sipxcom Documentation

Release 20.04

Support Team

Oct 29, 2021

CONTENTS:

CHAPTER

ONE

HISTORY

The code that now makes up the sipXcom project was first contributed to SIPfoundry by Pingtel Corp in 2004 as the sipXpbx open source project Pingtel organized and funded the creation of SIPfoundry, but at the time it was an independent legal not-for-profit corporation. The PBX code had been under development as the Pingtel SIPxchange product for a couple of years at that point, and was itself an extension of code from the Pingtel Expressa phone before that. The Pingtel Expressa phone was one of the phones that made the very first successful SIP calls between implementations from different vendors. For some time after that initial release, development of the Pingtel commercial product continued in parallel with the sipXpbx open source code base, with contributions continuing from the commercial version to the open source. Over time, the sipXpbx open source project gained attention from both users and developers. A number of active developers from outside Pingtel began contributing heavily, especially in those parts of the code base that were most useful to developers of SIP User Agents (phones and other endpoints). Because of the licensing structure that was used at that time, it was sometimes difficult to incorporate those changes into the commercial version at the same time, so the two code bases began to diverge - which in turn increased the complexity of creating and especially testing both the open source (sipXpbx) and commercial (SIPxchange) versions of the product. In the spring of 2007, we decided to change the project structure and licensing to reduce complexity and potential usage conflicts. Pingtel made a new contribution of code to SIPfoundry based on its previous commercial version to create the sipXecs project. Many of the developers who were focused on User Agents continued with those parts of the original code base as the sipXtapi project.

1.1 Commercial sponsor history

As noted above, this project began with contributions from Pingtel. Pingtel began life focused on the development of VoIP phones, and quickly narrowed that focus to SIP phones. The Pingtel Expressa phone was considered one of the best SIP phones in the business at one time, and nearly every SIP interoperability lab had at least one for testing (some probably still do). It was, alas, not a commercial success, and Pingtel eventually shifted its focus to the SIPxchange PBX product and the accompanying open source projects as described above.

In July of 2007, Pingtel was acquired by Bluesocket, which in turn sold all the assets of Pingtel to Nortel in August of 2008. Nortel had already been selling a product (Software Communications System - SCS) based on code from the sipXecs project for some time. When Nortel bought the assets of Pingtel, they also hired all of the developers from Bluesocket. Nortel greatly expanded the developer base and the features of both its SCS product and the sipXecs open source project. In January of 2009, Nortel declared bankruptcy; while in bankruptcy, development and contributions to the open source project continued, and in December of 2009, the commercial product was acquired by Avaya. In March 0f 2010, Avaya stopped contributing to the source code to the community. Members of the community came together (including several of the original Pingtel employees under the company name eZuce, Inc.) and until the beginning of February 2015 maintained the sipXecs project. Citing differences with SIPfoundry ownership, eZuce decided in February 2015 to fork sipXecs to form sipXcom. eZuce continued to improve the code base until August of 2020, when CoreDial LLC aquired eZuce.

CHAPTER

TWO

FAQ

2.1 Why sipxcom over asterisk, freepbx, etc?

The biggest difference is sipxcom proxy is a stateless proxy, where other proxies such as Asterisk are B2BUAs.

This means sipxcom is only involved in the call setup. It is never involved in relaying audio or video (RTP) media unless you're using a b2bua function, like *Conferencing* , *Voicemail*, *Auto Attendants*, or *Call Queue*. Once there is a 200 OK with SDP to a INVITE, and ACK to the 200 OK with SDP, the media (RTP) is direct between phone to phone.

Because of this sipxcom (on sufficient hardware) can handle 10s of thousands of SIP transactions per second, per proxy instance. Some services such as proxy and registrar can run on multiple servers, increasing capability and reliability.

Compare against Asterisk where their wiki indicates the calls per second rate is somewhere between 30 to 40 on a HP DL360, and it is standalone. At the time of this writing, that wiki entry was last modified Jan 31st of 2011. In 2011 a (Gen 7) HP DL360 would support a max of two Intel socket FCLGA1366 (Xeon 55xx) processors and 384GB of RAM.

CHAPTER

THREE

FEATURES

3.1 System Application Services

All the sipXcom application services are allocated to specific server roles. Using the centralized cluster management system each role can be instantiated on a dedicated server or several (all) roles can be run on a single server. Configuration of all services and participating servers is fully automatic and Web UI based.

- SIP Session Router, optionally geo-redundant and load sharing
- Media server for unified messaging and IVR (auto-attendant) services
- Conferencing server based on FreeSWITCH
- XMPP Instant Messaging (IM) and presence server (based on Openfire)
- Contact center (ACD) server
- Call park / Music on Hold (MoH) server
- Presence server (Broadsoft and IETF compliant resource list server for BLF)
- Shared Appearance Agent server to support shared lines (BLA)
- Group paging server
- SIP trunking server (media anchoring and B2BUA for SIP trunking & remote worker support)
- Call Detail Record (CDR) collection & processing server
- Third party call control (3PCC) server using REST interfaces
- Management and configuration server
- Process management server for centralized cluster management

3.2 SOA Architecture / Business Process Integration using Web Services

- Web Services SOAP interfacefor key administrative functions
- Web Services REST interface for user portal functions and third party call control
- All components centrally managed using XML RPC
- Google Web Toolkit (GWT)

3.3 Core Calling Features (Telephony Features)

- Transfer (consultative & blind)
- Call coverage
- Call hold / retrieve
- Consultation hold
- Music on Hold for IETF standards compliant phones
- User-specific MoH files
- MoH music from an external streaming source
- Admin or user configurable Busy Lamp Field (BLF) presence and softkeys
- Shared Line Appearance / Bridged Line Appearance (Polycom only)
- Uploadable music file
- 3-way / 5-way video and voice conference on the phone
- Call pickup (global and directed call pickup)
- Call park & retrieve
- Hunt groups
- Intercom with auto-answer (bi-directional)
- SIP URI dialing
- CLID (Calling Line Identification)
- CNIP (Calling party Name Identification Presentation)
- CLIP (Call Line Identification Presentation)
- CLIR (Call Line Identification Restriction)
- Per gateway CLIP manipulation
- Call waiting / retrieve
- Do not Disturb (DnD)
- Forward on busy, no answer, do not disturb
- Multiple line appearances
- Multiple calls per line
- Multiple station appearance
- **Outbound call blocking - Calls from phones to PSTN numbers, or classes of numbers, can be blocked based on:**
  - The destination of the call; for example, when a user or device cannot initiate an international long distance call.
  - The source of the call; for example, when a lobby phone can only initiate calls to internal numbers.
- Click-to-call
- Redial
- Call history (dialed, received, missed)

- Auto off-hook / ring down
- Incoming only
- Configuration of individual Speed Dial softkeys
- Auto-generation of directory information

3.4 E911 Emergency Response

- Internal notification using email and SMS

3.5 Remote Branch office support

- Centralized deployment: Branch only provides phones and optionally PSTN gateway for failover, reduced WAN BW consumption or E911 calls
- Distributed deployment: Branch provides full call server with SIP site-to-site dialing between offices
- Branch office locations can be defined in the mgmt UI with a postal address
- Users, phones, gateways, SBC, and servers can be assigned to a branch location
- A PSTN gateway can be available for calls that originate in a specific branch only or for general use
- Source routing allows call routing based on location (branch local calls are routed through local gateway preferably)
- Branch postal address automatically proliferates to user's office address
- Survivable branch configuration possible with Audiocodes gateways SAS functionality (auto-configured)
- Certain sipXcom services can be deployed in the branch as part of the cluster (e.g. conferencing)

3.6 Enterprise Instant Messaging (IM) and Presence

- XMPP based IM and presence server based on Openfire
- Supports XMPP standards based clients
- Auto-configuration of user's IM accounts
- Auto-configuration of IM user groups
- Personal group chat room for every user auto-configured
- Federation of phone presence with IM presence
- Customizable "on the phone" presence status message
- Dynamic call routing based on user's presence status
- Message archiving and search for compliance (pending)
- Server-to-server XMPP federation
- Optional secure client connections
- Client-to-client file transfer

- Group chat rooms
- XMPP search
- Integration of user profile information and avatar (pending)

3.7 Personal Assistant IM Bot

- My Buddy Personal Assistant feature
- Dynamic call control using IM
- Dynamic conference management using IM
- Unified messaging management using IM
- Call history / missed calls
- Call initiation using corporate dialplan
- Corporate directory look-ups

3.8 Presence and IM Federation

- Server side federation with other public XMPP IM systems
- Allows group chat sessions across systems
- Allows message archiving (if enabled) across systems
- User self-administration of credentials for other IM systems

3.9 Fixed Mobile Convergence (FMC) Application

- 3rd Party FMC application with the following functionality:
- Enterprise number dialing
- System call-back saves on wireless toll charges
- Corporate directory look-ups
- Call history
- Presence sharing
- IM

3.10 Web Conferencing & Collaboration

- Commercial options available through eZuce's viewme and viewme Cloud products

3.11 User Self-Control (User Web Configuration Portal)

- Every user on the system gets access to a personal Web user portal for self-management and control
- Management of unified messaging (voicemail)
- Configuration of unified messaging preferences
- Time based find-me / follow-me
- Flexible configuration of call forwarding
- Management of personal profile data including avatar
- Personal call history
- Personal phone book, speed dial and presence management
- Click-to-call
- Individual phone management
- Personal auto-attendant
- Management of personal IM account
- Personal MoH music upload and preferences

3.12 Superior Voice Quality

- Peer-to-peer media routing for best quality (media not routed through the sipXcom server)
- Unmatched voice quality with lowest delay and jitter
- Support for any codec supported by the phone or gateway (including video)
- Support for HD Voice (Polycom and other phones)
- Codec negotiation (no transcoding required)
- Conferencing, auto-attendant and voicemail support HD voice w/ transcoding if necessary

3.13 User Management

- Create a user, provision a phone and assign a line in only three clicks - easy!
- Numeric or alpha-numeric User ID
- User PIN management (UI or TUI)
- Aliasing facility (numeric and alpha-numeric aliases)
- Extension and alias uniqueness assurance
- Management or auto-assignment of user's IM ID and display name
- Automatic IM buddy list creation based on user groups
- Granular per user permissions

3.14 Call permissions

- 900 Dialing
- International Dialing
- Long Distance Dialing
- Mobile Dialing
- Local Dialing
- Toll Free Dialing

3.15 System permissions

- User has voicemail inbox
- User listed in auto-attendant directory
- User can record system prompts
- User has superuser access
- User allowed to change PIN from TUI
- User can use Microsoft Exchange VM
- User has a personal auto-attendant
- Custom permissions as defined by the admin
- Supervisor permission for groups (e.g. Call Center supervisor)
- Management of user contact record (user profile)
- Comprehensive profile data
- Work and home address
- In-building location information
- Assistant information
- Support for avatar including support for gravatar
- SIP password management for security
- User groups with group properties
- **Per user call forwarding (follow me)**
  - To local extension, PSTN number, or SIP address
  - Based on user or admin defined time schedules
  - Parallel or serial ring
  - Allows definition of ring time before trying next number
  - Allows several forwarding destinations
  - Follow-me configuration using user portal
- Extension pool with automatic assignment
- Per user Caller ID (CLID) assignment

3.16 Dial Plan

- Easy to use GUI based dial plan manipulation
- Time-based dialing rules with different admin defined schedules
- Rules based least cost routing
- Dynamic call routing based on user's IM presence status
- Directly route to voicemail on IM status DND
- Dynamically add forwarding destination based on phone number in custom presence status
- Automatic gateway redundancy and fail-over
- Specific E911 routing
- Permission based rules
- Prefix manipulation
- Dialplan templating for international dial plans
- Built-in support for U.S., German, Swiss, and Polish local dial plans (Any other local dial plan can be added as a plugin)
- Specify internal extension length
- Specific rule for site-to-site call routing between SIP systems
- Redirector plugins - any imaginable dial rule can be added as a plugin

3.17 Internet Calling

- Ability to configure SIP URI based call routing to other domains
- Specific SBC selection for call routing
- Configuration of native NAT traversal w/ optionally redundant media anchoring if necessary
- Media anchoring supports voice and video for any codec

3.18 Directory, Softkeys, Speed Dial

- Automated generation of directory information per user or per user group
- Support for complete contact information and user profile, including avatar
- Crreation and Management of many different directories (per user, per user group, per location, etc.)
- Upload of contacts from GMail and Outlook
- User management of directory information
- Automated provisioning of directory information into user's phones
- Allows adding contacts to the directory from a .csv file (Excel)
- User configurable speed dial (internal / external numbers, SIP URIs)
- Speed dial generated server side and backed up

- Auto-provisioning of speed dial to phones
- User configuration of Busy Lamp Field (BLF) to monitor presence of other users or phones (e.g. attendant console)

3.19 PSTN Trunking

- Unlimited number of PSTN gateways and trunk lines
- Supports most SIP compliant gateways (e.g. Audiocodes, Mediatrix, Sangoma, Patton, etc.)
- Gateways can be in any location
- Gateway selection per dialing rule
- Source routing of calls so that calls can be routed through a local gateway to save WAN bandwidth
- DID
- Local DID per gateway
- DNIS
- **CLIP Management**
  - User CLIP
  - Gateway default CLIP
  - Prefix stripping / appending
- Per gateway CLIR
- **Automatic Route Selection (ARS)**
  - Implemented with XML-formatted mapping rules.
  - Mapping values re-write SIP URLs to specify the next hop or destination for a SIP message that has been received by the Communications Server component.
  - Direct messages to different SIP/PSTN trunk gateways, either on premise or at a remote premise location, based on any portion of SIP URL or E.164 number.
  - Route messages to commercial SIP/PSTN service providers, which reduces or eliminates the need for on-premise trunk gateways.
- Least-cost routing (LCR)
- Automatic failover if unavailable
- Automatic failover if busy
- Inbound FAX support
- Mixing of PSTN and SIP trunks with least cost routing

3.20 SIP Trunking

- Basic SIP trunking gateway w/ NAT traversal
- Remote worker support w/ near-end and far-end NAT traversal and auto-detection

- **ITSP templates for simplified configuration. Interop (not certified) with the following ITSPs. Many other ITSP are compa**
  - BT (UK)
  - AT&T
  - Bandwidth.com
  - CBeyond
  - Bandtel
  - CallWithUs
  - Eutelia (Italy)
  - LES.NET
  - SIPcall (Switzerland)
  - Vitality
  - VOIPUser (UK)
  - VOIP.MS
  - Appia
- SIP interop with Nortel CS1000 R6
- SIP call origination & termination
- Branch office routing
- Proxy to proxy interconnect using ACLs
- Least-cost-routing (LCR)
- Mixing of PSTN trunks with SIP trunks
- TLS support for secure signaling
- Route header for flexible call routing through an SBC
- Flexible rules for SBC selection (route selection)
- Support for Skype for Business SIP trunking

3.21 Integration with Microsoft Active Directory and Exchange

- **Synchronization with Microsoft Active Directory**
  - Using LDAP interface
  - On demand or automatically based on a schedule
  - Graphical query design combines ease of use with flexibility
  - Allows preview of records to be imported
- **Dialplan integration with Microsoft Exchange voicemail server**
  - Allows mixed environment with groups of users on Exchange or the sipXcom VM server
  - Permission based selection of VM server per user or user group
  - Automatic dialplan routing to Exchange VM

- Enables sll speech based Exchange capabilities

3.22 Supported Softclients

3.22.1 Combined SIP and XMPP clients

Jitsi and Counterpath Bria clients can be used with the provisioning server for automated mass deployment of SIP and XMPP account setup.

- Counterpath Bria professional
- Jitsi

3.22.2 XMPP (IM only) clients

- Pidgin
- Trillian
- Spark

3.22.3 Analog Gateways (FXO and FXS)

- Supports any SIP compliant FXO or FXS gateway
- Analog fax machines FXS gateways
- Analog cordless phone support with FXS gateways
- Plug & play management of many analog gateway models

3.23 Performance

- Unlimited number of simultaneous calls (voice, HD voice, video) - only depends on LAN/WAN bandwidth
- 54,000 BHCC, 120,000 BHCC two-way redundant (depends on server HW)
- Up to three-way redundant configuration using cluster mgmt Web GUI
- Up to 10,000 users per dual-server HA system
- Tested up to 10,000 IM users
- 450 simultaneous calls through the SIP trunking gateway require < 20% CPU on dual core system
- Up to 500 simultaneous conferencing ports per server
- Up to 300 media server ports for unified messaging (supports 15,000 users)
- Automatic time distribution of re-registration and subscription events

3.24 High Availability

- Optionally fully redundant call control system
- Geo-redundant SIP session manager
- Based in DNS SRV (no cluster required)
- Load balance under normal operating conditions
- Geographic dispersion of redundant systems
- Real-time synchronization of state information
- Automatic recovery after server failure
- Reports on load distribution

3.25 Call Detail Records collection and reporting

- Call State Events (CSE) collected for all signaling activity
- Processing of CSEs into CDRs
- All data stored in a database at all times
- Flexible report generation using Jasper Reports, built-in
- Supports redundant call control
- Determines and records call type information
- Internal / external calls
- Calls to specific sipXcom services
- Collates call legs
- Historic Call Detail Record reporting in real-time
- Additional reports using call type info
- Monitoring of currently active (on-going) calls
- Export of active and historic CDRs to Excel (.csv file)
- Direct database access for reporting application (e.g. Crystal Reports, Jasper Reports)
- SOAP Web Services access to CDR data
- Individual call history per user in the user portal

3.26 Security

- All outbound calls authenticated
- Secure user password management
- DoS attack prevention
- HTTPS secure Web access
- TLS based signaling for SIP trunks

- HTTPS secures non-SIP communication between sipX components.
- HTTPS secures communications between sipX components and admin and user consoles.
- Secure channel for retrieving messages from voicemail repository.
- HTTP digest authentication for SIP signaling, as specified in RFC 2617, is used for authentication challenges between SIP endpoints and sipX components.
- HTTP digest implementation supports MD5.

3.27 System Administration Features

- Browser based configuration and management
- Several admin accounts
- Notification when new version or patches are available
- GUI based software upgrade
- GUI based certificate management
- LDAP integration
- Integration with Microsoft Exchange 2007 for voicemail and Active Directory
- SOAP Web Services interface
- CSV import and export of user and device data
- Administration of Instant Messaging (IM) and Presence settings
- Integrated backup & restore
- Scheduled backups
- **Diagnostics**
  - Display active registrations
  - Display job status
  - Status of services
  - Snapshot logs for debugging
  - Logging (customizable log levels, message log per service)
  - Display active calls
- Domain Aliasing
- Support for DNS SRV
- Support for DNS NAPTR based call routing
- **Automatic restart after power failure**
  - Single sipXcom application can start all other application processes associated with starting up sipXcom, including dependent processes that must be started in particular order.
  - Configured from browser interface
- Login history report (successful and unsuccessful)
- Automated testing of network services (DHCP, DNS, NTP, TFTP, FTP, HTTP) for proper configuration

3.28 Plug & Play Device Management

- Auto-discovery of phones & gateways on the LAN
- Auto-registration of Polycom phones simplifies installation
- Plug & play management of phones
- Plug & play management of PSTN gateways
- Auto-generation of phone / gateway config profile
- Auto-pickup of profile by phone / gateway
- Centralized management of all the parameters
- Centralized backup and restore of all the configs
- Auto-generation of lines by assigning users to devices
- Device group management & properties
- Firmware upgrade management

3.29 Unified Messaging (Voicemail)

- Integrated unified messaging system
- Localized per user by installing language packs
- Number of voicemail boxes only limited by disk size (tested up to 10,000)
- Performance tested up to 300 simultaneous calls (ports) on dual core server
- IMAP back-end connection
- Acts as an IMAP client into MSFT Exchange and other compatible email systems
- User manageable credentials for IMAP federation
- Properly controls MWI on the phone when message is “read” using the email client
- Browser based user portal for unified messaging management
- RSS feed for new messages
- Message Waiting Indication (MWI)
- User configurable distribution lists
- Group and system distribution lists
- Unified Messaging:
- Email notification of new voicemail messages
- Forwarding of message as .wav file
- Supports several parallel notifications
- IMAP client into Exchange
- Per user selectable templates for email format used when forwarding voicemail
- Manage folders: Folders for message organization
- Manage greetings: Multiple customizable greetings

- Operator escape from anywhere
- Remote voicemail access using a phone
- SOA Web Services (REST) access to messages and greetings
- Unlimited number of inboxes
- Auto-removal of deleted messages

3.30 Personal Auto Attendant

- User configurable personal auto-attendant for every user on the system
- Up to 10 individual forwarding choices (keys 0 through 9)
- User can record greeting that corresponds with key configuration
- Individual zero-out to a personal assistant or receptionist
- Individual selection of language based on installed language packs
- Personal greeting

3.31 Auto Attendant Features

- Unlimited number of auto-attendants
- Dial by extension and name
- Night and holiday service
- Special auto-attendant
- Transfer on invalid response
- Nested auto-attendants (multi-level)
- **Fully customizable actions:**
  - Operator
  - Dial by Name
  - Repeat Prompt
  - Voicemail login
  - Disconnect
  - Auto-Attendant
  - Goto Extension
  - Deposit Voicemail
- Uploadable custom prompts
- Configurable DTMF handling

3.32 Presence Server Features

- Compatible with Broadsoft or IETF implementations
- Centralized management of resource lists for dialog events
- Busy Lamp Field (BLF) feature based on presence
- Used to support shared lines (BLA)
- Presence federated with IM presence to show "on the phone" status
- Support for 3rd party Attendant Consoles (such as Voice Operator Panel)

3.33 Hunt Groups

- Unlimited number of hunt groups
- Serial and parallel forking (rings sequentially or at the same time)
- Configurable ring time per attempt
- Enable / disable user call forwarding rules while hunting
- Flexible configuration of destination if no answer

3.34 Call Park Server

- Unlimited number of park orbits
- Visual indication on the phone of the state of the park orbit using the presence server (BLF)
- Music on park
- Uploadable music file
- Configurable call retrieve code
- Configurable call retrieve timeout
- Automatic park timeout with configurable time
- Configurable park escape key
- Allow multiple calls on one orbit

3.35 Group Paging Server

- Integrated group paging server
- Unlimited number of paging groups
- Supports regular SIP phones using auto-answer
- Supports dedicated in-ceiling devices (SIP)
- Configurable paging prefix

3.36 Conferencing Server

- Voice conferencing server that can run on the same sipXcom server or on dedicated hardware
- Support for voice conferencing
- Each user on the sipXcom system can have a personal conference bridge
- Recording of conference calls
- Dynamic conference controls from the user's Web portal (user portal)
- Dynamic conference control using IM
- Participant entry / exit messages
- Roll call
- Mute, isolate, disconnect, invite
- Association of personal conference bridge with personal group chat room
- Automatic migration of group chat to a voice conference using the @conf directive
- Support for HD Audio and transcoding if necessary
- Support for up to 500 ports of conferencing, dependent on hardware
- Configurable DTMF keys for conference controls using the TUI
- A sipXcom IP PBX system can have more than one conference server if more capacity is needed
- All conferencing servers and services centrally managed and configured
- Conferencing based on FreeSWITCH

3.37 Call Queueing (ACD)

- ACD server collocated or on a different server hardware
- Several (unlimited) queues per server
- Several lines per queue
- Support trunk lines (many calls per line) or single call per line
- Dedicated overflow queues or overflow to hunt group or voicemail
- Configurable call routing scheme per queue:
- Ring all
- Circular
- Linear
- Longest idle
- Agent presence monitor using presence server
- Separate welcome and queue audio
- Call termination tone or audio
- Configurable answer mode
- Agent wrap-up time

- Auto sign-out of agents if calls are not answered
- Configurable maximum ring delay
- Configurable maximum queue length
- Configurable maximum wait time until overflow condition
- Unlimited number of agents per queue

3.38 sipXcom Managed Devices

Almost any SIP compatible phone works with sipXcom if configured manually (i.e. by logging into the phone's Web interface to configure it one phone at a time). The following devices are plug & play managed automatically and centrally by sipXcom:

- Polycom SoundPoint all models (IP 301, 320, 330, 430, 450, 501, 550, 560, 601, 650, 670)
- Polycom SoundStation IP 4000, 6000, 7000 SIP
- Polycom VVX phones (300/310, 400/410, 500, 600, 1500)
- Audiocodes gateways MP112, MP114, MP118, MP124 FXS
- Audiocodes gateways FXO and PRI/BRI
- Counterpath Bria Professional

3.39 sipXcom Managed Devices (Community supported)

Community supported means that the phone plugin for plug & play management is provided as is. These phone plugins are provided and maintained by community members. Some system functionality might not be implemented or supported.

- Aastra 53i, 55i, 57i
- Snom 300, 320, 360, 370 up to firmware 7.x
- Grandstream BudgeTone, HandyTone
- Grandstream GXP2000, GXP1200, GXP2010, GXP2020
- Grandstream GXV3000 Video Phone
- Hitachi IP3000 and IP5000 WiFi phones
- Cisco ATA 186/188
- Cisco 7960, 7940, 7912, 7905
- Cisco 7911, 7941, 7945, 7961, 7965, 7970, 7975
- ClearOne MaxIP Conference Phone
- LG-Nortel LG 6804, 6812, 6830
- Nortel video phone 1535
- Linksys ATA 2102, ATA 3102
- Linksys SPA8000
- Linksys SPA901, SPA921, SPA922, SPA941, SPA942, SPA962

- Nortel 1120 / 1140 SIP
- G-Tec AQ10x, HL20x, VT20x

3.40 Centrally Managed sipXcom Distributed System (cluster)

- Automated installation and configuration of a distributed system with specific server roles
- Automated and central configuration of a high-availability redundant sipXcom system
- Allows for dedicated server hardware for conferencing, voicemail, ACD Call Center, and Call Control
- All configuration for remote servers is centrally generated and distributed securely

3.41 SIP Implementation

This is probably quite an incomplete list. In any case, sipXcom IP PBX is fully SIP standards compliant.

- RFC 3261 Session Initiation Protocol using both UDP and TCP transports
- **Advanced call control using RFCs**
  - RFC 3515 Refer Method
  - RFC 3891 Referred-By header
  - RFC 3892 Replaces header
- **Provide for consultative and blind transfer and third party call controls**
  - Blind transfer (Unannounced) to a different phone without speaking to the other phone prior to transfer.
  - Consultative transfer (announced) to a different phone without speaking to the other phone prior to transfer.
- RFC 3263 Locating SIP Servers - use of DNS SRV records for call routing control and server redundancy.
- RFC 3581 Symmetric Response Routing (rport)
- RFC 3265 SIP Event Notification - for phone configuration and
- RFC 3842 Voice mail message waiting indication (MWI)
- RFC 3262 Reliable Provisional Responses
- RFC 2833 Out-of-band DTMF tones
- RFC 3264 Offer/Answer model for SDP for Codec Negotiation
- RFC 2617 HTTP Authentication: Basic and Digest Access Authentication
- RFC 3327 Path header
- RFC 3325 P-Asserted identity
- RFC 4235 An INVITE-Initiated Dialog Event Package for the Session Initiation Protocol (SIP)
- RFC 4662 A Session Initiation Protocol (SIP) Event Notification Extension for Resource Lists
- RFC 2327 SDP: Session Description Protocol
- RFC 3326 The Reason Header Field for the Session Initiation Protocol (SIP)

- Early media (SDP in 180/183)
- Delayed SDP (SDP in ACK)
- Re-INVITE: Codec change, hold, off-hold
- Route/Record-Route header fields
- Configurable RTP/RTCP ports
- Configurable SIP ports
- BLA support
- RFC 3680: A Session Initiation Protocol (SIP) Event Package for Registrations
- RFC 3265: Session Initiation Protocol (SIP)-Specific Event Notification
- draft-ietf-sipping-dialog-package-06
- draft-anil-sipping-bla-02
- XMPP Compliance
- RFC 3920: XMPP Core
- RFC 3921: XMPP IM
- XEP-0030: Service Discovery
- XEP-0077: In-Band Registration
- XEP-0078: Non-SASL Authentication
- XEP-0086: Error Condition Mappings
- XEP-0073: Basic IM Protocol Suite
- XEP-0004: Data Forms
- XEP-0045: Multi-User Chat
- XEP-0047: In-Band Bytestreams
- XEP-0065: SOCKS5 Bytestreams
- XEP-0071: XHTML-IM
- XEP-0096: File Transfer
- XEP-0115: Entity Capabilities
- XEP-0004: Data Forms
- XEP-0012: Last Activity
- XEP-0013: Flexible Offline Message Retrieval
- XEP-0030: Service Discovery
- XEP-0033: Extended Stanza Addressing
- XEP-0045: Multi-User Chat
- XEP-0049: Private XML Storage
- XEP-0050: Ad-Hoc Commands
- XEP-0054: vcard-temp
- XEP-0055: Jabber Search

- XEP-0059: Result Set Management
- XEP-0060: Publish-Subscribe
- XEP-0065: SOCKS5 Bytestreams
- XEP-0077: In-Band Registration
- XEP-0078: Non-SASL Authentication
- XEP-0082: Jabber Date and Time Profiles
- XEP-0086: Error Condition Mappings
- XEP-0090: Entity Time
- XEP-0091: Delayed Delivery
- XEP-0092: Software Version
- XEP-0096: File Transfer
- XEP-0106: JID Escaping
- XEP-0114: Jabber Component Protocol
- XEP-0115: Entity Capabilities
- XEP-0124: HTTP Binding
- XEP-0128: Service Discovery Extensions
- XEP-0138: Stream Compression
- XEP-0163: Personal Eventing via Pubsub
- XEP-0175: Best Practices for Use of SASL ANONYMOUS

CHAPTER

FOUR

PLANNING

4.1 Gathering Information

You will need to gather a lot of information about the existing infrastructure, physical locations, configuration of equipment, etc. Know who is responsible at the site for any equipment or service configuration changes that may be necessary. Below are a few suggestions to begin with.

4.1.1 Network Diagram

Find or create a detailed network diagram that includes the make/model of all switches, routers, firewalls, DNS and DHCP servers. Review the current state of exiting wiring, patch panels, distribution closets, etc.

4.1.2 IP Addressing

Understand and document the existing network IP addressing scheme, VLANs used, routing, etc.

4.1.3 Internal/External DNS

The SIP domain name is important to establish up front. The sipXcom DNS service will consider itself authoritative for whatever the domain name is, so be mindful of any conflicts with existing DNS zones. For example if the existing network domain name is the same as your SIP domain name you may need to create MX or important A records manually on the sipXcom side.

4.1.4 SSL Certificates

Is there an existing SSL certificate? Who is the provider or technical contact for site certificates?

4.1.5 Telephony Provider

How will the site connect to the PSTN? Will it be through an analog gateway or SIP trunk? Gather technical contact information for the telco provider, make/model of the gateway or SBC, firmware versions, etc.

4.1.6 Client Requirements

Will users register from inside the private network, outside the private network, or both? What make and model phones will be used? Are any existing phones running current firmware versions? Are there any intercom, interconnect, or custom requirements at the site?

4.1.7 Security Concerns

If there will be remote workers (users registering across the WAN and outside the servers local network), what security measures are in place to protect the SIP proxies and registrars? Who is the technical contact for IT or telecom security at the site? Will there be signaling or audio encryption requirements (SIPS/SRTP)?

CHAPTER

FIVE

INSTALLATION

Note:

- All servers in the cluster should have a static IP address.
- The server(s) must have only one active NIC or IP interface.
- Only IPv4 is supported. Disabling IPv6 on the NIC during OS install is recommended.
- Review the partition sizes if automatic partitioning is used.

5.1 Recommended Specs

- 2x CPU/vCPU
- 8GB RAM
- 50GB or larger disk

5.2 Operating System

Recent sipXcom RPMs will only install on top of CentOS 7.x with amd64/x86_64 architecture. We recommend using the CentOS minimal ISO.

5.2.1 Disk Partitioning Recommendations

- 1GB ext2 for the /boot partition with the boot flag set
- swap partition equal to the system RAM size
- Allocate the rest of the free space for the root (/) partition as a LVM volume, XFS formatted

Warning: If the disk is larger than 50G and you use automatic partitioning, most of the space will be allocated to /home rather than /.

5.3 Downloading RPMs

Run yum update to update OS packages first. Reboot if you need to after:

```
yum update -y
reboot
```

Install wget:

```
yum install wget -y
```

Add the sipxcom 20.04 repository file beneath /etc/yum.repos.d, then run yum update to update available packages:

```
wget -P /etc/yum.repos.d/ http://download.sipxcom.org/pub/sipXecs/20.04-centos7/
↪sipxecs-20.04.0-centos.repo
yum update
```

Install the sipxcom packages:

```
yum install sipxcom -y
```

CHAPTER

SIX

SETUP SCRIPT

6.1 Preparation

If you haven't done so already, update the OS packages first and reboot after:

```
yum update -y
reboot
```

The script will ask you about the SIP and network domain name to begin with. The server will build a DNS zone and all records required based upon the names inputed.

Warning: Use all lower case as you input hostname, network domain, and SIP domain. DNS records are built based upon your inputs. Any whitespace, extra periods, etc will cause the resulting DNS zone to be invalid. Additionally SIP URIs are case sensitive. For example, sip:MATT@example.org is not the same as sip:matt@example.org or sip:Matt@EXAMPLE.ORG.

If you are using external DNS servers then all records for the zone should exist on the external DNS server.

