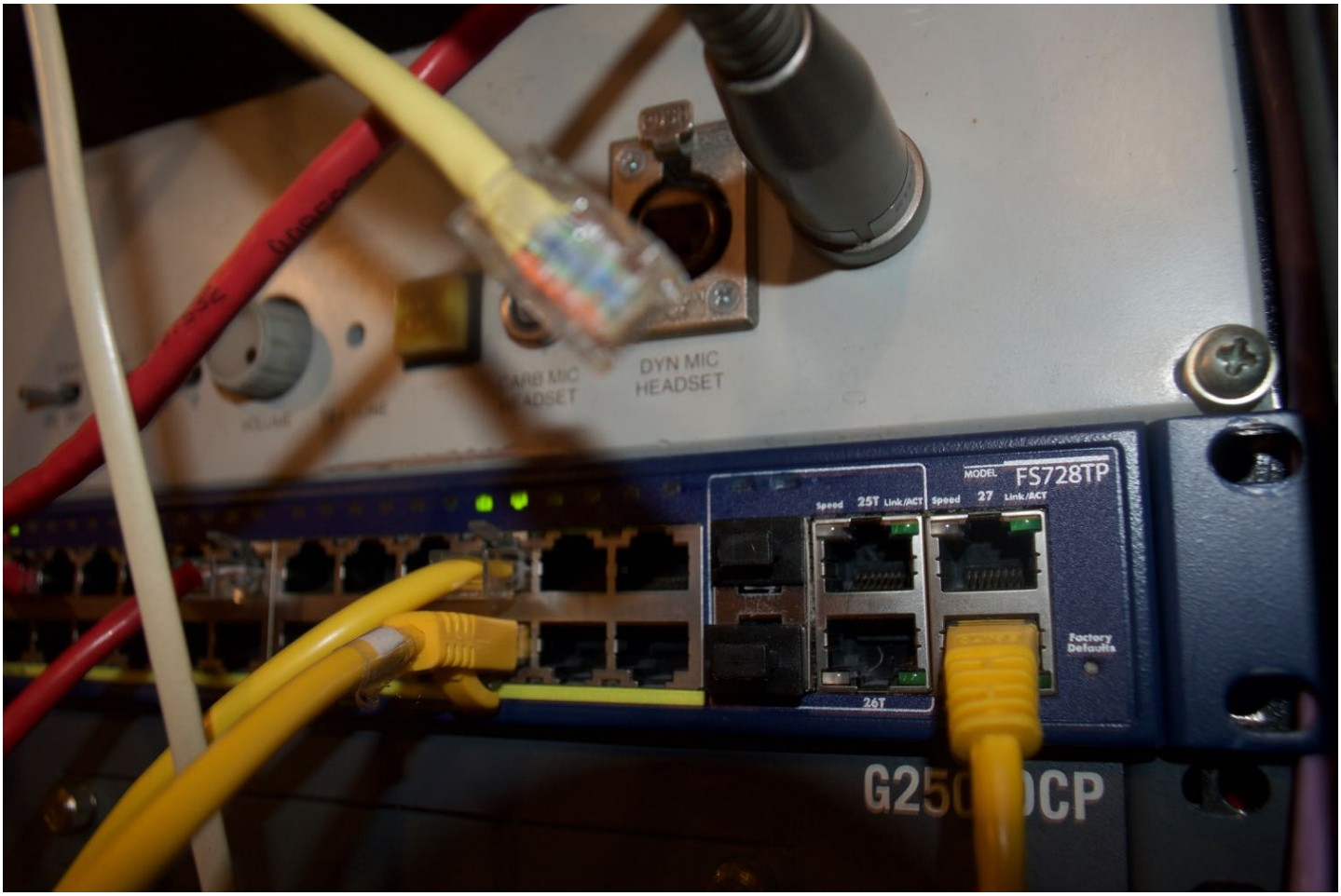

CLICKFORD MEDIA SERVICES

HOW TO IMPLEMTENT CISCO UNIFIED COMMUNICATIONS MANAGER AT HOME

FEBRUARY 2021

The documentation is Version 2021.1. The typeface used in this document are: Palintino for Chapter Header, and secondary heading; condensed Garamond as the body text; Courier for “screen printouts”.

