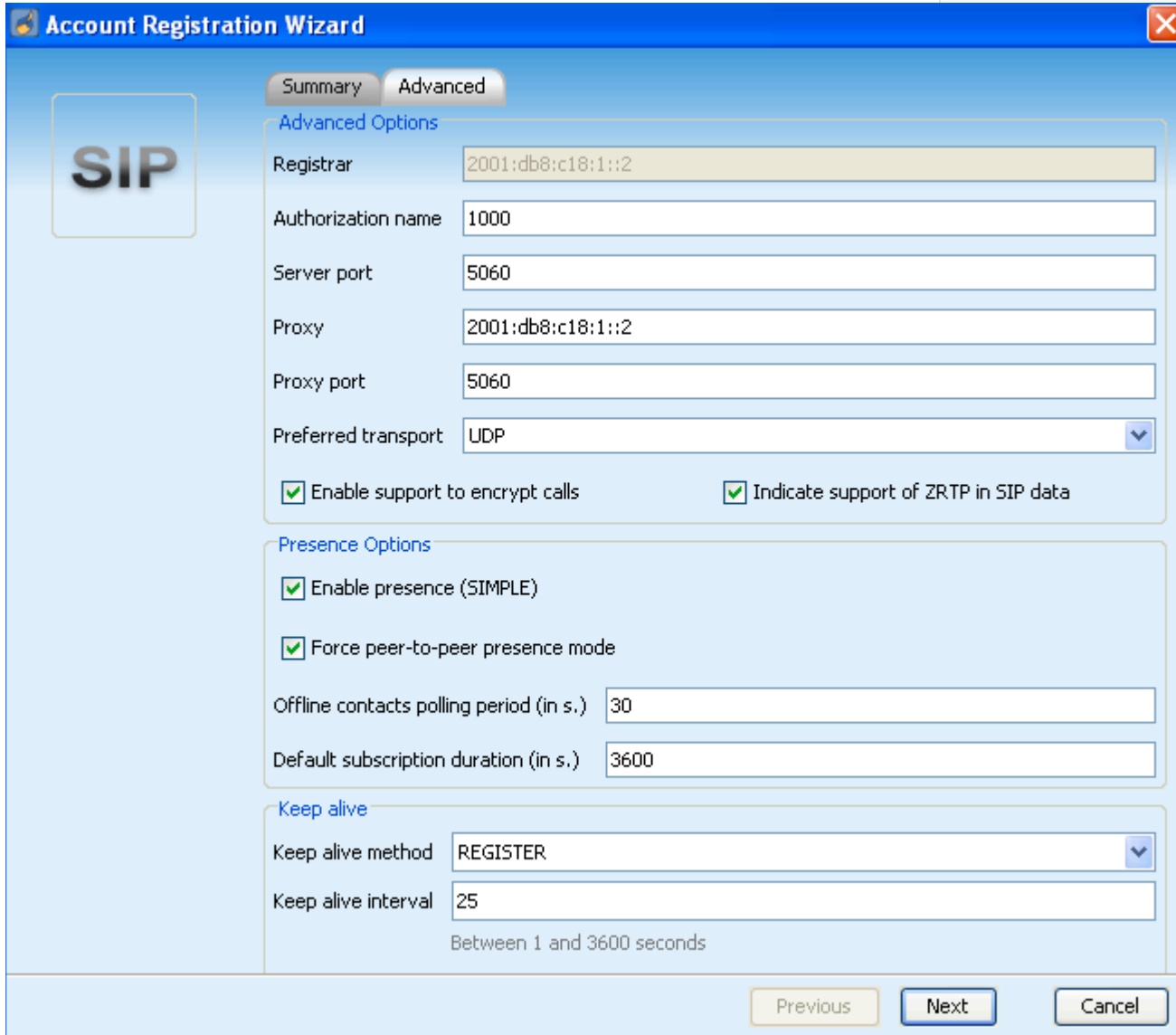
SIP communicator is completely Open Source and is freely available under the terms of the GNU lesser General Public License.

The reason SIP communicator is used that it also has IPv6 capability. It is installed on Windows XP machine which has a forced IP address of 2001:db8:c18:1::3 and default gateway as 2001:db8:c18:1::2. This Windows XP machine is connected to SIP proxy machine which has FreeSWITCH installed on it.

Screenshots below explain the configuration of the SIP communicator:
To configure SIP communicator you need to click on File and click on Account Registration Wizard.



In Account Registration Wizard, click on the Advanced tab and fill up the options as shown in the screenshot below,



The screenshot shows the 'Account Registration Wizard' window with the 'SIP' icon on the left. The 'Advanced' tab is selected, showing the following configuration options:

- Advanced Options:**
 - Registrar: 2001:db8:c18:1::2
 - Authorization name: 1000
 - Server port: 5060
 - Proxy: 2001:db8:c18:1::2
 - Proxy port: 5060
 - Preferred transport: UDP
 - Enable support to encrypt calls
 - Indicate support of ZRTP in SIP data
- Presence Options:**
 - Enable presence (SIMPLE)
 - Force peer-to-peer presence mode
 - Offline contacts polling period (in s.): 30
 - Default subscription duration (in s.): 3600
- Keep alive:**
 - Keep alive method: REGISTER
 - Keep alive interval: 25 (Between 1 and 3600 seconds)

Navigation buttons at the bottom: Previous, Next, Cancel.

SIP PROXY SERVER – FreeSWITCH

Freeswitch is the proxy server being used in this project. The Freeswitch is installed on an Ubuntu machine. The Ubuntu has three interface cards installed on it; however we are just using two interfaces, eth0 and eth1.

eth0 is connected to the Cisco router 2821, it has a forced IP address of 10.1.31.15 and a default gateway of 10.1.31.1.

eth1 is connected to the Windows XP and has an IPv6 address forced on it, which is 2001:db8:c18:1::2. Figure shown below shows the configuration on the Ethernet interfaces.

```

varun@varun-desktop: ~
File Edit View Terminal Tabs Help
varun@varun-desktop:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:da:20:ac:de
          inet addr:10.1.31.15  Bcast:10.1.31.255  Mask:255.255.255.0
          inet6 addr: fe80::250:daff:fe20:acde/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:173518 errors:0 dropped:0 overruns:0 frame:0
          TX packets:165073 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:21238280 (20.2 MB)  TX bytes:18838044 (17.9 MB)
          Interrupt:16 Base address:0xe000

eth1      Link encap:Ethernet  HWaddr 00:02:b3:e7:eb:ca
          inet6 addr: 2001:db8:c18:1::2/64 Scope:Global
          inet6 addr: fe80::202:b3ff:fee7:ebca/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:1840 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2168 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:791852 (773.2 KB)  TX bytes:868929 (848.5 KB)

eth2      Link encap:Ethernet  HWaddr 00:0c:6e:da:8f:cb
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:19 Base address:0x6000

```

Now we are going to discuss the FreeSWITCH configurations:

As we have already discussed important configuration files in Chapter 3, this time we are going to discuss only the files which need to be configured in Freeswitch. One has to tell FS about where to be listening for SIP registrations and other SIP traffic. That is done in the /conf/sip_profiles directory. One profile is for internal traffic (internal.xml); the other is for outside traffic (external.xml). However what



we have here is just a LAN so we won't be needing external.xml. There is another profile in SIP_profiles which is used for IPv6 (internal-ipv6.xml) which we will be configuring as well.

- **internal.xml** – In the internal.xml, we'll make some changes to accommodate the LAN network. It is a big file but just a small tweak needs to be made, we would edit the file and search for rtp-ip, sip-ip, ext-rtp-ip and ext-sip-ip, respectively. Initially, “value=\${local_ip_v4}” in these four lines. Change the “value=” as below, using our own LAN IP addresses.

```
<param name="rtp-ip" value="10.1.31.15"/>
<param name="sip-ip" value="10.1.31.15"/>
<param name="ext-rtp-ip" value="10.1.31.15"/>
<param name="ext-sip-ip" value="10.1.31.15"/>
```

- **internal-ipv6.xml** – In this xml file as well we need to change the value of rtp-ip, sip-ip, ext-rtp-ip and ext-sip-ip but with the IPv6 address forced on the eth1 as this would be responsible for the LAN phone's registration, which is SIP communicator in our case. However, ext-rtp-ip and ext-sip-ip are disabled in the file by default which needs to be enabled by removing the quotes.

```
<param name="rtp-ip" value="2001:db8:c18:1::2"/>
<param name="sip-ip" value="2001:db8:c18:1::2"/>
<param name="ext-rtp-ip" value="2001:db8:c18:1::2"/>
<param name="ext-sip-ip" value="2001:db8:c18:1::2"/>
```

As our SIP proxy server is connected to the Cisco, we should be able to make it talk to the Freeswitch. For Cisco to talk to FS, we need to create a gateway profile in the /conf/sip_profile/internal so that Cisco can register with FS.