If you are going to use the internal DNS you should change the server to point to itself for DNS resolution before running the installer script. To do so, use the nmtui utility:

```
nmtui
```

Select the interface configuration and set the Primary DNS as the local server IP. Save and quit, then restart the network service:

```
service network restart
```

6.2 Running the script

To begin the installation run:

```
sipxecs-setup
```

The script will disable SElinux and reboot automatically. Press any key to initiate the reboot:

```
Checking SELinux...
Detected SELinux enforcing, setting SELinux to disabled
```

(continues on next page)

(continued from previous page)

```
A reboot is required to apply SELinux changes. Please login as root and run sipxecs-
↪setup after the reboot to continue setup.
Press any key to reboot the system now.
```

Login as root and run the setup script again:

```
sipxecs-setup
```

The first question is if you need to change the network interface configuration. See the *Preparation* section above regarding the DNS servers. Press y to enter nmtui and make changes, or n to continue on.:

```
SELinux not set to enforcing
Network settings:
IP address    : 192.168.1.31
Would you like to configure your system's  network settings? [ enter 'y' or 'n' ] :
```

The second question is if this is the first server in the cluster. Answer y if it is, n if it is not. Complete the first server before adding secondaries.

```
Is this the first server in your cluster? [ enter 'y' or 'n' ] :
```

The third question is the hostname. Press enter if the existing name looks ok.:

```
Configuring as the first server...
Enter just the host name of this computer?. Example: myhost. [ press enter for
↪'sipxcom1' ] :
```

The fourth question is the network domain name. Press enter if the existing name looks ok.:

```
Enter just the domain name of your network? Example: mydomain.com [ press enter for
↪'home.mattkeys.net' ] :
```

The fifth and sixth question is the SIP domain name and realm. This is the domain the DNS SIP SRV records will be built for.:

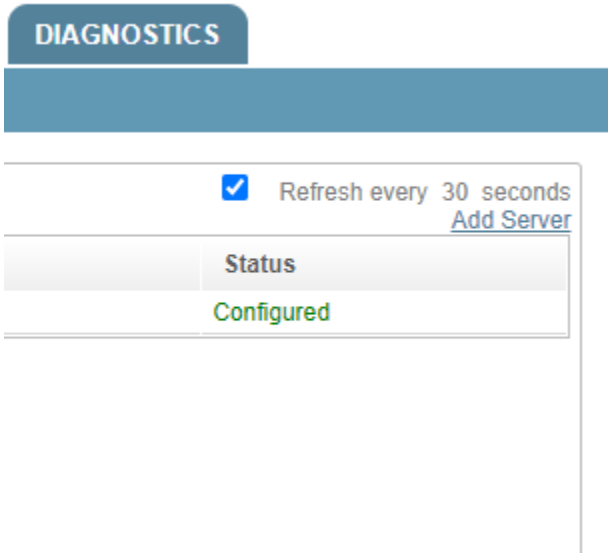
```
Tip: Use 'sipxcom1.home.mattkeys.net' as your SIP domain if you are
setting up for the first time or if you know you are only going to setup one
server. This can make configuration easier.  You can always change the value
later.
Enter SIP domain name [ press enter for 'home.mattkeys.net' ] :
Enter SIP realm [ press enter for 'home.mattkeys.net' ] :
```

The seventh and final question asks if you need to make any changes to your input choices. Press n if everything is correct.:

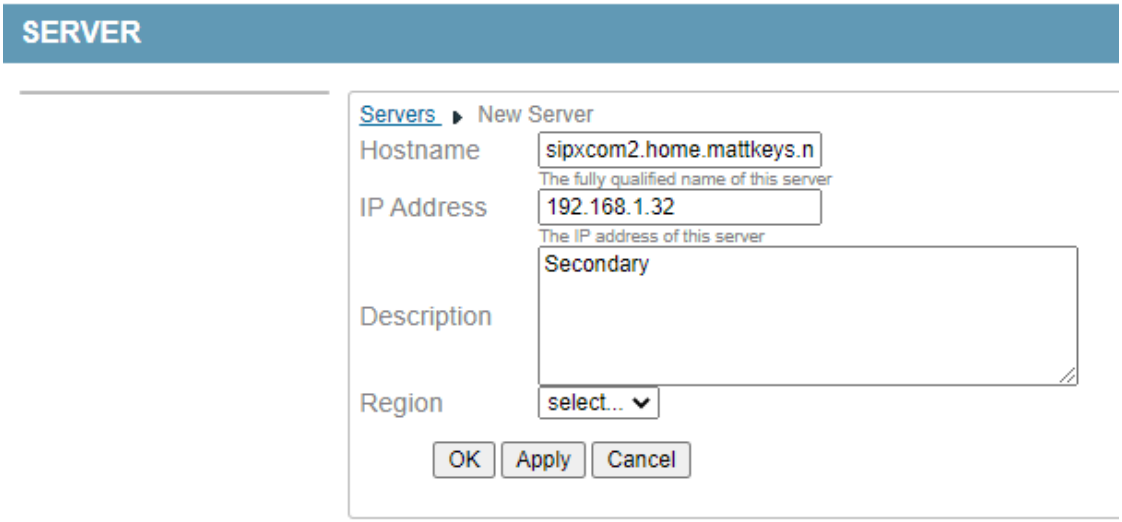
```
Application settings:
Primary server : yes
Host           : sipxcom1
SIP Domain     : home.mattkeys.net
Network Domain : home.mattkeys.net
Would you like to change your application settings? [ enter 'y' or 'n' ] :
```

6.3 Adding secondary servers

Once the setup is complete on the primary server you can add secondary servers. To do so navigate to *Servers*.



Click the 'Add Server' link at the top-right of the page. Enter the FQDN, IP, and description of the server you are adding.



The sipxcom RPMs should be installed on the secondary just as the primary during the *Installation* step.

Running the script on the secondary servers is similar to the primary. The script will first disable SElinux. Press any key to reboot:

```
Checking SELinux...
Detected SELinux enforcing, setting SELinux to disabled
A reboot is required to apply SELinux changes. Please login as root and run sipxecs-
↪setup after the reboot to continue setup.
Press any key to reboot the system now.
```

Run the sipxecs-setup script again after reboot. The second question is if the network settings are correct:

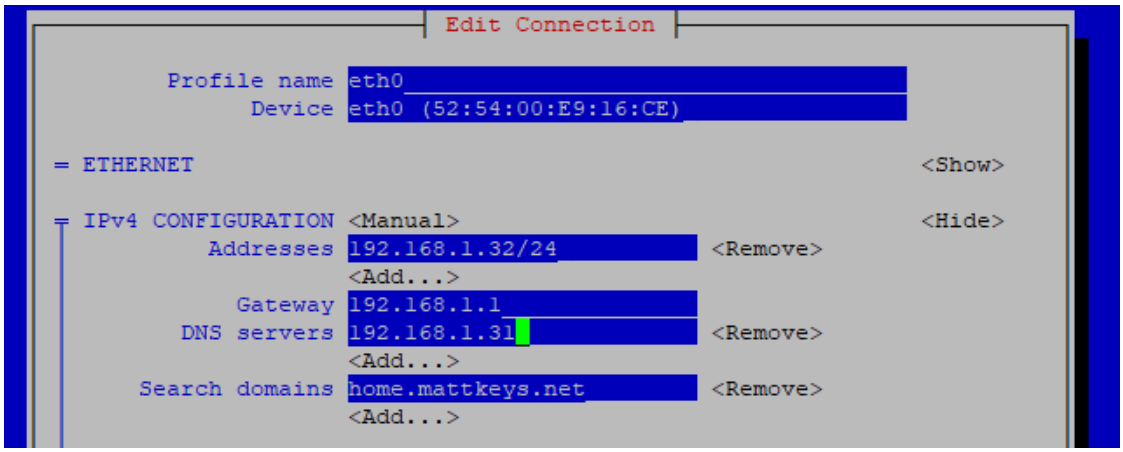
```
SELinux not set to enforcing
Network settings:
```

(continues on next page)

(continued from previous page)

```
IP address    : 192.168.1.32
Would you like to configure your system's  network settings? [ enter 'y' or 'n' ] :
```

Answer Y and point the server to the primary server IP for the primary DNS server entry.

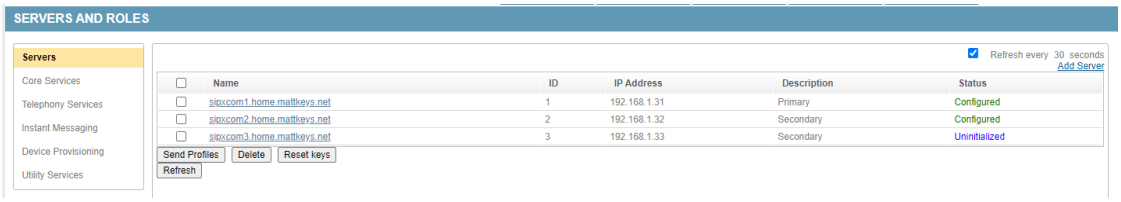


Save, then answer N when prompted again if you want to make network changes.

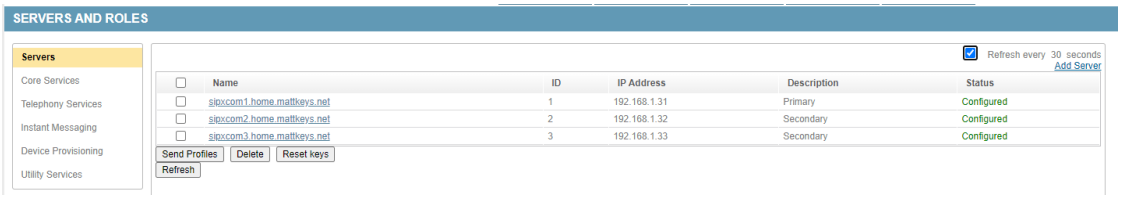
The final set of questions:

```
Is this the first server in your cluster? [ enter 'y' or 'n' ] : n
Configuring as an additional server...
Enter ip address or fully qualified host name of the primary server : 192.168.1.31
Enter the numeric id assigned to this server by the administration server : 2
Application settings:
Primary server : no
Location ID    : 2
Master         : 192.168.1.31
Would you like to change your application settings? [ enter 'y' or 'n' ] : n
```

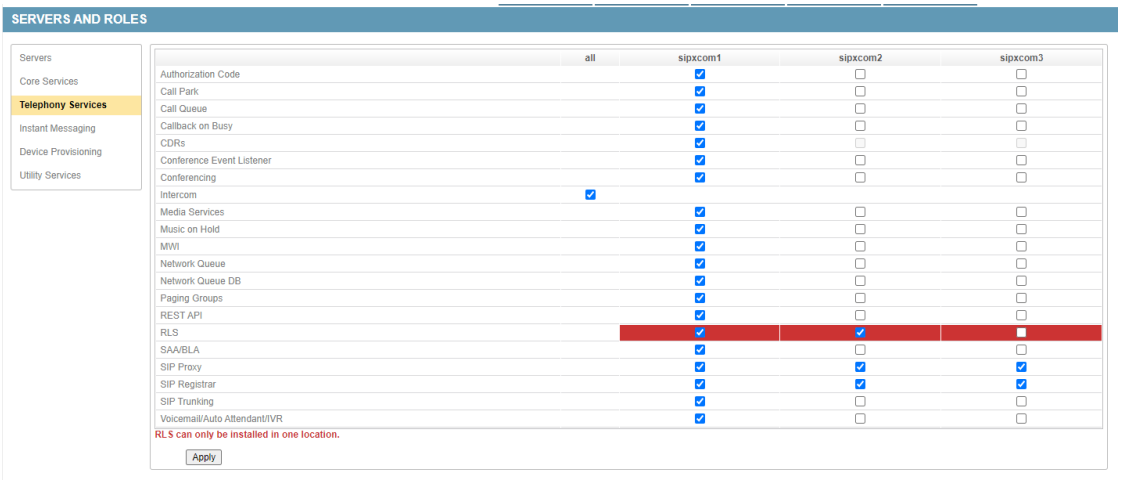
You should see the "Status" field change from "Uninitialized" to "Configured" after this step.



Repeat these steps on additional defined secondaries until all servers are listed as "Configured".



You may now select services to run on the secondaries. Some services can only run on the primary server.


SERVERS AND ROLES
Servers
Core Services
Telephony Services
Instant Messaging
Device Provisioning
Utility Services
all
sipxcom1
sipxcom2
sipxcom3
Authorization Code
Call Park
Call Queue
Callback on Busy
CDRs
Conference Event Listener
Conferencing
Intercom
Media Services
Music on Hold
MWI
Network Queue
Network Queue DB
Paging Groups
REST API
RLS
SAA/BLA
SIP Proxy
SIP Registrar
SIP Trunking
Voicemail/Auto Attendant/IVR
RLS can only be installed in one location.
Apply

CHAPTER

SEVEN

SIPXCOM WEBUI

The sipxcom webui is controlled by the sipxconfig service. Restarting the sipxconfig service will not interrupt calls. If you find the webui unresponsive or there is a "internal error", try on the command line as root:

```
service sipxconfig restart
```

sipxconfig is a java jetty fronted by apache2. Self signed SSL certificates are used by default, so you can expect to see a warning about this in the web browser upon first login.

Note: Firefox or Chrome are the recommended browsers to interact with the webui. Issues have been reported with Internet Explorer and Edge.

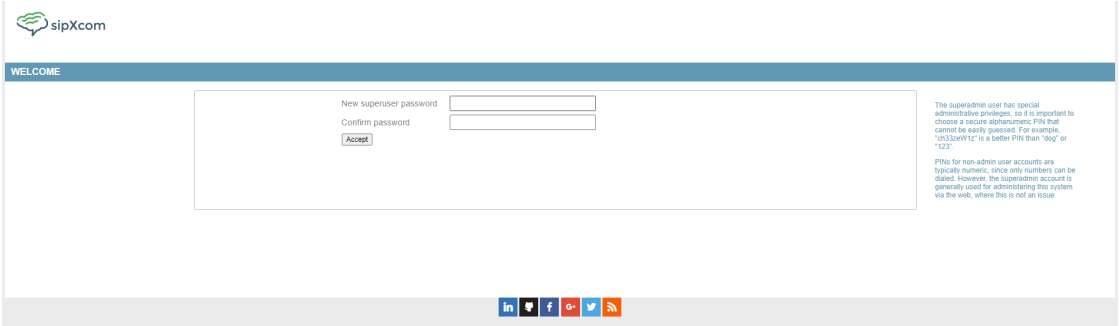
Lets Encrypt Certificates are now available for use in the webui. The domain validation component requires that server have an A record in public DNS, and be accessible on port 80 (http) and 443 (https) from the public internet. If a Lets Encrypt certificate is used the server will automatically renew the SSL certificate – for free! The CA is also trusted by all browsers.

Other options are to purchase a certificate from a trusted CA, use your own CA, or ignore the warning.

7.1 Initial Login

Warning: Beware of browser auto-fill features unintentionally changing important fields like Password, PIN, or SIP password!

The default administrative account is "superadmin". The first login to the webui will prompt to set the superadmin password.



7.1.1 Resetting the superadmin password

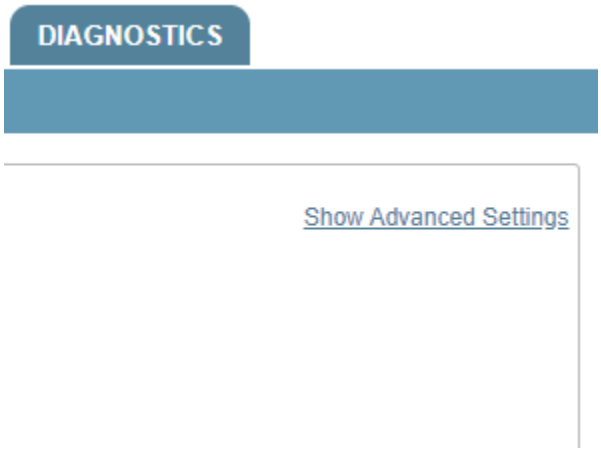
If you need to reset the superadmin password, issue on the command line as root:

```
service sipxconfig reset-superadmin
```

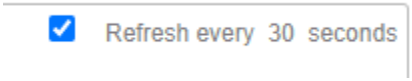
Restart your browser and upon next login use a blank password for the superadmin password.

7.1.2 Show Advanced Settings

Some fields or options are hidden by default. For example, the user SIP password on the user properties page. Use the "Show Advanced Settings" option on the page to see all available settings and options of the page.



Warning: Avoid leaving your browser open for extended periods of time on pages that automatically refresh, such as diagnostics - registrations page and features - conferencing. Another option is to uncheck the option to automatically refresh.



Repeated requests to these pages (and others such as running large CDR reports) may result in a sipxconfig java out of memory error. Try a 'service sipxconfig restart' as the root user on the primary (webui) server if you encounter that. This will restart only the webui service and related components. It is not disruptive to telephony services.

Note: The sipxconfig maximum java heap (**-Xmx**) size is 1GB by default. The value can be changed but will likely be overwritten upon any future sipxconfig rpm upgrades.

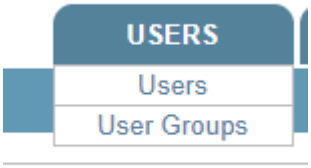
It is configured in the /etc/init.d/sipxconfig file:

```
 Command="$JavaCmd \
-Dprocname=${procNameId} \
-XX:MaxPermSize=128M \
-Xmx1024m \
```

The java manual page (man java) suggests the maximum value would be 2048m:

```
-Xmxn
   Specifies the maximum size, in bytes, of the memory allocation pool. This value␣
↪must a multiple of 1024 greater than 2 MB. Append the letter k or K to indicate␣
↪kilobytes, or m or M to indicate megabytes. The default
   value is chosen at runtime based on system configuration.
   For server deployments, -Xms and -Xmx are often set to the same value. See Garbage␣
↪Collector Ergonomics at http://docs.oracle.com/javase/7/docs/technotes/guides/vm/gc-
↪ergonomics.html
   Examples:
   -Xmx83886080
   -Xmx81920k
   -Xmx80m
   On Solaris 7 and Solaris 8 SPARC platforms, the upper limit for this value is␣
↪approximately 4000 m minus overhead amounts. On Solaris 2.6 and x86 platforms, the␣
↪upper limit is approximately 2000 m minus overhead
   amounts. On Linux platforms, the upper limit is approximately 2000 m minus␣
↪overhead amounts.
```

7.2 Users Tab



The users tab includes the Users, and User Group menu items.

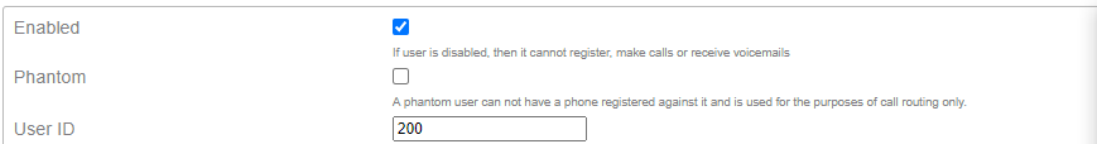
7.2.1 Users

RFC-3261 section 4 provides an excellent overview if you're completely new to SIP.

There are two types of users - regular or phantom. Both will terminate the User ID field, and anything in the Alias field, as the user portion of a SIP URI.

Phantom Users

Phantoms are not allowed to register to the system. They are only used for call routing purposes, or as a general voicemail box target.



Warning: **A phantom user is not allowed to register to the system. Changing a normal user to a phantom user can cause a REGISTER and SUBSCRIBE flood from any phones that were assigned to that user. This also applies to disabling (unchecking the "Enabled" option in the user profile) or deleting the user.** You must first remove line assignments from any phones assigned to that user. Next send profiles to the phones, which should remove any current registrations of the user. Verify beneath users - $user - registrations. Don't forget about any FXS gateways that may need to be manually configured. Only after you have checked these things should you delete, disable, or convert a user to phantom.

User ID

The "User ID" field is the internal extension number. It is typically numerical, as this is what other registered extensions would dial to call this user.

Note: DIDs should not be entered into this field. Use the alias field instead for that.

Alias Field

Non-numerical entries such as "matt" and DIDs are usually added in the aliases field. If you have a large number of DIDs to manage, consider using *DID Pool* feature instead of terminating them here.

Warning: When terminating DIDs it is important to list all possible variations of the DID. For example, in the United States the DID could be presented as 7 digit, 10 digit, 11 digit, or 11 digit prefixed with a +. To terminate all variations of the (fake) DID 4235551212 I'd need to list:

```
5551212 4235551212 14235551212 +14235551212
```

A missing entry here might instead match the outbound dial plan, which would introduce a signaling loop between outbound (egress) and inbound (ingress) traffic.

Password Field

This is the password the user will use to log into the sipxcom webui or using XMPP Instant Messaging chat client such as Pidgin. Jitsi and CounterPath Bria also have XMPP capabilities built-in.

Voicemail PIN

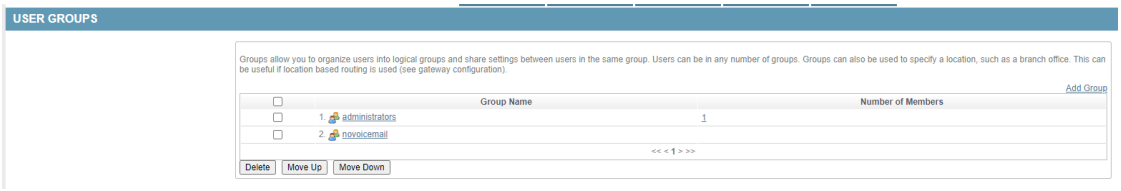
The Voicemail PIN is the numerical passcode the user enters to access voicemail.

SIP Password

The SIP password is the password a phone or soft phone uses register the line. The user ID field is the username the phone or softphone would use.

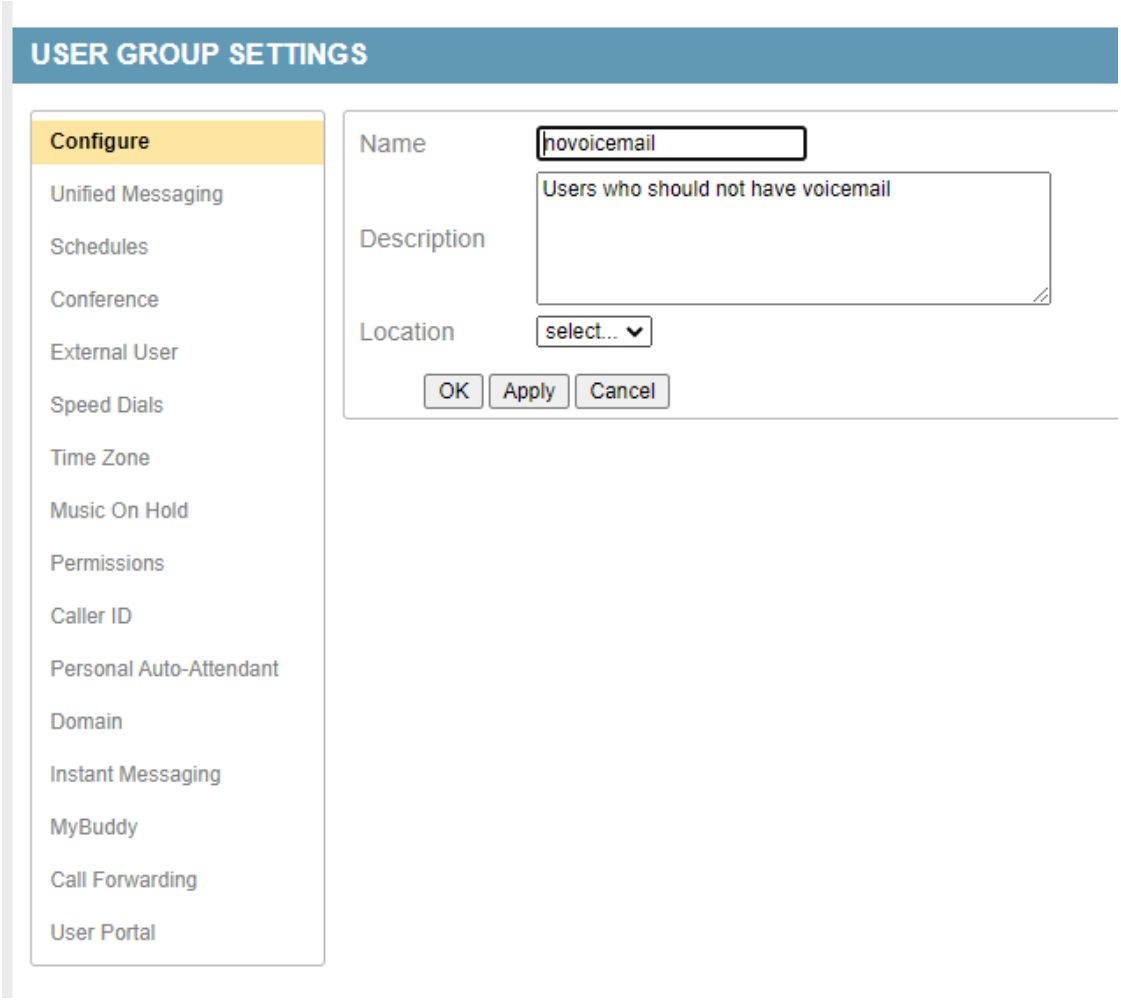
7.2.2 User Groups

User groups are a way to organize users into logical groupings in order to share common settings between the members of that group. There is an administrators group created by default, which the superadmin user is a member of.



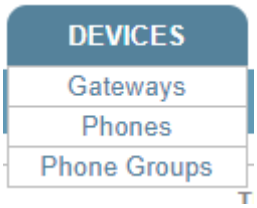
User Group Settings

User groups are a powerful tool for keeping the system easy to manage. The common settings available are Unified Messaging, Schedules, Conference, External User, Speed Dials, Music On Hold, Permissions, Caller ID, Personal Auto Attendant, Instant Messaging, Call Forwarding, and User Portal.



For example, a “novoicemail” group where the administrator has unchecked the “Voicemail” permission in the group properties beneath the Permissions tab.

7.3 Devices Tab



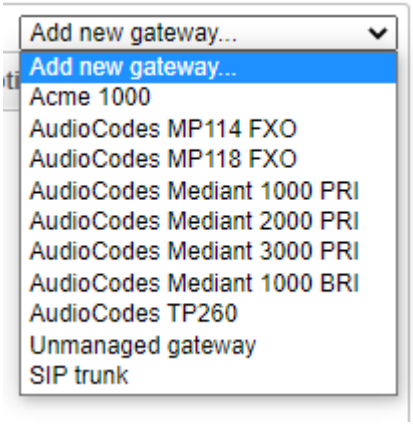
The devices tab includes Gateways, Phones, and Phone Groups menu items.

Sipxcom classifies physical equipment into two areas – managed and unmanaged. Generally speaking managed devices are devices the system can generate configuration files for. Unmanaged devices must be manually configured.

7.3.1 Gateways

Gateways provide connectivity out of the system such as out to the Public Switched Telephone Network (PSTN), or to interconnect with another PBX.

Managed or Unmanaged Gateway?



- A unmanaged gateway is usually something like an AudioCodes gateway, an SBC, or a different PBX. It is a device the server should be aware of to allow traffic, but a device the server cannot directly interact with to change configuration or restart. For example sipxcom can generate the .INI file that you would load on a AudioCodes gateway, but it cannot directly change the configuration of that gateway live or reboot it remotely.
- A managed gateway example would be a SIP Trunk to a ITSP. SIP trunks can be configured with or without authentication.

Note: If SIP trunk is selected the system will use the sipxbridge service to communicate with the ITSP by default. Sipxbridge listens on port 5080, so your ITSP should SIP traffic to port 5080 instead of port 5060.

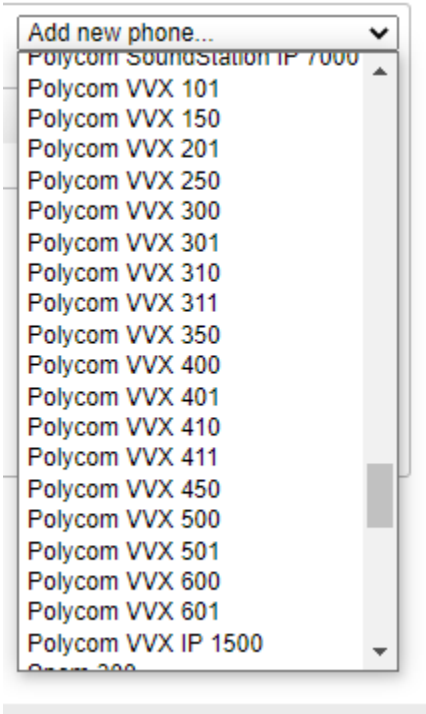
Note: Unmanaged gateways such as AudioCodes gateways should send SIP traffic to the proxy service, which is listening on port 5060. Unmanaged gateways cannot be configured to authenticate.

Warning: You should only allow connections to 5080 or 5060 (and generally any labeled PUBLIC in the firewall rules page) from trusted source IPs. Do not expose them to the entire public internet!

7.3.2 Phones

The phones page is used to define phones to be managed by the server.

Managed or Unmanaged Phone?



- **Unmanaged phones** - It is not required to define the phone in order for that phone to register to the system. For example you only need the user ID and SIP password to register a Jitsi or CounterPath Bria soft phone.
- **Managed Phones** - These are phones sipxcom has a template for and that you want to centrally manage from the sipxcom webui.

When "send profiles" is used this regenerates the *.cfg files for that device, then attempts to send it a reboot command. The configuration files are stored beneath /var/sipxdata/configserver/phone/profile/tftproot/. The reboot command is sent via a "check-sync" event SIP NOTIFY like:

```
2020-10-20T03:23:12.890728Z:337948:OUTGOING:INFO:sipx.home.mattkeys.net:SipClientTcp-
↪5043:7ff0ea672700:sipxproxy:SipUserAgent::sendTcp TCP SIP User Agent sent message:
----Local Host:192.168.1.14---- Port: -1----
----Remote Host:192.168.1.126---- Port: 5060----
NOTIFY sip:200@192.168.1.126;transport=tcp;x-sipX-nonat;sipXecs-CallDest=INT SIP/2.0
Record-Route: <sip:192.168.1.14:5060;lr>
Call-Id: b9d6a0732edf685a09aae8a24d61b3ce@192.168.1.14
Cseq: 100 NOTIFY
From: <sip:~~id~config@sipx.home.mattkeys.net>;tag=206087292
To: <sip:~~in~0004f28034d2@home.mattkeys.net>
```

(continues on next page)

(continued from previous page)

```
Via: SIP/2.0/TCP 192.168.1.14;branch=z9hG4bK-XX-2ff7BwYh_bvFZp7K5e28muVtGg~
↪P6CvTajezxHXKaDzP0uqhA
Via: SIP/2.0/UDP 192.168.1.14:5180;branch=z9hG4bK98271829329565e17ea8a136c99f2dd63434
Max-Forwards: 19
Contact: <sip:192.168.1.14:5180;transport=udp;x-sipX-nonat>
Event: check-sync
Subscription-State: Active
Content-Length: 0
Date: Tue, 20 Oct 2020 03:23:12 GMT
X-Sipx-Spiral: true
```

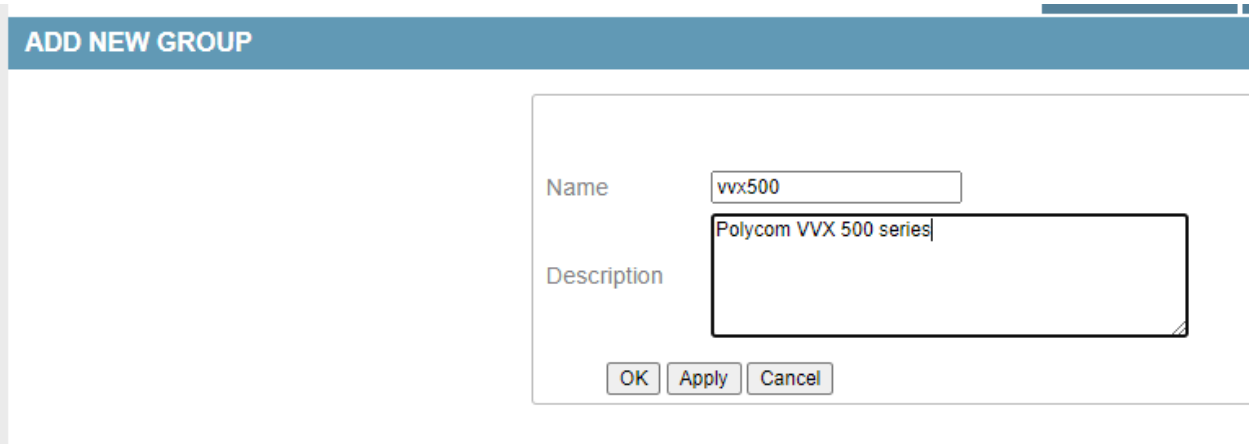
The phone must have a registered line in order to receive that SIP NOTIFY message, and the phone must be configured to use the correct provisioning protocol and server IP in order to download the files from the server.

Note: Don't forget to configure your DHCP scopes with option 160 for http:// and https:// provisioning server addresses. For tftp:// or ftp:// provisioning addresses use option 66. If both are specified a Polycom will prefer option 160 by default. It's also a good idea to specify option 42 for NTP servers. See also the *Date and Time* section to point sipxcom to those same NTP server(s).

7.3.3 Phone Groups

Phone groups are useful to group models together for similar configuration options. Other reasons might include:

- **Incompatible firmware between devices** - Polycom SoundPoint IP series and Polycom VVX series phones run incompatible firmware with each other, so it would be useful to group them separately.
- **Incompatible features or physical capabilities** - There may be enough difference in the capabiilities or features of a series of models to necessitate separate grouping between the series. For example, the difference of color displays on the VVX 500 and the grayscale displays of the VVX 300/301s models.
- **Production vs testbeds** - Another reason would be to test the latest version firmware on a smaller subset of phones – production vs testbed.
- **Physical Location** – For example all VVX 500s at the Central Office.



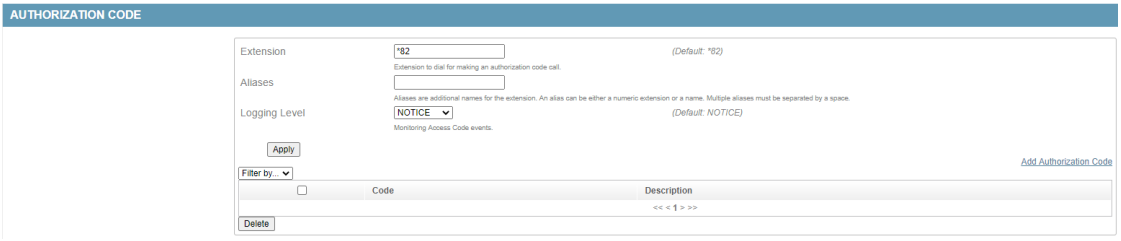
Note: A good minimal practice is to create a group for each model you have in use.

7.4 Features Tab

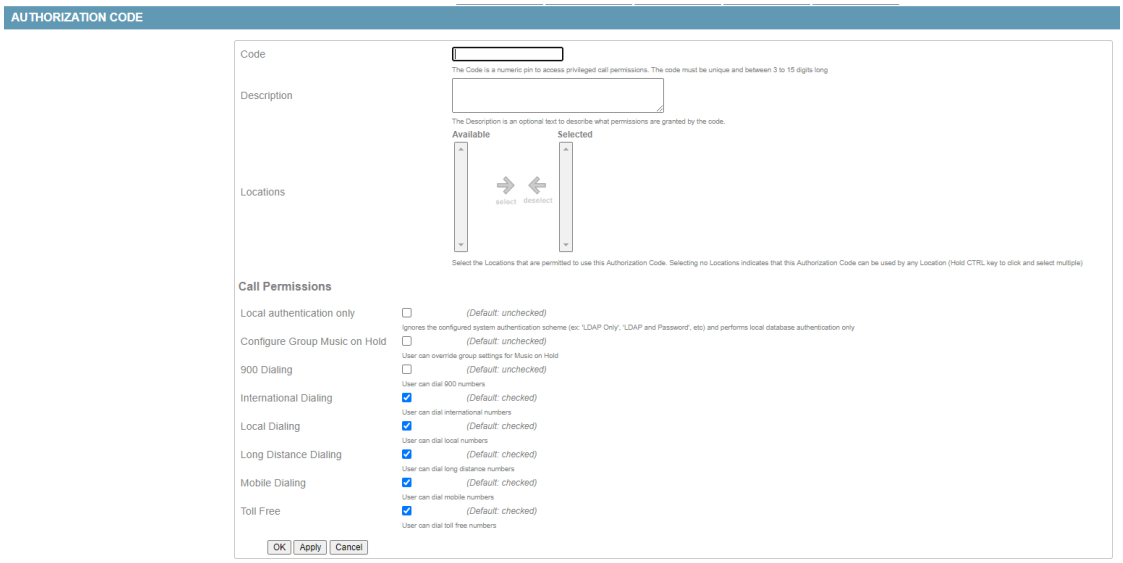
The features tab includes Auth Codes, Auto Attendants, Call Park, Call Queue, Callback on Busy, Conferencing, Hunt Groups, Intercom, Music on Hold, Paging Groups, and Phonebook menu items.