Published and edited with <3 dually in the Rigid State of New Hampshire and the lovely Minibrick State.

i Introduction

The future is all IP

The Future will be Rewritten to be all VOIP.

Cisco is the future.

Just kidd.

In seriousness, if you're all wired at home, or you are interested in wiring up your home for multi line telephony or have the ability to answer calls from a number of phones or internally call people from within... I think given the consolidation and the access to them, the recommended path is to Cisco. As much as I can't stand a lot of their technology, you do not need to need to have everything running on Cisco to do Cisco telephony. Being frank. I have switches using Netgear, and I have some third party endpoints.

The advantage with Cisco is

- The "Gold-standard" in VOIP
- Much easier to acquire, at a reasonable secondhand price (other devices that may be even better like MutiVOIP gateways, are really expensive)
- Gets your feet wet into other platforms such as wireless, firewalls (ASA), etc.
- Voice quality sounds better than any standard POTS phones, using the same POTS wiring from your phone company
- They are more open to the standards-based, meaning you don't have to be a Cisco showoff at the tables, desks. You could use SIP phones from other vendors now than historically.

I am writing this because a lot of the content out there is **sh*t.** Whether it's written by guys who haven't had proofread his own copy, or someone with U.S. English as a Second Language; or people who are so elitist (bloggers that card-carry) their Cisco certs, they're like "it is what it is!"

First off, this is the third attempt in understanding Cisco IPT and this comes from a telephony point of view, with the legacy digital and analog systems. I tried this in 2011, to a massive fail, another attempt in 2015 (while it worked, it was still buggy and I corrupted the router) and by 2019, I started to "settle" with Cisco. The cutover was done in early 2020, and for the first time in 5 years, there has been a functioning telephone system that has done both internal calling and handling outside calls. It's too long of why that was, and that has been previously been documented. All with trial and error (and countless web searches, and I will refuse to say "Google it" if you're learning.) And the Cisco textbooks are very condescending in tone.

There's also a sidenote in 2016 to 2017, there was an attempt to use an Asterisk distro using it on a VM instance. I had some third party SIP phones, but learned the hard way that some SIP phones are more cumbersome than others.

It is recommend that before starting to work with Cisco Voice; learn the following:

- The general knowledge of Cisco's IOS
- How to set up Ethernet ports, on the routing; and switching
- How to set up Virtual Local Area Networks or VLANs to appropriately set up voice, data, or even test VLANs
- How to set up Telnet, and most importantly SSH.
- To start off, it's probably best to use the infamous Cisco console cable and use a serial port or a USB to Serial adaptor if your computer doesn't have one.
- The confusing language of ephones, voice pools, and what's telephony-service and what's not before putting all this together
- that SCCP stands for Skinny Call Control Protocol, that is Cisco's Proprietary protocol; and Session Initiation Protocol stands for SIP; and this where third party phones could come in. OR could be used as a "VOIP trunk" from an Internet Telephony Service Provider or ITSP.

IMPORTANT: You will need Packet Voice Data Modules or **PVDMs**, they look like RAM sticks, but they are DSPs. They vary, but one that costs about $15 on the secondhand sites, it can vary but simplicity is the theme. If you do not have PVDM**, the telephony-service will not work at all.** And if an eBay seller explicitly sells you a router with CME and it you get a no PVDM message, I would fight to get a partial refund by claiming "not as described" because **you cannot use CME/telephony service if you don't have a PVDM -at all**. If an item is advertising CME abilities, or has CME installed, then gawddang it; it needs a PVDM!

Conventions

Important information will be sidenoted with the phrase **Important**

What Cisco calls a "Caveat" is our phrase sidenoted for **Caution**

`Terminal printouts are in this font; explicit` **`commands to input into a terminal is in boldface`**

CHAPTER 1

CME - Not to be Confused with Chicago Merc. Exchange

This document's focus will be on Cisco IP Telephony or Unified Communications, or whatever the brand of the day.