- **00_internal.xml** – We created a new xml file by the name 00_internal.xml file in the /conf/sip_profiles/internal. If you check internal.xml there is a preprocess mentioned which includes all the xml file in internal folder which is in the sip_profile.

```
<X-PRE-PROCESS cmd="internal/*.xml">
```

The xml file is mentioned below

```
<include>
  <gateway name="10.1.31.15">
    <param name="username" value="1000"/>
    <param name="password" value="1234"/>
    <param name="expire-seconds" value="60"/>
    <param name="register" value="true"/>
    <param name="register-transport" value="udp"/>
    <param name="retry-seconds" value="30"/>
    <param name="caller_id_in_from" value="false"/>
    <param name="contact-params" value="tport=5060"/>
    <param name="ping" value="25">
  </gateway>
</include>
```

That's the whole file. The "name" has to be something DNS can translate or an actual IP address. Username and password is something which I setup on the Cisco which you will come across when we will discuss about Cisco's configuration.

- **conf/dialplan/default.xml** – To make this gateway work, we add the following extension to the "context" in the file as mentioned below. This directs FS to use a specific gateway.

```
<extension name="cisco">
  <condition field="destination_number" expression="^(200[0-9])">
    <action application="set" data="effective_caller_id_number=1000"/>
    <action application="bridge"
data=sofia/gateway/10.1.31.15/$1@10.1.31.1/>
  </condition>
</extension>
```

These are the main files which need to be configured as far as FreeSwitch is



concerned.

Cisco 2821 Router

In the Cisco router we are using two Fast Ethernet ports and one Gigabit Ethernet port. The Gigabit Ethernet is connected to Freeswitch. Therefore the interface has been assigned an IP address of 10.1.31.1.

```
!  
interface GigabitEthernet0/0  
ip address 10.1.31.1 255.255.255.0  
duplex auto  
speed auto  
!
```

I had to create DHCP services on both the Fast Ethernet interfaces. IP address of the voice vlans were defined on the router.

```
!  
no ip dhcp conflict logging  
ip dhcp excluded-address 192.168.20.1 192.168.20.50  
ip dhcp excluded-address 192.168.10.1 192.168.10.50  
!  
ip dhcp pool data  
network 192.168.20.0 255.255.255.0  
default-router 192.168.20.1  
lease infinite  
!  
ip dhcp pool voice  
network 192.168.10.0 255.255.255.0  
default-router 192.168.10.1  
lease infinite
```



And the Vlans were also created to distinguish between the Voice and Data:

```
!  
interface Vlan10  
  description voip vlan  
  ip address 192.168.10.1 255.255.255.0  
!  
interface Vlan20  
  description data vlan  
  ip address 192.168.20.1 255.255.255.0  
!
```

Since, I was using two IP phones so both the Fast Ethernet interfaces have the same configurations.

```
!  
interface FastEthernet0/1/0  
  switchport mode trunk  
  switchport voice vlan 10  
!  
interface FastEthernet0/1/1  
  switchport mode trunk  
  switchport voice vlan 10  
!
```

Switchport voice vlan is used to instruct the IP phone to forward all the voice traffic through the VLAN and switchport mode trunk is used for letting VLAN information pass between the interfaces.

This was the setup for IP phones and the Freeswitch. But the issue over here is that since we are using Cisco IP phones on the router side which come with default Skinny protocols and the Freeswitch is using SIP. We could have flashed the IP phones with SIP files but that is not an easy thing to do and can be quiet tedious. So



instead of flashing the phone I used the following command.

```
!  
voice service voip  
  allow-connections h323 to h323  
  allow-connections h323 to sip  
  allow-connections sip to h323  
  allow-connections sip to sip  
sip  
!
```

These commands can be used to use Skinny protocol with SIP as Skinny is interoperable with H.323.

Now we needed to define the patterns for outgoing calls. These are configured as dial peers as shown below:

```
!  
dial-peer voice 10 voip  
  destination-pattern 1000  
  no voice-class sip anat  
  session protocol sipv2  
  session target ipv4:10.1.31.15  
  session transport udp  
  dtmf-relay rtp-nte  
  codec g711ulaw  
  no vad  
!
```

Destination pattern has only 1000 mentioned here because I was using only one extension which happened to be 1000 (as shown in the screenshot of the SIP communicator), though this is not right thing to do.



In freeswitch we created a gateway which happens to be Cisco router. We need to configure SIP-UA in the Cisco so that it can register with the SIP registrar which is Freeswitch.

```
!  
sip-ua  
authentication username 1000 password 7 091D1C5A4D  
retry invite 3  
retry register 10  
timers connect 100  
registrar ipv6:[2001:db8:c18:1::2] expires 3600  
sip-server ipv4:10.1.31.15  
!
```

Username and password mentioned above, are already in XML we created by the name of 00_internal.xml in Freeswitch. The password is not shown by default, even though it can be changed as and when required.

The configuration mentioned below is the information required by the IP phones connected to the L2 links of the router along with phone information required for the registration, files which are needed to be flashed on to the phone when a the IP phone loads up,among other information.

```
!  
telephony-service  
em logout 0:0 0:0 0:0  
max-ephones 15  
max-dn 24  
ip source-address 192.168.10.1 port 2000  
load 7941 usbflash0:SCCP41.8-5-2SR1S  
max-conferences 8 gain -6  
transfer-system full-consult  
create cnf-files version-stamp 7960 Nov 26 2009 13:09:38
```



```
!  
!  
ephone-dn 1  
  number 2001  
  name Talwar,V  
!  
!  
ephone-dn 2  
  number 2002  
  name MINT  
!  
!  
ephone 1  
  device-security-mode none  
  mac-address 001B.D512.4E3A  
  type 7941  
  button 1:1  
!  
!  
!  
ephone 2  
  device-security-mode none  
  mac-address 001B.D512.3D35  
  type 7941  
  button 1:2  
!
```

This finishes our configuration for the devices used in the network.

Chapter 5 Conclusion

Even though, the world sooner or later would speed up the transition from IPv4 to IPv6 and thus, the VoIP systems too. But as mentioned before there are still many issues that are being faced while transitioning from IPv4 to IPv6. After reaching the end of this project I did face various issues which are as follows:

- There are many operating systems which already provide support for IPv6 but communication hardware and software providers have been less enthusiastic and have done nothing to very less to provide support for IPv6. An example for that would be Asterisk, they still have no known IPv6 support for their softswitch.
- There is not many open source softphones available which have IPv6 support. I tried Linphone, X-lite and kphone. They all had some or the other issue with IPv6 support.
- Even though I was able to make calls using Freeswitch but it had its own issues. Firstly it is primarily dependent on IPv4, if someone is using a pure IPv6 based network, Freeswitch becomes ineffective.
- Since there are not many open source softwares available thus this results in huge costs and complexities as well, to upgrade their existing systems. In which, not all organizations would have an interest, considering that the world economy is still on recovery path.

Appendix

Here are the complete FreeSwitch configurations:

```
****conf/dialplan/default.xml****
```

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<!--
```

NOTICE:

This context is usually accessed via authenticated callers on the sip profile on port 5060 or transferred callers from the public context which arrived via the sip profile on port 5080.

Authenticated users will use the user_context variable on the user to determine what context they can access. You can also add a user in the directory with the cidr= attribute acl.conf.xml will build the domains ACL using this value.

```
-->
```

```
<!-- http://wiki.freeswitch.org/wiki/Dialplan_XML -->
```

```
<include>
```

```
<context name="default">
```

```
<extension name="cisco">
```

```
<condition field="destination_number" expression="^(200[0-9])">
```

```
<action application="set" data="effective_caller_id_number=1000"/>
```

```
<action application="bridge" data="sofia/gateway/10.1.31.15/$1@10.1.31.1"/>
```

```
</condition>
```

```
</extension>
```

```
<extension name="unloop">
```

```
<condition field="{unroll_loops}" expression="^true$"/>
```

```
<condition field="{sip_looped_call}" expression="^true$">
```

```
<action application="deflect" data="{destination_number}"/>
```

```
</condition>
```

```
</extension>
```

<!-- Example of doing things based on time of day.

year = 4 digit year. Example year="2009"
yday = 1-365
mon = 1-12
mday = 1-31
week = 1-52
mweek= 1-6
wday = 1-7
hour = 0-23
minute = 0-59
minute-of-day = 1-1440

Example:

```
<condition minute-of-day="540-1080"> (9am to 6pm EVERY day)
do something ...
</condition>
-->
<extension name="tod_example" continue="true">
  <condition wday="2-6" hour="9-18">
<action application="set" data="open=true"/>
  </condition>
  </extension>
```

<!-- Example of routing based on holidays

This example covers all US Federal holidays except for inauguration day.