7.4.1 Auth Codes

Authorization codes provide the ability to a user to initiate a call that requires permissions to which it is normally not allowed.



Auth Code options



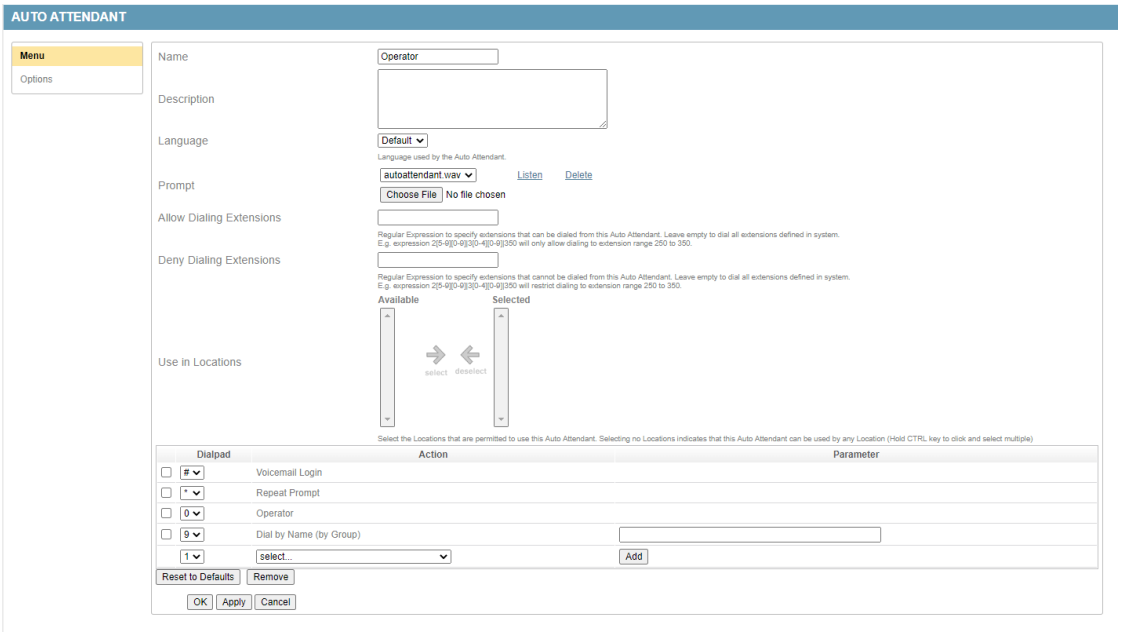
7.4.2 Auto Attendants

Sipxcom includes a multi-level auto attendant service.

Auto attendants provide automatic answering of incoming calls, dial-by-name directory, automated transfer to extension, access to voicemail remotely, and transfer to other auto attendants. For good auto attendant design, try to avoid nesting more than two auto attendants menus deep. This also applies to hunt groups.

Note: Consider using the more powerful *Call Queue* feature instead of nesting AAs.

By default a "Operator" and "After Hours" attendant are created. See the *Dial Plans* section on assigning extension numbers to auto attendants.



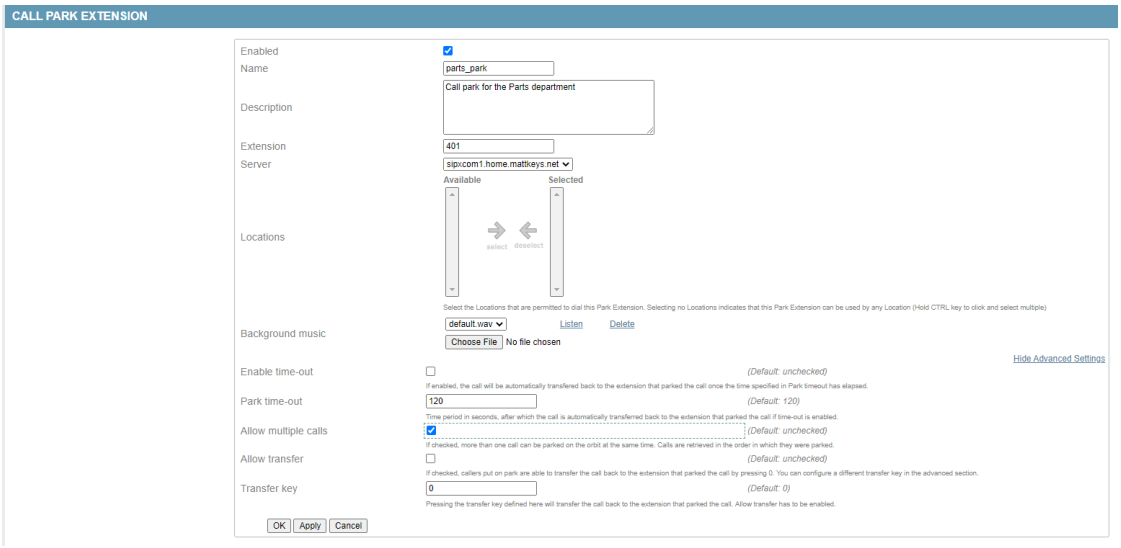
Note: wav files must be in the appropriate format of RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 8000 Hz

Note: There are three major standards for DTMF interpretation: RFC-2833 (inband within RTP), RFC-2976 (out of band SIP INFO), and RFC-3265 (out of band SIP NOTIFY). The AA/IVR only supports RFC-2833 by default.

Note: The auto attendant cannot dial a PSTN number as the target. It does not have dial plan permissions to use gateways. It can call a phantom user however, and that phantom user can call forward to the PSTN.

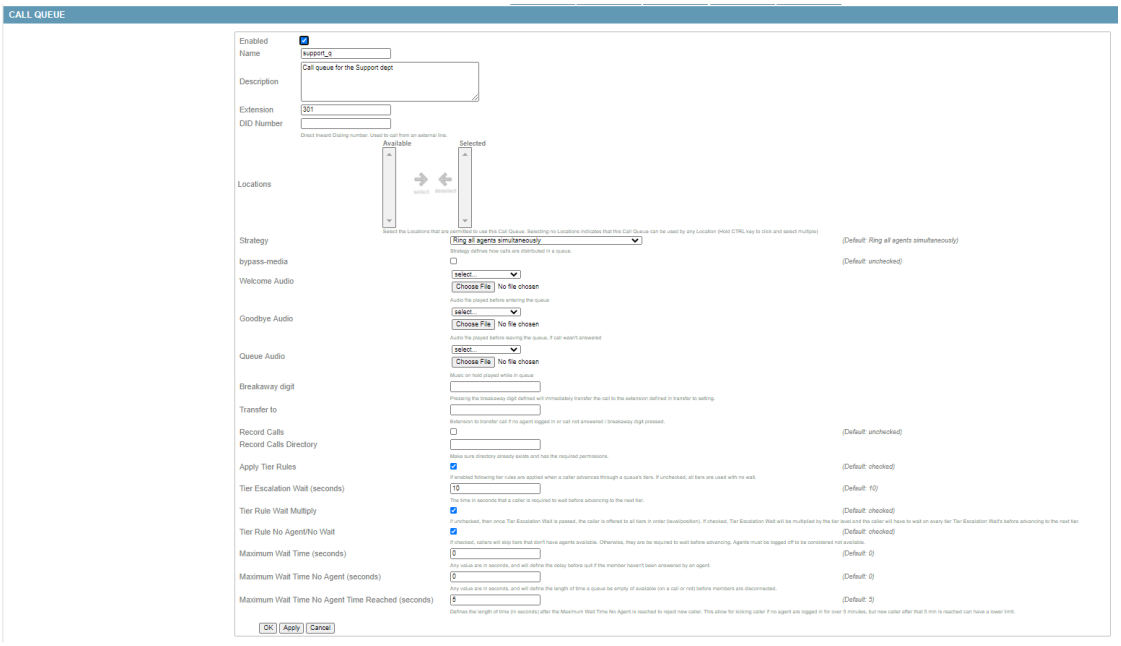
7.4.3 Call Park

The call park feature enables the transfer of calls to an extension. Calls can be retrieved after parking by pressing *4 followed by the extension number.



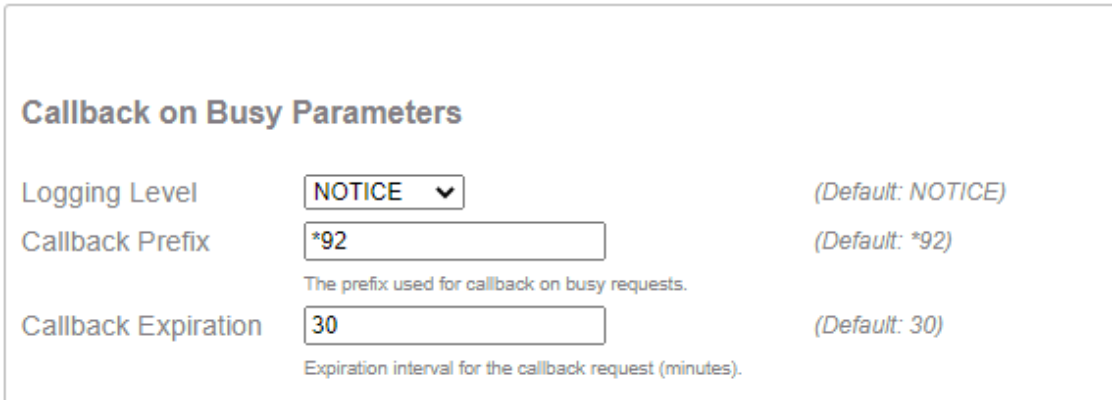
7.4.4 Call Queue

The call queue feature leverages the FreeSWITCH inbound call queueing application, mod_callcenter. This provides lightweight call center functionality by distributing the calls to agents using various scenarios and rules.



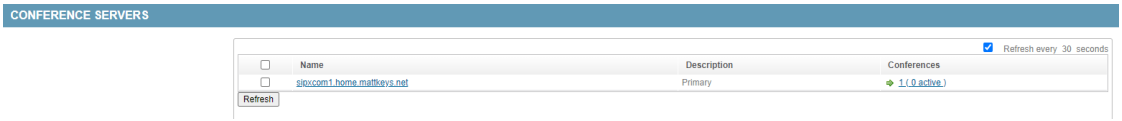
7.4.5 Callback on Busy

The Callback on Busy feature enables a caller to dial the callback prefix and an intended user number. When the intended user is available it will initiate a call between the two users, provided that the callback request has not expired. To request a callback, you need to dial the callback prefix (default *92) with the extension you want a callback from (example *92200).



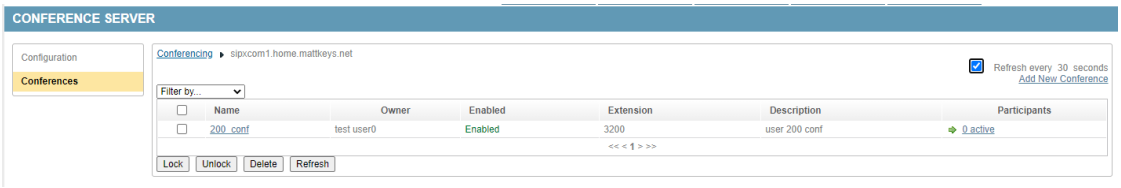
7.4.6 Conferencing

The conferencing feature leverages the FreeSWITCH inbound and outbound conference bridge service, mod_conference. You can create as many conferences as you like, but take care not to overcommit the system resources.



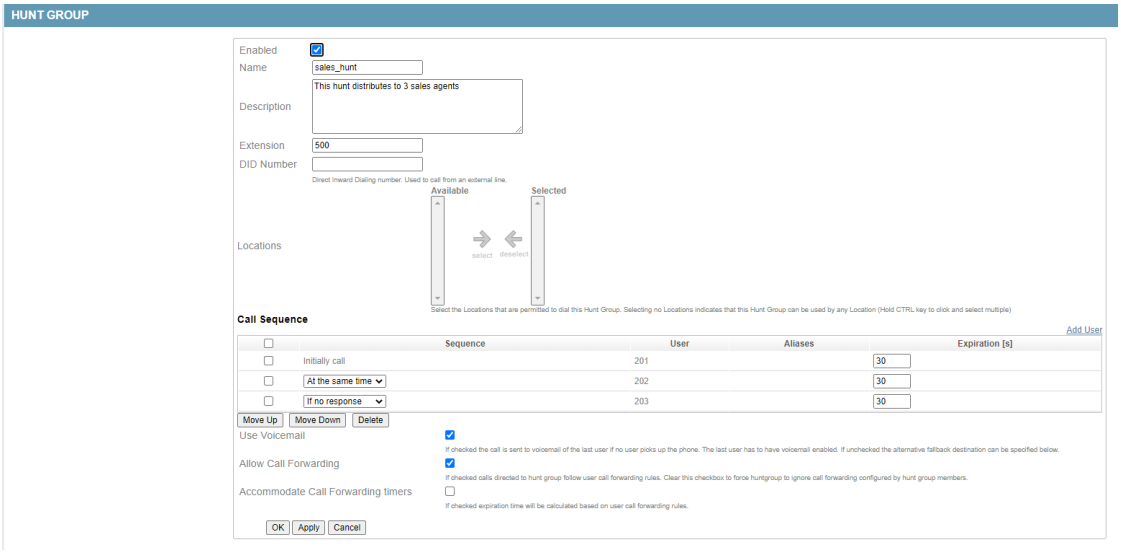
Adding a Conference Room

Conferences can be added from the features - conference - $server page, or beneath the user properties.



7.4.7 Hunt Groups

Hunt groups distribute a given inbound call to members of the group in either a broadcast-like “at the same time”, a sequential “if no response” manner, or combination of both.



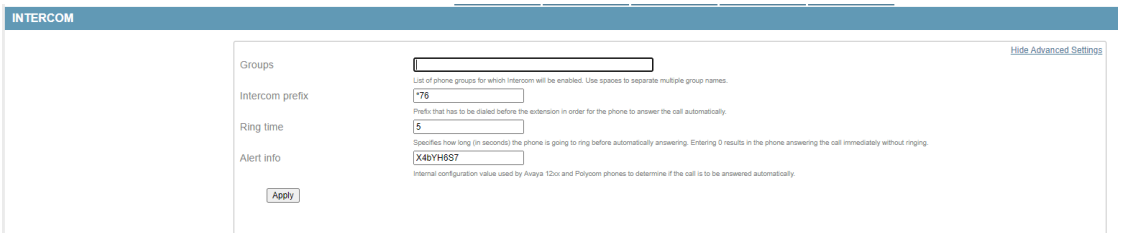
Warning: A hunt group should not exceed more than 5 members.

Signaling delay to endpoints is a common problem with hunt groups. This is due to the nature of the signaling involved. The larger the hunt group becomes, the greater the chance there will be a signaling delay issue. This may manifest as calls the hunt group members are unable to answer (call was CANCELed or answered elsewhere already). Hunt groups can be nested however this practice is also strongly discouraged for the same reasons.

Note: If more than 5 members or nesting is needed consider using the more powerful *Call Queue* feature instead, which utilizes FreeSwitch mod_callcenter instead of burdening the proxy.

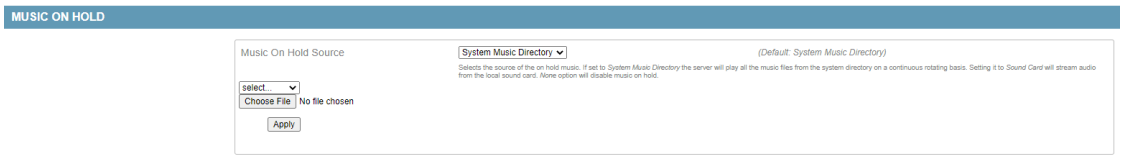
7.4.8 Intercom

Intercom is only supported on devices that can be configured to automatically answer incoming calls. The intercom call can be initiated from any phone. To configure intercom create a new phone group and specify dial prefix.

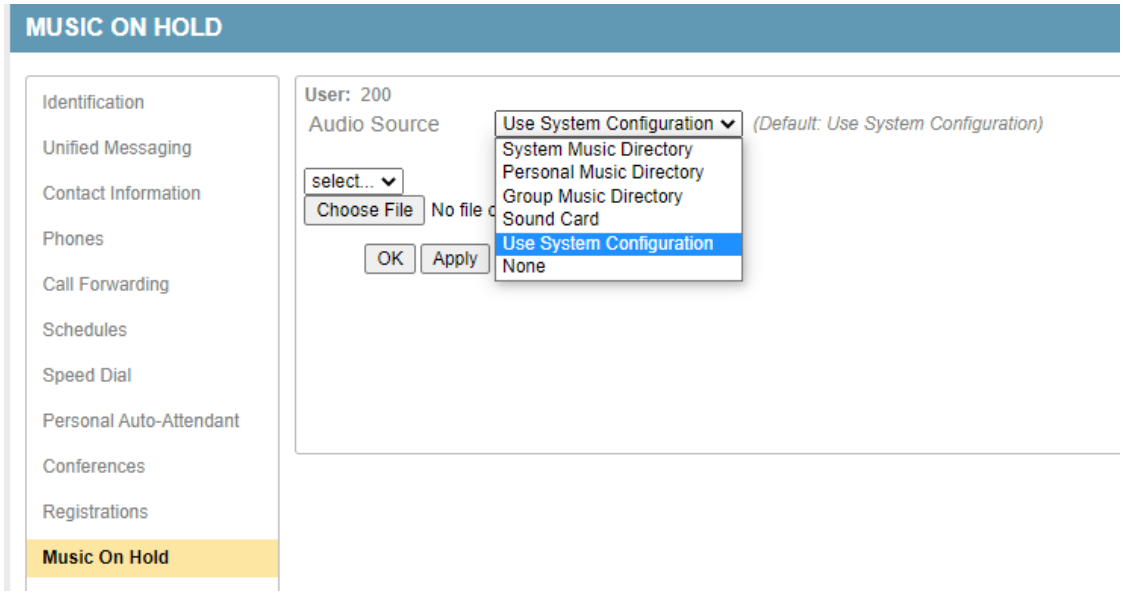


7.4.9 Music on Hold

Music on Hold (MoH) is supported on any phone model that implements IETF RFC-7008 . When incoming call is put on hold the caller will hear music from the source selected on this page. Files can be uploaded to system music directory, and existing files can be deleted. The default MoH files are Creative Commons licensed sound files that included in FreeSWITCH packages.



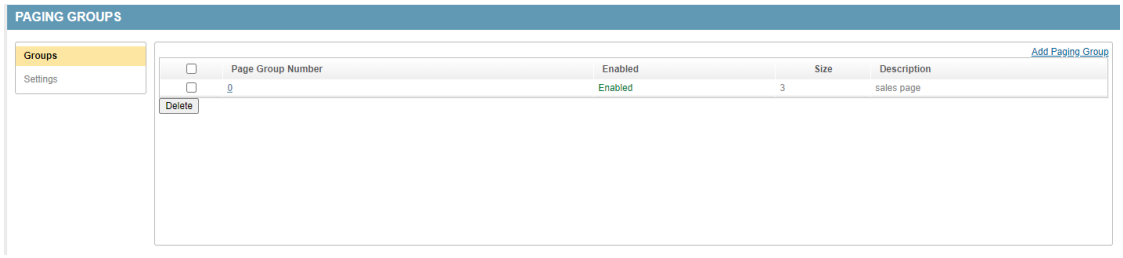
Users can also upload their own Music on Hold.



Note: wav files must be in the appropriate format of RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 8000 Hz

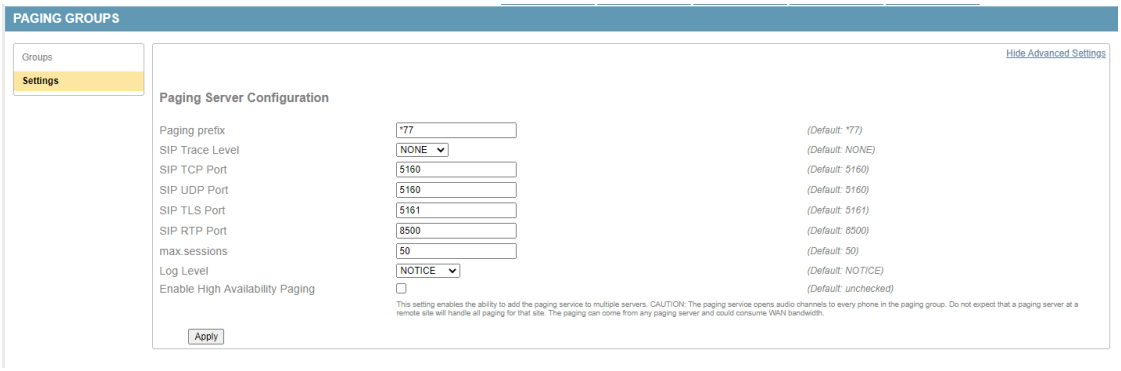
7.4.10 Paging Groups

The paging group contains a list of extensions to call when the paging prefix followed by the paging group number is dialed. You can make changes to the paging server configuration without affecting the running server. The paging server will be automatically restarted when the configuration is changed.

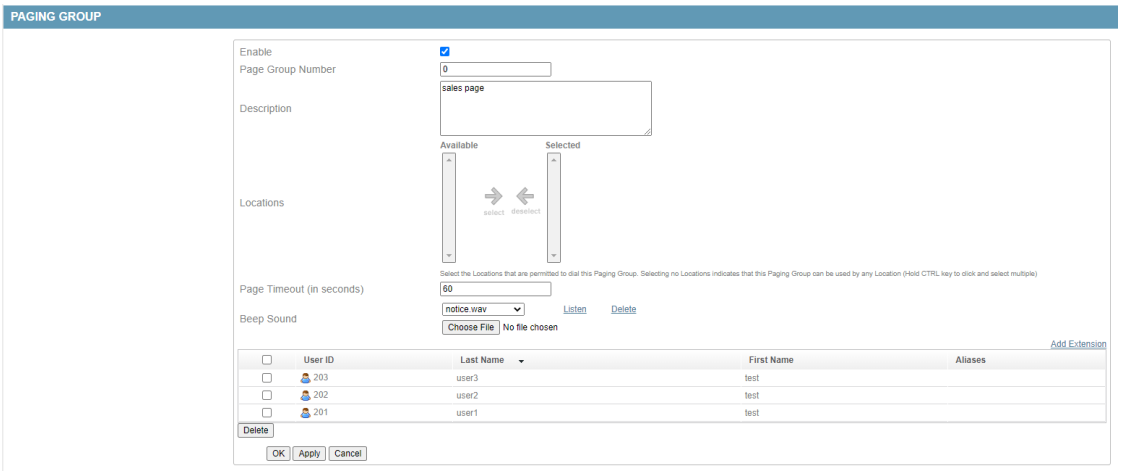


Paging Prefix

The page group number represents the digits that follow the prefix. For example, with the prefix set to *77, when dialing "*770" will invoke the page to page group 0.

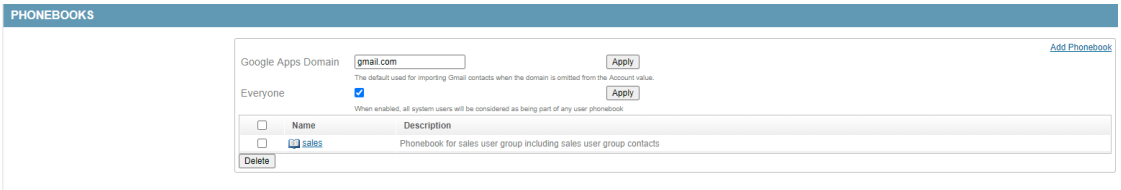


Paging Group options



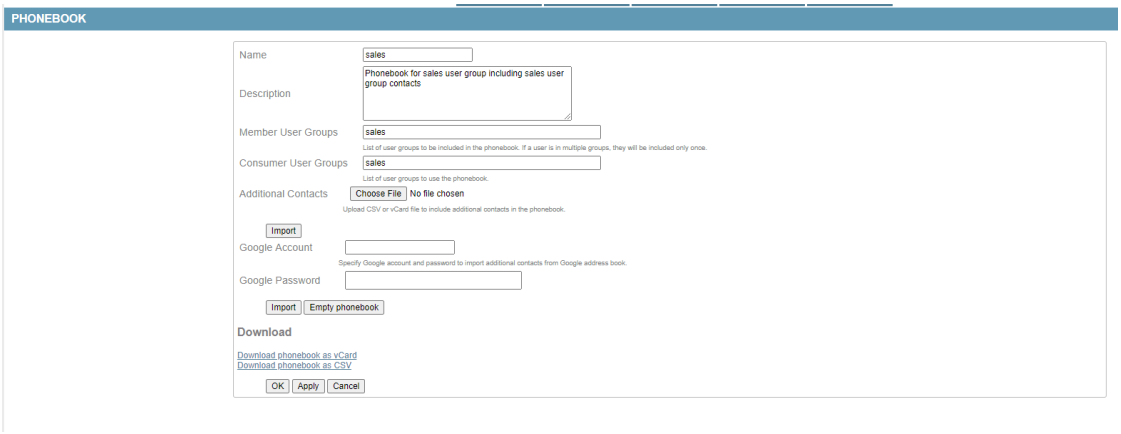
7.4.11 Phonebooks

The phonebook feature allows for central management of phone number directories. This allows users to look up phone extensions and numbers by name and dial directly from the directory. The administrator can create different directories per department, user group, or for individual users. In addition to maintaining a list of internal users, lists of external phone numbers can be imported as well. At present this feature is supported on Polycom and Snom phones.



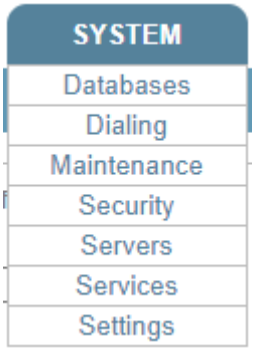
Phonebook Options

The administrator can specify a Google Apps Domain to append to contact information when the domain is ommited from an account name. When the Everyone check box is ticked, the system will automatically add all system users to any user phonebooks.



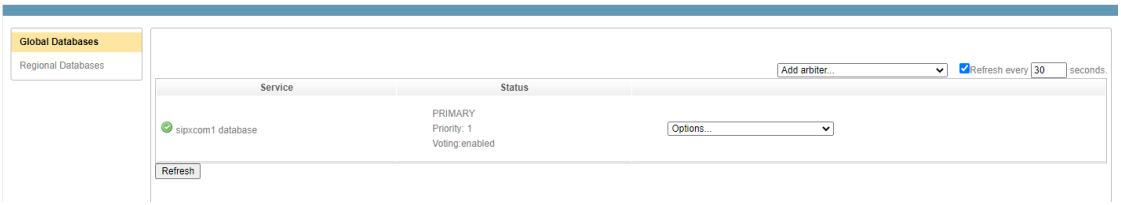
7.5 System Tab

The system tab includes the Databases, Dialing, Maintenance, Security, Servers, Services, and Settings menu options.



7.5.1 Databases

This is the configuration page for the MongoDB global and regional databases. A global database on each server in the cluster is an optional configuration. If running as a cluster MongoDB requires an odd number of servers in the replica set. For example 3 servers, each server running a full global databse. Or you could have 3 servers, two of them running a full global database and one server running arbiter service. See the MongoDB Manual on Replication . We also recommend reviewing the MongoDB Production Notes, particularly the part about NUMA hardware.



Database Settings

This page allows you to tweak settings of the mongo driver.

Increasing the Query Read Timeout and Query Write Timeout may be necessary if you have slow disks or a heavy disk IO frequently. If these are exceeded a warning is broadcast on the command line, for example:

```
Broadcast message from systemd-journald@sipxcom1.home.mattkeys.net (Sun 2020-10-18␣
↪17:29:30 EDT):
sipXproxy[25189]: ALARM_MONGODB_SLOW_READ Last Mongo read took a long time: document:␣
↪node.subscription delay: 1698 milliseconds

Message from syslogd@sipxcom1 at Oct 18 17:29:30 ...
sipXproxy[25189]:ALARM_MONGODB_SLOW_READ Last Mongo read took a long time: document:␣
↪node.subscription delay: 1698 milliseconds
```

7.5.2 Dialing

Dial Plans is the only sub menu item.

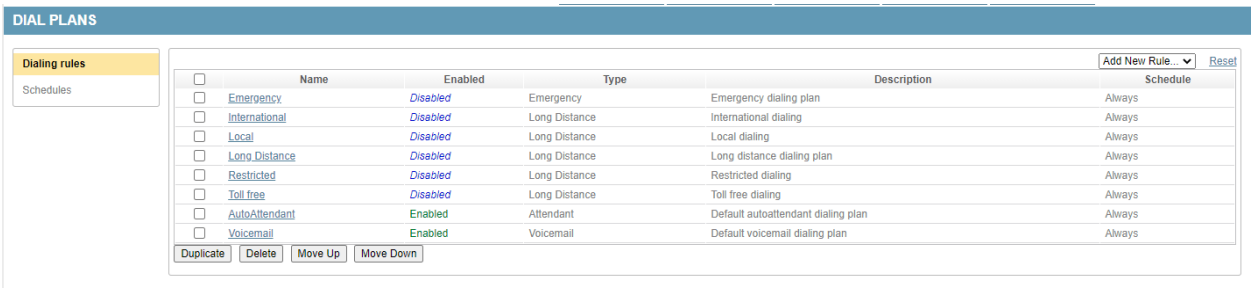
Dial Plans

This page allows you to utilize any gateways defined in the system, or to perform dial string manipulation to and from gateways. By default eight plans are created as templates. These are Emergency, International, Local, Long Distance, Restricted, Toll Free, AutoAttendant, and Voicemail.

Warning: Avoid the use of whitespace or special characters in the dial plan names. Dial plan configuration is written to files such as /etc/sipxpbx/mappingrules.xml on the filesystem. Special characters or whitespace may interfere with sipxconfig reading in, or writing out, data correctly to those files. To avoid this problem use lower case only and replace spaces with underscores in field entries, e.g. local_dialing.

Warning: Avoid creating dial plan entries that have no permission requirements. If you do, you may be creating an opportunity for your dial plan and gateways to be exploited.

Note: Ordering matters. Dial plan entries are read from top to bottom. Rules at the top are processed before the rules at the bottom.



7.5.3 Maintenance

The maintenance tab includes Backup, Import/Export, and Restore menu options.

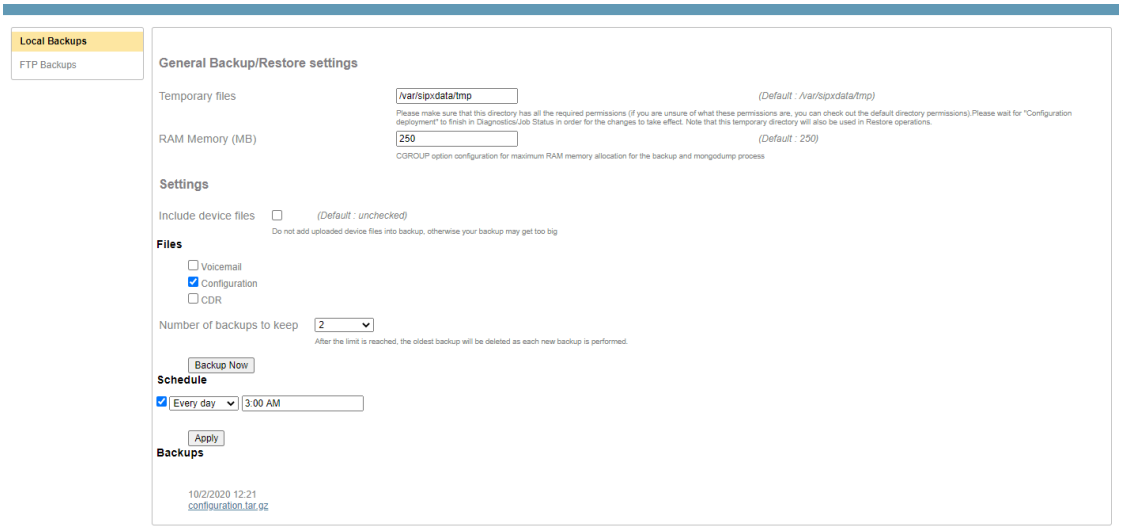
Backup

The Backup page has two tabs - Local or FTP backups.

Note: The backup log is /var/log/sipxpbx/backup.log. Check it for clues if you're having problems. Filenames (uploaded prompts, music on hold, etc) with whitespace or special characters can cause problems with archive creation.

Local

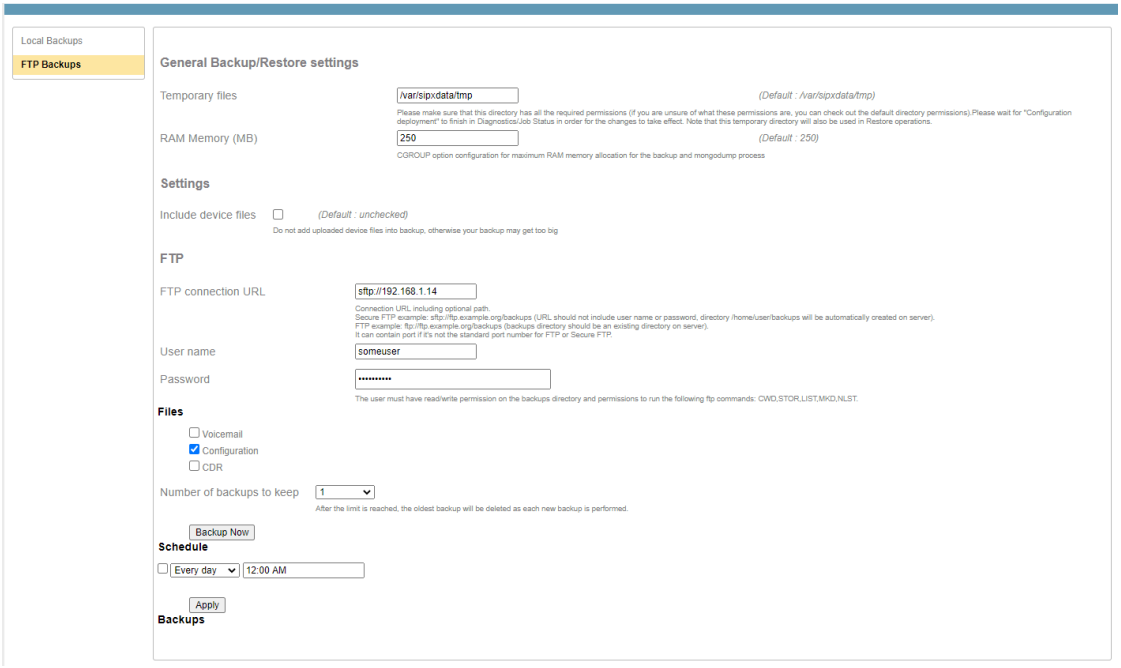
Local backups are stored on the server disk.



Warning: Creating a backup archive can use a lot of disk space, especially if voicemail is selected. **If you run out of free disk space all services will halt!** We strongly recommend setting "number of backups to keep" to 5 or less.