First off, Cisco's **Unified Communication Manager Express** was once known as Cisco **CallManager Express**; that was once known as Cisco's **IOS Telephony Service** (may run into the ITS phrase somewhere along the line), that was based on an acquisition by one of the 145 companies Cisco bought since the corporate inception to recent years. The latter brand is still well known in the IOS command prompts, which could confuse you. This runs on various routers, is part of an IOS train that ends on a filename with "enterprise" in the main IOS router binaries.

It can also run Survivable Remote Site Telephony or SRST. If you are running the PBX/software switch, the big Unified Communication Manager, you should know about SRST. As a sidenote, SRST basically a handicapped UCM, though if you're a small site, like your house, you'll see similarities in commands.

The Home CME Rig

Included in these examples are

- Cisco 7900 Series IP Phones (will be programmed as **ephones)** Cisco 7970, 7961, 7965 using Skinny Call Control Protocol (SCCP)
- 2 Polycom IP 311 SoundPoint desksets using Session Imitation Protocol or SIP
- 2 Avaya 9611 VOIP sets using SIP firmware
- 1 Cisco 2801 Integrated Services Router, using Cisco IOS 11 and CME 8.5
- 1 2 port FXO Voice Interface Card or VIC.
- Netgear GS108 POE Managed Ethernet Switch
- Netgear GS728TPv3 Gigabit switch, (acting as a core switch between the machines and edge users.)

After doing general configuration on Cisco routers and switches, it's now time to introduce VOIP to the router (and I say "VOIP" to stay consistent to character capitalization) as there is some protocol-specific things you should be doing first.

After logging into Executive Privilege mode; and going into config mode like you see here

```
CORE1#
```

At this point you want to enable VOIP services across your router enter:

```
voice service voip
```

enter the following commands

```
CORE1#(voice-serv-voip) allow-connections h323 to h323
CORE1#(voice-serv-voip) allow-connections h323 to sip
CORE1#(voice-serv-voip) allow-connections sip to h323
CORE1#(voice-serv-voip) allow-connections h323 to sip
```

This will allow SCCP sets to talk to SIP sets and vice versa. If you don't have SIP phones or trunks, then do not enter the said commands, leave it to "h323 to h323".

Caution! Beginning with Cisco CME 8x, they require the router to "bind" the source interface to prevent VOIP hacks, and disruptions. Unfortunately, not all applications are doing this, and by "binding" this means that only CME traffic can live and move across those interfaces (wheher it's VLAN or a dedicated interface like a Fast Ethernet). If you are on vacay and want to call your folks without making an actual telephone call, it's best you use a VPN and a firewall on both sides. **You should never put your source-interface on an external facing IP address!**

For reliability purposes across the system, it's best you enter the following (adjust IP address accordingly to your environment, this is an example.) Ensure you're in config mode, and exited out of any sub-prompts (single string command)

```
sccp local FastEthernet0/0 sccp ccm 172.16.18.2 identifier 1
version 7.0 sccp ip precedence 1
```

If you have other Cisco IOS routers with Voice support, you could actually make it be a backup system.

I did this as well to ensure reliability (note: this is a single string of commands)

```
sccp ccm group 1  bind interface FastEthernet 0/0  associate ccm 1
priority 1  registration timeout 30  keepalive timeout 30  connect
interval 3  switchback method graceful
```

Another important feature for the overall routing of calls is called a "dial peer". While it can be very confusing at first; put it this way, VOIP is a multi stack protocol that can go amonst various mediums (a single line telephone to a T1 circuit, or a Foreign Exchange Station to call to a SIP PBX set up for voicemail.) Because it can do a lot; the router needs to know when it listens to DTMF commands, to know what to do with it.