```
-->
<extension name="holiday_example" continue="true">
  <condition mday="1" mon="1">
<!-- new year's day -->
<action application="set" data="open=false"/>
  </condition>
  <condition wday="2" mweek="3" mon="1">
<!-- martin luther king day is the 3rd monday in january -->
<action application="set" data="open=false"/>
```



```
</condition>
<condition wday="2" mweek="3" mon="2">
<!-- president's day is the 3rd monday in february -->
<action application="set" data="open=false"/>
</condition>
<condition wday="2" mon="5" mday="25-31">
<!-- memorial day is the last monday in may (the only monday between the 25th and the 31st)
-->
<action application="set" data="open=false"/>
</condition>
<condition mday="4" mon="7">
<!-- independence day -->
<action application="set" data="open=false"/>
</condition>
<condition wday="2" mweek="1" mon="9">
<!-- labor day is the 1st monday in september -->
<action application="set" data="open=false"/>
</condition>
<condition wday="2" mweek="2" mon="10">
<!-- columbus day is the 2st monday in october -->
<action application="set" data="open=false"/>
</condition>
<condition mday="11" mon="11">
<!-- veteran's day -->
<action application="set" data="open=false"/>
</condition>
<condition wday="5-6" mweek="4" mon="11">
<!-- thanksgiving is the 4th thursday in november and usually there's an extension for black
friday -->
<action application="set" data="open=false"/>
</condition>
<condition mday="25" mon="12">
<!-- Christmas -->
<action application="set" data="open=false"/>
</condition>
</extension>

<extension name="global-intercept">
```



```

    <condition field="destination_number" expression="^886$">
    <action application="answer"/>
    <action application="intercept" data="{hash(select/{domain_name}-last_dial/global)}"/>
    <action application="sleep" data="2000"/>
    </condition>
</extension>

<extension name="group-intercept">
    <condition field="destination_number" expression="^\*8$">
    <action application="answer"/>
    <action application="intercept" data="{hash(select/{domain_name}-
last_dial/{callgroup})}"/>
    <action application="sleep" data="2000"/>
    </condition>
</extension>

<extension name="intercept-ext">
    <condition field="destination_number" expression="^\*\*(\d+)$">
    <action application="answer"/>
    <action application="intercept" data="{hash(select/{domain_name}-last_dial_ext/{1})}"/>
    <action application="sleep" data="2000"/>
    </condition>
</extension>

<extension name="redial">
    <condition field="destination_number" expression="^870$">
    <action application="transfer" data="{hash(select/{domain_name}-
last_dial/{caller_id_number})}"/>
    </condition>
</extension>

<extension name="global" continue="true">
    <condition field="{call_debug}" expression="^true$" break="never">
    <action application="info"/>
    </condition>
<!--

```

This is an example of how to auto detect if telephone-event is missing and activate inband detection



```

-->
<!--
  <condition field="{switch_r_sdp}" expression="a=rtpmap:(\d+)\stelephone-event/8000"
break="never">
  <action application="set" data="rtp_payload_number=$1"/>
  <anti-action application="start_dtmf"/>
  </condition>
-->
  <condition field="{sip_has_crypto}"
expression="^(AES_CM_128_HMAC_SHA1_32|AES_CM_128_HMAC_SHA1_80)$"
break="never">
  <action application="set" data="sip_secure_media=true"/>
  <!-- Offer SRTP on outbound legs if we have it on inbound. -->
  <!-- <action application="export" data="sip_secure_media=true"/> -->
  </condition>

  <condition>
  <action application="hash" data="insert/{domain_name}-
spymap/{caller_id_number}/{uuid}"/>
  <action application="hash" data="insert/{domain_name}-
last_dial/{caller_id_number}/{destination_number}"/>
  <action application="hash" data="insert/{domain_name}-last_dial/global/{uuid}"/>
  </condition>
</extension>

<!-- If sip_req_host is not a local domain then this has to be an external sip uri -->

<extension name="external_sip_uri" continue="true">
  <condition field="source" expression="mod_sofia"/>
  <condition field="{outside_call}" expression="^{}/>
  <condition field="{domain_exists({sip_req_host})}" expression="true">
<anti-action application="bridge" data="sofia/{use_profile}/{sip_to_uri}"/>
  </condition>
</extension>

<!--
  Snom button demo, call 9000 to make button 2 mapped to transfer the current call to a
conference

```

```
-->
```

```
<extension name="snom-demo-2">
  <condition field="destination_number" expression="^9001$">
    <action application="eval" data="{snom_bind_key(2 off DND ${sip_from_user}
${sip_from_host} ${sofia_profile_name} message notused)"/>
    <action application="transfer" data="3000"/>
  </condition>
</extension>
```

```
<extension name="snom-demo-1">
  <condition field="destination_number" expression="^9000$">
<!--<key> <light> <label> <user> <host> <profile> <action_name> <action>-->
    <action application="eval" data="{snom_bind_key(2 on DND ${sip_from_user}
${sip_from_host} ${sofia_profile_name} message api+uuid_transfer ${uuid} 9001)"/>
    <action application="playback" data="{hold_music}"/>
  </condition>
</extension>
```

```
<extension name="eavesdrop">
  <condition field="destination_number" expression="^88(.*)$|^*0(.*)$">
<action application="answer"/>
<action application="eavesdrop" data="{hash(select/${domain_name}-spymap/$1)"/>
  </condition>
</extension>
```

```
<extension name="eavesdrop">
  <condition field="destination_number" expression="^779$">
<action application="answer"/>
<action application="set" data="eavesdrop_indicate_failed=tone_stream://%(500, 0, 320)"/>
<action application="set" data="eavesdrop_indicate_new=tone_stream://%(500, 0, 620)"/>
<action application="set" data="eavesdrop_indicate_idle=tone_stream://%(250, 0, 920)"/>
<action application="eavesdrop" data="all"/>
  </condition>
</extension>
```

```
<extension name="call_return">
  <condition field="destination_number" expression="^*69$|^869$|^lcr$">
```



```
<action application="transfer" data="{hash(select/{domain_name}-
call_return/{caller_id_number})}"/>
  </condition>
</extension>

<extension name="del-group">
  <condition field="destination_number" expression="^80(\d{2})$">
<action application="answer"/>
<action application="group"
data="delete:$1@{domain_name}:{sofia_contact({sip_from_user}@{domain_name})}"/
>
<action application="gentones" data="% (1000, 0, 320)"/>
  </condition>
</extension>

<extension name="add-group">
  <condition field="destination_number" expression="^81(\d{2})$">
<action application="answer"/>
<action application="group"
data="insert:$1@{domain_name}:{sofia_contact({sip_from_user}@{domain_name})}"/>
<action application="gentones" data="% (1000, 0, 640)"/>
  </condition>
</extension>

<extension name="call-group-simo">
  <condition field="destination_number" expression="^82(\d{2})$">
<action application="bridge"
data="{leg_timeout=15,ignore_early_media=true}{group(call:$1@{domain_name})}"/>
  </condition>
</extension>

<extension name="call-group-order">
  <condition field="destination_number" expression="^83(\d{2})$">
<action application="bridge"
data="{leg_timeout=15,ignore_early_media=true}{group(call:$1@{domain_name}:order)}"
/>
  </condition>
</extension>
```

```

<extension name="extension-intercom">
  <condition field="destination_number" expression="^8(10[01][0-9])$">
<action application="set" data="dialed_extension=$1"/>
<action application="export" data="sip_auto_answer=true"/>
<action application="bridge" data="user/${dialed_extension}@${domain_name}"/>
  </condition>
</extension>

<!--
dial the extension (1000-1019) for 30 seconds and go to voicemail if the
call fails (continue_on_fail=true), otherwise hang up after a successful
bridge (hangup_after-bridge=true)
-->
<extension name="Local_Extension">
  <condition field="destination_number" expression="^(10[01][0-9])$">
<action application="set" data="dialed_extension=$1"/>
<action application="export" data="dialed_extension=$1"/>
<!-- bind_meta_app can have these args <key> [a|b|ab] [a|b|o|s] <app> -->
<action application="bind_meta_app" data="1 b s execute_extension::dx XML features"/>
<action application="bind_meta_app" data="2 b s
record_session::${base_dir}/recordings/${caller_id_number}.${strftime(%Y-%m-%d-%H-
%M-%S)}.wav"/>
<action application="bind_meta_app" data="3 b s execute_extension::cf XML features"/>
<action application="set" data="ringback=${us-ring}"/>
<action application="set" data="transfer_ringback=${hold_music}"/>
<action application="set" data="call_timeout=30"/>
<!-- <action application="set" data="sip_exclude_contact=${network_addr}"/> -->
<action application="set" data="hangup_after_bridge=true"/>
<!--<action application="set"
data="continue_on_fail=NORMAL_TEMPORARY_FAILURE,USER_BUSY,NO_ANSWER,
TIMEOUT,NO_ROUTE_DESTINATION"/> -->
<action application="set" data="continue_on_fail=true"/>
<action application="hash" data="insert/${domain_name}-
call_return/${dialed_extension}/${caller_id_number}"/>
<action application="hash" data="insert/${domain_name}-
last_dial_ext/${dialed_extension}/${uuid}"/>

```



```
<action application="set"
data="called_party_callgroup=${user_data(${dialed_extension}@${domain_name} var
callgroup)"/>
<!--<action application="export"
data="nolocal:sip_secure_media=${user_data(${dialed_extension}@${domain_name} var
sip_secure_media)"/>-->
<action application="hash" data="insert/${domain_name}-
last_dial/${called_party_callgroup}/${uuid}"/>
<action application="bridge" data="user/${dialed_extension}@${domain_name}"/>
<action application="answer"/>
<action application="sleep" data="1000"/>
<action application="voicemail" data="default ${domain_name} ${dialed_extension}"/>
</condition>
</extension>

<extension name="group_dial_sales">
<condition field="destination_number" expression="^2000$">
<action application="bridge" data="${group_call(sales@${domain_name})}"/>
</condition>
</extension>

<extension name="group_dial_support">
<condition field="destination_number" expression="^2001$">
<action application="bridge" data="group/support@${domain_name}"/>
</condition>
</extension>

<extension name="group_dial_billing">
<condition field="destination_number" expression="^2002$">
<action application="bridge" data="group/billing@${domain_name}"/>
</condition>
</extension>

<!-- voicemail operator extension -->
<extension name="operator">
<condition field="destination_number" expression="^(operator|0)$">
<action application="set" data="transfer_ringback=${hold_music}"/>
<action application="transfer" data="1000 XML features"/>
```

```
</condition>
</extension>
```

```
<!-- voicemail main extension -->
<extension name="vmain">
  <condition field="destination_number" expression="^vmain$|^4000$|^*98$">
<action application="answer"/>
<action application="sleep" data="1000"/>
<action application="voicemail" data="check default ${domain_name}"/>
  </condition>
</extension>
```

```
<!--
```

This extension is used by mod_portaudio so you can pa call sip:someone@example.com mod_portaudio will pass the entire string to the dialplan for routing.