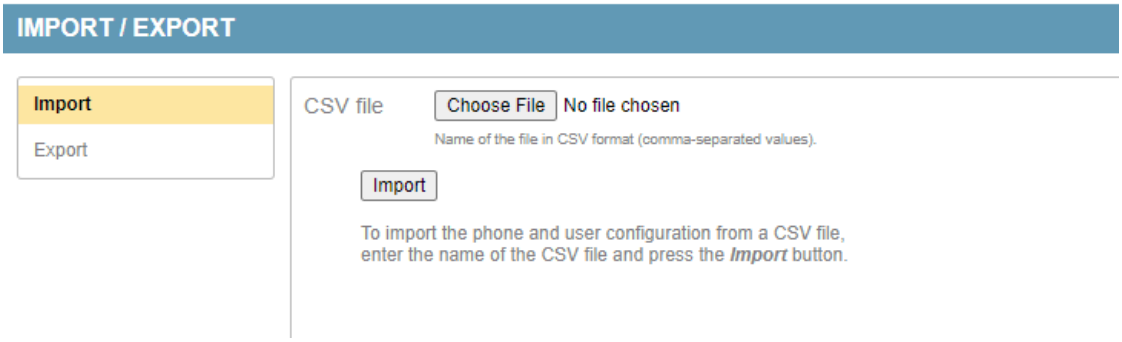
FTP

FTP backups can be used to transfer the backup automatically to a FTP (use the uri ftp://) or SFTP (use the uri sftp://) server.



7.5.4 Import/Export

Import



Phone and user data can be imported from a CSV file (comma separated values), which is compatible with most spreadsheet applications. The CSV should have a title line and the following fields:

- User name
- PIN
- Voice-mail PIN
- SIP password
- First name
- Last name

- User alias
- EMail Address
- User group
- Phone serial number
- Phone model
- Phone group
- Phone description
- IM ID

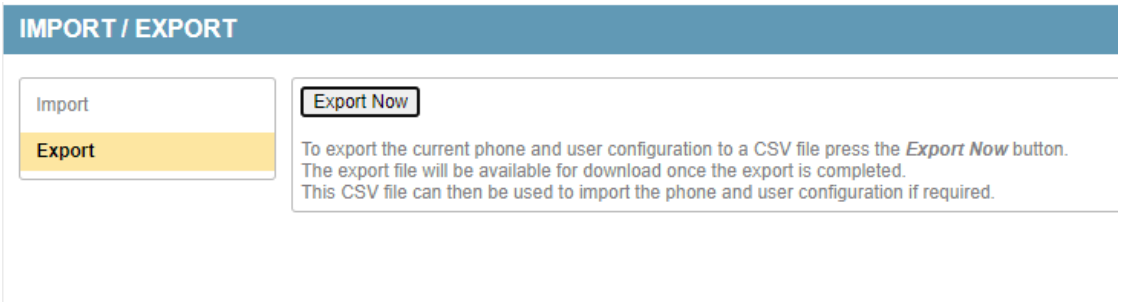
All CSV header fields as of 20.04:

```
User name,PIN,Voicemail PIN,SIP password,First name,Last name,User alias,EMail␣
↪address,User group,Phone serial number,Phone model,Phone group,Phone description,Im␣
↪Id,Salutation,Manager,EmployeeId,Job Title,Job department,Company name,Assistant␣
↪name,Cell phone number,Home phone number,Assistant phone number,Fax number,Did␣
↪number,Alternate email,Alternate im,Location,Home street,Home city,Home state,Home␣
↪country,Home zip,Office street,Office city,Office state,Office country,Office zip,
↪Office mail stop,Twitter,Linkedin,Facebook,Xing,Active greeting,Email voicemail␣
↪notification,Email format,Email attach audio,Alternate email voicemail notification,
↪Alternate email format,Alternate email attach audio,Internal Voicemail Server,
↪Caller ID,Block Caller ID,Additional phone settings,Additional line settings,Auth␣
↪Account Name,EMail address aliases,Custom 1,Custom 2,Custom 3
```

Each line from imported file will result in creation of the phone and the user assigned to that phone. If user group or phone group fields are not empty, the newly created user and phone will be added to respective groups. Groups will be created if they do not exist already.

If the user with the same username is already present, this system will update existing user instead of creating a new one. The same is true for phones: if the phone with the same serial number already exist it'll be updated. Only user name and phone serial number are obligatory fields. You can leave the remaining fields empty - in which case this system will not overwrite their values.

Export



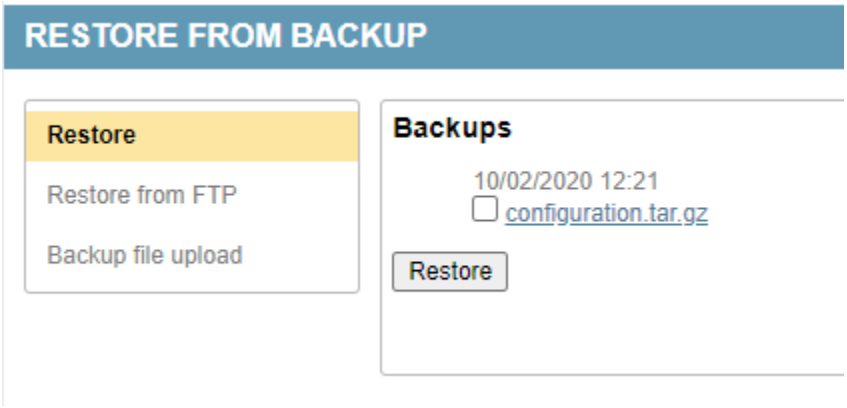
7.5.5 Restore

The restore feature allows administrators to restore configuration data, voicemail data, or CDR data. The gzip archives are created by the Backup feature.

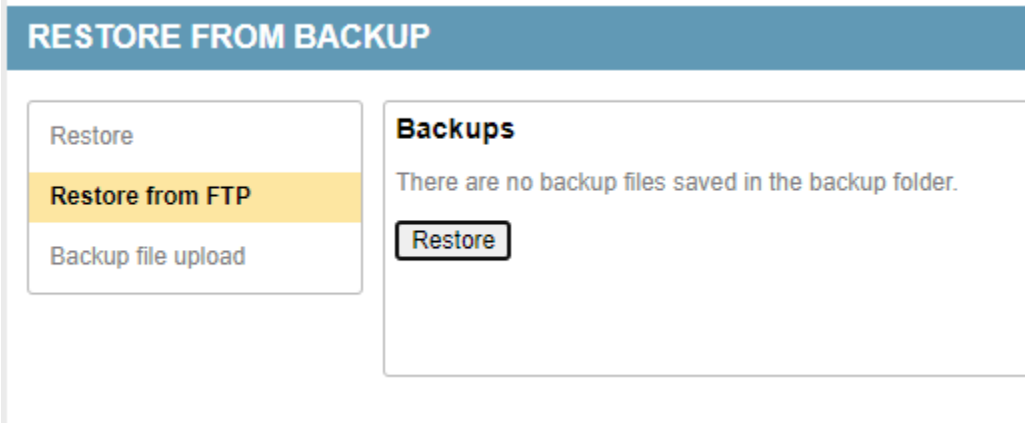
Note: Backup archives from very old installations (prior to 14.04) may need to be restored in a series of incremental steps. In those cases a CSV restore of only user and phone data may be more appropriate.

Restore

The Restore page reads from the local backups folder by default.

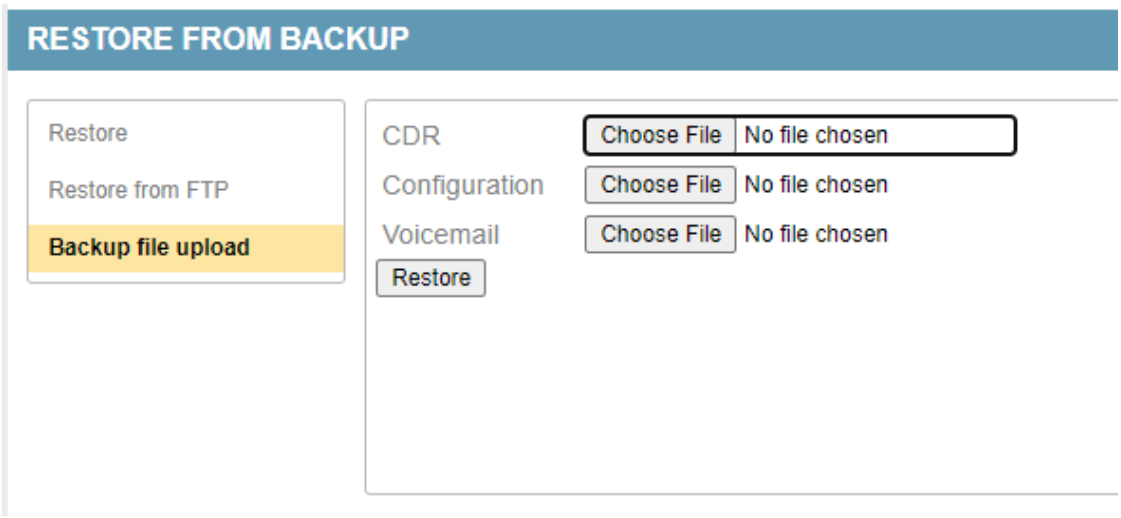


Restore from FTP

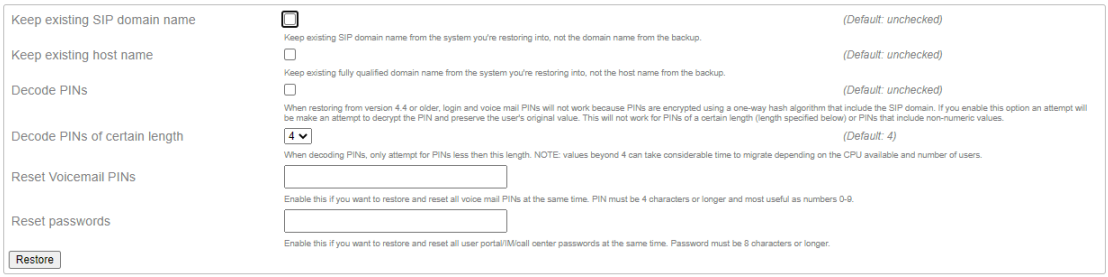


Backup file upload

Upload configuration, voicemail, or CDR data archives.



The options presented upon a configuration restore are to keep the existing SIP domain, keep the existing hostname, decode voicemail PINs or specify the voicemail PIN length, reset all voicemail PINs, and reset user passwords (user portal/IM password).

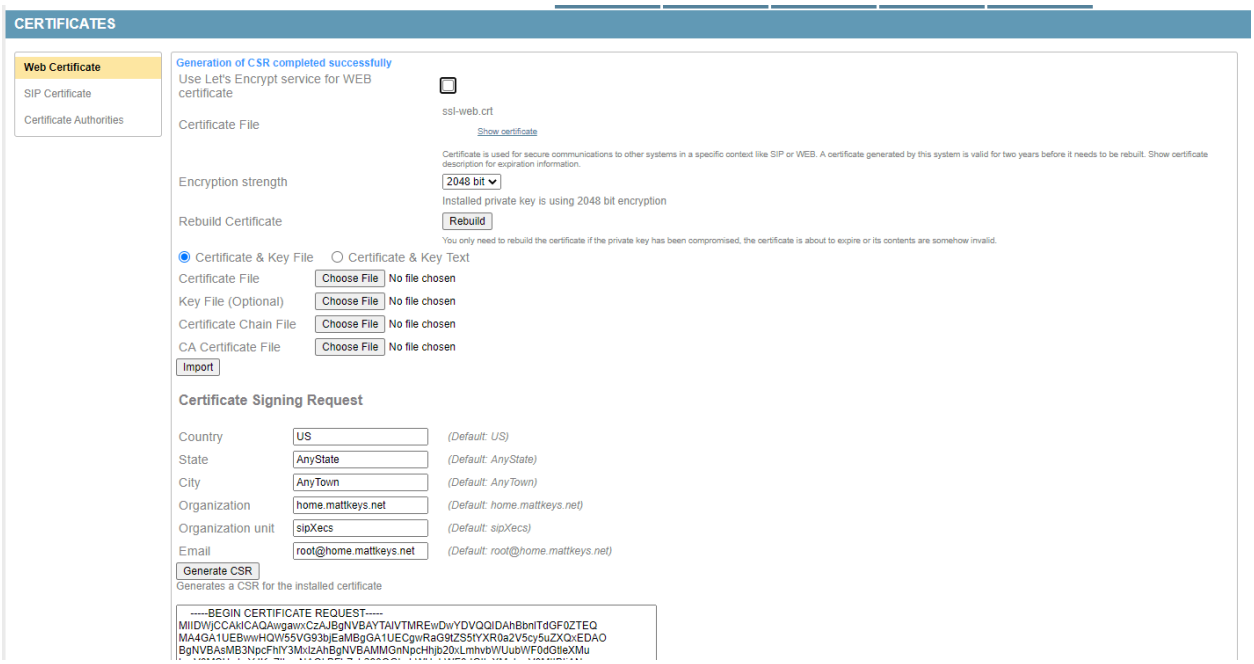


7.5.6 Security

Certificates

The system - certificates page has three tabs to the left, Web Certificate, SIP Certificate, and Certificate Authorities (CAs). The web certificate is used for https on the webui and device provisioning, SIP certificate for SIPS (SIP+TLS) connections, and the Certificate Authorities to load CA or any intermediary certificates.

Warning: Use of sips (5061) is not recommended because not all services work properly with it. If this is important to you we recommend offloading TLS (sips) on the way to the sipxcom with a Session Border Controller (SBC).

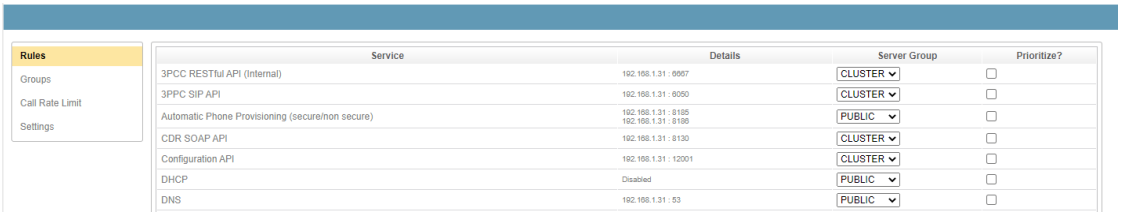


7.5.7 Firewall

Sipxcom includes a generic yet powerful firewall based upon netfilter iptables.

Rules

Rules are determined automatically based on what services are running.

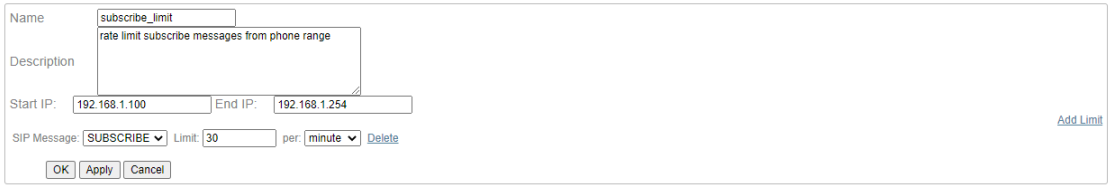


Groups

PUBLIC is all addresses (0.0.0.0), CLUSTER is only the servers in the sipxcom cluster. You can also add custom groups.

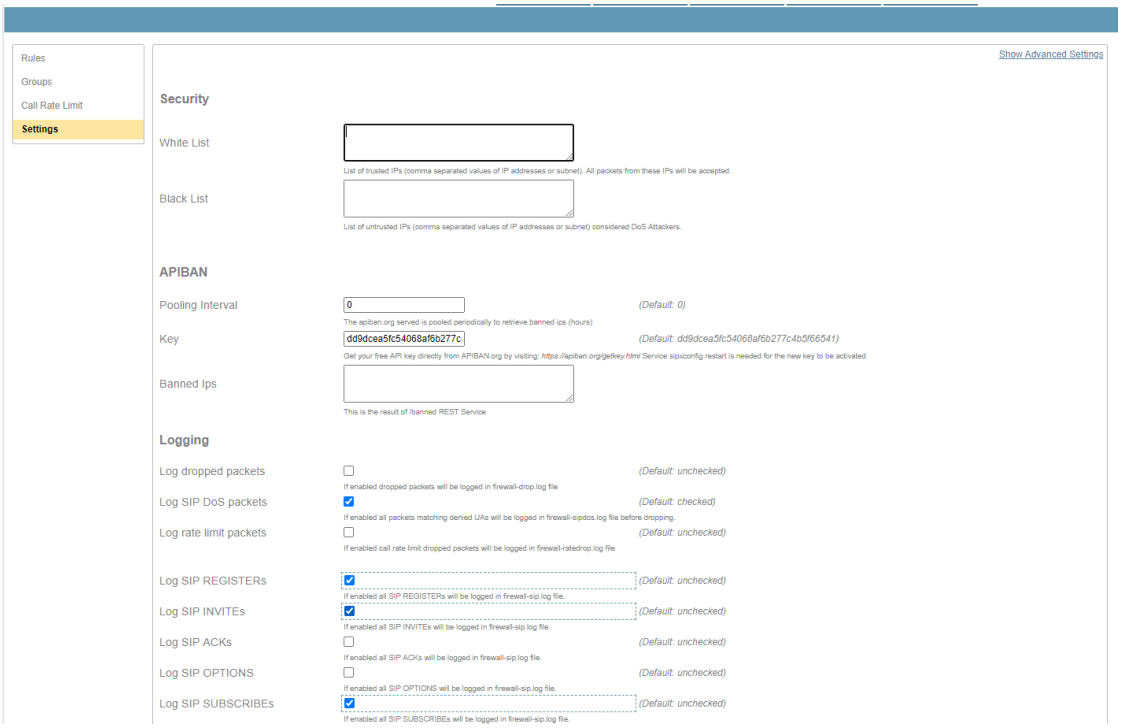
Call Rate Limit

Create Call Rate Limit Rule in order to prevent DoS attacks or to limit SIP traffic for the defined range of IPs. Leave end IP empty in case you want to define call rate limit for a single IP address or for a subnet.



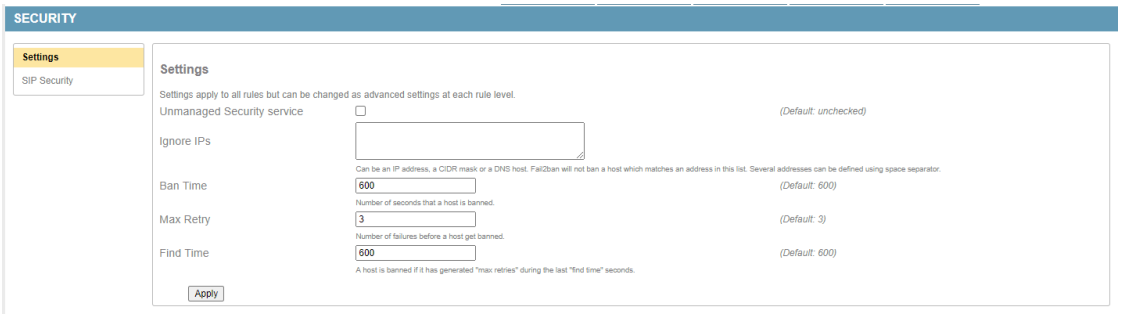
Settings

Use the settings page to add IPs or IP ranges (in CIDR format) to the whitelist (always allow), blacklist (always block), or new in 20.04 you can add the blacklist from LODs API Ban. Also important on this page are the "Log xxx" options. These are required for SIP Security mechanisms and rating limiting.



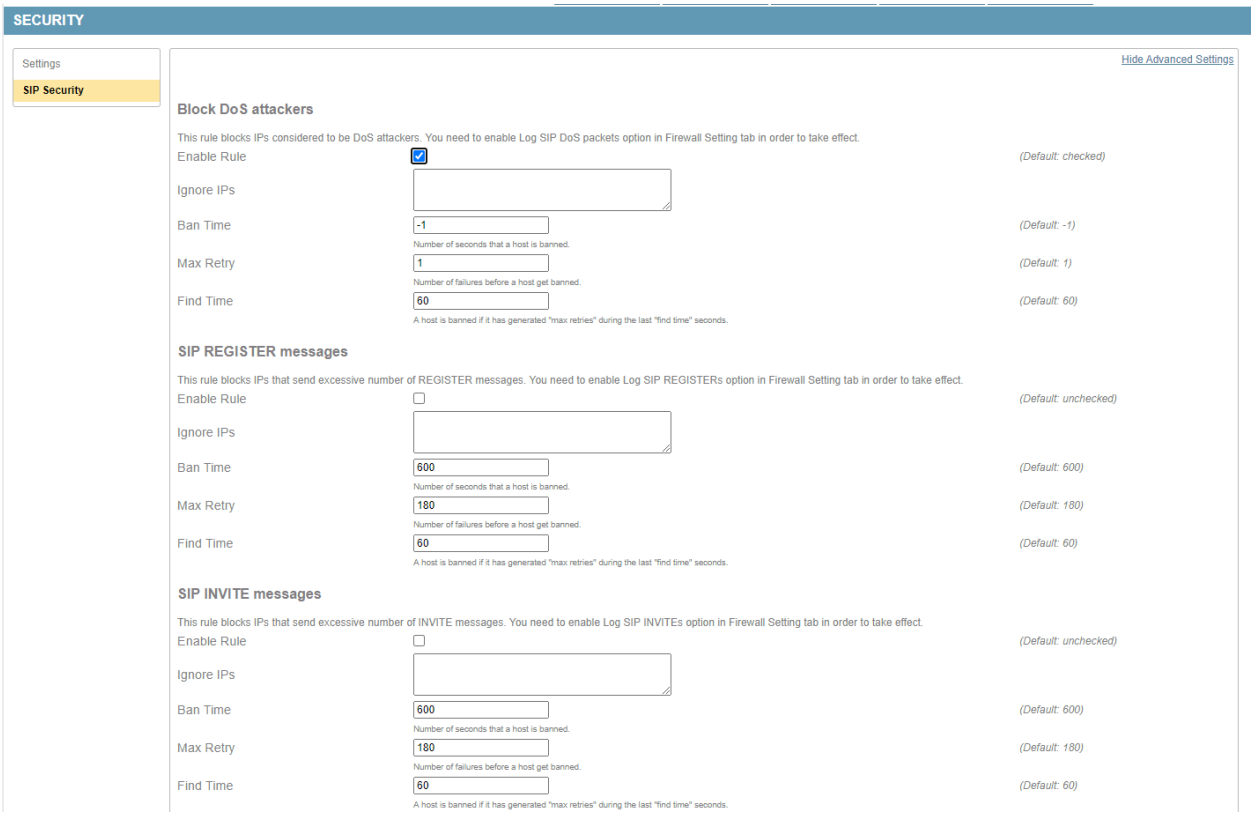
SIP Security

The SIP security page uses fail2ban to automatically ban IPs that have exceeded the thresholds defined. It does so by adding a rule to iptables to deny the source address all destinations.



Usage of these mechanisms requires additional logging (see Firewall Settings section).

Warning: Do not use this feature if a Session Border Controller (SBC) is in use! All SIP traffic will originate from the SBC in that case and you wouldn't want to ban that. The SBC should have rules in place to protect sipxcom. Use the *sipcodes.sh* script to verify those rules are working.



TLS Peers

To allow calls from an authenticated peer to use resources that require permissions, add the domain as a Trusted Peer and configure the permissions for it. The peer must use TLS to communicate to this system, and the Certificate Authority used to sign certificates must be installed on both systems.

7.5.8 Servers

The Servers page includes six tabs on the left: Servers, Core Services, Telephony Services, Instant Messaging, Device Provisioning, and Utility Services.

About sipxsupervisor (CFEngine)

Sipxsupervisor uses CFEngine, a configuration management and automation framework, to define the desired state and configuration of each server. The sipxsupervisor service (cfengine agent) running on each server ensures compliance. The key exchange model used for this process is based on that used by openssh. The keys are exchanged during the intitial *setup script* (just after the *Is this the first server in the cluster?* question).

Note: If a server in the cluster is showing "Uninitialized" in the status field, that generally indicates the primary (webui) server has lost communication with the cfengine agent running on that server. To correct the problem try issuing on the affected server:

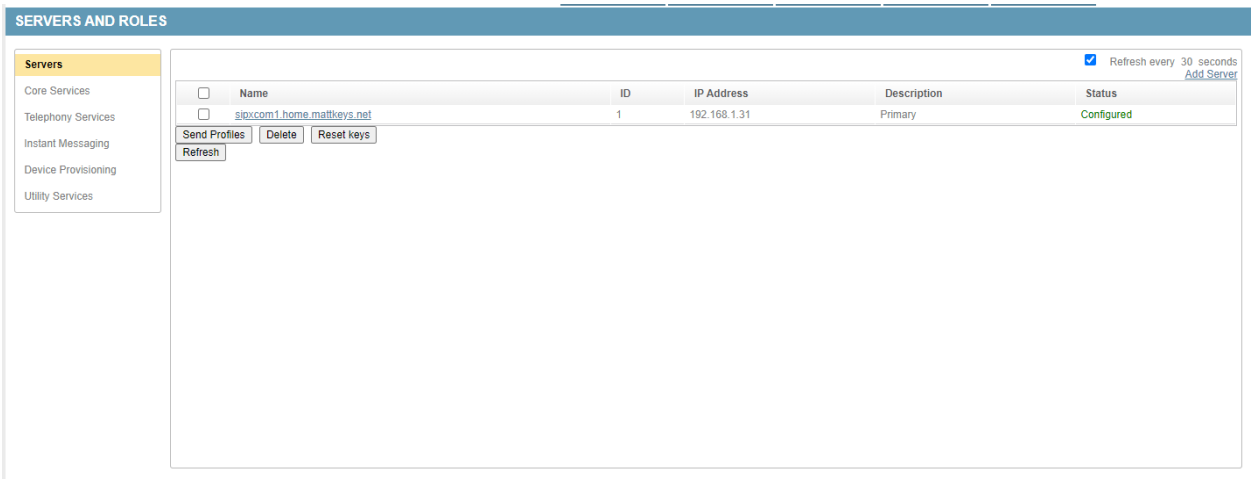
```
service sipxsupervisor restart
```

It expected to see Uninitialized status if you have defined the server in the webui, but have yet to run the sipxecs-setup script on the server to complete the key exchange.

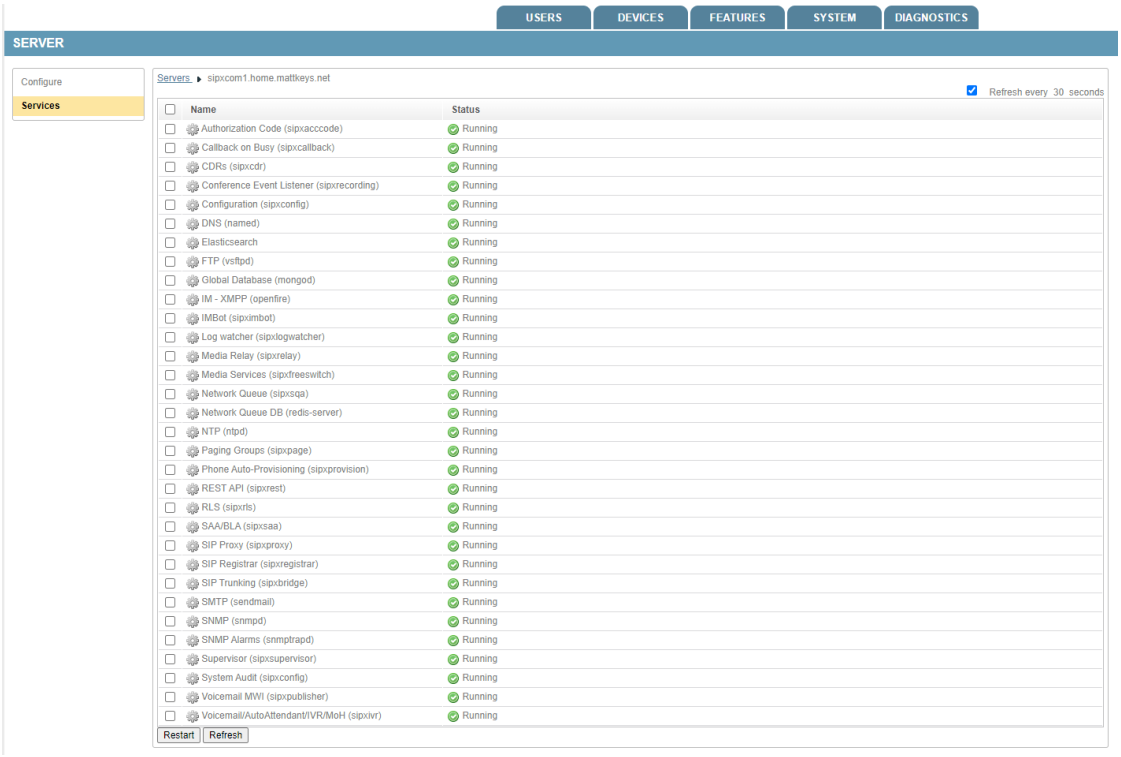
Warning: Do not alter the sshd (or firewall, network, etc) configuration in such a way that would prevent root login between the servers. This will break sipxsupervisor (cfengine) communication. Running other configuration management agents such as Puppet or Chef will also conflict with sipxsupervisor (cfengine).

Servers

This page lists each server in the cluster as a hyperlink. The status field indicates if sipxsupervisor (cfengine) is responding and healthy on that server.



By clicking the link of a server, you can restart any service on that server. This is also accomplished by communicating with sipxsupervisor (cfengine) on that server.



Sending Server Profiles

The send profiles button forcefully redeploys all configuration, for all services, to the selected servers. The cfengine term for this is configuration convergence. Any affected services will be restarted (if required) automatically by the agent.

Note: Upon sending server profiles you can verify the connection took place by monitoring (tail -f) /var/log/messages of the server(s). The log entry should look similar to:

```
Oct 19 18:16:53 sipxcom1 cf-serverd[16545]: Accepting connection from "192.168.1.31"
```

Reset Keys

The Reset Keys option attempts to re-establish stored authentication key pairs used by sipxsupervisor (cfengine) agents in the cluster. The key exchange model is based on that used by OpenSSH. It is a peer to peer exchange model, not a central certificate authority model.

Note: To verify the network path is good, ssh as root to each server in the cluster, and from each server in the cluster.

Warning: Only use Reset Keys if the key pairs that were established during initial setup have changed. You should have a good explaination as to why the stored keys have changed.

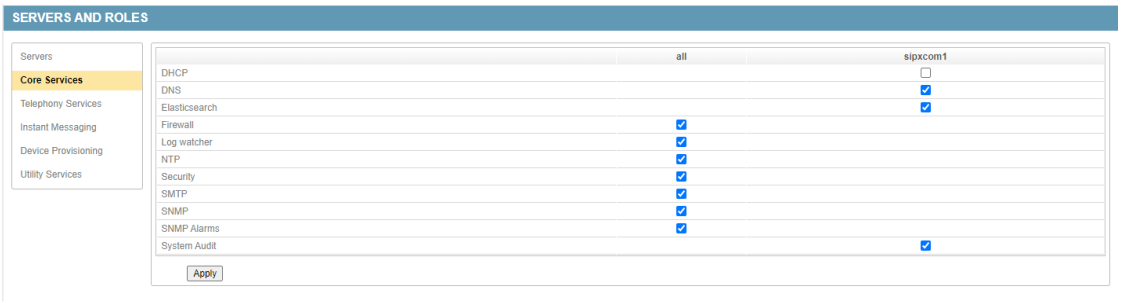
If needed you can force the sipxsupervisor (cfengine) agent of any server to run configuration convergence by issuing on the command line:

```
sipxagent
```

Similar to *Sending Server Profiles*, you can verify by checking for agent connection log entries in /var/log/messages of the server(s).

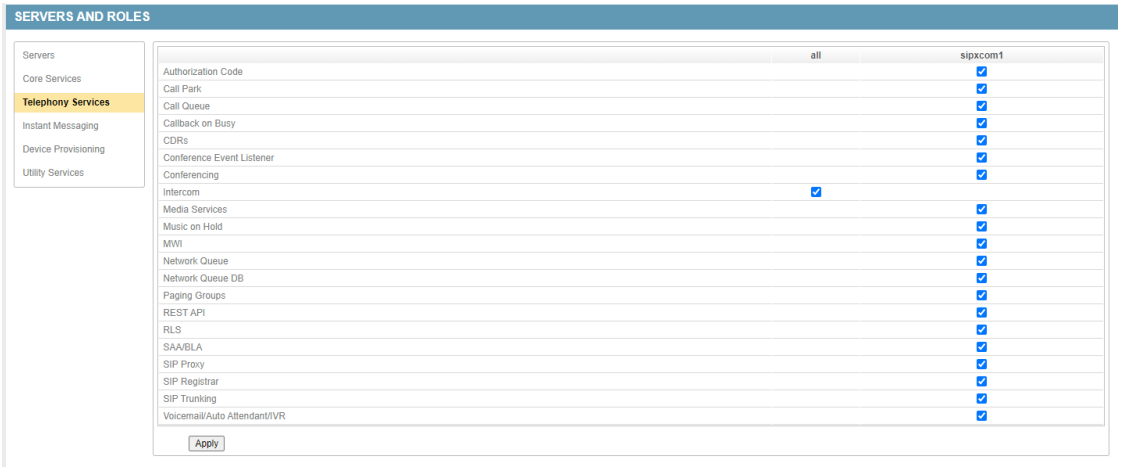
Core Services

This page allows you to enable or disable DHCP, DNS, Elasticsearch, Firewall, Log watcher, NTP, SIP Security, SMTP, SNMP, SNMP Alarms, and System Audit. A sipxcom feature might require one or more of these services to be enabled. Some services can only run on the primary server.



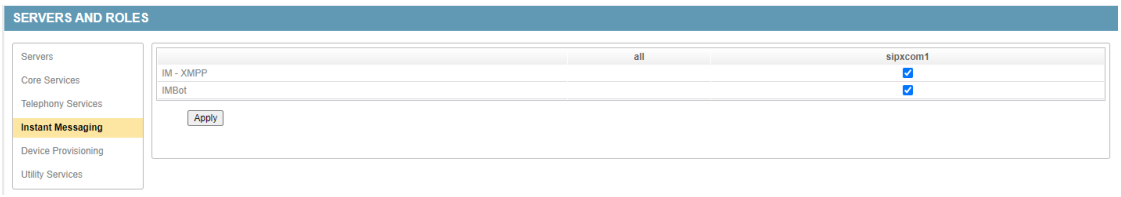
Telephony Services

This page lists all telephony related services, and what server it is enabled on in the cluster. Some services can only run on the primary server. Some services require other services to be enabled. The webui should refresh in that case and highlight service(s) required.



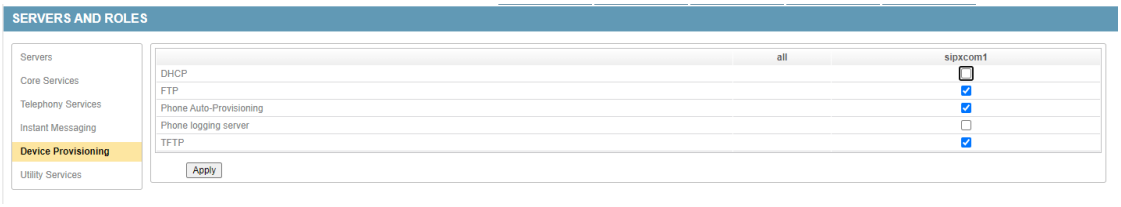
Instant Messaging

This page allows you to enable or disable the OpenFire XMPP server service. The My Buddy function is dependant upon the IMBot service being enabled. The IMBot depends upon the IM-XMPP feature.