In this example I have it set to a FXO so when any user of any telset wants to dial out, they can. And for each type of number, needs it's own dial-peer. Even if you have Comcast, Charter, or some residential telephone line where any domestic number won't be a charge, you'll still need to keyboard exercises to basic calling

In configuration mode: enter

```
CORE1#(config)dial-peer voice 201 pots
```

I treated the tag like it was a Trunk Access Code; it's an admin friendly tag… doesn't do anything if you dialed 201 literally

```
CORE1#(dial-peer) description Local Outbound
```

It's recommended to add a description so you know what the peer will do

```
CORE1#(dial-peer) preference 1
```

I personally added this to make it work; consult other notes if this doesn't fit your needs

```
CORE1#(dial-peer) destination-pattern 8[2-9].......
```

To mimic the traditional Trunk Access Code, just add your preferred number before you add the string… but…

```
CORE1#(dial-peer) forward-digits 7
```

For a 7 digit number ensure you explicIty say "7" rather than "all" because then it will dial that TAC.

```
CORE1#(dial-peer)incoming called-number ..........
```

I believe this is the way to route calls or allow to the phone to interpret the inbound CLID number. I copied this from a config example but didn't explain the details.

```
CORE1#(dial-peer) port 0/1/0
```

You're telling the dial peer to port on a voice card to dial out. In this example, it's *gateway* 0, *slot* 1, *port* 0, that matches to an 2 port FXO card

In the example for 10 digits, if you have free long distance on the same trunk (especially with many US cable providers… I just used a 9 as a TAC because these phone systems predate low cost calling, and as such this environment is the example in this entire document:

```
CORE1#(config) dial-peer voice 208 pots
CORE1#(dial-peer) preference 1
CORE1#(dial-peer)destination-pattern 81..........
CORE1#(dial-peer)port 0/1/0
CORE1#(dial-peer)forward-digits 11
```

In North America, literally 10 plus 1 (as in our country code) equals 11. Ensure the "1" gets forwarded out too!

If you live in an area (especially in a deep-red, politically rigid state in the U.S.) where PBX systems are likely to kill you more than some creepy man with a domestic violence rap sheet – sometimes known as Kari's Law - you could enter something along the lines of below

Note: Please consult with local regulations on how the emergency services number should be dialed and/or be tested or how to handle inadvertent dialing. The law doesn't indicate homes and residences are required to comply; but if you are so scared, check with a legal-eagle first.

```
CORE1#(config)  dial-peer voice 911 pots
CORE1#(dial-peer) preference 1
CORE1#(dial-peer) destination-pattern 911
CORE1#(dial-peer) port 0/1/0
CORE1#(dial-peer) forward-digits all
```

CHAPTER 2

Setting up Cisco CME without the *setup* command

For versions prior to 8x, the "setup" command has been "depreciated", but yet it's still seen on the Cisco router. It literally gets bitchy and basically tells you to eff-off and configure it the more complex way.
Also for $85 on the auction site named after the Ebola virus in the summer of 2018, the seller it did include the GUI. While I had an image for the GUI, it was for Release 4. And yet all the nerds love to brag doing everything in terminal.

I also defend GUI because it also helps you for the small things, like changing the time at the end of the Daylight time, if you happened to fudge it up, or you want to change the music on hold, stuff like that. I consider the GUI to be the admin, and the terminal for heavier lifting tasks like the routing, etc.

If you so choose to, you're going t have to set things up line by line.

If you have not assigned a DHCP pool yet, and the VOIP VLAN is separate to your data VLAN; you're going have to enter (if you don't have a DHCP server for the sets)

```
CORE1#(config)ip dhcp pool ITS
```

I used "ITS" for the sake of legacy uses because this is what would've been entered for the name if you had "setup" still in use.

```
CORE1#(dhcp) network 172.18.2.0 255.255.255.0
```

Network means the server will spit out IP addresses along the network it's bounded on, by the IP address set up on the interface. I used FastEthernet 0/0 because I felt internal/private LAN should be on the lowest port possible. Your option 150, handles the TFTP server, which will need it's own discussion later.

```
CORE1#(dhcp) option 150 ip 172.18.2.2
```

The CLI equivalent to DHCP Option 150 to pull phone files via TFTP to the IP address that's in your router (that's recommend)

```
CORE1#(dhcp) default router 172.18.2.2
```

I did use the same IP address as the CME LAN, for the sake of stability

```
CORE1#(dhcp) dns-server 172.18.2.16
```

Ideal for not having to put IP addresses on everything, but not required. This was tied to an internal DNS server.