```
-->
```

```
<extension name="sip_uri">
  <condition field="destination_number" expression="^sip:(.*)$">
<action application="bridge" data="sofia/${use_profile}/${1}"/>
  </condition>
</extension>
```

```
<!--
```

start a dynamic conference with the settings of the "default" conference profile in conference.conf.xml

```
-->
```

```
<extension name="nb_conferences">
  <condition field="destination_number" expression="^(30\d{2})$">
<action application="answer"/>
<action application="conference" data="$1-${domain_name}@default"/>
  </condition>
</extension>
```

```
<extension name="wb_conferences">
  <condition field="destination_number" expression="^(31\d{2})$">
<action application="answer"/>
<action application="conference" data="$1-${domain_name}@wideband"/>
  </condition>
```




```

</extension>

<extension name="uwb_conferences">
  <condition field="destination_number" expression="^(32\d{2})$">
<action application="answer"/>
<action application="conference" data="$1-${domain_name}@ultrawideband"/>
  </condition>
</extension>
<!-- MONO 48kHz conferences -->
<extension name="cdquality_conferences">
  <condition field="destination_number" expression="^(33\d{2})$">
<action application="answer"/>
<action application="conference" data="$1-${domain_name}@cdquality"/>
  </condition>
</extension>

<!-- dial the FreeSWITCH conference via SIP-->
<extension name="freeswitch_public_conf_via_sip">
  <condition field="destination_number" expression="^9(888|1616|3232)$">
<action application="export" data="hold_music=silence"/>
<!--
  This will take the SAS from the b-leg and send it to the display on the a-leg phone.
  Known working with Polycom and Snom maybe others.
-->
<!--
<action application="set" data="exec_after_bridge_app=${sched_api(+4 zrtp expand
uuid_display ${uuid} \${uuid_getvar(\${uuid_getvar(${uuid} signal_bond)} zrtp_sas1_string
)} \${uuid_getvar(\${uuid_getvar(${uuid} signal_bond)} zrtp_sas2_string )} )}"/>
<action application="export" data="nolocal:zrtp_secure_media=true"/>
-->
<action application="bridge" data="sofia/${use_profile}/${1}@conference.freeswitch.org"/>
  </condition>
</extension>

<!--
This extension will start a conference and invite a group.
At anytime the participant can dial *2 to bridge directly to the boss.
All other callers are then hung up on.

```

```
-->
<extension name="mad_boss_intercom">
  <condition field="destination_number" expression="^0911$">
    <action application="set" data="conference_auto_outcall_caller_id_name=Mad Boss1"/>
    <action application="set" data="conference_auto_outcall_caller_id_number=0911"/>
    <action application="set" data="conference_auto_outcall_timeout=60"/>
    <action application="set" data="conference_auto_outcall_flags=mute"/>
    <action application="set"
data="conference_auto_outcall_prefix={sip_auto_answer=true,execute_on_answer='bind_meta
_app 2 a s1 transfer::intercept:${uuid} inline'}"/>
    <action application="set" data="sip_exclude_contact=${network_addr}"/>
    <action application="conference_set_auto_outcall" data="${group_call(sales)}"/>
    <action application="conference"
data="madboss_intercom1@default+flags {endconf|deaf}"/>
  </condition>
</extension>
```

<!--

This extension will start a conference and invite a few of people.
 At anytime the participant can dial *2 to bridge directly to the boss.
 All other callers are then hung up on.

-->

```
<extension name="mad_boss_intercom">
  <condition field="destination_number" expression="^0912$">
    <action application="set" data="conference_auto_outcall_caller_id_name=Mad Boss2"/>
    <action application="set" data="conference_auto_outcall_caller_id_number=0912"/>
    <action application="set" data="conference_auto_outcall_timeout=60"/>
    <action application="set" data="conference_auto_outcall_flags=mute"/>
    <action application="set"
data="conference_auto_outcall_prefix={sip_auto_answer=true,execute_on_answer='bind_meta
_app 2 a s1 transfer::intercept:${uuid} inline'}"/>
    <action application="set" data="sip_exclude_contact=${network_addr}"/>
    <action application="conference_set_auto_outcall" data="loopback/9999"/>
    <action application="conference"
data="madboss_intercom2@default+flags {endconf|deaf}"/>
  </condition>
</extension>
```



```

<!--This extension will start a conference and invite several people upon entering -->
<extension name="mad_boss">
  <condition field="destination_number" expression="^0913$">
<!--These params effect the outcalls made once you join-->
<action application="set" data="conference_auto_outcall_caller_id_name=Mad Boss"/>
<action application="set" data="conference_auto_outcall_caller_id_number=0911"/>
<action application="set" data="conference_auto_outcall_timeout=60"/>
<action application="set" data="conference_auto_outcall_flags=none"/>
<!--<action application="set" data="conference_auto_outcall_announce=say:You have been
called into an emergency conference"/>-->
<!--Add as many of these as you need, These are the people you are going to call-->
<action application="conference_set_auto_outcall" data="loopback/9999"/>
<action application="conference" data="madboss3@default"/>
  </condition>
</extension>

<!-- a sample IVR -->
<extension name="ivr_demo">
  <condition field="destination_number" expression="^5000$">
    <action application="answer"/>
    <action application="sleep" data="2000"/>
<action application="ivr" data="demo_ivr"/>
  </condition>
</extension>

<!-- Create a conference on the fly and pull someone in at the same time. -->
<extension name="dynamic_conference">
  <condition field="destination_number" expression="^5001$">
<action application="conference"
data="bridge:mydynaconf:sofia/${use_profile}/1234@conference.freeswitch.org"/>
  </condition>
</extension>

<extension name="rtp_multicast_page">
  <condition field="destination_number" expression="^pagegroup$|^7243$">
<action application="answer"/>
<action application="esf_page_group"/>
  </condition>

```

```
</extension>
```

```
<!--
```

Parking extensions... transferring calls to 5900 will park them in a queue.

```
-->
```

```
<extension name="park">
```

```
  <condition field="destination_number" expression="^5900$">
```

```
<action application="set" data="fifo_music=${hold_music}"/>
```

```
<action application="fifo" data="5900@${domain_name} in"/>
```

```
  </condition>
```

```
</extension>
```

```
<!--
```

Parking pickup extension. Calling 5901 will pickup the call.

```
-->
```

```
<extension name="unpark">
```

```
  <condition field="destination_number" expression="^5901$">
```

```
<action application="answer"/>
```

```
<action application="fifo" data="5900@${domain_name} out nowait"/>
```

```
  </condition>
```

```
</extension>
```

```
<!--
```

Valet park retrieval, works with valet_park extension below.

Retrieve a valet parked call by dialing 6000 + park number + #

```
-->
```

```
<extension name="valet_park">
```

```
  <condition field="destination_number" expression="^(6000)$">
```

```
<action application="answer"/>
```

```
<action application="valet_park" data="valet_parking_lot ask 1 11 10000 ivr/ivr-  
enter_ext_pound.wav"/>
```

```
  </condition>
```

```
</extension>
```

```
<!--
```

Valet park 6001-6099. Blind x-fer to 6001, 6002, etc. to valet park the call.

Dial 6001, 6002, etc. to retrieve a call that is already valet parked.

After call is retrieved, park extension is free for another call.



```
-->
<extension name="valet_park">
  <condition field="destination_number" expression="^(60\d[1-9])$">
<action application="answer"/>
<action application="valet_park" data="valet_parking_lot $1"/>
  </condition>
</extension>
```

<!--
This extension is used with Snom phones.

Set a function key to park+lot (lot being a number or name.)
Set type to Park+Orbit. You can then park and pickup using
the softkey on the phone. Should work with other phones.

```
-->
<extension name="park">
  <condition field="source" expression="mod_sofia"/>
  <condition field="destination_number" expression="park\+(\d+)">
<action application="fifo" data="$1@${domain_name} in undef $$ {hold_music}"/>
  </condition>
</extension>
<!--
```

The extension is parking pickup with a to param of the fifo we are calling
Some phones send things like orbit= and you can extract that info.