Device Provisioning

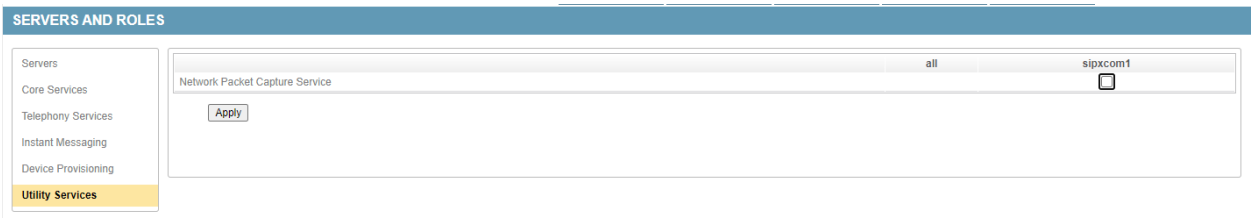
This page allows you to enable or disable the DHCP, FTP, Phone Auto Provisioning (HTTP/HTTPS), Phone Logging (syslog server), or Trivial FTP (tftp) services.



Utility Services

This page allows you to enable automatic packet captures using tcpdump.

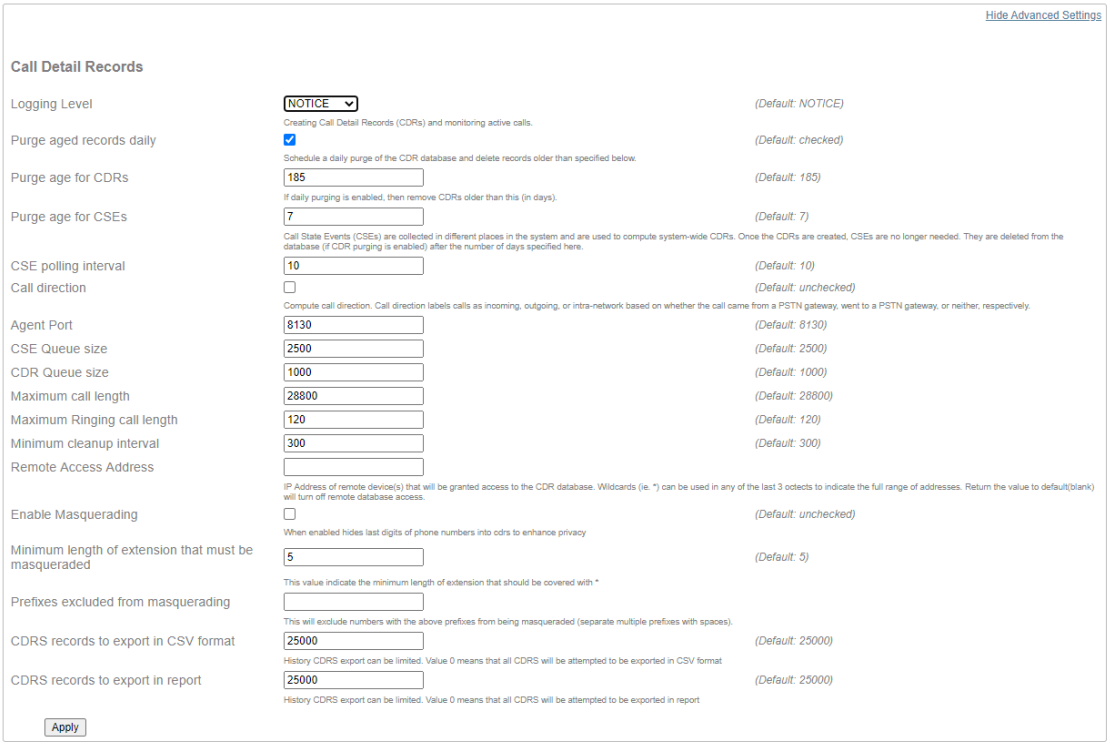
Warning: This is resource intensive and should only be enabled to assist in troubleshooting a problem that you can replicate. If you enable this don't forget to review settings beneath *Diagnostics - Network Packet Capture - Configure tab*.



7.5.9 Services

The Services menu has the CDR, Conference Event Listener, DNS, FTP Server, Instant Messaging, Log Watcher, Media Services, MWI, My Buddy, Phone Provision, Rest Server, SAA/BLA, Service Msg Queue, RLS, SIP Proxy, SIP Registrar, SIP Trunk, SNMP, and Voicemail options.

CDR



Conference Event Listener

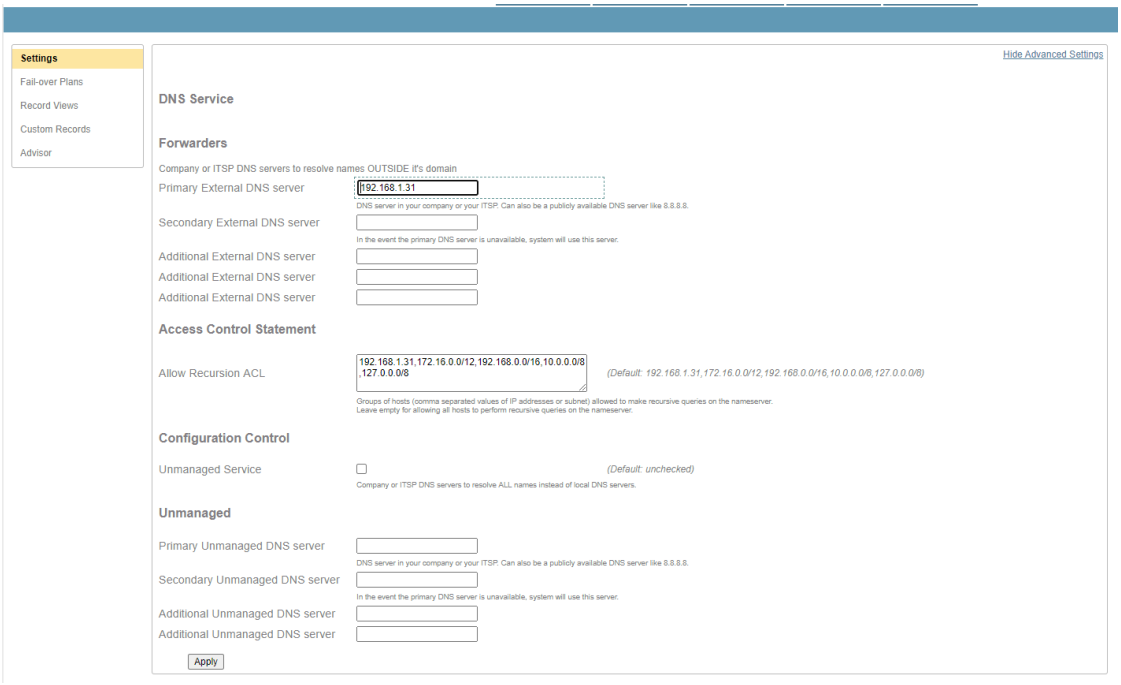
DNS

The DNS menu has five tabs to the left: Settings, Fail-over Plans, Record Views, Custom Records, and Advisor.

Settings tab

If using Managed DNS, sipxcom (sipxsupervisor/cfengine) will manage the DNS zone file, which is stored beneath /var/named/, and the DNS server configuration file /etc/named.conf.

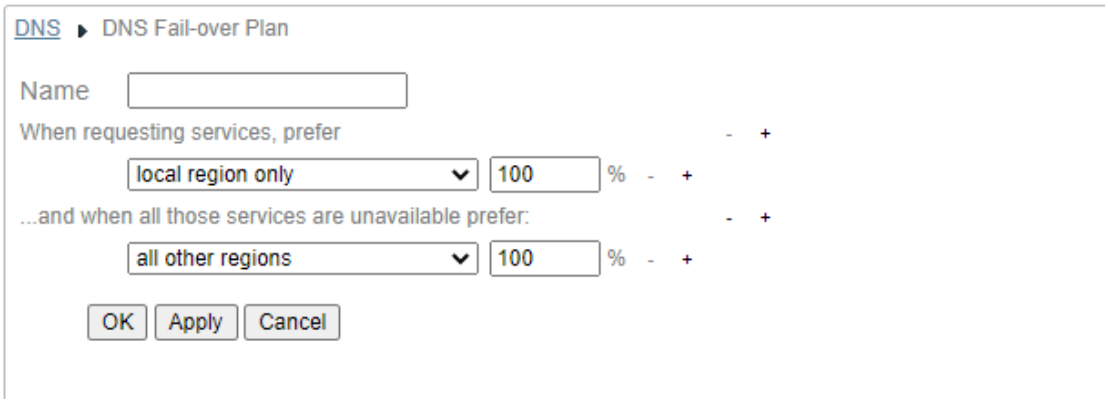
If using Unmanaged DNS, sipxcom (sipxsupervisor/cfengine) will not change the zone file or /etc/named.conf.



Fail-over Plans

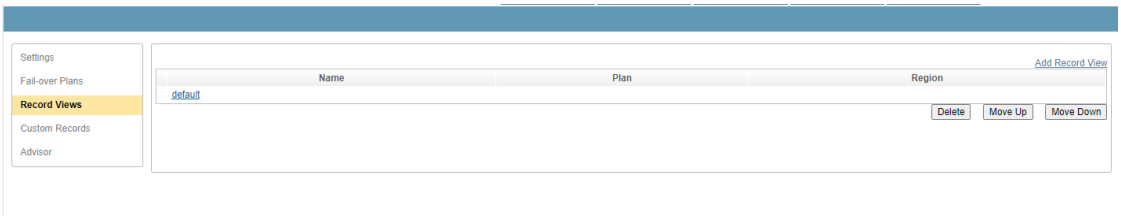
Fail-over plans control what services are used and when and how much traffic they receive. Fail-over plans are used in DNS record views and they can be reused for many views.

A fail-over plan controls how traffic flows into and through the cluster when there is a server or network failure. This can also be used when you want to distribute traffic unevenly through your system to account for resource constraints or various other reasons. It's important to understand that regardless of the failover plan, once traffic hits a server the services that are local to that server will be preferred. For example, you may have a SIP proxy take 1% of the traffic, but once the SIP REGISTER message enters that server it will use the local registrar. The failover plan will only be used if the local registrar does not respond.

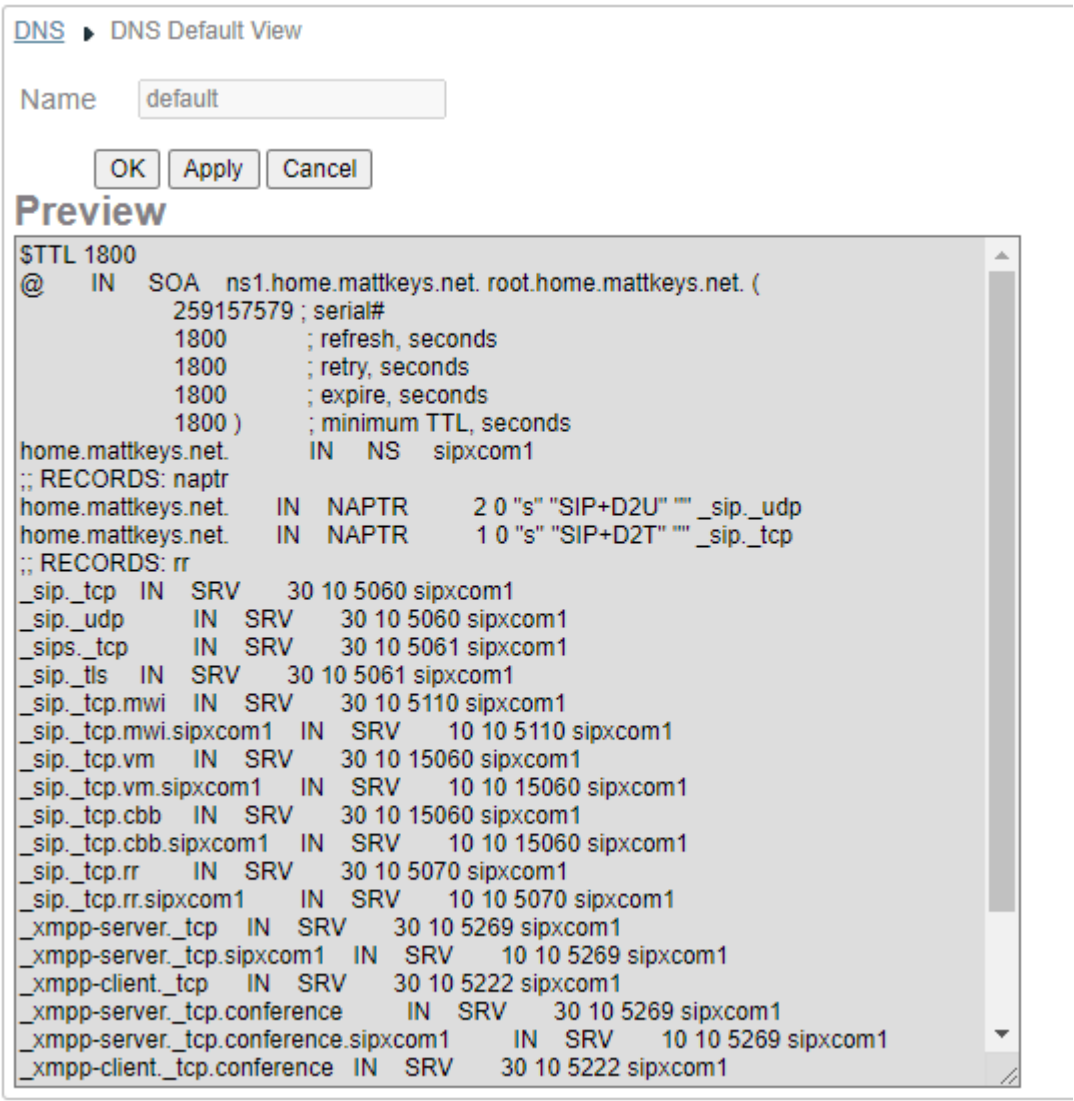


Record Views

Record views allow you to have a different set of DNS records for a region of your network.



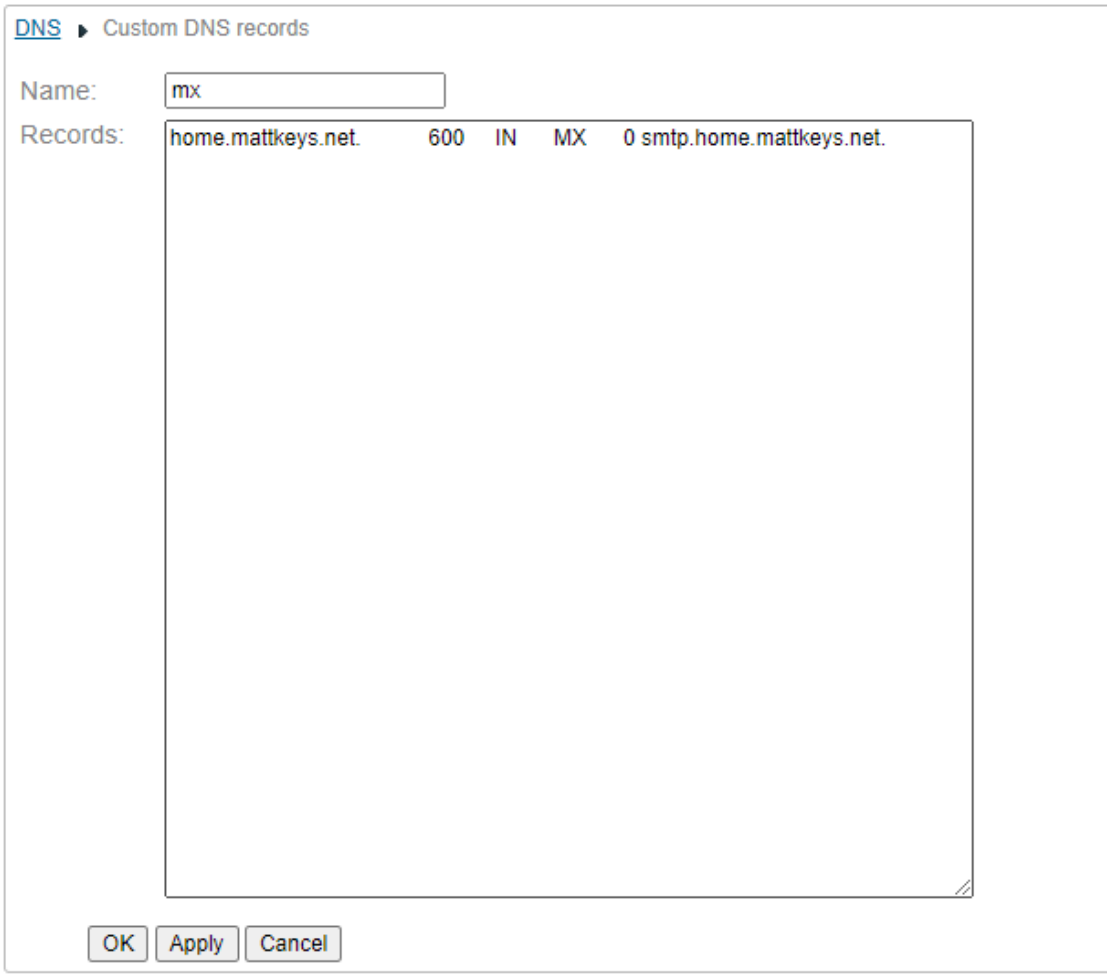
The default SRV record priority and weight will distribute traffic evenly among all cluster members.



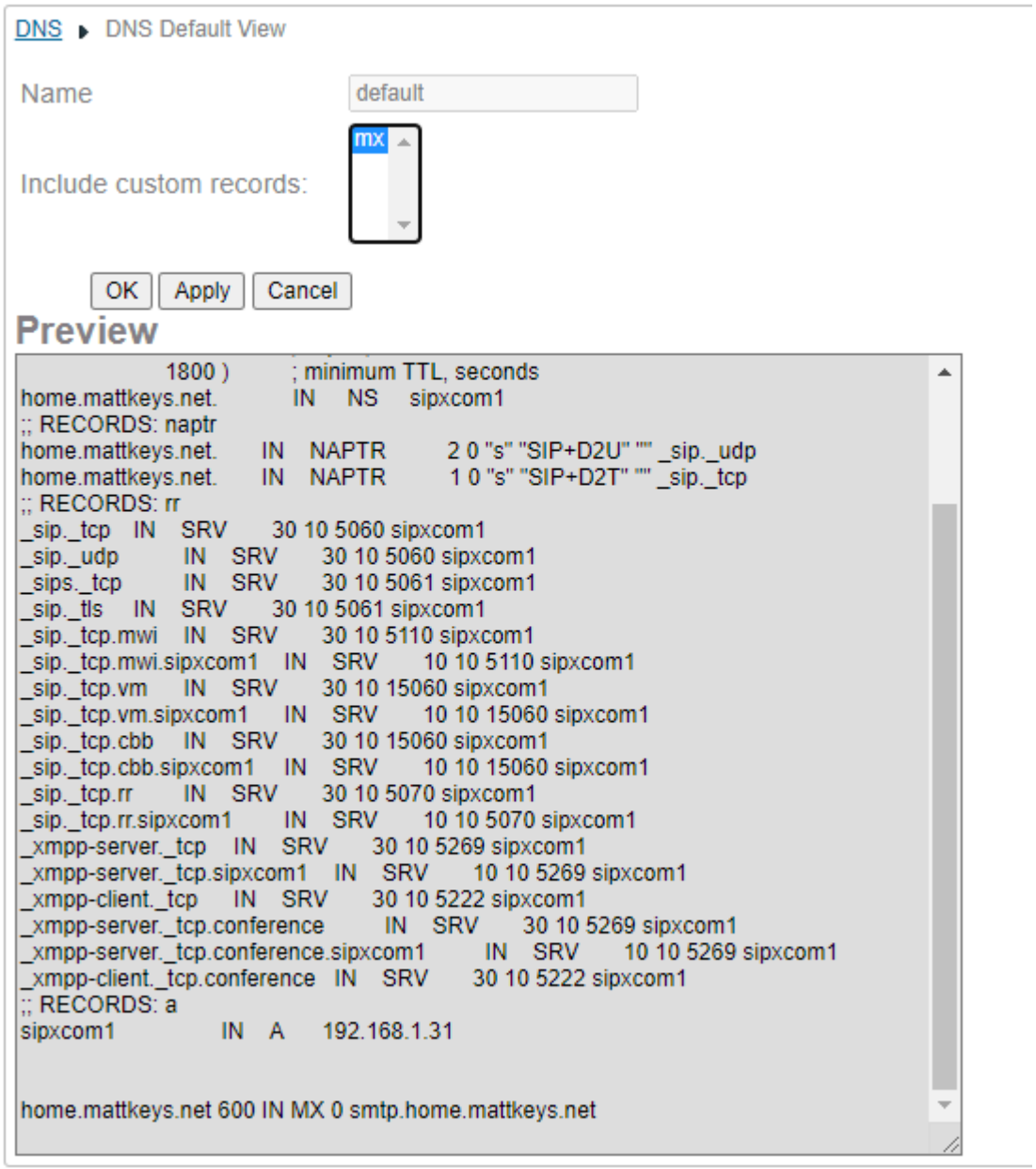
Custom Records

You'll need to add records for entries missing from the default plan such as MX or A records.

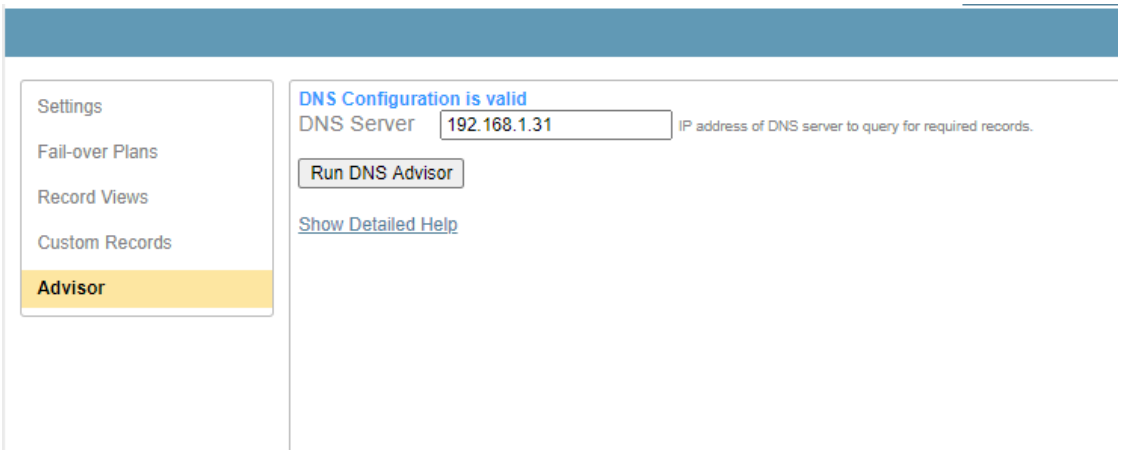
Click 'Add custom record' to create new zone entries.



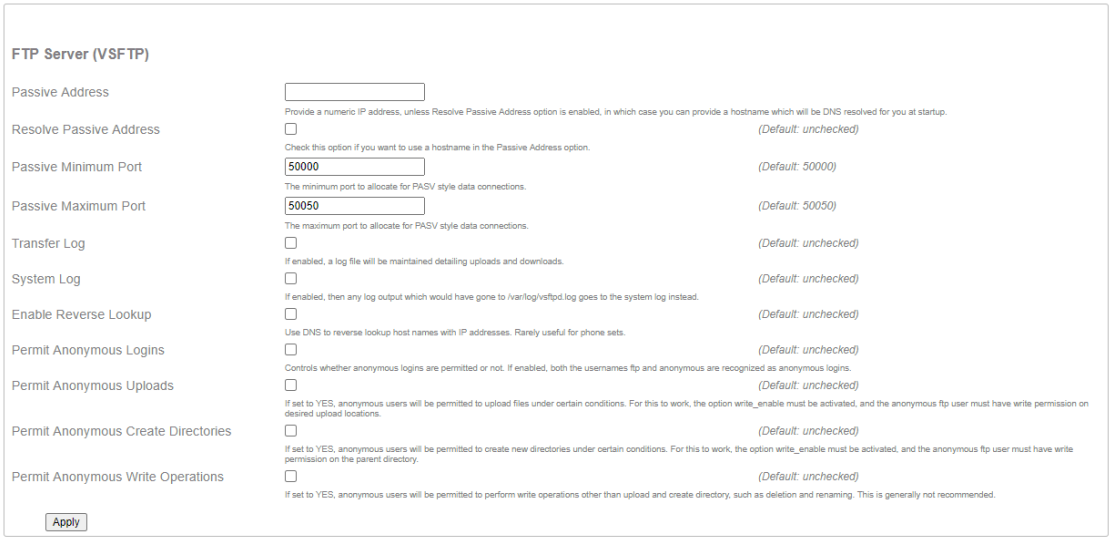
After saving the new record(s), navigate to *Record Views* and click the zone. Highlight the records to add into the zone next, then click apply. The preview should now dispaly the new records at the bottom of the zone.


DNS
DNS Default View
Name
default
Include custom records:
mx
OK
Apply
Cancel
Preview
1800) ; minimum TTL, seconds
home.mattkeys.net. IN NS sipxcom1
;; RECORDS: naptr
home.mattkeys.net. IN NAPTR 2 0 "s" "SIP+D2U" "" _sip._udp
home.mattkeys.net. IN NAPTR 1 0 "s" "SIP+D2T" "" _sip._tcp
;; RECORDS: rr
_sip._tcp IN SRV 30 10 5060 sipxcom1
_sip._udp IN SRV 30 10 5060 sipxcom1
_sips._tcp IN SRV 30 10 5061 sipxcom1
_sip._tls IN SRV 30 10 5061 sipxcom1
_sip._tcp.mwi IN SRV 30 10 5110 sipxcom1
_sip._tcp.mwi.sipxcom1 IN SRV 10 10 5110 sipxcom1
_sip._tcp.vm IN SRV 30 10 15060 sipxcom1
_sip._tcp.vm.sipxcom1 IN SRV 10 10 15060 sipxcom1
_sip._tcp.cbb IN SRV 30 10 15060 sipxcom1
_sip._tcp.cbb.sipxcom1 IN SRV 10 10 15060 sipxcom1
_sip._tcp.rr IN SRV 30 10 5070 sipxcom1
_sip._tcp.rr.sipxcom1 IN SRV 10 10 5070 sipxcom1
_xmpp-server._tcp IN SRV 30 10 5269 sipxcom1
_xmpp-server._tcp.sipxcom1 IN SRV 10 10 5269 sipxcom1
_xmpp-client._tcp IN SRV 30 10 5222 sipxcom1
_xmpp-server._tcp.conference IN SRV 30 10 5269 sipxcom1
_xmpp-server._tcp.conference.sipxcom1 IN SRV 10 10 5269 sipxcom1
_xmpp-client._tcp.conference IN SRV 30 10 5222 sipxcom1
;; RECORDS: a
sipxcom1 IN A 192.168.1.31
home.mattkeys.net 600 IN MX 0 smtp.home.mattkeys.net

Advisor



FTP Server



Instant Messaging

Log Watcher

Service that reads incoming log messages and reacts accordingly. Typically used to trigger a SNMP alarms.

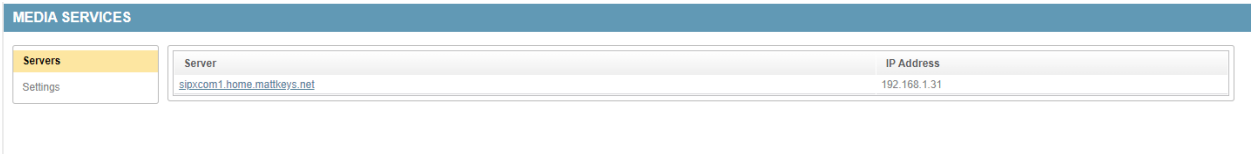
Note: This setting does not change the log verbosity of other services.

Media Services

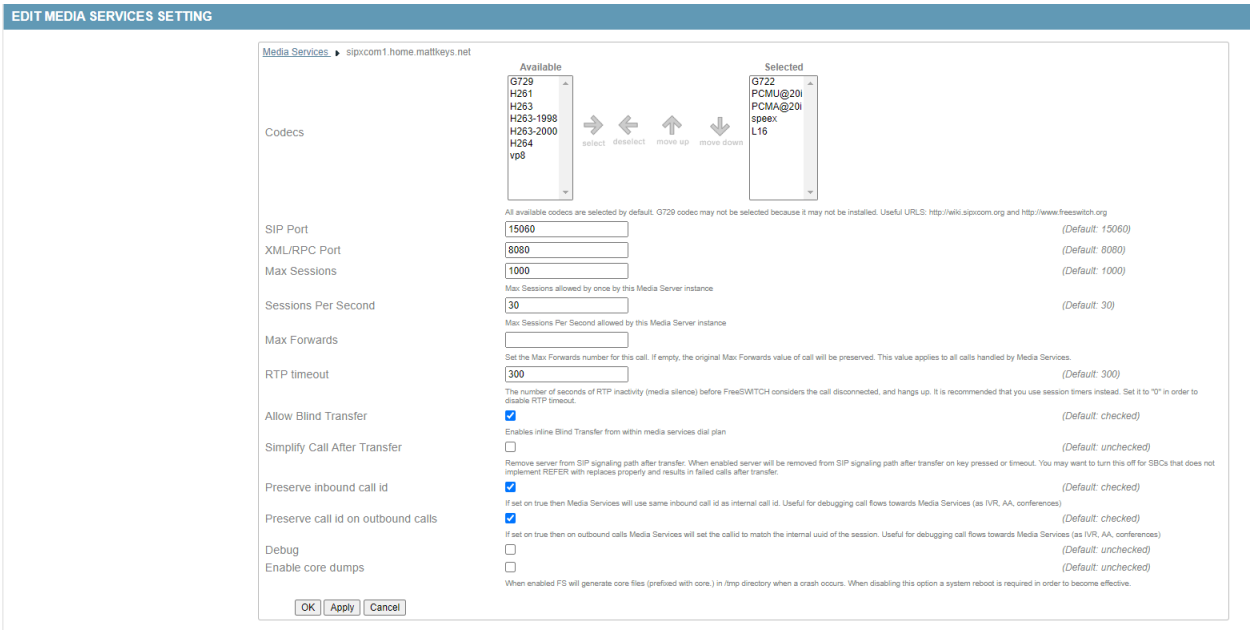
This is the configuration page for Media Services (FreeSWITCH). WAV or MP3 files can be used for prompts, MoH, and voicemail recordings. Audacity is a good program to use to create or convert media files. Another popular utility is Sound Exchange (sox).

Note: wav files must be in the appropriate format of RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 8000 Hz

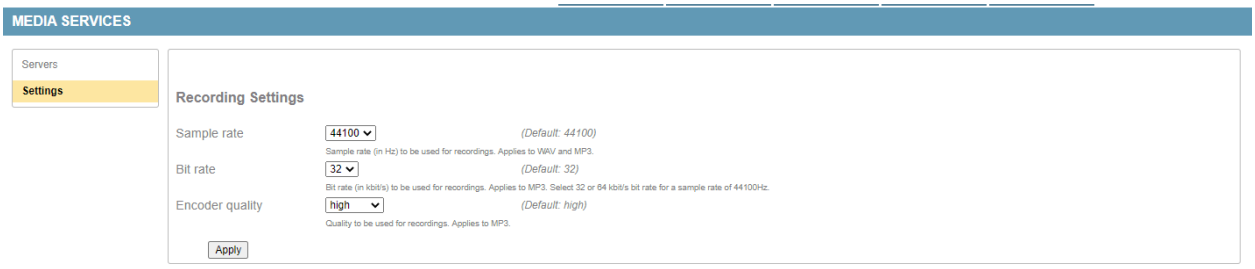
An easy way to record a file quickly is to use the Voicemail Attachment option beneath the user profile Unified Messaging tab. The file attachment will be in the proper format. By default 8 is the voicemail prefix. So if your extension is 200, dial 8200 to immediately deposit a voicemail to yourself.



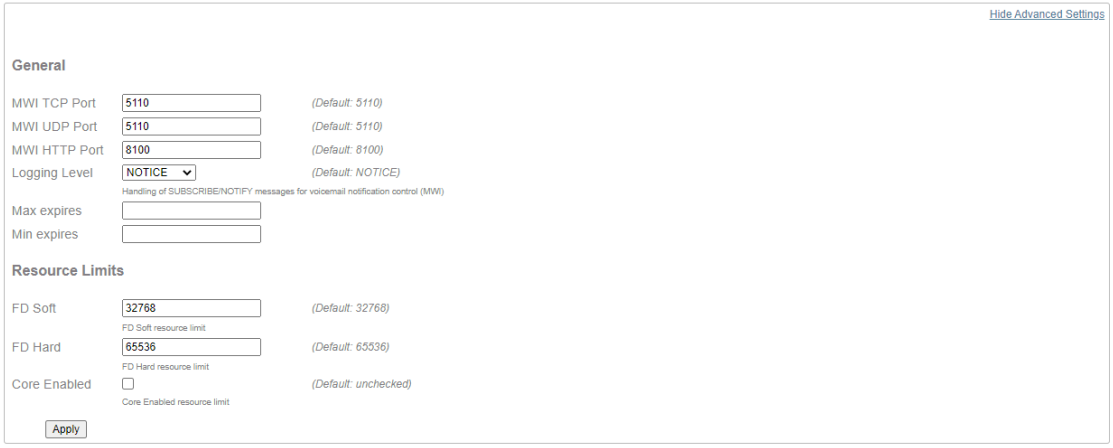
Each server running media services (freeswitch) in the cluster is listed and can be individually edited.



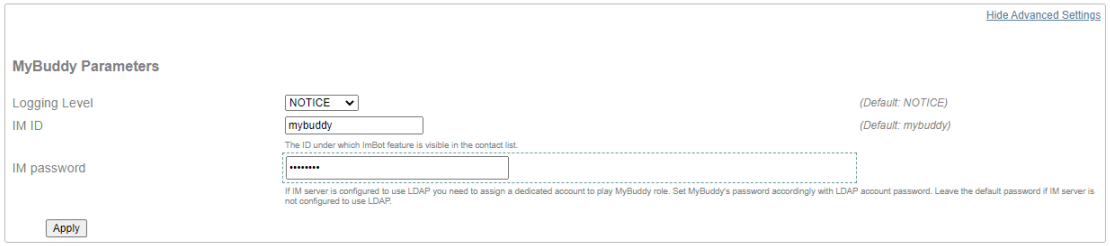
The Settings tab on the left are global settings.