Note: It is strongly recommended if you don't have DNS server to use a "proxy" or the local IP address of a WAN facing router. Please do not use Well Known DNS servers in the heart of local appliances to prevent any disruption, hacks, and other nefarious behavior if the device or router gets attacked.

Now enter telephony service by typing in:

```
CORE1#(config) telephony-service
```

Now you want to tell the router how many Cisco SCCP phones you want to have, say

```
CORE1(telephony)# max-ephones 24
```

For the number of extensions, double it plus a few more (especially if you want to do paging and Key system functionality)

```
CORE1(telephony)# max-dn 80
```

you want to remind the telephony service where its assigned on

```
CORE1(telephony)# ip source-address 172.18.1.2 port 2000
```

where port 2000 is the default pipe for Cisco's SCCP

```
CORE1(telephony)# system-message  (('')) Merry Halloween (('))
```

Do you want to change the line of "Cisco Unified CME" (that on the big CallManager it's known by "Your Current Options" above the softkeys?) Only CME or an Asterisk release can this line be redefined.

```
CORE1(telephony)# voicemail 8*97
```

The voicemail button could in theory dial any pre-defined number, and I just used the most likely default carrier number if you don't have Unity Express installed. When this was tied to the Museum's Definity G3 PBX, it could dial into AUDIX. (Another subject for another day.)

```
CORE1(telephony)# moh music-on-hold.au
```

ensure your music-on-hold.au meets Cisco's spec and it's living on the flash drive

Entering IP phones is not done in telephony-service at all. This could be because it could be used for SRST functionality, and they kept that prompt at the lowest level. In theory; you can't do SRST and CME at the same time.

CHAPTER 3

The Numbers Game

There are several types of telephone numbers, known as "tags" to identify the individual extensions, sets, etc.; similar to how an Internet routing path works.

ephone for **[Selsius]** Ethernet Phone, or known today as Cisco IP Phone (the company Cisco acquired in the late 1990s)
ephone-dn for Directory Number (easy if you came off from Nortel)
voice register pool for a SIP Phone, whether it's a Cisco or not
voice-register-dn the number for the SIP phones or devices
SIP and Voice Registers are in a separate chapter As of this writing in 2020, the CME releases prior to 9 are essentially becoming End of Life; and SIP is now becoming the only protocol Cisco will officially support, no new models since the mid 2010s uses SCCP exclusively; **it is important to note that** CME doesn't work well with SIP natively if you have the 7900 series with SIP firmware.

Since version 11, it only handles SIP at the desktop level, SCCP only works if you still have the VG200 analog phone gateways; or the smaller FXS cards. The more recent, 7800, 8800 and 9900 models **only** runs on SIP protocols. In reality, SIP is much more mature (but still not the best VOIP protocol around), and the newer generations of Cisco IP Phones with the SIP stack reflect it. The 7900 was never really intended to take all the lack of benefits to SIP and not only that the very original generations the 79×0 models limited it's use to support BLFs, the speaker was half duplex, and the audio quality was at level of a POT set, over a set that had a Skinny firmware; that did the polar opposite.

Trunk mode: Key System ("Call on Line 1")

As much as I admire the 1A2 systems, and my poor fine motor skills, I don't want to forget history of the largest types of systems, PBX are actually a microcosm in the telephony world. If you got a 796x, this will be easy; without loosing button space for other line appearances.

In this example, this would be a phantom extension

```
CORE1(config)# ephone-dn 60
CORE1(ephone-dn)# number 800
CORE1(ephone-dn)# label Line A
CORE1(ephone-dn)# name Line A
```

If you have *Call Waiting* and you enabled *Switchhook* flash in telephony-service, you could set this up to be a dual-line, to achieve this, to do that, all you would need to do is add "ephone-dn 60 dual-line" to the end if you did this already, and made a mistake you will need to remove it (using "no ephone-dn 60" as the example) and reenter appropriately.