```
-->
<extension name="unpark">
  <condition field="source" expression="mod_sofia"/>
  <condition field="destination_number" expression="^parking$"/>
  <condition field="{sip_to_params}" expression="fifo\=(\d+)">
<action application="answer"/>
<action application="fifo" data="$1@${domain_name} out nowait"/>
  </condition>
</extension>
```

<!--
This extension is used with Linksys phones.



Set a Phone tab option Call Park Serv to yes. You can park and pickup using soft keys "park" and "unpark" found during active call when moving navigation button. The other option is to use phone's star codes (defaults to *38 and *39).

-->

```
<extension name="park">
  <condition field="source" expression="mod_sofia"/>
  <condition field="destination_number" expression="callpark"/>
  <condition field="{sip_refer_to}">
<expression><![CDATA[<sip:callpark@${domain_name};orbit=(\d+)>]]></expression>
<action application="fifo" data="$1@${domain_name} in undef $$ {hold_music}"/>
  </condition>
</extension>
```

<!--

This extension is used with Linksys phones.

The extension is parking pickup with a to param of the fifo we are calling. Linksys sends orbit=<parkingslotnumber> and we extract that info.

-->

```
<extension name="unpark">
  <condition field="source" expression="mod_sofia"/>
  <condition field="destination_number" expression="pickup"/>
  <condition field="{sip_to_params}" expression="orbit\=(\d+)">
<action application="answer"/>
<action application="fifo" data="$1@${domain_name} out nowait"/>
  </condition>
</extension>
```

<!--

Here are some examples of how to override the ringback heard by the far end. You have two variables that you can use to override this.

ringback - used when a call isn't answered. (early media)
 transfer_ringback - used when the call is already answered. (post answer)

-->



```
<!-- Demonstration of how to override the ringback in various situations -->
<extension name="wait">
  <condition field="destination_number" expression="^wait$">
<action application="pre_answer"/>
<action application="sleep" data="20000"/>
<action application="answer"/>
<action application="sleep" data="1000"/>
<action application="playback" data="voicemail/vm-goodbye.wav"/>
<action application="hangup"/>
  </condition>
</extension>

<extension name="fax_receive">
  <condition field="destination_number" expression="^9978$">
<action application="answer" />
<action application="playback" data="silence_stream://2000"/>
<action application="rxfax" data="/tmp/rxfax.tif"/>
<action application="hangup"/>
  </condition>
</extension>

<extension name="fax_transmit">
  <condition field="destination_number" expression="^9979$">
<action application="txfax" data="/tmp/txfax.tif"/>
<action application="hangup"/>
  </condition>
</extension>

<!-- Send a 180 and let the far end generate ringback. -->
<extension name="ringback_180">
  <condition field="destination_number" expression="^9980$">
<action application="ring_ready"/>
<action application="sleep" data="20000"/>
<action application="answer"/>
<action application="sleep" data="1000"/>
<action application="playback" data="voicemail/vm-goodbye.wav"/>
<action application="hangup"/>
  </condition>
```



```
</extension>
```

```
<!-- Send a 183 and send uk-ring as the ringtone. (early media) -->
```

```
<extension name="ringback_183_uk_ring">
```

```
  <condition field="destination_number" expression="^9981$">
```

```
<action application="set" data="ringback=${uk-ring}"/>
```

```
<action application="bridge" data="loopback/wait"/>
```

```
  </condition>
```

```
</extension>
```

```
<!-- Send a 183 and use music as the ringtone. (early media) -->
```

```
<extension name="ringback_183_music_ring">
```

```
  <condition field="destination_number" expression="^9982$">
```

```
<action application="set" data="ringback=${hold_music}"/>
```

```
<action application="bridge" data="loopback/wait"/>
```

```
  </condition>
```

```
</extension>
```

```
<!-- Answer the call and use music as the ringtone. (post answer) -->
```

```
<extension name="ringback_post_answer_uk_ring">
```

```
  <condition field="destination_number" expression="^9983$">
```

```
<action application="set" data="transfer_ringback=${uk-ring}"/>
```

```
<action application="answer"/>
```

```
<action application="bridge" data="loopback/wait"/>
```

```
  </condition>
```

```
</extension>
```

```
<!-- Answer the call and use music as the ringtone. (post answer) -->
```

```
<extension name="ringback_post_answer_music">
```

```
  <condition field="destination_number" expression="^9984$">
```

```
<action application="set" data="transfer_ringback=${hold_music}"/>
```

```
<action application="answer"/>
```

```
<action application="bridge" data="loopback/wait"/>
```

```
  </condition>
```

```
</extension>
```

```
<extension name="ClueCon">
```

```
  <condition field="destination_number" expression="^9991$">
```




```
<action application="set" data="effective_caller_id_name=ClueCon IVR"/>
<action application="bridge" data="sofia/$$ {domain}/2000@bkw.org"/>
</condition>
</extension>
```

```
<extension name="show_info">
<condition field="destination_number" expression="^9992$">
<action application="answer"/>
<action application="info"/>
<action application="sleep" data="250"/>
<action application="hangup"/>
</condition>
</extension>
```

```
<extension name="video_record">
<condition field="destination_number" expression="^9993$">
<action application="answer"/>
<action application="record_fsv" data="/tmp/testrecord.fsv"/>
</condition>
</extension>
```

```
<extension name="video_playback">
<condition field="destination_number" expression="^9994$">
<action application="answer"/>
<action application="play_fsv" data="/tmp/testrecord.fsv"/>
</condition>
</extension>
```

```
<extension name="delay_echo">
<condition field="destination_number" expression="^9995$">
<action application="answer"/>
<action application="delay_echo" data="5000"/>
</condition>
</extension>
```

```
<extension name="echo">
<condition field="destination_number" expression="^9996$">
<action application="answer"/>
```

```

<action application="echo"/>
  </condition>
</extension>

<extension name="milliwatt">
  <condition field="destination_number" expression="^9997$">
<action application="answer"/>
<action application="playback" data="tone_stream://%(251,0,1004);loops=-1"/>
  </condition>
</extension>

<extension name="tone_stream">
  <condition field="destination_number" expression="^9998$">
<action application="answer"/>
<action application="playback"
data="tone_stream://path=${base_dir}/conf/tetris.ttml;loops=10"/>
  </condition>
</extension>

<!-- install zrtp_agent.lua into scripts (ZRTP == 9787) -->
<extension name="zrtp_enrollement">
  <condition field="destination_number" expression="^9787$">
<action application="answer"/>
<action application="lua" data="zrtp_agent.lua"/>
  </condition>
</extension>

<!--
You will no longer hear the bong tone. The wav file is playing stating the call is secure.
The file will not play unless you have both TLS and SRTP active.
-->

<extension name="hold_music">
  <condition field="destination_number" expression="^9999$"/>
  <condition field="${sip_has_crypto}"
expression="^(AES_CM_128_HMAC_SHA1_32|AES_CM_128_HMAC_SHA1_80)$">
<action application="answer"/>
<action application="execute_extension" data="is_secure XML features"/>

```



```
<action application="playback" data="$$ {hold_music}"/>
<anti-action application="set" data="zrtp_secure_media=true"/>
<anti-action application="answer"/>
<anti-action application="playback" data="silence_stream://2000"/>
<anti-action application="execute_extension" data="is_zrtp_secure XML features"/>
<anti-action application="playback" data="$$ {hold_music}"/>
  </condition>
</extension>
```

<!--

You can place files in the default directory to get included.

-->

```
<X-PRE-PROCESS cmd="include" data="default/*.xml"/>
```

<!--

WARNING WARNING WARNING WARNING WARNING WARNING WARNING
WARNING WARNING WARNING WARNING

Anything you put below this line will usually get ignored due to the file in
default/99999_enum.xml as it will transfer the call to the enum dialplan.

WARNING WARNING WARNING WARNING WARNING WARNING WARNING
WARNING WARNING WARNING WARNING

-->

<!--

```
<extension name="refer">
  <condition field="{sip_refer_to}">
<expression><![CDATA[<sip:${destination_number}@${domain_name}>]]></expression>
  </condition>
  <condition field="{sip_refer_to}">
<expression><![CDATA[<sip:(.*)@(.*)>]]></expression>
  <action application="set" data="refer_user=$1"/>
  <action application="set" data="refer_domain=$2"/>
  <action application="info"/>
  <action application="bridge" data="sofia/${use_profile}/${refer_user}@${refer_domain}"/>
  </condition>
</extension>
```



```

-->
<!--
This is an example of how to override the RURI on an outgoing invite to a registered contact.
-->
<!--
<extension name="ruri">
  <condition field="destination_number" expression="^ruri$">
    <action application="bridge"
data="sofia/${ruri_profile}/${ruri_user}${regex(${sofia_contact(${ruri_contact})}|^[^@]+(.*)
|%1)}"/>
  </condition>
</extension>