Message Waiting Indicator (MWI)



My Buddy

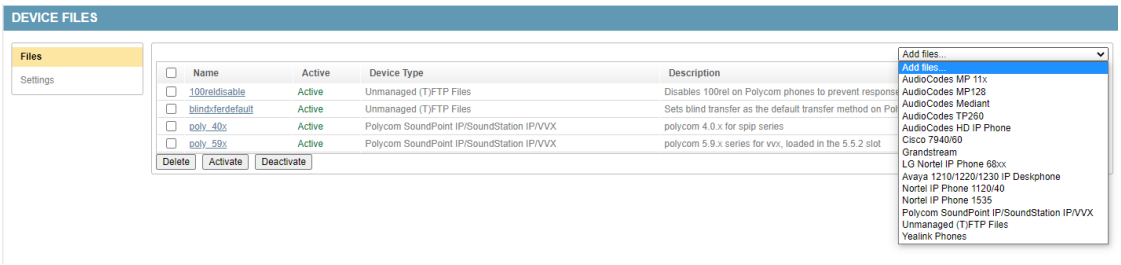


Phone Provision

The Provision Menu is used to upload phone firmware. There are two menu options of Device Files and Settings.

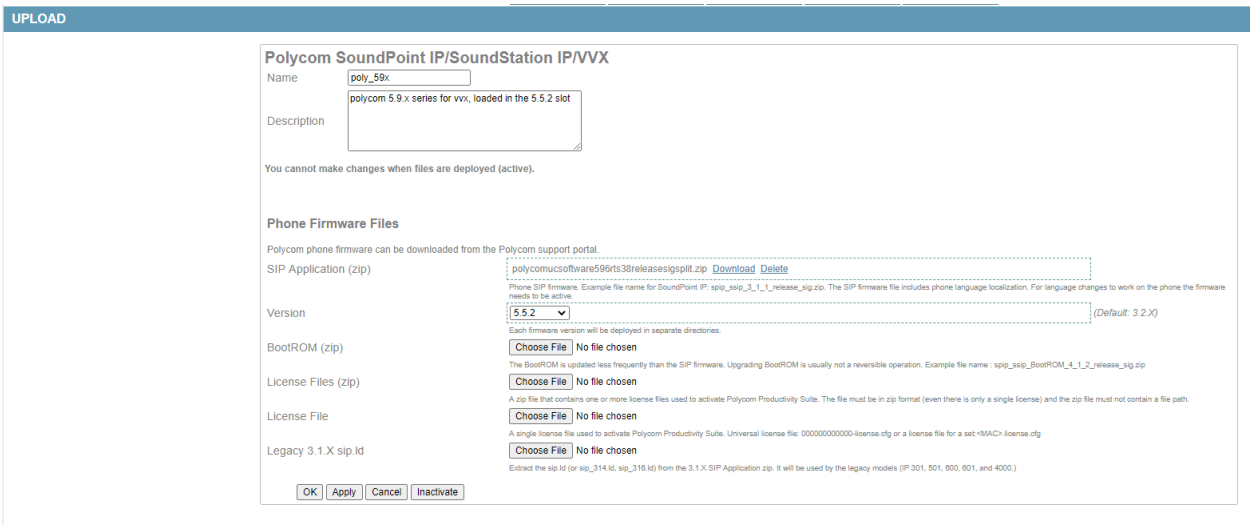
Note: All generated phone configuration files are beneath /var/sipxdata/configserver/phone/profile/tftproot/. Any phone custom configuration (Unmanaged TFTP) files and uploaded firmware archives are deployed beneath that path as well.

Device Files



To find the latest GA for your model Polycom visit the Polycom firmware matrix. There are two pages, one for SoundPoint and SoundStation IP models, and the other for VVX models.

The Version drop-down is only a label to distinguish a set of files. It is not required to match the actual firmware version.

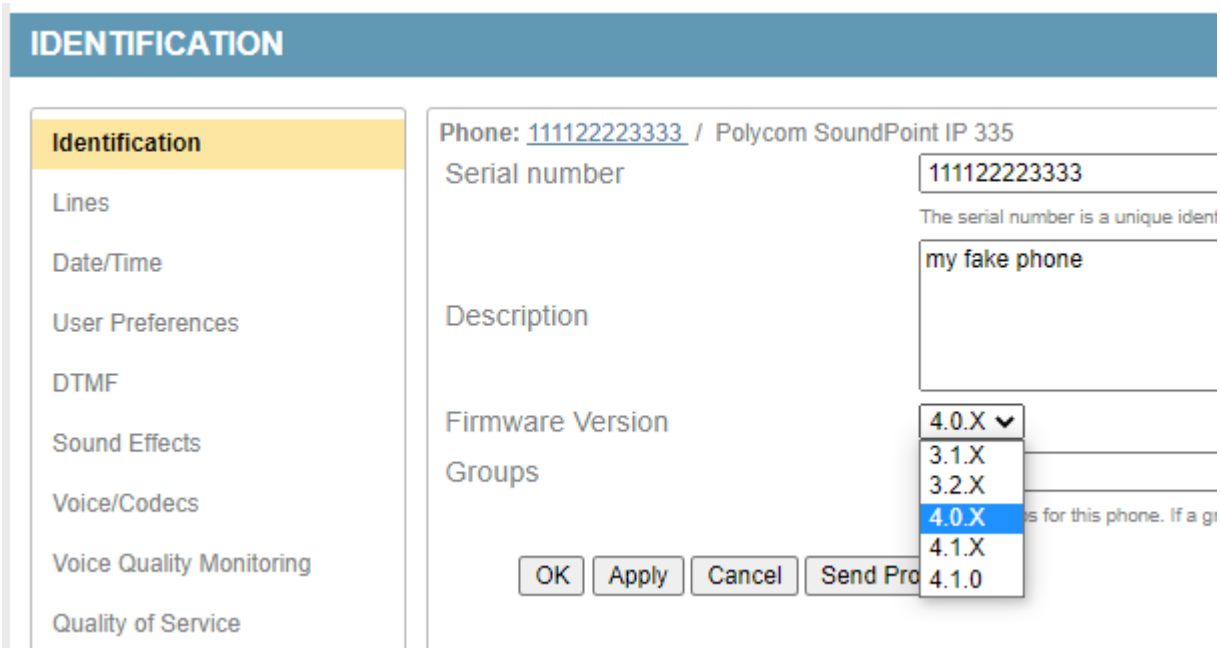


The firmware application zip is extracted to the server filesystem beneath whatever the label was set to:

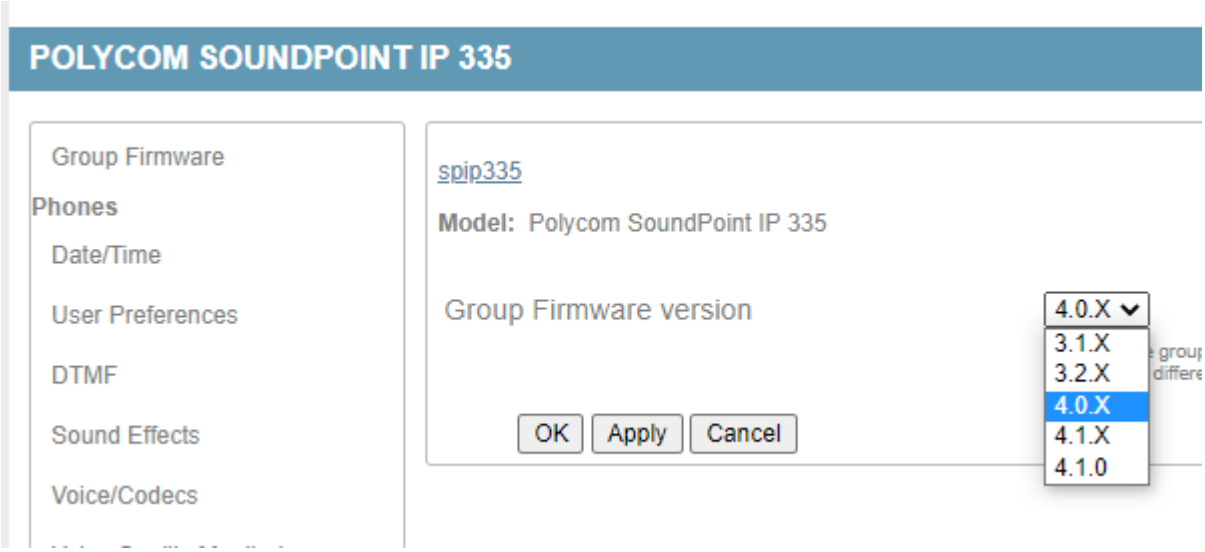
```
# ls -l /var/sipxdata/configserver/phone/profile/tftproot/polycom
total 8
drwxr-xr-x 5 sipx sipx 4096 Oct  2 10:02 4.0.X
drwxr-xr-x 5 sipx sipx 4096 Oct  2 10:02 5.5.2
```

You can load any version, but if there is a difference document that in the description field. If there is a large difference there will likely be missing features or configuration options. Try to stay in the ballpark (3.2.x, 4.0.x, 5.x) if possible. In the screenshot below I have version 5.9.6 firmware loaded into the 5.5.2 slot.

This label must match in the phone profile.



There is also a setting for this at the phone group level.

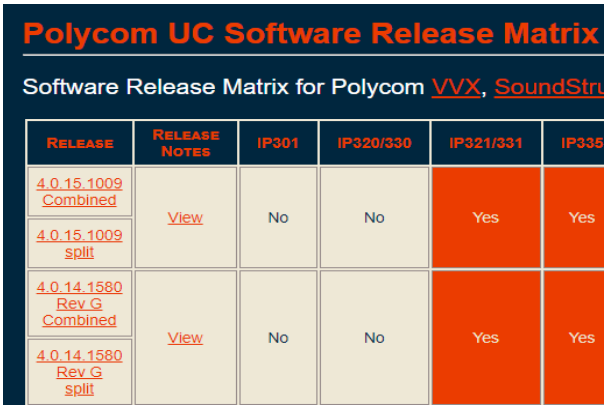


Note: Be cautious of conflicting group firmware versions if the phone is a member of multiple phone groups. When troubleshooting it is a good idea to remove the phone from all phone groups and just use the phone level setting so there is no doubt what it will download.

The bootrom is only required for special circumstances. For example, when upgrading SPIP phones between version 3.2.x and version 4.0.x a special upgrader bootrom is required. A separate downgrader bootrom may be required for the inverse from 4.0.x to 3.2.x. Check the release notes of the firmware version you're interested in for instructions.

Warning: Using a incompatible version firmware application archive or bootrom will result with the phone being stuck in a reboot loop.

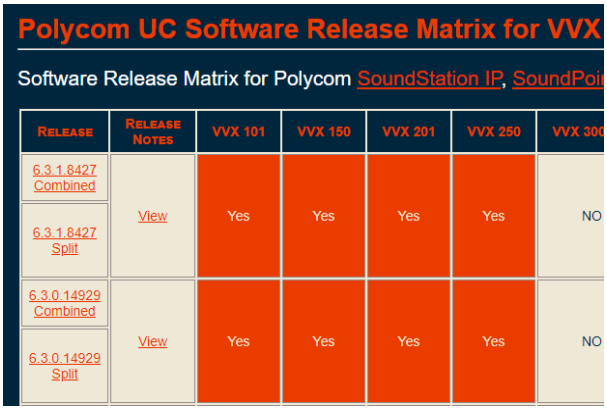
Note: On the Polycom matrix page, the release notes link should be just to the right of the version.



Polycom UC Software Release Matrix

Software Release Matrix for Polycom VVX, SoundStru

Release	Release Notes	IP301	IP320/330	IP321/331	IP335
4.0.15.1009 Combined 4.0.15.1009 split	View	No	No	Yes	Yes
4.0.14.1580 Rev G Combined 4.0.14.1580 Rev G split	View	No	No	Yes	Yes



Polycom UC Software Release Matrix for VVX

Software Release Matrix for Polycom SoundStation IP, SoundPoi

Release	Release Notes	VVX 101	VVX 150	VVX 201	VVX 250	VVX 300
6.3.1.8427 Combined 6.3.1.8427 Split	View	Yes	Yes	Yes	Yes	NO
6.3.0.14929 Combined 6.3.0.14929 Split	View	Yes	Yes	Yes	Yes	NO

Example custom configuration files

If a needed feature or configuration option is missing from the device profile a custom configuration file may be a possible workaround. For example:

`This file sets blind transfer as the default transfer method on Polycom SPIP and VVX phones.` Blind should always work, consultative (also known as attended) transfers have limitations. For example, **you cannot consultative transfer to a voicemail or conference target (anything freeswitch), and possibly PSTN targets (depending on your pstn gateway)**.

`This file removes 100rel support from a Polycom SPIP or VVX phone.` This would prevent the phone from responding to PRACKs.

Building a custom configuration file

The Polycom UC Software Administrator Guide describes all the options available and what template should be used.

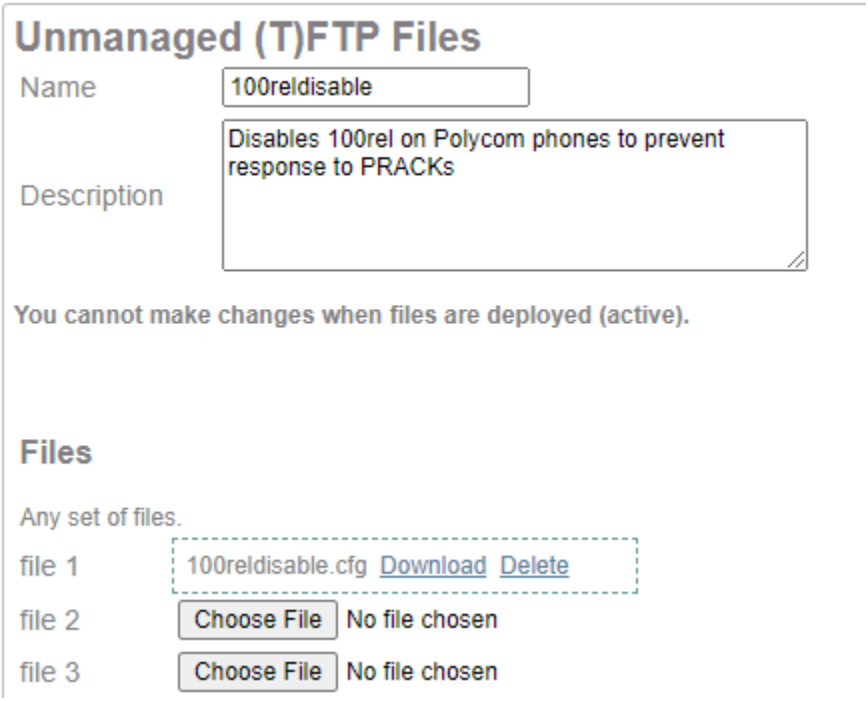
The templates can be found on the server filesystem after you've uploaded the application zip:

```
# ls -l /var/sipxdata/configserver/phone/profile/tftproot/polycom/4.0.X/Config/
total 3764
-rw-r--r-- 1 sipx sipx    2322 Oct  2 10:02 applications.cfg
-rw-r--r-- 1 sipx sipx   13927 Oct  2 10:02 device.cfg
-rw-r--r-- 1 sipx sipx   22823 Oct  2 10:02 features.cfg
-rw-r--r-- 1 sipx sipx    1162 Oct  2 10:02 H323.cfg
-rw-r--r-- 1 sipx sipx 3605355 Oct  2 10:02 polycomConfig.xsd
-rw-r--r-- 1 sipx sipx    9393 Oct  2 10:02 reg-advanced.cfg
-rw-r--r-- 1 sipx sipx     529 Oct  2 10:02 reg-basic.cfg
-rw-r--r-- 1 sipx sipx   31638 Oct  2 10:02 region.cfg
-rw-r--r-- 1 sipx sipx     739 Oct  2 10:02 sip-basic.cfg
-rw-r--r-- 1 sipx sipx   21262 Oct  2 10:02 sip-interop.cfg
-rw-r--r-- 1 sipx sipx  104003 Oct  2 10:02 site.cfg
-rw-r--r-- 1 sipx sipx    5271 Oct  2 10:02 video.cfg
-rw-r--r-- 1 sipx sipx     505 Oct  2 10:02 video-integration.cfg
```

Uploading a custom configuration file

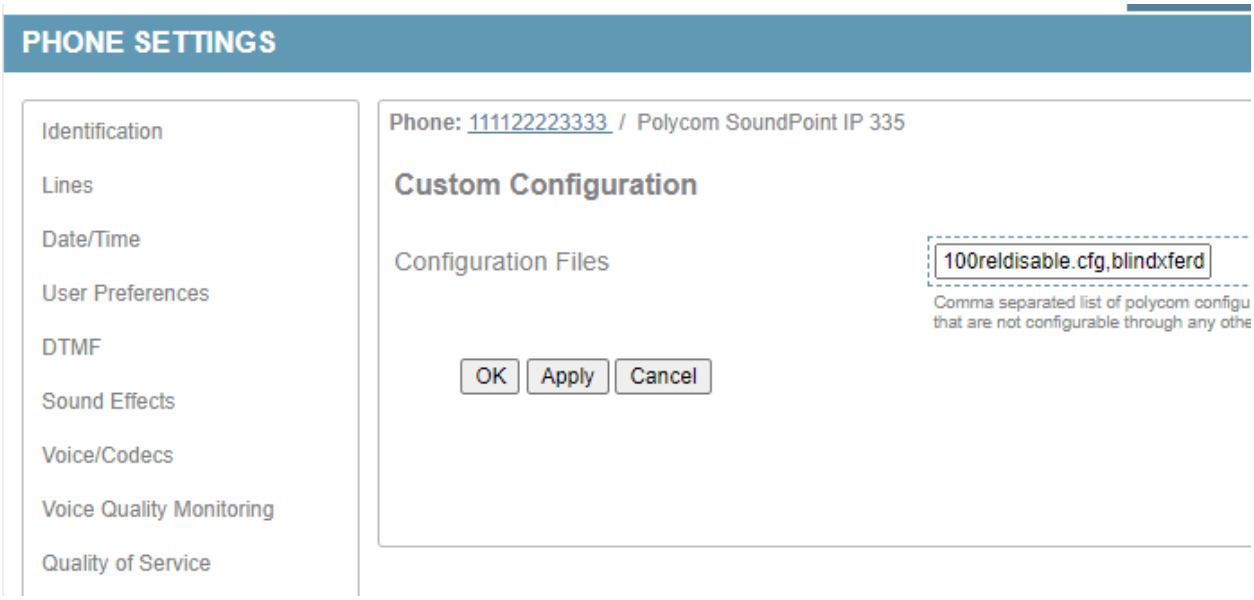
Use the "Unmanaged TFTP Files" option to upload the custom configuration file(s). You can upload multiple files in one, or create multiple individual unmanaged tftp files entries. Don't forget to provide a description as to what it is

supposed to do.



Adding the custom config to a phone

Navigate to devices - phones, select the phone, then select the 'custom configuration' tab in the phone profile. You can also set this at the phone group level. Type the filename exactly (case sensitive) as it exists on the filesystem. If you have multiple custom config files, use a comma with no space between filenames. Apply to save, then send profiles to the phone.



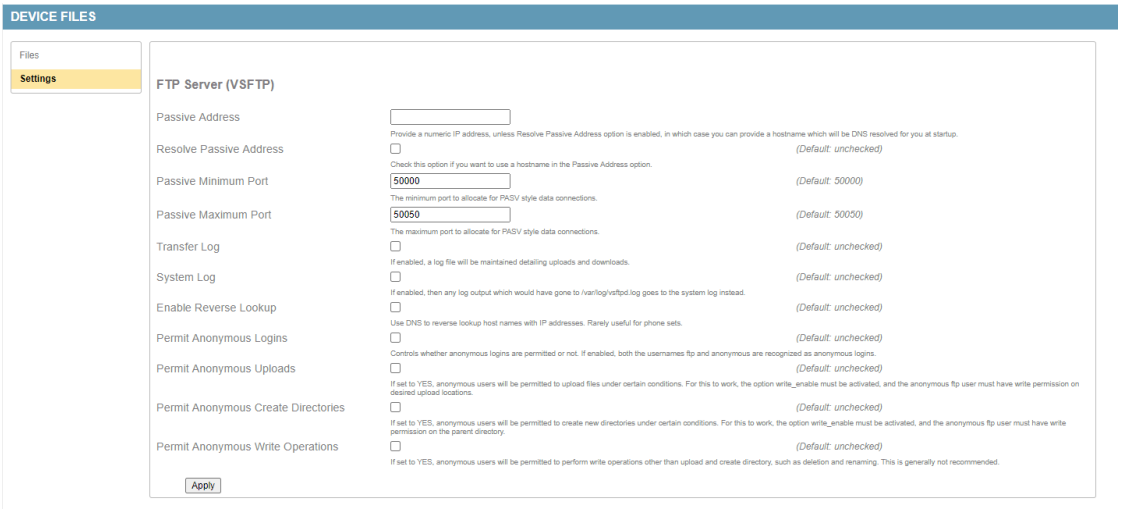
Upon sending profiles to the phone the custom configuration files are added to the $mac.cfg

```
# grep "CONFIG_FILES" /var/sipxdata/configserver/phone/profile/tftproot/111122223333.
↪cfg
  CONFIG_FILES="100reldisable.cfg,blindxferdefault.cfg,[PHONE_MAC_ADDRESS]-sipx-
↪applications.cfg,[PHONE_MAC_ADDRESS]-sipx-features.cfg,[PHONE_MAC_ADDRESS]-sipx-reg-
↪advanced.cfg,[PHONE_MAC_ADDRESS]-sipx-region.cfg,[PHONE_MAC_ADDRESS]-sipx-sip-basic.
↪cfg,[PHONE_MAC_ADDRESS]-sipx-sip-interop.cfg,[PHONE_MAC_ADDRESS]-sipx-site.cfg,
↪[PHONE_MAC_ADDRESS]-sipx-video.cfg"
```

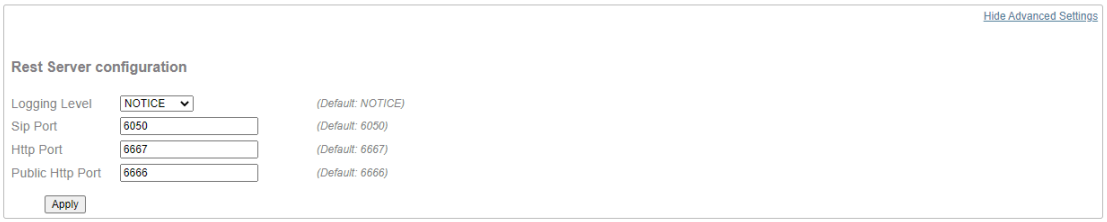
Note: The phone must be configured (possibly manually) to download from the server. Polycom phones use DHCP option 66 (tftp:// or ftp://), or option 160 (http:// or https://) for the provisioning server address. If both are specified option 160 is preferred. It's also a good idea to specify option 42 for NTP servers.

The Settings Tab

The settings tab allows fine tuning of the FTP (vsftpd) service.

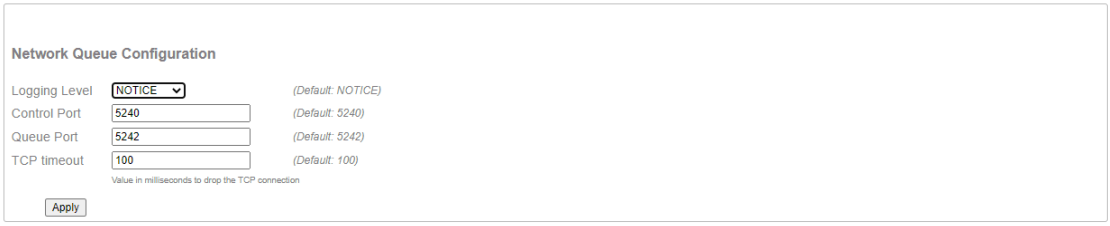


Rest Server



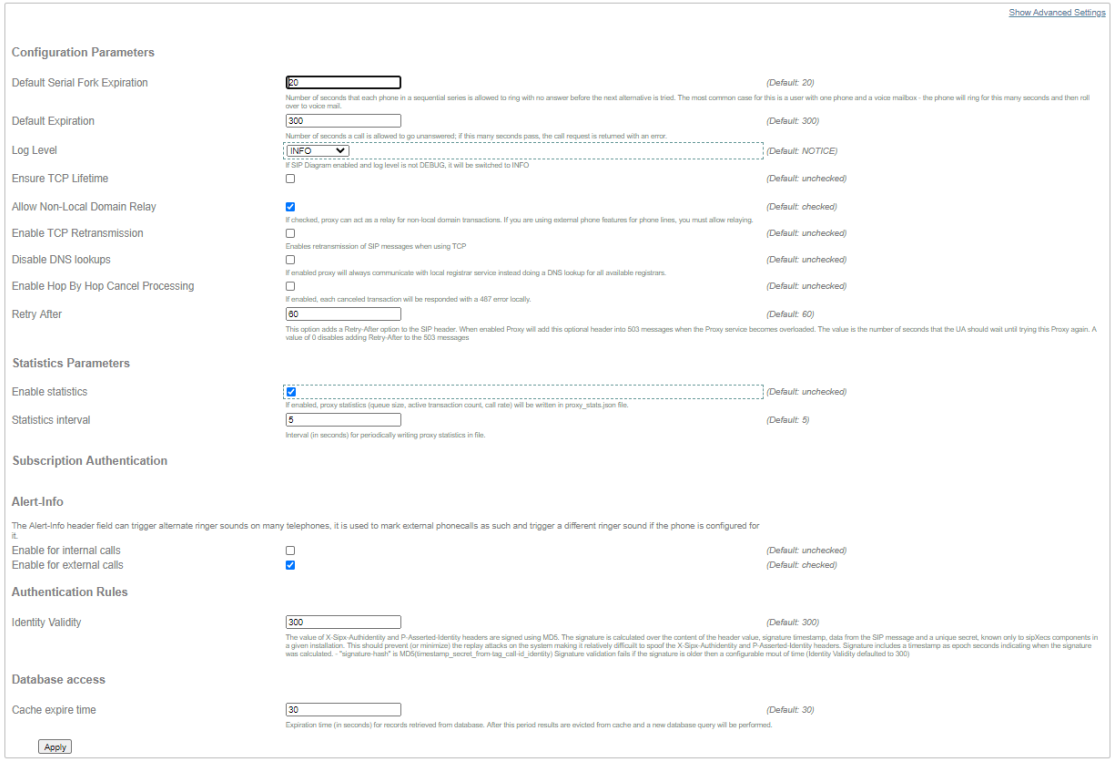
SAA/BLA

Service Msg Queue



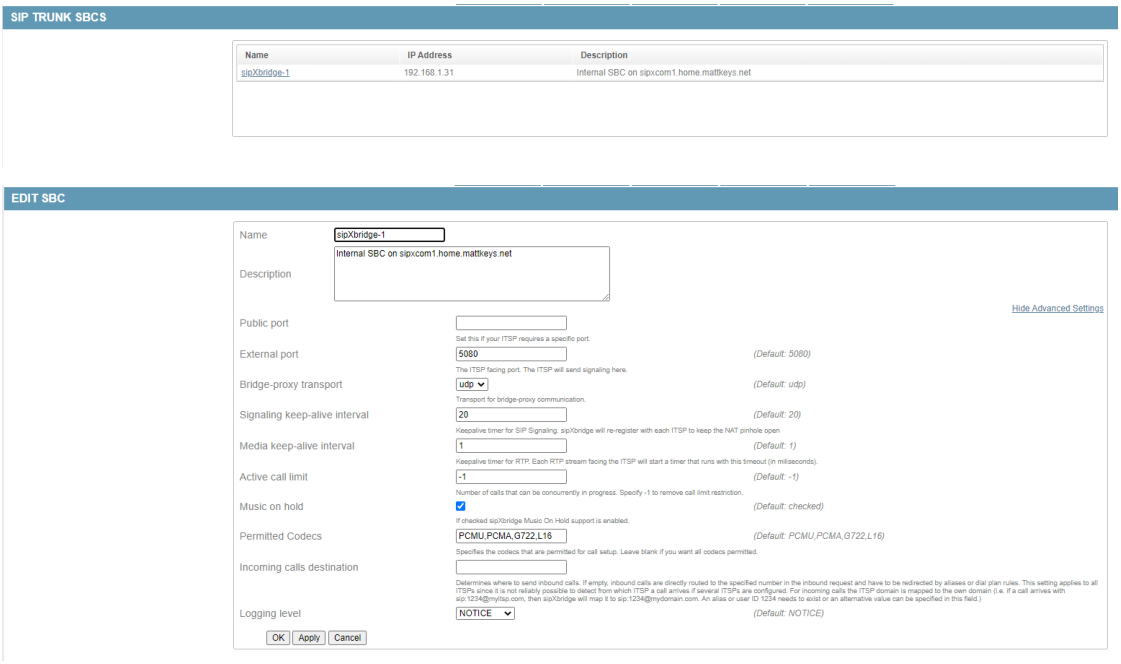
RLS

SIP Proxy

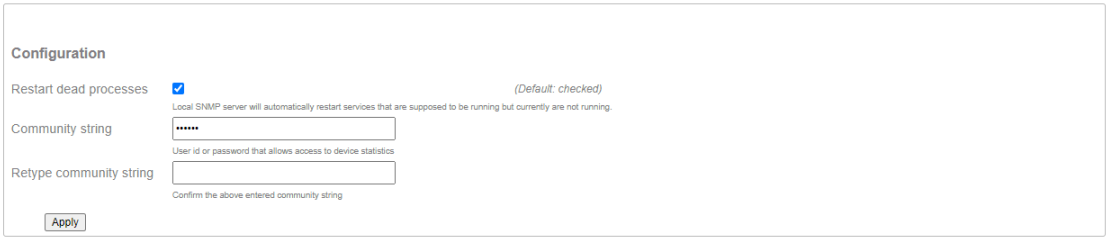


SIP Registrar

SIP Trunk



SNMP



Voicemail

7.5.10 Settings

The Settings menu includes Admin, Authentication, Date and Time, DID Pool, Domain, Extension Pool, Internet Calling, Localization, Locations, NAT Traversal, Permissions, and Regions menu options.

Admin

Authentication

LDAP/Active Directory is the only menu item.

LDAP/Active Directory

This page is used to manage (read only) Lightweight Directory Access Protocol (LDAP) or Microsoft Active Directory (AD) connections. There are two tabs to the left, Configuration and Management Settings.

Values which can be imported are listed in the following table along with the recommended AD attribute.

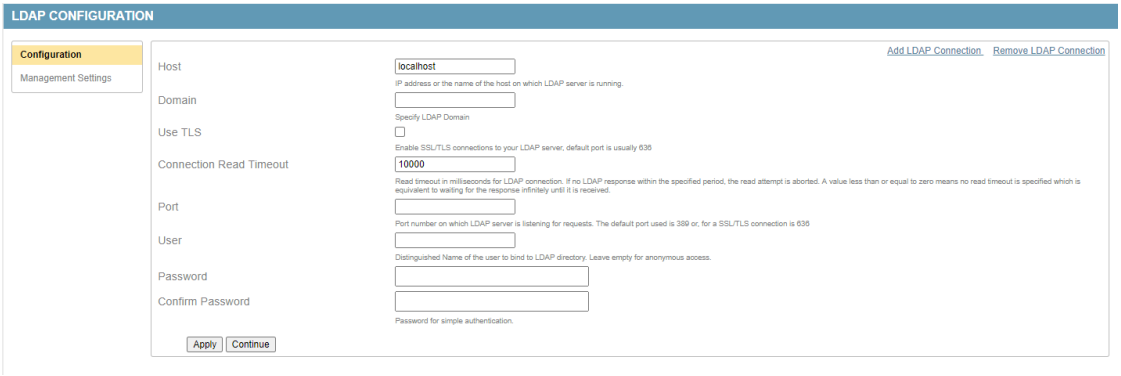
Value	Description
User ID	Represents the user ID. The value **must** be unique.
First name	The first name of the user.
Last name	The last name of the user.
Alias	If this has more than one value a separate alias will be created for each. You can map multiple LDAP attr
Email address	If this has more than one value a separate alias will be created for each.
User Groups	If this has more than one value the user will be added to multiple groups. Groups are created as necessary
Voicemail PIN	The user PIN code to access voicemail. When blank imported users are assigned a default PIN.
Default PIN	The default (voicemail) PIN code.
Confirm Default PIN	Confirmation of the above.
SIP Password	If blank or not mapped sipxcom will automatically generate a random SIP password for each imported us
IM ID	The instant message (IM) ID.
Job Title	The user job title. The value is saved under the user contact information.
Department	The user job department.
Company Name	The name of the company.

Table 1 -

Assistant Name	The user assistant or secretary name.
Mobile Phone	The user mobile phone number.
Home Phone Number	The user home phone number.
Assistant phone number	The assistant or secretary phone number.
Fax Number	The users fax number.
Alternate email	Alternative email addresses for the user.
Alternate IM Account	Alternative IM accounts
Location	The user location.
Home Address	
Street	
City	
State	
Country	
Zip Code	
Office Address	
Street	
City	
State	
Country	
Zip Code	

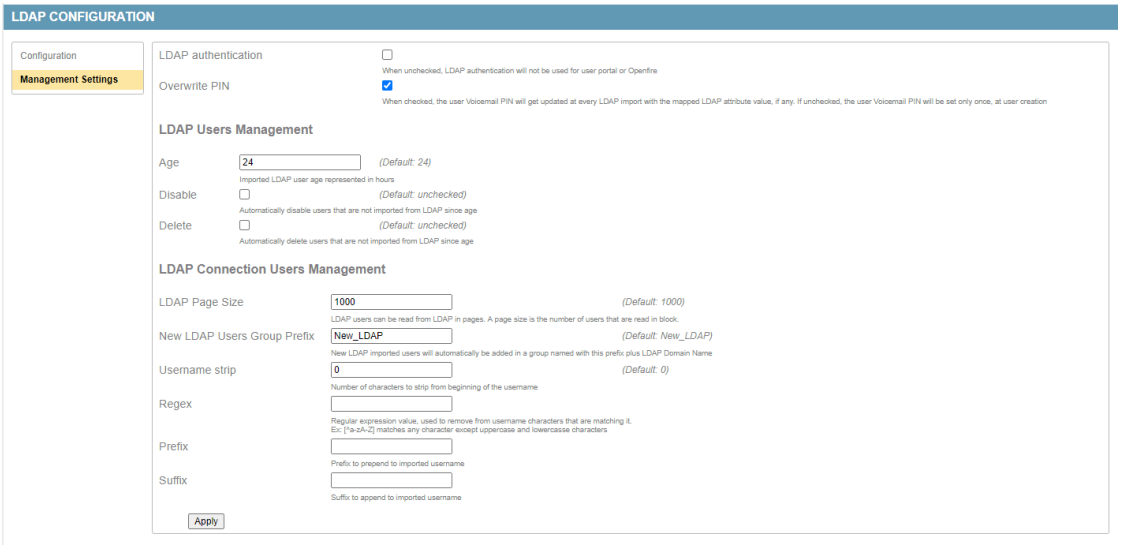
Configuration

As a good security practice you should create a user in LDAP or AD with read only permissions for the sole purpose of syncing data to sipxcom. It is also a good idea to keep service or admin level accounts in their own OU.