FXO Configuration (voice-port)

Now it's time to define that FXO port to match that phantom extension. Now you'll branch out to the trunk level and assign various commands to make this work:

```
CORE1(config)# voice-port 0/1/0
```

Which means on the Cisco 2801, it's on *router* 0, *slot* 1, *port* 0 (the female jack), similar to 01A0401 on say a large Avaya PBX.

```
CORE1(voice-port)# supervisory-disconnect anytone

CORE1(voice-port)# groundstart auto-tip

CORE1(voice-port)# timeouts-interdigit 6

CORE1(voice-port)# timeouts call-disconnect 28
```

I use this because the Xifinity Digital Voicemail, and if no one answers by the 6th ring, it stops ringing. If you do not have this type of timeout, the router "answers" the call and rings to the command below. If you do not insert this, the phones will "ring off the hook". In fact it concerned one of my grandmother's doctors who just happened to call on the same day of the final cutover, and I realized I omitted it.

```
CORE1(voice-port)# connection plar opx 200
```

This means it will connect to a **Private Line Automatic Ringdown** to **Off Premise Extension** of 200, the ephone-dn 60 for the phantom "Line 1". Ironically OPX must think the extensions is off premise to the router's mind.

```
CORE1(voice-port)# description Comcast POTS  caller-id
enable
```

If say you wanted only one phone to answer the call (which this would act like a PBX, where a single point of entry), you'd use the DN that is of the extension you wanted to answer, such as say 101. For the home, it's easier to lump sets on a phantom extension in case of some network flare up that caused a phone to go down, someone unplugged a set, etc. I'm a clumsy guy, and if you had one set go down, a busy signal would generate on the other end, because no device is able to ring.

There was some issues where the phones would ring once every 10 minutes or so. Upon a google search, I found entering the following in the config mode, this helped

```
CORE1(voice-port)#  voice class custom-cptone Comcast

dualtone disconnect

frequency 480 625
```

ephone-dn (Extension Number)

As previously explained, the ephone-dn is for the extension. For CME versions 8 and higher, there is more than just an option **dual-line**, but an **octo-line**. Octo means 8 different instances of calls can come into a CME instance without a busy signal.

Note: **octo-lines are recommended for operator positions or meet-me conferencing.**

Though in Cisco, your call appearances are virtual and requires juggling using softkeys and no hard keys like how Avaya is known for. With SIP, you could return back to the Avaya-style as call-waiting or appearance buttons are device-centric and not phone system specific, which in a lot of ways could be easier. Octo-line is ideal for call-parking, which will be discussed in the future.

Depending on that PVDM that's required to anything, even if you were just playing with commands, the more instances of calls will chew up resources on the PVDM (this is a buffer card to convert voice into packets and vice versa), even if you are within the limits. This is because after you assign a Cisco IP Phone, the ephone command already populates the lines after configuring, so that means that phone will have dedicated lines literally.

The SCCP protocol can support up to 200 simultaneous calls, and versions 8 and higher are limited to 8 because of CME running under the router. In the full fledged CallManager or UCM, all the call processing is software based and is done by the server. CME does not work that way, hence why you loose the theoretical by 192 instances. There are loopholes to mimic call appearances but it's outside the scope of the beginners guide.

Setting up Cisco IP Phones

If you did the **ephone-dns** for all your relevant extensions; and you're in no hurry to add BLFs, you could *in theory* use auto-reg-ephone in the telephony-service function, and plug in the phones one at a time, to then to match extensions to phones. But if you want to do more, one at a time maybe your only bet is to follow this guide.

For all intensive purposes, start with one. In config mode enter the following

```
CORE1(config)# ephone 1

CORE1(ephone)# mac-address 0000.0000.0000
```

In mac-address use the MAC that is on your VOIP set that you're adding. Also ensure it's split up into threes and add a "." Every 4 characters in the hardware address.

```
CORE1(ephone)# type 7970
```

Not specifying type may result in unpredictable results.

```
CORE1(ephone)# button 1:1
```

```
CORE1(ephone)# button 3m70
```

For all intensive purposes, "button 1:1" means Button 1 is tied to Directory Number 1, extension 100; button "3m70" is functioning as BLF for the phantom extension of 700 that's acting as a Key line and is **monitoring** the line; you can access it and it will ring when the number receives a call.