<extension name="7004">
  <condition field="destination_number" expression="^7004$">
    <action application="set" data="ruri_profile=default"/>
    <action application="set" data="ruri_user=2000"/>
    <action application="set" data="ruri_contact=1001@${domain_name}"/>
    <action application="execute_extension" data="ruri"/>
  </condition>
</extension>
-->

<!-- SEE WARNING ABOVE IF YOU ARE TRYING TO ADD EXTENSIONS HERE! -->

</context>
</include>

```

conf/sip_profiles/internal/00_internal.xml

```

<include>
  <gateway name="10.1.31.15">
    <!--// account username *required* //-->
    <param name="username" value="1000"/>
    <!--// auth realm: *optional* same as gateway name, if blank //-->
    <!--<param name="realm" value="asterlink.com"/>-->
    <!--// username to use in from: *optional* same as username, if blank //-->

```



```

<!--<param name="from-user" value="cluecon"/>-->
<!--// domain to use in from: *optional* same as realm, if blank ///-->
<!--<param name="from-domain" value="asterlink.com"/>-->
<!--// account password *required* ///-->
<param name="password" value="1234"/>
<!--// extension for inbound calls: *optional* same as username, if blank ///-->
<!--<param name="extension" value="cluecon"/>-->
<!--// proxy host: *optional* same as realm, if blank ///-->
<!--<param name="proxy" value="asterlink.com"/>-->
<!--// send register to this proxy: *optional* same as proxy, if blank ///-->
<!--<param name="register-proxy" value="mysbc.com"/>-->
<!--// expire in seconds: *optional* 3600, if blank ///-->
<param name="expire-seconds" value="60"/>
<!--// do not register ///-->
<param name="register" value="true"/>
<!-- which transport to use for register -->
<param name="register-transport" value="udp"/>
<!--How many seconds before a retry when a failure or timeout occurs -->
<param name="retry-seconds" value="30"/>
<!--Use the callerid of an inbound call in the from field on outbound calls via this gateway -->
<param name="caller-id-in-from" value="false"/>
<!--extra sip params to send in the contact-->
<param name="contact-params" value="tport=5060"/>
<!--send an options ping every x seconds, failure will unregister and/or mark it down-->
<param name="ping" value="25"/>
</gateway>
</include>

```

```

****conf/sip_profiles/internal.xml****

```

```

<profile name="internal">

```

```

<!--

```

This is a sofia sip profile/user agent. This will service exactly one ip and port.
In FreeSWITCH you can run multiple sip user agents on their own ip and port.

When you hear someone say "sofia profile" this is what they are talking about.

```

-->

```



```
<!-- http://wiki.freeswitch.org/wiki/Sofia_Configuration_Files -->
<!--aliases are other names that will work as a valid profile-->
<aliases>
  <!--
  <alias name="default"/>
  -->
</aliases>
<!-- Outbound Registrations -->
<gateways>
  <X-PRE-PROCESS cmd="include" data="internal/*.xml"/>
</gateways>

<domains>
  <!-- indicator to parse the directory for domains with parse="true" to get gateways-->
  <!--<domain name="$$ {domain} " parse="true"/>-->
  <!-- indicator to parse the directory for domains with parse="true" to get gateways and alias
every domain to this profile -->
  <!--<domain name="all" alias="true" parse="true"/>-->
  <domain name="all" alias="true" parse="false"/>
</domains>

<settings>
  <!--
When calls are in no media this will bring them back to media
when you press the hold button.
  -->
  <!--<param name="media-option" value="resume-media-on-hold"/> -->
  <!--
This will allow a call after an attended transfer go back to
bypass media after an attended transfer.
  -->
  <!--<param name="media-option" value="bypass-media-after-att-xfer"/>-->
  <!-- <param name="user-agent-string" value="FreeSWITCH Rocks!"/> -->
  <param name="debug" value="0"/>
  <!-- If you want FreeSWITCH to shutdown if this profile fails to load, uncomment the next
line. -->
  <!-- <param name="shutdown-on-fail" value="true"/> -->
```

```

<param name="sip-trace" value="no"/>
<param name="log-auth-failures" value="true"/>
<param name="context" value="public"/>
<param name="rfc2833-pt" value="101"/>
<!-- port to bind to for sip traffic -->
<param name="sip-port" value="$$ {internal_sip_port}"/>
<param name="dialplan" value="XML"/>
<param name="dtmf-duration" value="100"/>
<param name="inbound-codec-prefs" value="$$ {global_codec_prefs}"/>
<param name="outbound-codec-prefs" value="$$ {global_codec_prefs}"/>
<param name="rtp-timer-name" value="soft"/>
<!-- ip address to use for rtp, DO NOT USE HOSTNAMES ONLY IP ADDRESSES -->
<param name="rtp-ip" value="10.1.31.15"/>
<!-- ip address to bind to, DO NOT USE HOSTNAMES ONLY IP ADDRESSES -->
<param name="sip-ip" value="10.1.31.15"/>
<param name="hold-music" value="$$ {hold_music}"/>
<param name="apply-nat-acl" value="auto"/>
<!--<param name="aggressive-nat-detection" value="true"/>-->
<!--

```

There are known issues (asserts and segfaults) when 100rel is enabled.
It is not recommended to enable 100rel at this time.

```

-->
<!--<param name="enable-100rel" value="true"/>-->
<!-- Enable Compact SIP headers. -->
<!--<param name="enable-compact-headers" value="true"/>-->
<!--

```

enable/disable session timers

```

-->
<!--<param name="enable-timer" value="false"/>-->
<!--<param name="minimum-session-expires" value="120"/>-->
<param name="apply-inbound-acl" value="domains"/>
<!--

```

This defines your local network, by default we detect your local network and create this localnet.auto ACL for this.

```

-->
<param name="local-network-acl" value="localnet.auto"/>
<!--<param name="apply-register-acl" value="domains"/>-->
<!--<param name="dtmf-type" value="info"/>-->

```



```

<param name="record-path" value="$$ {base_dir}/recordings"/>
<param name="record-template"
value="${caller_id_number}.${target_domain}.${strftime(%Y-%m-%d-%H-%M-
%S)}.wav"/>
<!--enable to use presence -->
<param name="manage-presence" value="true"/>
<!--<param name="manage-shared-appearance" value="true"/>-->
<!-- used to share presence info across sofia profiles -->
<!-- Name of the db to use for this profile -->
<!--<param name="dbname" value="share_presence"/>-->
<!--<param name="presence-hosts" value="$$ {domain}"/>-->
<!-- ***** -->

<!-- This setting is for AAL2 bitpacking on G726 -->
<!-- <param name="bitpacking" value="aal2"/> -->
<!--max number of open dialogs in proceeding -->
<!--<param name="max-proceeding" value="1000"/>-->
<!--session timers for all call to expire after the specified seconds -->
<!--<param name="session-timeout" value="120"/>-->
<!-- Can be 'true' or 'contact' -->
<!--<param name="multiple-registrations" value="contact"/>-->
<!--set to 'greedy' if you want your codec list to take precedence -->
<param name="inbound-codec-negotiation" value="generous"/>
<!-- if you want to send any special bind params of your own -->
<!--<param name="bind-params" value="transport=udp"/>-->
<!--<param name="unregister-on-options-fail" value="true"/>-->

<!-- TLS: disabled by default, set to "true" to enable -->
<param name="tls" value="$$ {internal_ssl_enable}"/>
<!-- additional bind parameters for TLS -->
<param name="tls-bind-params" value="transport=tls"/>
<!-- Port to listen on for TLS requests. (5061 will be used if unspecified) -->
<param name="tls-sip-port" value="$$ {internal_tls_port}"/>
<!-- Location of the agent.pem and cafile.pem ssl certificates (needed for TLS server) -->
<param name="tls-cert-dir" value="$$ {internal_ssl_dir}"/>
<!-- TLS version ("sslv23" (default), "tlsv1"). NOTE: Phones may not work with TLSv1 -->
<param name="tls-version" value="$$ {sip_tls_version}"/>

```




```
<!-- turn on auto-flush during bridge (skip timer sleep when the socket already has data)
(reduces delay on latent connections default true, must be disabled explicitly)-->
<!--<param name="rtp-autoflush-during-bridge" value="false"/>-->

<!--If you don't want to pass through timestamps from 1 RTP call to another (on a per call
basis with rtp_rewrite_timestamps chanvar)-->
<!--<param name="rtp-rewrite-timestamps" value="true"/>-->
<!--<param name="pass-rfc2833" value="true"/>-->
<!--If you have ODBC support and a working dsn you can use it instead of SQLite-->
<!--<param name="odbc-dsn" value="dsn:user:pass"/>-->

<!--Uncomment to set all inbound calls to no media mode-->
<!--<param name="inbound-bypass-media" value="true"/>-->

<!--Uncomment to set all inbound calls to proxy media mode-->
<!--<param name="inbound-proxy-media" value="true"/>-->

<!--Uncomment to let calls hit the dialplan *before* you decide if the codec is ok-->
<!--<param name="inbound-late-negotiation" value="true"/>-->

<!-- this lets anything register -->
<!-- comment the next line and uncomment one or both of the other 2 lines for call
authentication -->
<!-- <param name="accept-blind-reg" value="true"/> -->