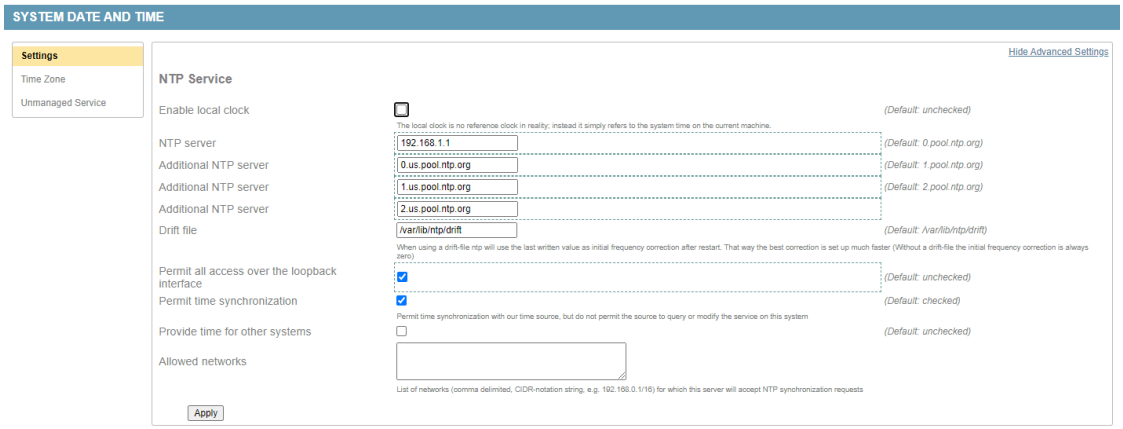
- **Host**: Enter the IP address or fqdn of the server running LDAP/AD services.
- **Domain**: This specifies the user domain. The value is saved as a user setting and can be used for sipxcom web portal authentication (as user@domain or domain\user as the username).
- **Use TLS**: Enable or disable SSL/TLS connections to your LDAP/AD server (ldaps://).
- **Connection Read Timeout**: If there is no response from the LDAP server within the specified period, the read attempt is aborted. A value less than or equal to 0 will disable the read timeout and wait indefinately. The default value is 10 seconds.
- **Port**: The port number on which the LDAP/AD server is listening. The default port for LDAP is 389 or 636 for LDAPS.
- **User/Password/Confirm Password**: Credentials of the read only user that was created for the purpose of ldap/ad sync.

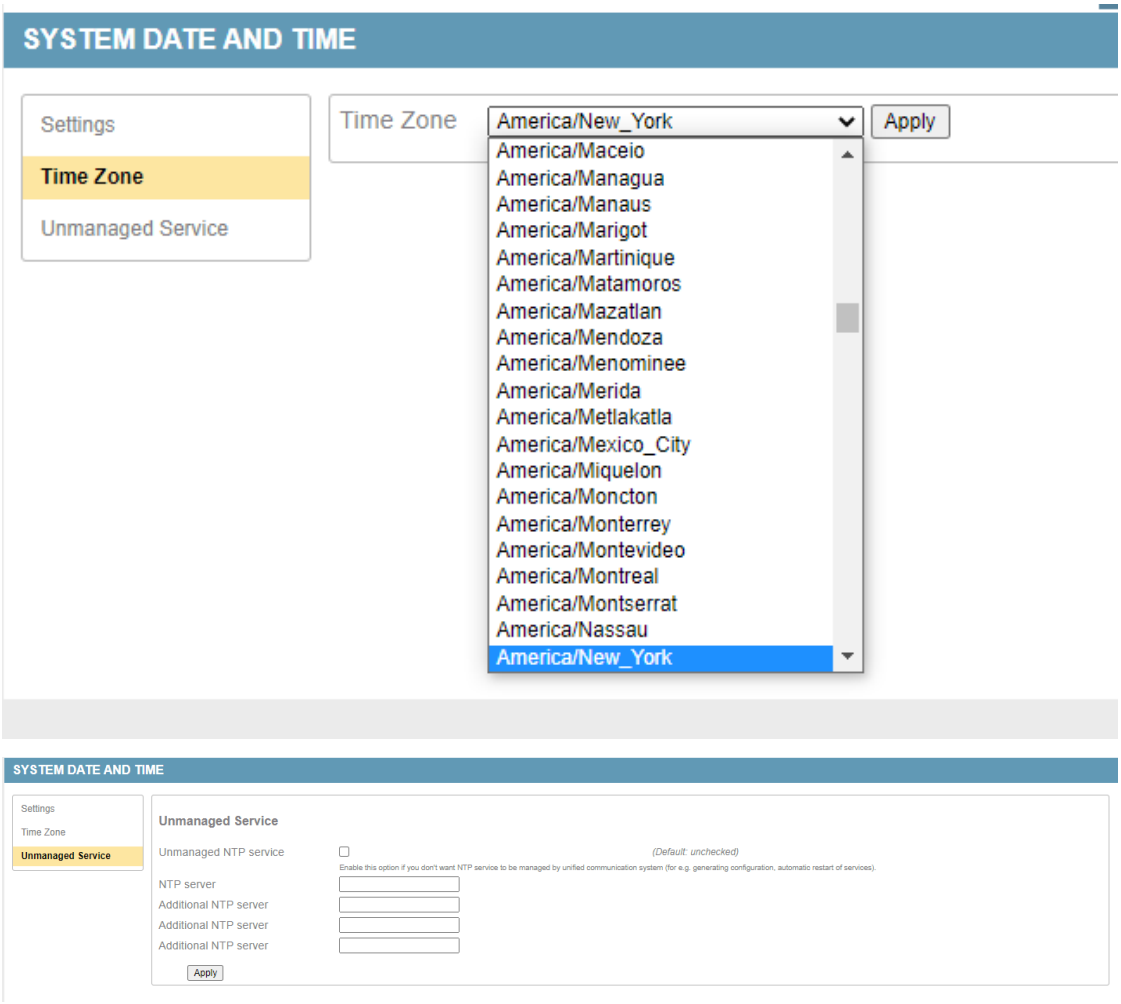
Management Settings



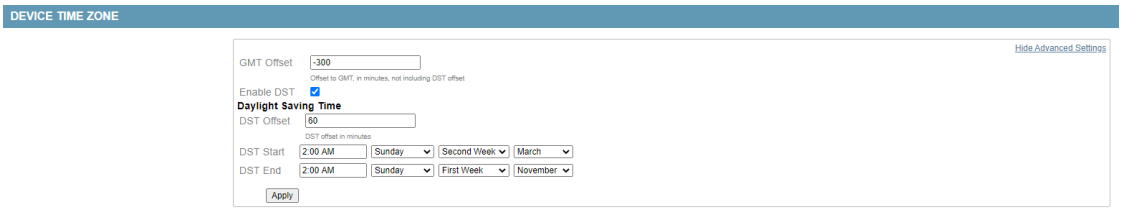
Date and Time

Date and Time is the NTP service configuration. There are three tabs to the left, Settings, Time Zone, and Unmanaged Service.

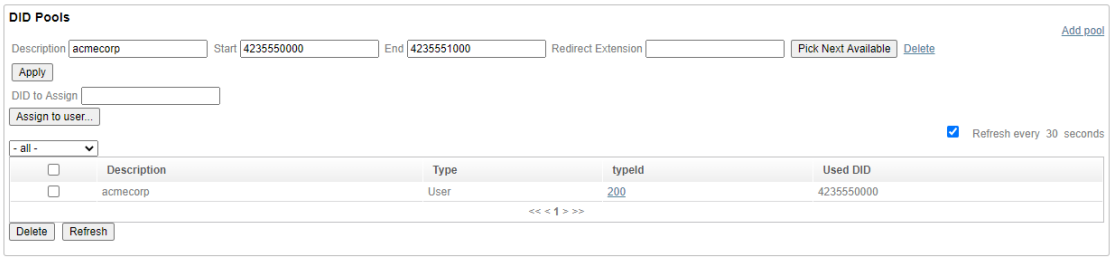




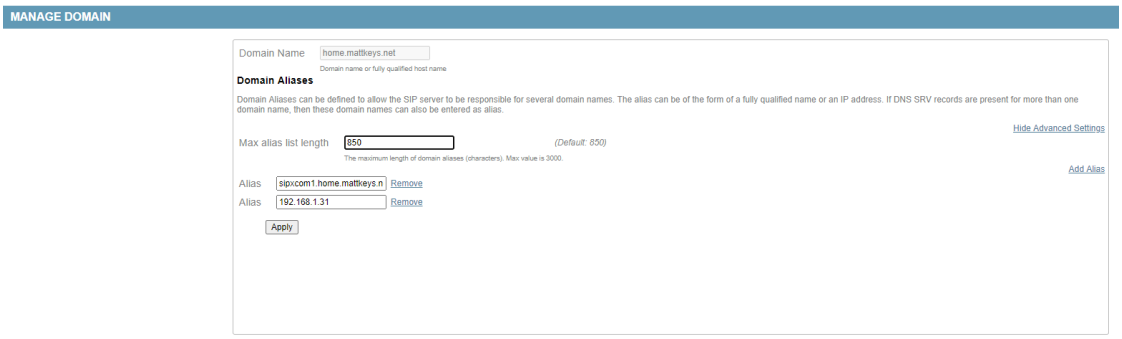
Device Time Zone



DID Pool

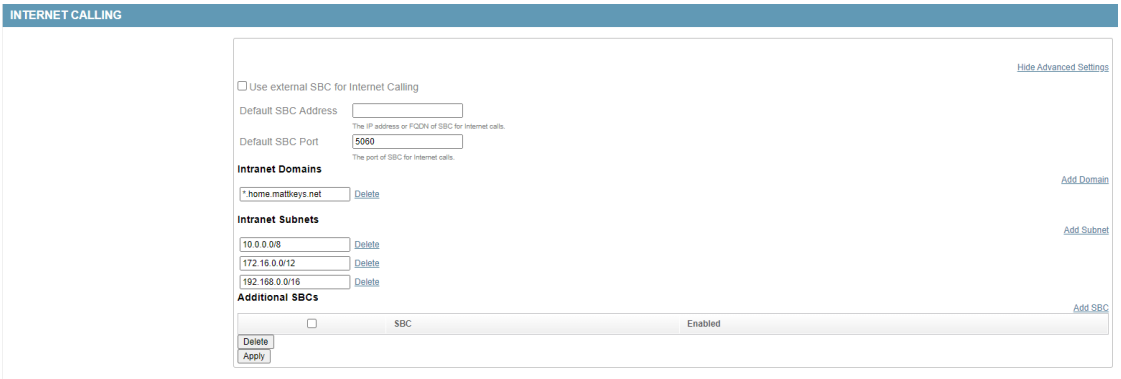


Domain

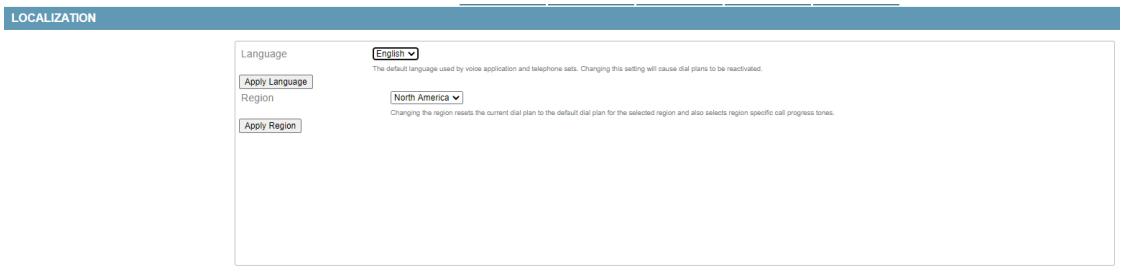


Extension Pool

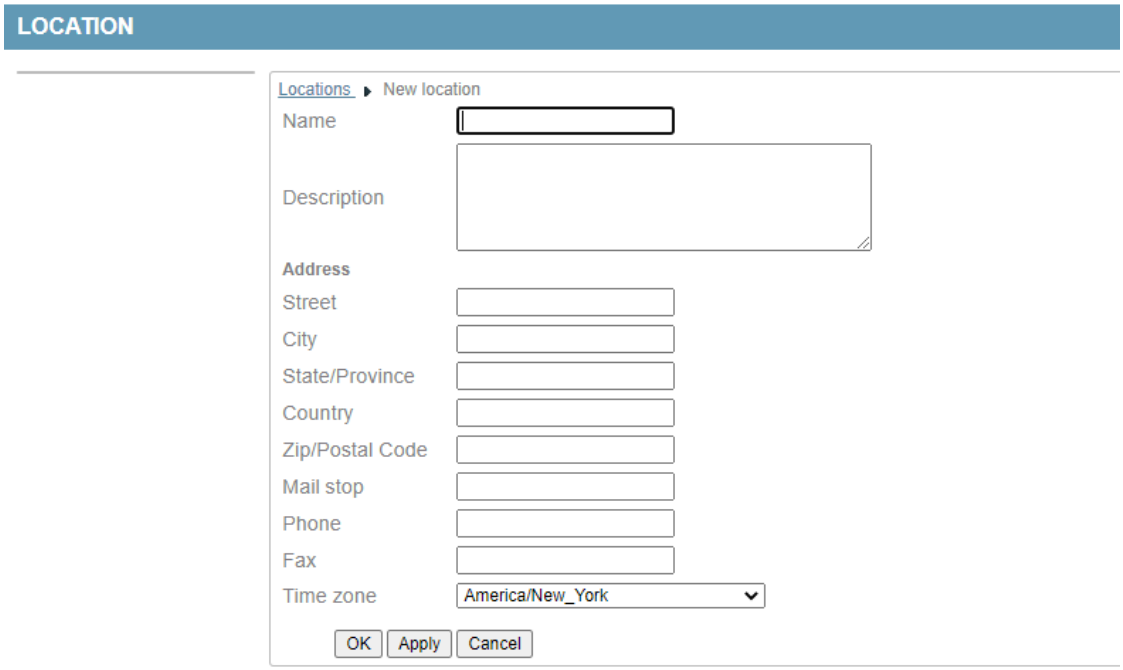
Internet Calling



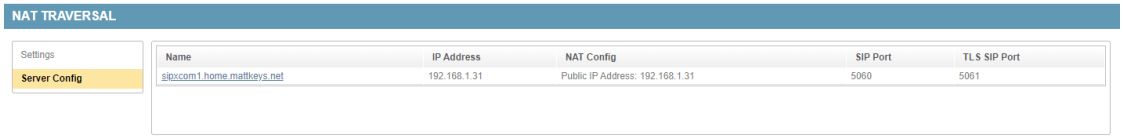
Localization

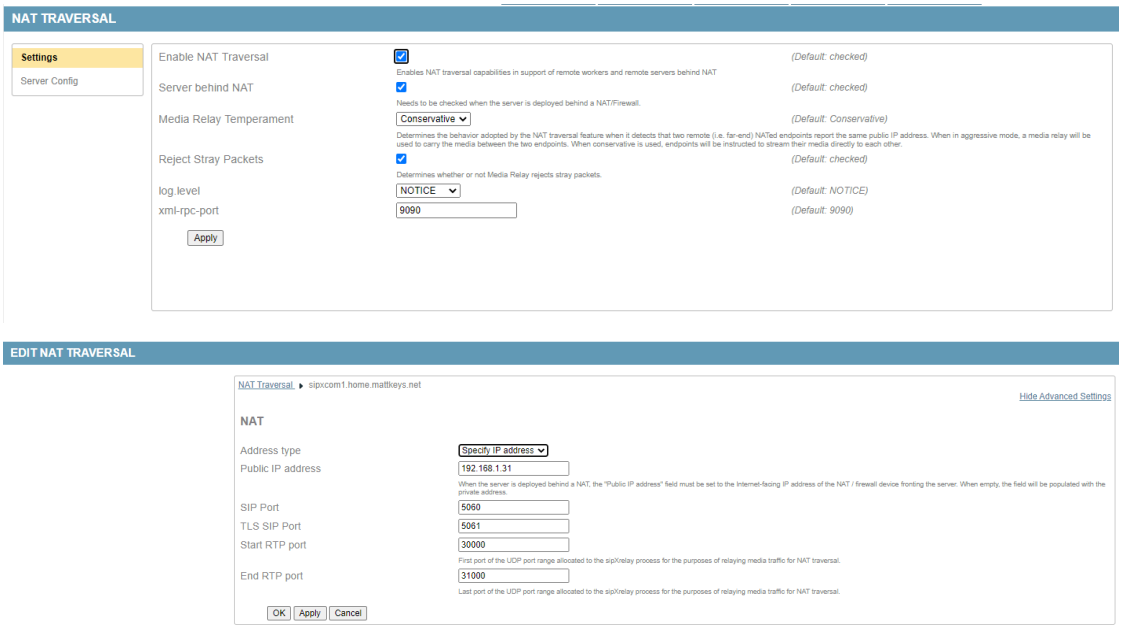


Locations

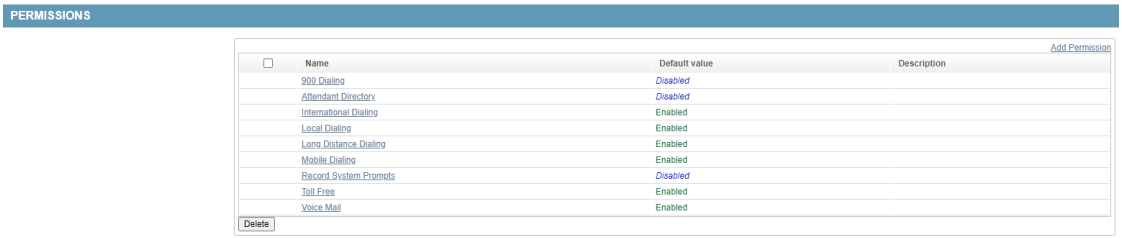


NAT Traversal





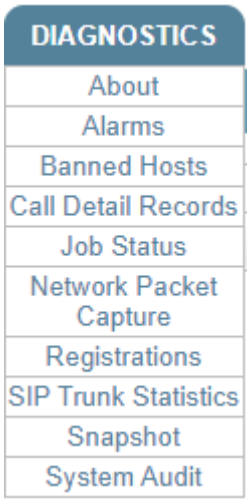
Permissions



Regions

Regions are used to organize servers into groups based on your network topology. Servers with the same region are generally located on the same LAN and have very low latency between the servers. Regions are used to determine how local databases and DNS is configured.

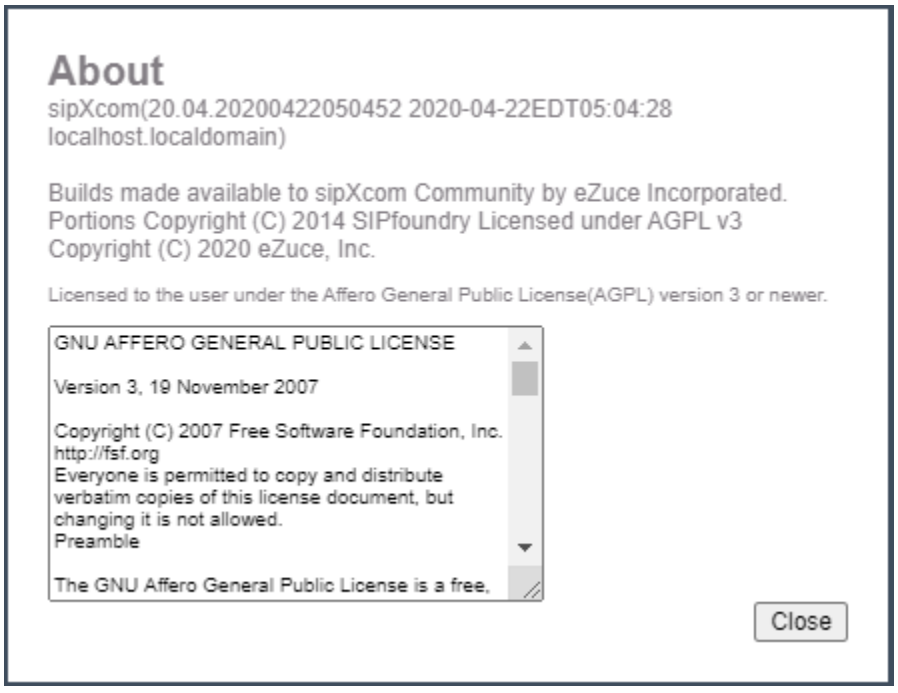
7.6 Diagnostics Tab



The diagnostics tab includes About, Alarms, Banned Hosts, Call Detail Records, Job Status, Network Packet Capture, Registrations, SIP Trunk Statistics, Snapshot, and System Audit menu options.

7.6.1 About

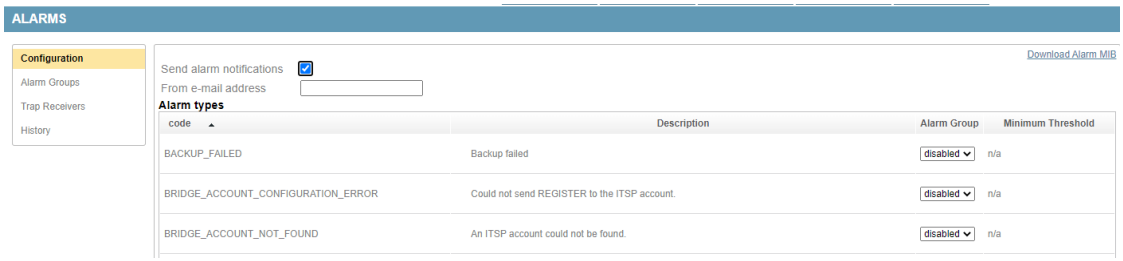
The About option displays the current version and license information.



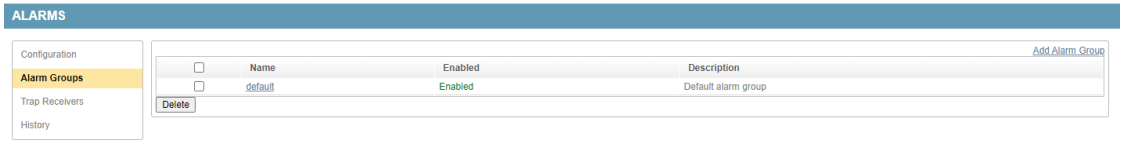
7.6.2 Alarms

The Alarms page has four tabs to the left - Configuration, Alarm Groups, Trap Receivers, and History.

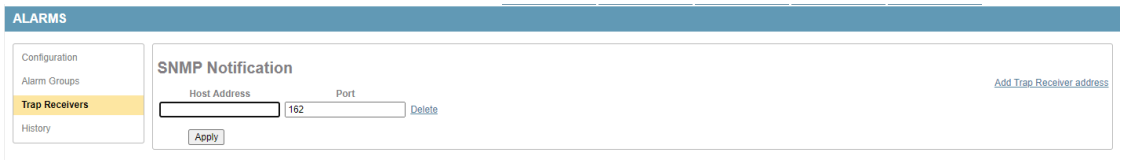
Configuration



Alarm Groups



Trap Receivers



History

7.6.3 Banned Hosts

This page displays IPs that have been banned by the SIP Security rules. You can also unban hosts from this page.

7.6.4 Call Detail Records

The Call Detail Records page has three tabs to the left - Active, Historic, and Reports.

Active

Historic

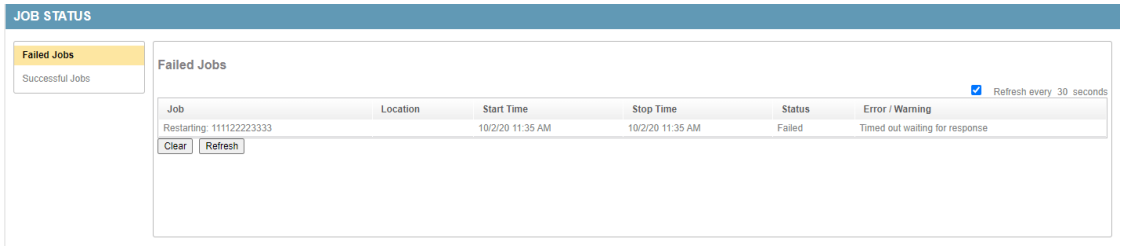
Reports

7.6.5 Job Status

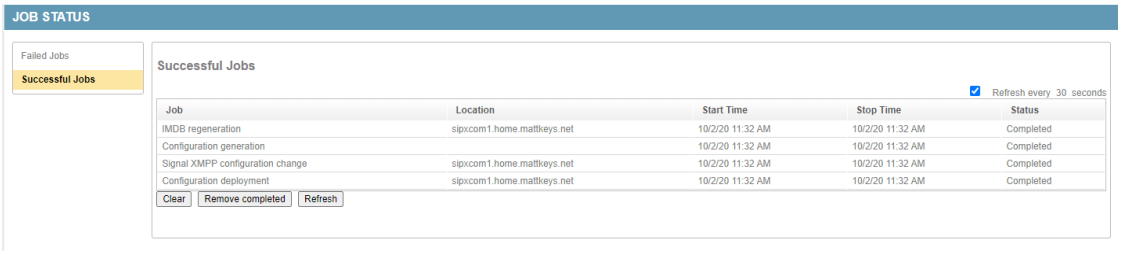
There are two tabs to the left, Failed and Successful jobs.

Failed Jobs

Often sending profiles to a phone that is not currently registered, powered off, or disconnected from the network will trigger an entry here.



Successful Jobs

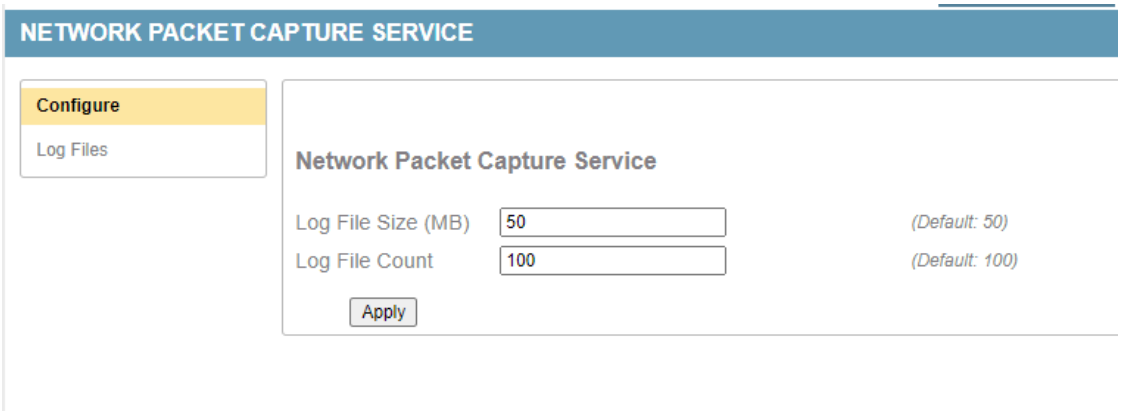


7.6.6 Network Packet Capture

Network Packet Capture can be used to automate rolling packet captures on all servers in the cluster.

Configure

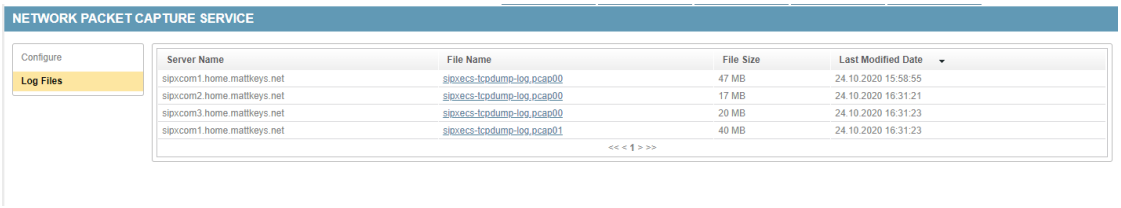
Configure the size of each pcap file and number of pcaps to keep.



Warning: **If you run out of free disk space all services will halt!** The default settings will consume 5GB (per server).

Log Files

The resulting pcap files are listed on this page for download.



7.6.7 Registrations

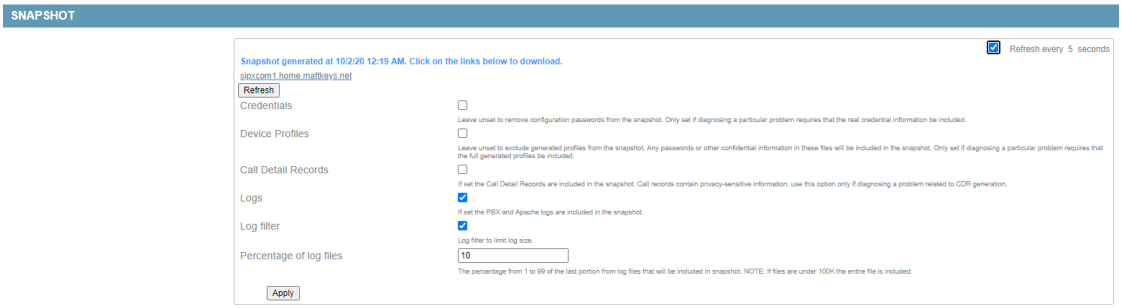
This page shows all current SIP registrations.

7.6.8 SIP Trunk Statistics

This page shows the current status of any SIP trunks that are using the sipxbridge service.

7.6.9 Snapshot

Snapshot archives contain the logs and configuration data needed to troubleshoot without having access to the command line of the server.



Note: The snapshot log filter percentage can only cover the current day logs. 99% will grab everything for today but will also increase the size of the resulting archive.

On Windows you will need a utility installed that can extract tar or gzipped archives. We recommend 7-zip . Notepad++ is recommended for viewing log data rather than Notepad, WordPad, or Word. WinMerge is recommended for side-by-side file comparisons.

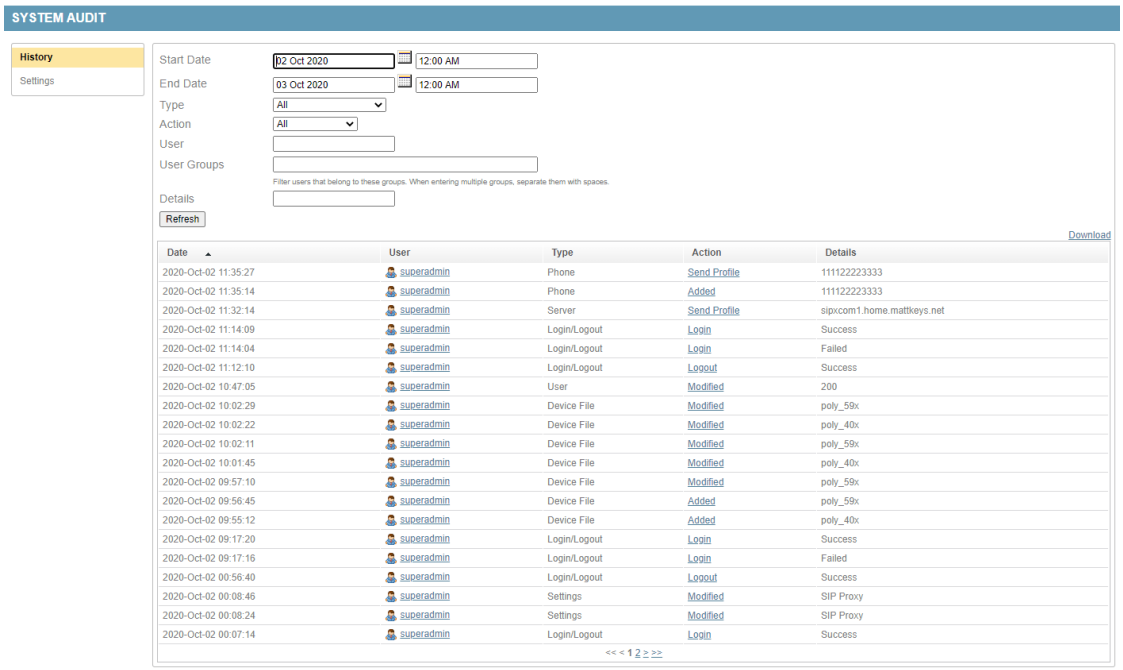
On Mac or Linux copy the archive to a location on the filesystem and use the following command:

```
tar -zxvf sipx-snapshot-host.domain.tar.gz
```

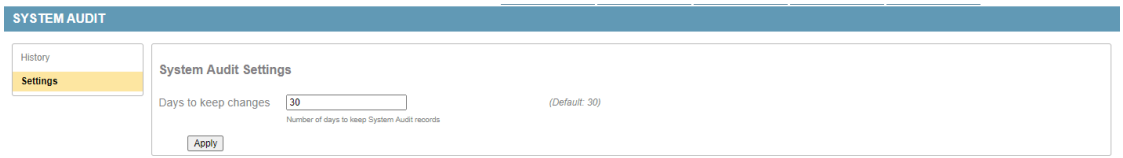
7.6.10 System Audit

System Audit keeps track of changes made in the webui, when the change was made, and the user that made the change. There are two tabs to the left, History and Settings.

History



Settings



CHAPTER

EIGHT

TROUBLESHOOTING

If you can't reproduce the problem it probably isn't worth troubleshooting it. Test multiple times, and from (or to) multiple locations.

For example, internal (extension to extension) calls only traverse the proxy/registrar and do not involve PSTN gateways. If a internal ext to ext call is working correctly, but a internal ext to external PSTN number is not working, the problem is likely the PSTN gateway rather than sipxcom.

Another example might be if the presence status is not updating for contacts on your phone, but other phones are fine. Rebooting the phone will force it to renew any line registrations and presence subscriptions. Try that first.

Narrow down where the problem may be by following a process of elimination.

8.1 Detailed Problem Description

Try to provide as much detail as possible. The more information the support team has to work with, the less likely they will have to ask for more information.

For example, instead of stating:

Bob, on extension 204, can't transfer a call to Sally on extension 205.

Provide significantly more detail by stating something like:

Bob (extension 204) received a call from 555-321-1234 at approximately 10:30 AM EST, which he then answered. Bob attempted to perform a consultative transfer to Sally (extension 205) but was prompted on her handset, "Cannot Complete Transfer."

Note how the time that the incident occurred and the transfer type are important. This helps the support team track down the issue.

Warning: There is a signaling difference between a blind transfer and a consultative/attended transfer. Blind transfer should work in every call scenario. Consultative transfer has limitations. For example, you cannot consultative transfer to a voicemail box or conference extension. **We strongly recommend using blind transfer by default.** Not all gateways can handle REFERs used in a consultative transfer, and PRACKs can cause audio issues. See the *Example custom configuration files* section for a custom config that sets blind transfer as default, or to disable phone response to PRACKs.

Note: "If no response" call forwards are similar to blind transfer. The call is not forked. "At the same time" call forwards are similar to consultative transfer. The call is forked. **We strongly recommend using 'if no response' instead of 'at the same time'.** Again not all gateways can handle REFERs, and PRACKs can cause audio issues.

8.2 Snapshot Covering the Time of the Incident

The following steps will ensure a good quality snapshot for support team to work with.

- Turn up logging to INFO for the proxy service
- Reproduce the problem
- Take a snapshot as soon as possible after

See the *Snapshot* section.

Note: The snapshot archive(s) are saved locally beneath the primary (webui) server /var/sipxdata/tmp/.