CHAPTER 4

TFTP: (Sometimes known as the "CUCM address")

TFTP and call management in Cisco land is the same. It's assumed you are treating your router to just do voice, and you're not mixing this with another network like an ASA or an AirPort Express with it's own IP network, etc. (I am not the only one that had this inexperience.) You use SolarWinds or tftp32d to *insert new files* to the Cisco router; then use the **tftp-server** command to serve the files for the actual sets themselves. You typically don't use the laptop/management PC's TFTP server to have the phones get their files. Why?

- --

TFTP and call management in Cisco land is the same.

- --

Now depending on the files, you're going to have to do this individually. If you have some mind in IOS, you can do in config mode, **tftp-server flash:loa** [first three letters of the file, then **Tab**] you can speed up this process. This flags the files living in the flash: directory this can be spit out to the TFTP server you have previously set up. Ensure that in config mode your **tftp-server source-interface** is set on the same network/subnet that the VOIP is running under.

Your going back to the telephony-service function yet again, this is where you enter in the "Loads" for your Skinny phone. *Why if say it works out of the box and it registers?* You may run into some bugs. My Cisco 7970 which I've had for years; didn't understand the quad-lines very well, and it locked up, and sometimes would constantly reload. The firmware dates back to CME 4 years and perhaps it needed a little more up to date code so it would work better.

You type in the telephony-service prompt in config mode the following:

```
CORE1(telephony-service)#  load 7970 [filename without the
.loads, or .default]
```

Note: change the model number if *different* from this example. There is roughly 6 files, and it's best you put them in the order that the firmware documentation has it.

The Cisco IP Phones basically phones home via TFTP and if it sees a new bootloader and firmware and checks against what it has, it should restart and attempt to upgrade.

There are catches and gotchas, ensure you have read Cisco's documentation on upgrading, because hopping well past major versions will make the phone become a brick, or totally just ignore the new files. For an example, you can't go to a firmware version 9 unless you have patched to 8 if the version is below 7. Understandable for IP appliances, yeah?

CHAPTER 5

Session Initiation Protocol

In 2021, as previously mentioned SIP is replacing most of the desktop phones; over time. SIP stands for Session Initiation Protocol. H323 or SIP is neither better or worse of standards, it only differs how the phones connect and disconnect calls. Cisco's SCCP uses H323 standard, even though the protocol itself is proprietary. Cisco refers to it's call controller by it's "TFTP Server" (see Chapter 4) as it handles the configuration (buttons and lines, sometimes for the user like backdrops and ring tones), the time clock (the time of day and day of the year), the signaling (the tones), and the transmission (the human conversation). SIP could do any of the four things and could be separate devices on separate networks and domain names (like a DNS address).

The basis of the SIP "stack"/protocol/extension supports

- Video chats
- Audio calls (err the traditional telephony)
- Instant Messaging (known as "texting", "messaging" or "chat" if someone has been born after 1994.)
- The basic SIP telephony stack supports essentially all standard 19 Custom Calling Features that the phone company used to provide separately, that many broadband phone companies provide for free or is included at a much lower price rate.
- The protocol literally revolves around Caller ID. "Display Name" and "SIP Alias" is equivalent to the "station-id name" and "station-id number" in the FXO ports respectively in the IOS configuration if you wanted to do ethical caller ID spoofing.)

As of mid 2020, the consumer equivalent or analogy to SIP is Zoom, that actually uses some or most of the SIP standards; while Skype is similar in nature, the difference is Skype is proprietary. Also after the acquisition by Microsoft, they turned Skype's signature P2P or Peer to Peer Protocol into a centralized protocol, basically Skinny for consumers and some businesses. (And a way for Microsoft to spy on their users…)

It's a blessing because Any Brand Company (ABC) SIP phones that adhere to the SIP specifications could work, but may work better on an ABC softswitch, or ABC could produce it's own SIP phones designed to work on any system (basically a glorified landline phone with an IP stack.) Likewise, Cisco, Avaya, Mitel, and others have added their own "stuff" in the SIP stack, albeit travels along the same pipe as SIP; mostly to replicate the traditional phone-system-type features that the standard SIP does not support.