<!-- accept any authentication without actually checking (not a good feature for most people)
-->
<!-- <param name="accept-blind-auth" value="true"/> -->

<!-- suppress CNG on this profile or per call with the 'suppress_cng' variable -->
<!-- <param name="suppress-cng" value="true"/> -->

<!--TTL for nonce in sip auth-->
<param name="nonce-ttl" value="60"/>
<!--Uncomment if you want to force the outbound leg of a bridge to only offer the codec
that the originator is using-->
<!--<param name="disable-transcoding" value="true"/>-->
<!-- Handle 302 Redirect in the dialplan -->
```

```

<!--<param name="manual-redirect" value="true"/> -->
<!-- Disable Transfer -->
<!--<param name="disable-transfer" value="true"/> -->
<!-- Disable Register -->
<!--<param name="disable-register" value="true"/> -->
<!-- Used for when phones respond to a challenged ACK with method INVITE in the hash --
>
<!--<param name="NDLB-broken-auth-hash" value="true"/>-->
<!-- add a ;received="<ip>:<port>" to the contact when replying to register for nat handling -
->
<!--<param name="NDLB-received-in-nat-reg-contact" value="true"/>-->
<param name="auth-calls" value="$$ {internal_auth_calls}"/>
<!-- Force the user and auth-user to match. -->
<param name="inbound-reg-force-matching-username" value="true"/>
<!-- on authed calls, authenticate *all* the packets not just invite -->
<param name="auth-all-packets" value="false"/>

<!-- external_sip_ip
Used as the public IP address for SDP.
Can be an one of:
    ip address          - "12.34.56.78"
    a stun server lookup - "stun:stun.server.com"
    a DNS name          - "host:host.server.com"
    auto                - Use guessed ip.
    auto-nat            - Use ip learned from NAT-PMP or UPNP
-->
<param name="ext-rtp-ip" value="10.1.31.15"/>
<param name="ext-sip-ip" value="10.1.31.15"/>

<!-- rtp inactivity timeout -->
<param name="rtp-timeout-sec" value="300"/>
<param name="rtp-hold-timeout-sec" value="1800"/>
<!-- VAD choose one (out is a good choice); -->
<!-- <param name="vad" value="in"/> -->
<!-- <param name="vad" value="out"/> -->
<!-- <param name="vad" value="both"/> -->
<!--<param name="alias" value="sip:10.0.1.251:5555"/>-->
<!--

```



These are enabled to make the default config work better out of the box.
If you need more than ONE domain you'll need to not use these options.

```
-->
<!--all inbound reg will look in this domain for the users -->
<param name="force-register-domain" value="$$ {domain}"/>
<!--force the domain in subscriptions to this value -->
<param name="force-subscription-domain" value="$$ {domain}"/>
<!--all inbound reg will stored in the db using this domain -->
<param name="force-register-db-domain" value="$$ {domain}"/>
<!--force suscription expires to a lower value than requested-->
<!--<param name="force-subscription-expires" value="60"/>-->
<!-- disable register and transfer which may be undesirable in a public switch -->
<!--<param name="disable-transfer" value="true"/>-->
<!--<param name="disable-register" value="true"/>-->

<!--
enable-3pcc can be set to either 'true' or 'proxy', true accepts the call
right away, proxy waits until the call has been answered then sends accepts
-->
<!--<param name="enable-3pcc" value="true"/>-->

<!-- use at your own risk or if you know what this does.-->
<!--<param name="NDLB-force-rport" value="true"/>-->
<!--
```

Choose the realm challenge key. Default is auto_to if not set.

auto_from - uses the from field as the value for the sip realm.
auto_to - uses the to field as the value for the sip realm.
<anyvalue> - you can input any value to use for the sip realm.

If you want URL dialing to work you'll want to set this to auto_from.

If you use any other value besides auto_to or auto_from you'll loose the ability to do multiple domains.

Note: comment out to restore the behavior before 2008-09-29



```
-->
<param name="challenge-realm" value="auto_from"/>
<!--<param name="disable-rtp-auto-adjust" value="true"/>-->
<!-- on inbound calls make the uuid of the session equal to the sip call id of that call -->
<!--<param name="inbound-use-callid-as-uuid" value="true"/>-->
<!-- on outbound calls set the callid to match the uuid of the session -->
<!--<param name="outbound-use-uuid-as-callid" value="true"/>-->
<!-- set to false disable this feature -->
<!--<param name="rtp-autofix-timing" value="false"/>-->

<!-- set this param to false if your gateway for some reason hates X- headers that it is
supposed to ignore-->
<!--<param name="pass-callee-id" value="false"/>-->

<!-- clear clears them all or supply the name to add or the name prefixed with ~ to remove
valid values:

clear
CISCO_SKIP_MARK_BIT_2833
SONUS_SEND_INVALID_TIMESTAMP_2833

-->
<!--<param name="auto-rtp-bugs" data="clear"/>-->

<!-- the following can be used as workaround with bogus SRV/NAPTR records -->
<!--<param name="disable-srv" value="false" />-->
<!--<param name="disable-naptr" value="false" />-->

<!-- The following can be used to fine-tune timers within sofia's transport layer
Those settings are for advanced users and can safely be left as-is -->

<!-- Initial retransmission interval (in milliseconds).
Set the T1 retransmission interval used by the SIP transaction engine.
The T1 is the initial duration used by request retransmission timers A and E (UDP) as
well as response retransmission timer G. -->
<!-- <param name="timer-T1" value="500" /> -->

<!-- Transaction timeout (defaults to T1 * 64).
```



Set the T1x64 timeout value used by the SIP transaction engine.

The T1x64 is duration used for timers B, F, H, and J (UDP) by the SIP transaction engine.

The timeout value T1x64 can be adjusted separately from the initial retransmission interval T1. -->

```
<!-- <param name="timer-T1X64" value="32000" /> -->
```

```
<!-- Maximum retransmission interval (in milliseconds).
```

Set the maximum retransmission interval used by the SIP transaction engine.

The T2 is the maximum duration used for the timers E (UDP) and G by the SIP transaction engine.

Note that the timer A is not capped by T2. Retransmission interval of INVITE requests grows exponentially

until the timer B fires. -->

```
<!-- <param name="timer-T2" value="4000" /> -->
```

```
<!--
```

Transaction lifetime (in milliseconds).

Set the lifetime for completed transactions used by the SIP transaction engine.

A completed transaction is kept around for the duration of T4 in order to catch late responses.

The T4 is the maximum duration for the messages to stay in the network and the duration of SIP timer K. -->

```
<!-- <param name="timer-T4" value="4000" /> -->
```

```
</settings>
```

```
</profile>
```

```
****conf/sip_profiles/internal-ipv6.xml****
```

```
<profile name="internal-ipv6">
```

```
<!--
```

This is an example of a sofia profile setup to listen on IPv6.

```
-->
```

```
<!-- http://wiki.freeswitch.org/wiki/Sofia\_Configuration\_Files -->
```



<!--aliases are other names that will work as a valid profile name for this profile-->

```

<settings>
  <!-- <param name="user-agent-string" value="FreeSWITCH Rocks!"/> -->
  <param name="debug" value="0"/>
  <param name="sip-trace" value="no"/>
  <param name="context" value="public"/>
  <param name="rfc2833-pt" value="101"/>
  <!-- port to bind to for sip traffic -->
  <param name="sip-port" value="$$ {internal_sip_port}"/>
  <param name="dialplan" value="XML"/>
  <param name="dtmf-duration" value="100"/>
  <param name="inbound-codec-prefs" value="$$ {global_codec_prefs}"/>
  <param name="outbound-codec-prefs" value="$$ {global_codec_prefs}"/>
  <param name="use-rtp-timer" value="true"/>
  <param name="rtp-timer-name" value="soft"/>
  <!-- ip address to use for rtp -->
  <param name="rtp-ip" value="2001:db8:c18:1::2"/>
  <!-- ip address to bind to -->
  <param name="sip-ip" value="2001:db8:c18:1::2"/>
  <param name="hold-music" value="$$ {hold_music}"/>
  <!--<param name="enable-100rel" value="false"/>-->
  <param name="apply-inbound-acl" value="domains"/>
  <!--<param name="apply-register-acl" value="domains"/>-->
  <!--<param name="dtmf-type" value="info"/>-->
  <param name="record-template"
value="$$ {base_dir}/recordings/$$ {caller_id_number}.$ {strftime(%Y-%m-%d-%H-%M-%S)}.wav"/>
  <!--enable to use presence and mwi -->
  <param name="manage-presence" value="true"/>
  <!-- This setting is for AAL2 bitpacking on G726 -->
  <!-- <param name="bitpacking" value="aal2"/> -->
  <!--max number of open dialogs in proceeding -->
  <!--<param name="max-proceeding" value="1000"/>-->
  <!--session timers for all call to expire after the specified seconds -->
  <!--<param name="session-timeout" value="120"/>-->
  <!--<param name="multiple-registrations" value="true"/>-->

```



```

<!--set to 'greedy' if you want your codec list to take precedence -->
<param name="inbound-codec-negotiation" value="generous"/>
<!-- if you want to send any special bind params of your own -->
<!--<param name="bind-params" value="transport=udp"/>-->
<!--<param name="unregister-on-options-fail" value="true"/>-->

<!-- TLS: disabled by default, set to "true" to enable -->
<param name="tls" value="$$ {internal_ssl_enable}"/>
<!-- additional bind parameters for TLS -->
<param name="tls-bind-params" value="transport=tls"/>
<!-- Port to listen on for TLS requests. (5061 will be used if unspecified) -->
<param name="tls-sip-port" value="$$ {internal_tls_port}"/>
<!-- Location of the agent.pem and cafile.pem ssl certificates (needed for TLS server) -->
<param name="tls-cert-dir" value="$$ {internal_ssl_dir}"/>
<!-- TLS version ("sslv23" (default), "tlsv1"). NOTE: Phones may not work with TLSv1 -->
<param name="tls-version" value="$$ {sip_tls_version}"/>

<!--If you don't want to pass through timestamps from 1 RTP call to another (on a per call
basis with rtp_rewrite_timestamps chanvar)-->
<!--<param name="rtp-rewrite-timestamps" value="true"/>-->
<!--<param name="pass-rfc2833" value="true"/>-->
<!--If you have ODBC support and a working dsn you can use it instead of SQLite-->
<!--<param name="odbc-dsn" value="dsn:user:pass"/>-->

<!--Uncomment to set all inbound calls to no media mode-->
<!--<param name="inbound-bypass-media" value="true"/>-->

<!--Uncomment to set all inbound calls to proxy media mode-->
<!--<param name="inbound-proxy-media" value="true"/>-->

<!--Uncomment to let calls hit the dialplan *before* you decide if the codec is ok-->
<!--<param name="inbound-late-negotiation" value="true"/>-->

<!-- this lets anything register -->
<!-- comment the next line and uncomment one or both of the other 2 lines for call
authentication -->
<!-- <param name="accept-blind-reg" value="true"/> -->

```



```

<!-- accept any authentication without actually checking (not a good feature for most people)
-->
<!-- <param name="accept-blind-auth" value="true"/> -->

<!-- suppress CNG on this profile or per call with the 'suppress_cng' variable -->
<!-- <param name="suppress-cng" value="true"/> -->

<!--TTL for nonce in sip auth-->
<param name="nonce-ttl" value="60"/>
<!--Uncomment if you want to force the outbound leg of a bridge to only offer the codec
that the originator is using-->
<!--<param name="disable-transcoding" value="true"/>-->
<!-- Used for when phones respond to a challenged ACK with method INVITE in the hash --
>
<!--<param name="NDLB-broken-auth-hash" value="true"/>-->
<!-- add a ;received="<ip>:<port>" to the contact when replying to register for nat handling -
->
<!--<param name="NDLB-received-in-nat-reg-contact" value="true"/>-->
<param name="auth-calls" value="$$ {internal_auth_calls}"/>
<!-- on authed calls, authenticate *all* the packets not just invite -->
<param name="auth-all-packets" value="false"/>
<param name="ext-rtp-ip" value="2001:db8:c18:1::2"/>
<param name="ext-sip-ip" value="2001:db8:c18:1::2"/>
<!-- rtp inactivity timeout -->
<param name="rtp-timeout-sec" value="300"/>
<param name="rtp-hold-timeout-sec" value="1800"/>
<!-- VAD choose one (out is a good choice); -->
<!-- <param name="vad" value="in"/> -->
<!-- <param name="vad" value="out"/> -->
<!-- <param name="vad" value="both"/> -->
<!--<param name="alias" value="sip:10.0.1.251:5555"/>-->
<!--
These are enabled to make the default config work better out of the box.
If you need more than ONE domain you'll need to not use these options.

-->
<!--all inbound reg will look in this domain for the users -->
<param name="force-register-domain" value="$$ {domain}"/>

```




```
<!--all inbound reg will stored in the db using this domain -->
<param name="force-register-db-domain" value="$$ {domain}"/>
<!-- disable register and transfer which may be undesirable in a public switch -->
<!--<param name="disable-transfer" value="true"/>-->
<!--<param name="disable-register" value="true"/>-->
<!--<param name="enable-3pcc" value="true"/>-->
<!-- use stun when specified (default is true) -->
<!--<param name="stun-enabled" value="true"/>-->
<!-- use stun when specified (default is true) -->
<!-- set to true to have the profile determine stun is not useful and turn it off globally-->
<!--<param name="stun-auto-disable" value="true"/>-->

<!-- the following can be used as workaround with bogus SRV/NAPTR records -->
<!--<param name="disable-srv" value="false" />-->
<!--<param name="disable-naptr" value="false" />-->

</settings>
</profile>
```

Below you will find the configurations of the Cisco 2821 router:

```
Router#sh running-config
Building configuration...

Current configuration : 2964 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname Router
!
```

MINT 709 Capstone Project



```
boot-start-marker
boot-end-marker
!
logging message-counter syslog
!
no aaa new-model
memory-size iomem 10
!
dot11 syslog
no ip source-route
!
!
ip cef
no ip dhcp conflict logging
ip dhcp excluded-address 192.168.20.1 192.168.20.50
ip dhcp excluded-address 192.168.10.1 192.168.10.50
!
ip dhcp pool data
  network 192.168.20.0 255.255.255.0
  default-router 192.168.20.1
  lease infinite
!
ip dhcp pool voice
  network 192.168.10.0 255.255.255.0
  default-router 192.168.10.1
  lease infinite
!
!
no ipv6 cef
!
multilink bundle-name authenticated
!
!
!
!
!
!
```

MINT 709 Capstone Project



```
voice service voip
  allow-connections h323 to h323
  allow-connections h323 to sip
  allow-connections sip to h323
  allow-connections sip to sip
  sip
  !
  !
  !
  !
  !
  !
  !
  !
  !
  !
  !
  !
  !
  !
  !
  !
  !
  !
  !
  !
  !
  !
  voice-card 0
  !
  !
  !
  !
  !
  archive
  log config
  hidekeys
  !
  !
  !
  !
  !
  !
  !
  class-map match-all voip
```

```
match cos 5
!  
!  
policy-map voice  
class voip  
set ip dscp ef  
  priority percent 60  
!  
!  
!  
!  
!  
interface GigabitEthernet0/0  
ip address 10.1.31.1 255.255.255.0  
duplex auto  
speed auto  
!  
interface GigabitEthernet0/1  
no ip address  
shutdown  
duplex auto  
speed auto  
!  
interface FastEthernet0/1/0  
switchport mode trunk  
switchport voice vlan 10  
!  
interface FastEthernet0/1/1  
switchport mode trunk  
switchport voice vlan 10  
!  
interface FastEthernet0/1/2  
!  
interface FastEthernet0/1/3  
!  
interface Serial0/0/0  
no ip address  
shutdown
```

MINT 709 Capstone Project



```
no fair-queue
clock rate 2000000
!
interface Vlan1
no ip address
!
interface Vlan10
description voip vlan
ip address 192.168.10.1 255.255.255.0
!
interface Vlan20
description data vlan
ip address 192.168.20.1 255.255.255.0
!
ip forward-protocol nd
no ip http server
no ip http secure-server
!
!
!
!
!
!
!
!
!
!
!
control-plane
!
!
!
ccm-manager fax protocol cisco
!
mgcp fax t38 ecm
!
!
!
dial-peer voice 10 voip
```

MINT 709 Capstone Project



```
destination-pattern 1000
no voice-class sip anat
session protocol sipv2
session target ipv4:10.1.31.15
session transport udp
dtmf-relay rtp-nte
codec g711ulaw
no vad
!
!
sip-ua
authentication username 1000 password 7 091D1C5A4D
retry invite 3
retry register 10
timers connect 100
registrar ipv6:[2001:db8:c18:1::2] expires 3600
sip-server ipv4:10.1.31.15
protocol mode dual-stack
!
!
!
telephony-service
em logout 0:0 0:0 0:0
max-ephones 15
max-dn 24
ip source-address 192.168.10.1 port 2000
load 7941 usbflash0:SCCP41.8-5-2SR1S
max-conferences 8 gain -6
transfer-system full-consult
create cnf-files version-stamp 7960 Nov 26 2009 13:09:38
!
!
ephone-dn 1
number 2001
name Talwar,V
!
!
ephone-dn 2
```

MINT 709 Capstone Project



```
number 2002
name MINT
!
!
ephone 1
device-security-mode none
mac-address 001B.D512.4E3A
type 7941
button 1:1
!
!
!
ephone 2
device-security-mode none
mac-address 001B.D512.3D35
type 7941
button 1:2
!
!
!
line con 0
line aux 0
line vty 0 4
login
!
scheduler allocate 20000 1000
end
```

Router#



References

- Cisco Voice over IP (CVOICE) student guide.
- Cisco IP Telephony (CIPT) student guide.
- FreeSWITCH.org
- Voice over IPv6: Architecture for next generation VoIP networks by Daniel Minoli
- Asteriskv6.org
- old.nabble.com
- RFCs 3262, 3264, 3265
- Wiki.freeswitch.org
- Cisco IOS Voice, Video and fax configuration guide