Nerds behind the invention basically didn't just think you could go to a Radio Shack and buy a $40 SIP phone to tie into your office's VOIP network (in fact an Avaya executive in 2005 had made such prediction… just that Radio Shack closed, and SIP phones started to make their appearances in some consumer fronts after) that it could also go the other way around. You could turn a Cisco SIP phone into a modern day landline; or any other set, with some expert configuration, this is where "flexibility" of the stack is touted for.

The nerds felt SIP would be the future as they felt, phones, wouldn't be around forever, and that apps would replace the hardware devices. Also the believers felt that telephony should adapt the web standards… because for an example when a user picks up a SIP phone and dials an extension or telephone number, what the backend does is it basically pulls up a URL that "resolves" a telephone number instead of a web page when using the omni bar.

Related to the CME: for all intensive purposes, the rest will assume you're using an IP connected phone that's closer to the functioning level of "standard" 2500-type telephone. The similar conventions of assigning a ol 2500 type set to a Voice Gateway (or FXS ports) should be the way think in this setup. Can you make your SIP set work more like a Skinny? This is where SIP gets a bit complex, you would have to go on the device side to make that possible.

Note: *As previously explained , the Cisco 7900s can do SIP, but it doesn't gel well CME; but newer 8900, 9900 and 8800 and 7800s could work a bit better though unverified by me.*

At the lowest config level mode, you want to enter the following commands (similar to telephony-service; you need to do this to get the SIP sets to work)

```
CORE1(config)# voice register global

CORE1#(config-voice-reg) mode cme

CORE1#(config-voice-reg) source-address 172.18.2.2 port 5060
```

Port 5060 is the default SIP port, ensure you have binded SIP as well as well as you did in H323

```
CORE1(config-voice-reg#) max-dn 52
CORE1(config-voice-reg#) max-pool 24
CORE1(config-voice-reg#) timezone 21
```

21 answers to US Eastern Time

```
CORE1(config-voice-reg#) ntp-server 172.18.1.2 mode
multicast
```

Like with the IP Phone (ephone) prompts; you want to do the same for the SIP phones, this time it's called **voice-register dn**. To create the first extension, use the following example in config mode

```
CORE1(config #)voice-register dn  40

CORE1(voice-reg-dn#) session-server 1

CORE1(voice-reg-dn#) number 200

CORE1(voice-reg-dn#) allow-watch
```

allow-watch the phone can function with BLF; monitoring off/on hook statues.
To setup the physical telsets; instead of ephone, it's called **voice register pool** (I know in Cisco's DHCP "pool" means *devices,* perhaps this is why it's called such.)

```
CORE1# (config) voice-register pool  1

CORE1# (voice-reg-pool) id mac 0000.0000.0000

CORE1# (voice-reg-pool) type P600

CORE1# (voice-reg-pool) number 1 dn 1

CORE1# (voice-reg-pool) username 200 password 6030000200
```

In this instance, I zero'd out the MAC address, supposedly the way the SIP sets authenticate, doesn't need MAC addresses (that is from the H323 days), it doesn't hurt. P600 is the defined model for Polycom's IP 600 line, and the model I used was the SoundPoint IP VVX 310, and nothing had changed for a number of years, what could hurt?

If the SIP set isn't exposed to the WAN or the wider Internet, there isn't a real concern for strict passwords, I treat it as PIN. And for this instance; my VOIP network is not exposed; so PINs are OK, the example mimics ISDN, such as the full "phone number". If there is WAN exposure, there should be firewall or proxy to **begin with** and using a proxy – this is where on the server side, could have the random password, then for the user side (or user-agents) you could use a PIN, especially for cube hoppers.

The following is unrelated to Cisco: but if you subscribe to "cloud phone service" and your router/firewall is not used to proxy VOIP calls, to strongly take a second look; particularly in the Polycom installs.

At this point you may need to consult your SIP phone's documentation on how the set should register into the Cisco. Most SIP-compliant phones should work, it's at the unique model and vendor where things have to be specific. That's outside the scope of this note. Things to look for is: the "proxy" address should be IP of the router; the "register" address should be the same. NTP or the time server should be the same as well. The username and password should meet what is set up in **voice register pool [tag #]** In other systems; some of the settings in the **dn** and **pool** would be reversed (username would go along with the extension, rather than the hardware.)