8.3 Call Packet Capture

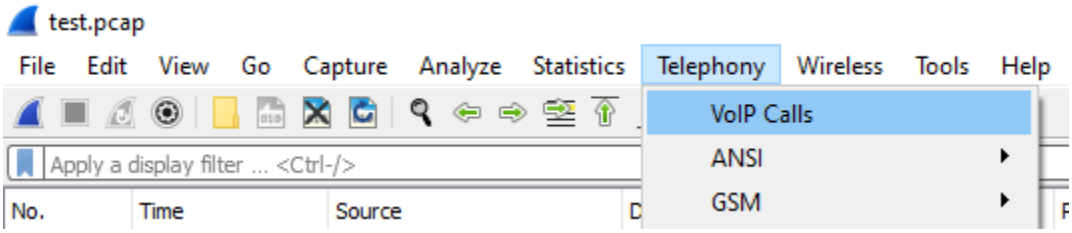
See *Utility Services* for automated packet capture. This is resource intensive, so it should only be enabled while troubleshooting. Another option is initiate the packet capture on the command line. Just before you begin to reproduce the issue, issue as root (on all servers running the proxy service):

```
tcpdump -s0 -n -i any -w ~/server1_example1.pcap
```

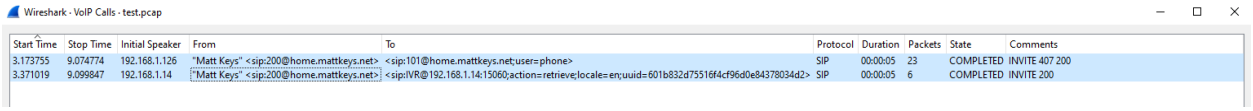
Notice the server name is in the filename so it can be easily distinguished. The CDR records may help you gather information, such as From:, To:, date/time of the call, or the Call-ID if you exported the CSV from CDRs.

8.3.1 Using Wireshark to view packet captures

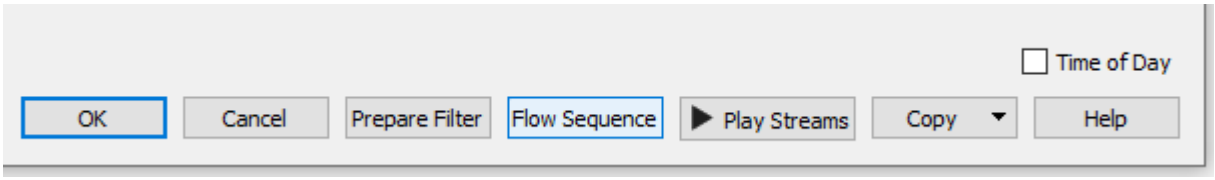
First open the packet capture file in Wireshark, then navigate to Telephony - Voip Calls.



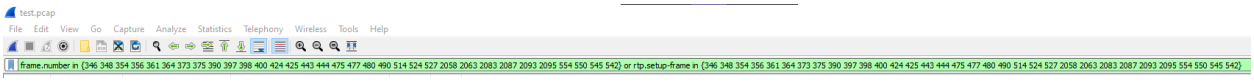
Calls within the packet capture are listed in the "Voip Calls" window. Hold down shift and click to highlight those you're interested in.



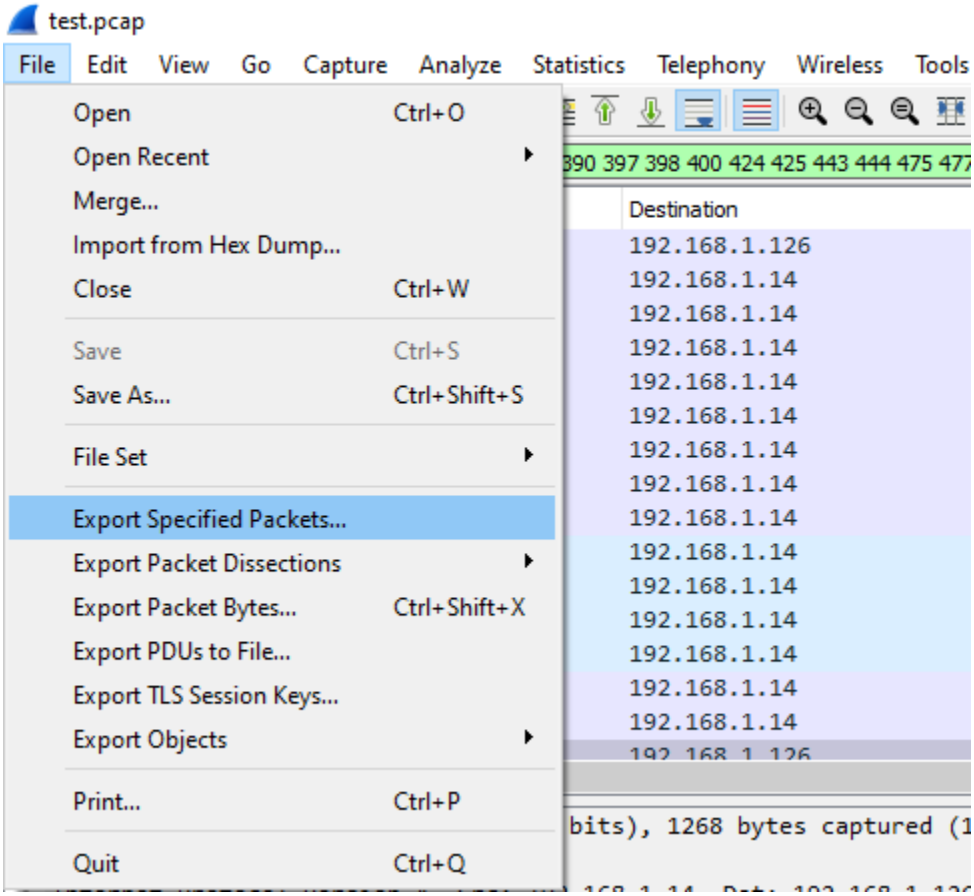
After highlighting calls you are interested in, options will become available at the bottom of the Voip Calls window.



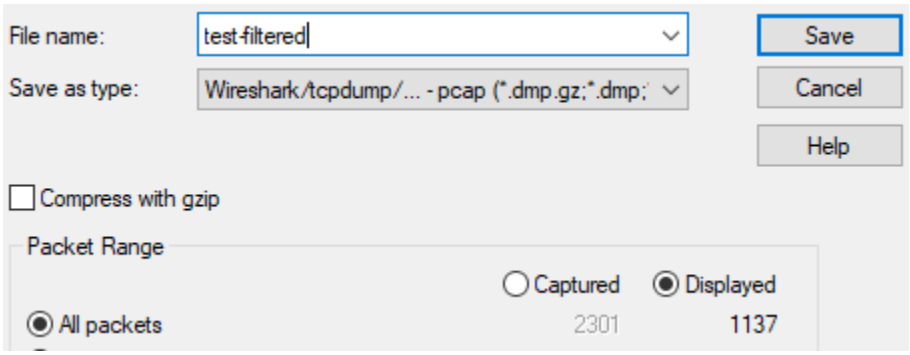
Prepare Filter will generate a filter in the main application window to display only SIP (and RTP packets if available) involved in the selected calls.



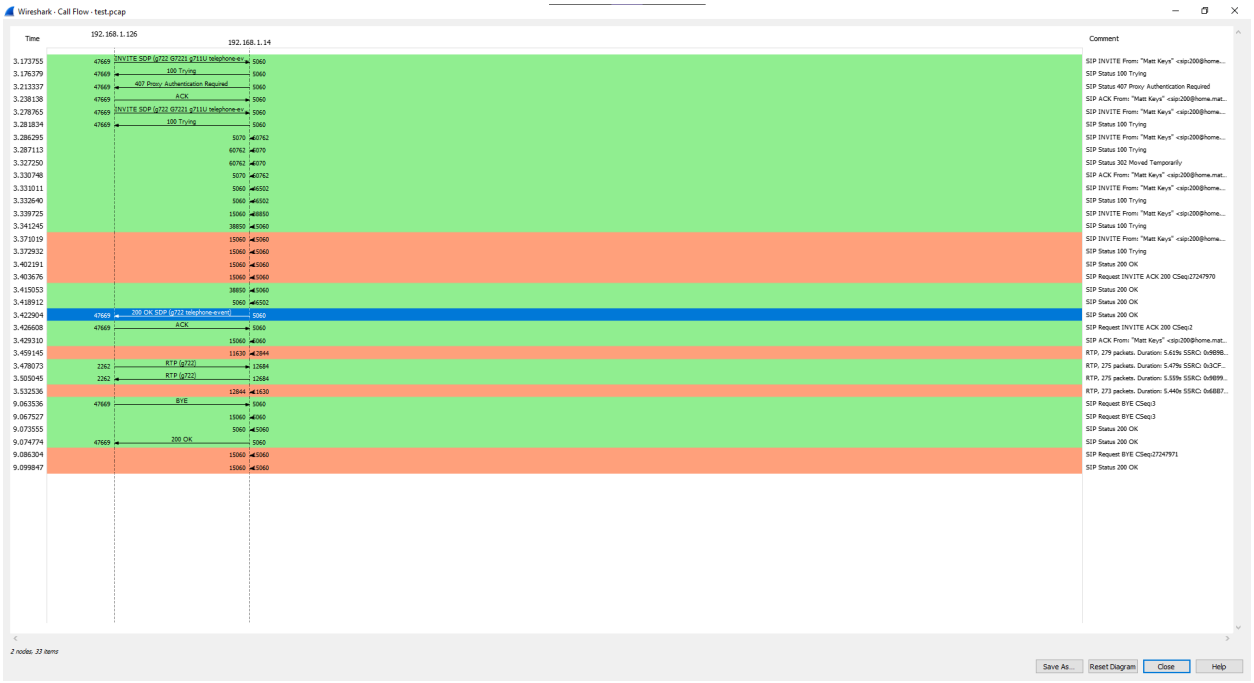
This can be used to create sanitized (smaller) exports. To export a sanitized pcap (after using **Prepare filter**), click File - Export Specified Packets.



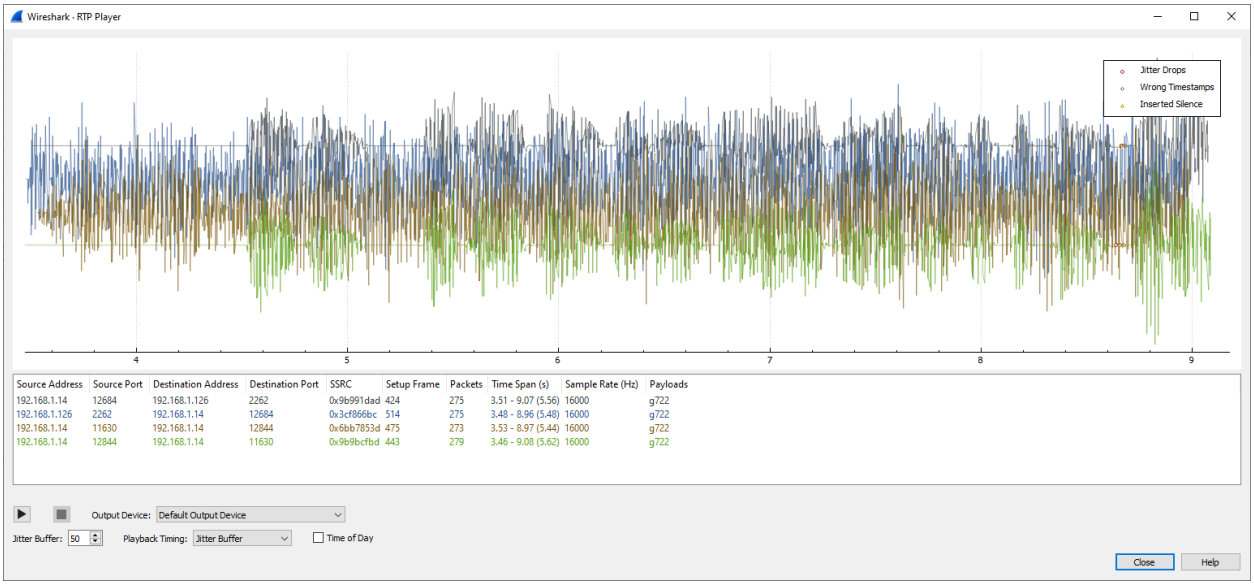
Next provide the output filename. The "Displayed" option should be selected. This exports only packets displayed in the main application window (after filtering).



- **Flow Sequence** will display the SIP ladder of the selected calls. Clicking any part of the ladder will move your placement in the main application window to that particular packet number.



- **Play Streams** is useful to view or listen to audio RTP involved in the call.



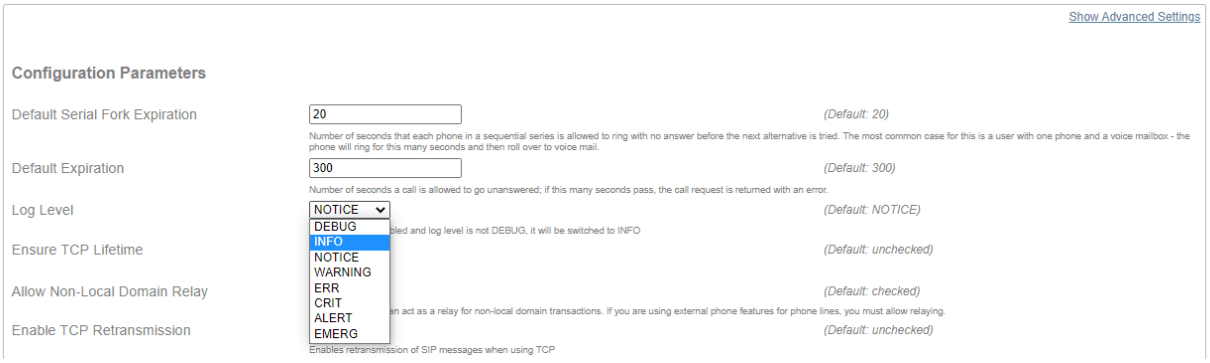
8.4 Logs

The first step is to gather logs. **All SIP signaling passes through the proxy service, but by default the log verbosity of the proxy service is set to NOTICE which does not display the SIP messages.**

Note: Each server has its own set of logs beneath /var/log/sipxpbx/. The sipXproxy.log on server1 will be different than on server2 or server3.

In order to see SIP messages in the sipXproxy.log file(s), increase the verbosity to INFO (or DEBUG) beneath System - Services - SIP Proxy - Log Level, then apply at the bottom of the page.

Note: This will restart the proxy service which may interrupt calls.



This will also enable use of the *sipcodes.sh* script, which is now included in the sipxcom (19.04 and above) rpms. It counts SIP messages in the logs, so log verbostiy must be at INFO or DEBUG to count them.

Note: Upon snapshot creation the sipcodes script is ran against the entire proxy log file. You'll always get the entire count (no matter what the log filter percentage was) in the snapshot as ./var/log/sipxpbx/sipcodes.log, given sipcodes had something to count (**proxy log must be at INFO or DEBUG**).

All sipxcom service logs are beneath /var/log/sipxpbx/. Other services such as apache2 (/var/log/httpd/), mongodb (/var/log/mongodb/), and postgresql (/var/lib/pgsql/data/pg_log/) logs are outside of that directory. A sipxcom snapshot is a handy way to grab everything that may be of use all at once.

Note: The logs are rotated every 24 hours. Rotated logs are renamed to be suffixed with the date, and may be compressed with gzip.

8.5 sipcodes.sh

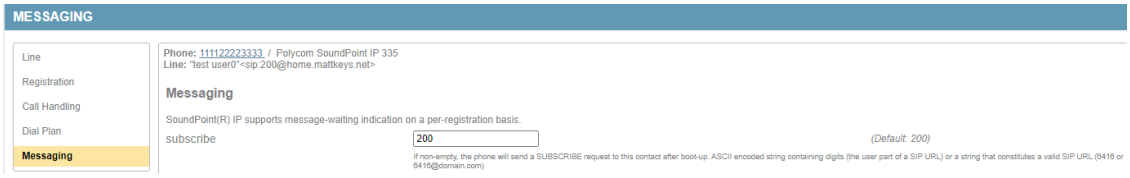
The sipcodes script (/usr/bin/sipcodes.sh) was developed to provide quick counts from the /var/log/sipxpbx/sipXproxy.log, or /var/log/sipxpbx/sipxbridge.log. The log verbosity of these services must be set to INFO or DEBUG in order to see and count the SIP signaling within the log.

A basic understanding of SIP signaling is required to understand and act upon the output of the script. In summary:

- A **REGISTER** is used to authenticate a SIP user. Within the **REGISTER** message, the **From:** header indicates the user attempting registration. The **Contact:** header provides the IP address of the phone or device sending the **REGISTER**. All registrations can be viewed beneath *Diagnostics - Registrations*, or for a single user in *Users Tab* - $user - Registrations.
- A **INVITE** is used when a user places a call. The **To:** header is the number dialed, the **From:** header is the user that sent it, and again the **Contact:** usually contains the IP of the device sending the message. INVITEs create entries in the *Call Detail Records*.
- A **SUBSCRIBE** is used when a SIP phone requests a feature on the server. Within the **SUBSCRIBE** message, the **Event:** header describes the service requested.

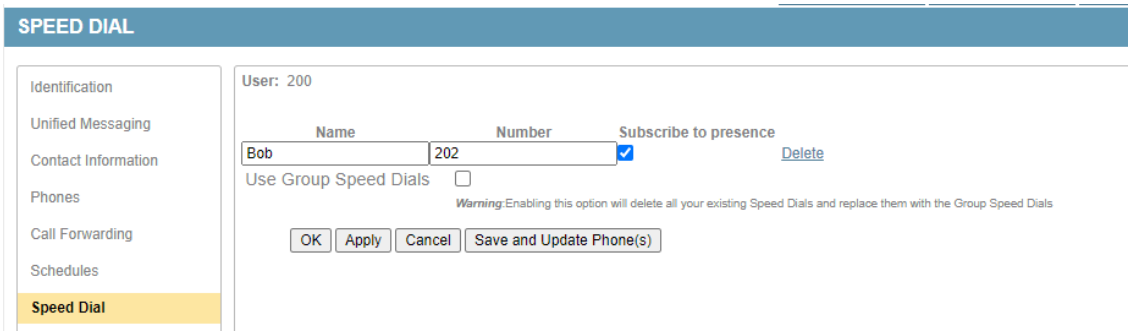
– **Event: message-summary** is used with the *Message Waiting Indicator (MWI)* service. A phone that supports this feature may have a light that blinks upon voicemail deposit.

  The MWI subscription of a Polycom phone is defined in the Users - $user - Phones - $phone - Lines - $line - Messaging tab.



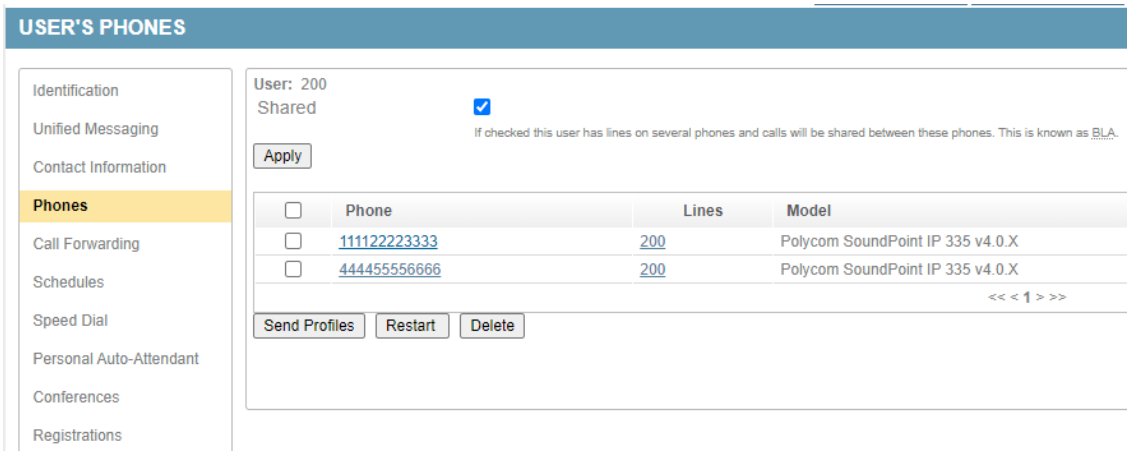
Note: To remove the MWI subscription empty the 'subscribe' field, apply, then send profiles to the phone. Repeat for each phone assigned.

– **Event: dialog** is used when the user is subscribed to the presence status of another user. These are defined in Users - $user - Speed Dials with the "Subscribe to presence" option checked.



Note: Speed dial entries **without** "Subscribe to presence" checked are saved in /var/sipxdata/configserver/phone/profile/tftproot/$mac-directory.xml. Speed dial entries **with** "Subscribe to presence" checked are stored in the SIPXCONFIG database.

– **Event: dialog;sla** is used with the *SAA/BLA* service. This allows a single extension to be assigned to multiple phones. The line status is shared between phones using this method. If the line is shared the 'Shared' box will be checked beneath Users - $user - Phones.



As with **INVITE** and **REGISTER**, the **Contact:** header in a **SUBSCRIBE** is usually the IP of the device

sending the message.

> **Warning:** Only those three **SUBSCRIBE Event:** headers are supported. If you find other Event: headers such as **Event: presence** or **Event: as-feature-event** you can safely assume the device is not configured properly.

There are more such as **OPTIONS** and **ACKs**, but these are usually not as important.

The first section of the sipcodes output describes the time period covered in the log file provided. Note that both proxy and bridge timestamps are in UTC/GMT. This portion is simply gathering and filtering output from the top (head) two lines and the last (tail) two lines of the log.

The next section describes overall SIP message counts mentioned previously, including OPTIONS and ACKs.

The "additional information" of each provides the most useful troubleshooting information. **The default output only displays the top 50 results.** If you have thousands of users or if there are more than 50 users/devices spamming the proxy, you may need to increase this (edit the "head -n50" parts of the script) to get a higher level perspective.

Users with the lowest counts in each column are likely users with a single phone assignment and properly configured. **Users with substantially higher counts than others should be suspected as misconfigured.**

For example:

- A user who has configured a incorrect SIP password on a gateway/phone will have a much higher **REGISTER** count than others because the gateway/phone will continually send requests and fail. A successful REGISTER does not send another request until the registration counter is near expiration (expiration values are typically somewhere between 300 to 3600 seconds). The only legitimate reason for a high **REGISTER** count is multiple phone assignment, or the device is configured manually with a shorter registration interval than everyone else.
- A user with a much higher **SUBSCRIBE Event: message-summary** count may not have voicemail permission, or even exist anymore, but the phone is still configured to subscribe to MWI. If the user doesn't exist or is disabled, you will probably see a high **REGISTER** count as well.

At the bottom of the output are some 400 and 500 series checks. A healthy output would be 0 on all of those. If devices are flooding the server it will fill the proxy message queue and prevent it from processing legitimate traffic. If you see a high count of 503 Service Unavailable responses this has happened (or is happening).

If you need to determine the IP address of a phone, the **Contact:** header may be used. The exception is if the device is behind a SBC (upper registered). For example, if there is a very high **REGISTER** count from user 200, who no longer exists in the system anymore but still assigned to a phone, you could manually filter the proxy log like:

```
grep "REGISTER sip" /var/log/sipxpbx/sipXproxy.log | grep "INCOMING" | grep "sip:200"␣
↪| syslogviewer --no-pager | grep "Contact:" | awk -F ";" '{ print $1 }' | sort |␣
↪uniq -c | sort -rn
```

8.6 DNS Checks

Use a desktop on the same network as your phones.

8.6.1 MS Windows

Use nslookup on the command line to check if DNS SRV records can be found.

```
C:\>nslookup
Default Server:  UnKnown
Address:  192.168.1.14

> set type=SRV
> _sip._tcp.home.mattkeys.net
Server:  UnKnown
Address:  192.168.1.14

_sip._tcp.home.mattkeys.net     SRV service location:
          priority       = 30
          weight         = 10
          port           = 5060
          svr hostname   = sipx.home.mattkeys.net
home.mattkeys.net       nameserver = sipx.home.mattkeys.net
sipx.home.mattkeys.net  internet address = 192.168.1.14
>
```

8.6.2 Mac, Linux/Unix

Use dig on the command line to check if DNS SRV records can be found.

```
# dig -t SRV _sip._tcp.home.mattkeys.net @192.168.1.14

; <<>> DiG 9.11.5-P4-5.1+deb10u2-Debian <<>> -t SRV _sip._tcp.home.mattkeys.net @192.
↪168.1.14
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2639
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: c0d5be2f1899987c18baf1545f96f47f46f6565a81ddbad1 (good)
;; QUESTION SECTION:
;_sip._tcp.home.mattkeys.net.   IN      SRV

;; ANSWER SECTION:
_sip._tcp.home.mattkeys.net. 1800 IN    SRV     30 10 5060 sipx.home.mattkeys.net.

;; AUTHORITY SECTION:
home.mattkeys.net.      1800    IN      NS      sipx.home.mattkeys.net.

;; ADDITIONAL SECTION:
sipx.home.mattkeys.net. 1800    IN      A       192.168.1.14

;; Query time: 1 msec
;; SERVER: 192.168.1.14#53(192.168.1.14)
;; WHEN: Mon Oct 26 12:08:31 EDT 2020
;; MSG SIZE  rcvd: 156
```

8.7 The Call-ID Header

The Call-ID header remains unique during any SIP message dialog. This allows you to use it as the search term in utilities such as grep to follow the dialog.

The easiest way to obtain the Call-ID of a call is probably using a CDR CSV export. The call ID is a column in the CSV output for each entry.

The other method is working with the proxy log data directly. **SIP messages are only visible if the proxy log is at INFO or DEBUG verbosity. It is at NOTICE by default, which does not display the SIP messages**.

The log verbosity setting is **not retrospective**. The setting must be at INFO or DEBUG *prior* to whatever it is you're trying to troubleshoot.

Note: If you can't reproduce the problem it probably isn't worth troubleshooting it. Test multiple times. If it is intermittent, and you are running telephony services on multiple servers, that may indicate the problem is isolated to a particular server in the cluster (check DNS SRV and network path).

If the message dialog is a INVITE, and the call is forked such as with a 'at the same time' call forward, the end of the Call-ID will be suffixed with -0, -1, and so on for as many call forks there were. When using grep against the log, use the first half (that is everything prior to the @) of the Call-ID as your search string. This will ensure you get the entire dialog in the grep output, including any forks.

For example user 201 called user 200. A call is a INVITE, and the To: is "sip:200". So we can show all Call-IDs To: 200 with the following:

```
# grep "INVITE sip:200" /var/log/sipxpbx/sipXproxy.log | syslogviewer --no-pager |␣
↪grep "Call-ID:" | sort -u
Call-ID: a52e3e8fa0e281c4ab51822b2d14f829@0:0:0:0:0:0:0:0
```

Next use the Call-ID in the output to fetch the entire dialog and redirect output to a log file like:

```
# grep "a52e3e8fa0e281c4ab51822b2d14f829" /var/log/sipxpbx/sipXproxy.log |␣
↪syslogviewer --no-pager > ~/a52e3e8fa0e281c4ab51822b2d14f829.log
```

Note: The syslogviewer utility is only available on systems with sipxcom rpms installed. It escapes all line returns (at each header) in the proxy log that would otherwise be considered one single line to tools like grep. In other words, syslogviewer makes reading the log human friendly. The –no-pager flag disables the pause at each page. This allows for redirection of the entire output to a small log file. As less is the default viewer, this allows for all the features of less. For example, use the / key to search and highlight occurrences of a search term.

```
# syslogviewer --help
Usage:
      syslogviewer [-h | --help] [-i | --indent] [-f[nn]]
              [of=output]
              [input]

      --help  Print this help message.
      --indent        Indent messages and continued lines.
      --no-pager      Do not pipe the output through a pager, even if it is a tty.
      -f[nn]  Fold lines that are over nn (default 80) characters.
              Implies --indent.
      if=<file>       input file name (deprecated - use file name directly).
      of=<file>       output file name (defaults to stdout)
```

(continues on next page)

(continued from previous page)

```
    input - input file name (defaults to stdin)

    If stdout is a tty, pipes output through the program specified
    by the PAGER environment variable (defaults to 'less').
```

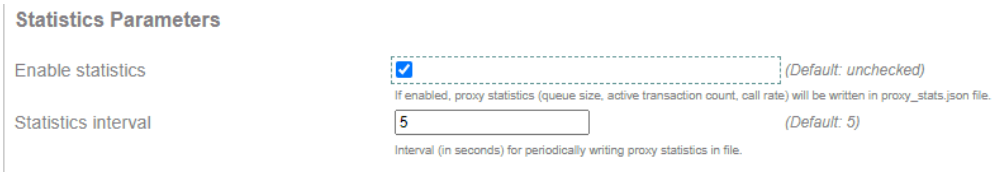
CHAPTER

NINE

MONITORING

There are several 3rd party options available to monitor the system, depending on your needs. **Do not run other configuration management agents such as Puppet or Chef on the server!** This is because CFengine (sipxsupervisor service) is already in use. Other configuration management agents will likely interfere with CFengine/sipxsupervisor functioning correctly.

- Sipxcom has built-in SNMP alarms for your convenience beneath Diagnostics - Alarms. Be sure to check those first.
- The sipcodes.sh script can also be used to produce high level SIP statistics ad hoc. Given the proxy log is at INFO or DEBUG verbosity the sipcodes script will automatically collect and report statistics upon snapshot collection as ./var/log/sipxpbx/sipcodes.log.
- The SIP proxy service has a option to save proxy statistics to a json file, /var/log/sipxpbx/proxy_stats.json.



- The /var/log/sipxpbx/proxy_stats.json file can be consumed by tools such as Grafana , ELK stacks, Graylog, Splunk, etc to create current or historical graphs.
- Nagios , Munin, Cacti and many others can also be used to monitor service status, server health, etc.

9.1 Nagios

This section provides example configuration of a Nagios Core to monitor sipxcom services.

9.1.1 Prerequisites

You'll need a separate server running Nagios Core, and administrative access to all the sipxcom servers. For this example I have compiled Nagios Core from source using the default settings rather than using a OS package. The path to files may vary if you have installed via rpm or apt. I will use the Nagios Remote Plugin Executor (NRPE) as a means to aggregate the checks, but there are alternatives available if NRPE doesn't suit your needs. Most of the checks used are available from the standard Nagios Plugins. If you want to expand on these there are many more available from the Nagios Exchange.

9.1.2 Overview

If compiled from source using the defaults, Nagios Core will install to /usr/local/nagios:

```
# tree --charset=ASCII -d nagios/
nagios/
|-- bin
|-- etc
|   `-- objects
|-- libexec
|-- sbin
|-- share
|   |-- contexthelp
|   |-- docs
|   |   `-- images
|   |-- images
|   |   `-- logos
|   |-- includes
|   |   `-- rss
|   |       `-- extlib
|   |-- js
|   |-- media
|   |-- ssi
|   `-- stylesheets
`-- var
    |-- archives
    |-- rw
    `-- spool
        `-- checkresults
```

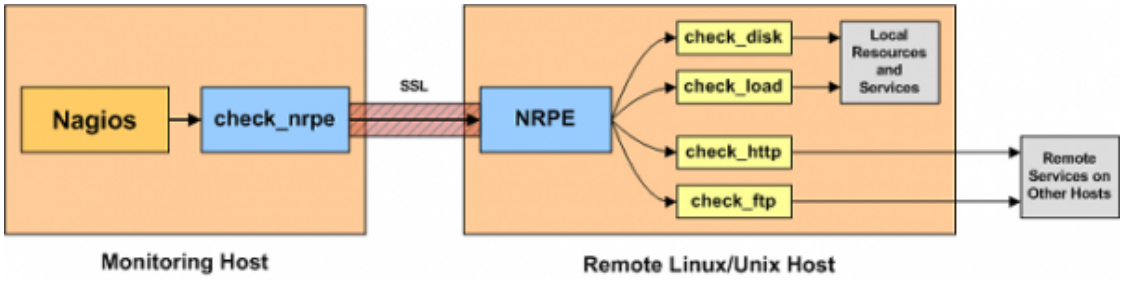
The etc/objects directory is where your host configuration files are stored. I recommend organizing hosts into groups beneath the objects directory, then grouping similar services beneath that. For example, if you have a group of three sipxcom servers at example.org:

```
$ mkdir /usr/local/nagios/etc/objects/example.org
$ mkdir /usr/local/nagios/etc/objects/example.org/sipx
$ touch /usr/local/nagios/etc/objects/example.org/sipx/sipx1.cfg
$ touch /usr/local/nagios/etc/objects/example.org/sipx/sipx2.cfg
$ touch /usr/local/nagios/etc/objects/example.org/sipx/sipx3.cfg
```

By structuring in this way the system administrator can quickly understand who it belongs to and what it does. This is also especially helpful if you intend on running Nagios in a multi tenant fashion.

9.1.3 Preparing a host for monitoring

Before stepping into the sipx1.cfg configuration on the Nagios server we'll need to prepare the sipxcom server(s) for our checks. Nagios Remote Plugin Executor (NRPE) works as an aggregate point for multiple check scripts.



You'll need to download and install both NRPE and the standard Nagios plugins on each host you intend on monitoring. After installing these you may wish to pause for a moment and review the check scripts now available under /usr/local/nagios/libexec. The NRPE configuration, /usr/local/nagios/etc/nrpe.cfg, was likely copied from the sample provided within the NRPE tarball. You should review this file for any environmental changes you may need to make such as partition locations:

```
command[check_hda1]=/usr/local/nagios/libexec/check_disk -w 20% -c 10% -p /dev/hda1
command[check_zombie_procs]=/usr/local/nagios/libexec/check_procs -w 5 -c 10 -s Z
command[check_total_procs]=/usr/local/nagios/libexec/check_procs -w 150 -c 200
```

The commands defined should match what is being called within the host configuration file. For example, checks for sipx1.example.org are defined on the nagios server in /usr/local/nagios/etc/objects/example.org/sipx/sipx1.cfg:

```
define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             Check Users
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_users
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             Check Swap
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_swap
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             Check Load
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_load
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             Check Boot Partition
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_boot
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             Check Root Partition
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_root
        }
```

(continues on next page)

(continued from previous page)

```
define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             Check Zombie Processes
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_zombie_procs
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             Check Total Processes
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_total_procs
        }
```

The check_command line is essentially "connect with NRPE and run xxx". Be sure that xxx is a defined within /usr/local/nagios/etc/nrpe.cfg of the host you are checking against. For things you want to execute from the Nagios server, make certain that you've defined those commands in the Nagios server /usr/local/nagios/etc/objects/commands.cfg. For example I defined the SSL certificate check on my Nagios server command.cfg:

```
define command {
        command_name    check_ssl_certificate
        command_line    $USER1$/check_ssl_certificate -H $HOSTADDRESS$ -c 3 -w 7
       }
```

But in /usr/local/nagios/etc/objects/example.org/sipx1.cfg, this is defined without the check_nrpe prefix so it will execute from the Nagios server rather than on the sipx1.example.org host:

```
define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             SSL Certificate Expiration
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_ssl_certificate
        }
```

9.1.4 Sipxcom services

Below are additional examples for sipx1.example.org that pertain to sipXcom/sipx services. These would be defined in /usr/local/nagios/etc/objects/example.org/sipx/sipx1.cfg on our Nagios server:

```
define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             NTP
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_ntp_time!0.5!1
        }
```

(continues on next page)

(continued from previous page)

```
define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             Check SIP Registration
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_sip_registration
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             SSH
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_ssh!-p 22
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             TFTP
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_tftp
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             FTP
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_ftp!-H sipx1.example.org -p 21
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             Check sipx Web UI
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_ui
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             Check XMPP
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_jabber
        }

define service{
        use                             generic-service
```

(continues on next page)

(continued from previous page)

```
        host_name                       sipx1.example.org
        service_description             TCP SIP SRV
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_tcp_sip_srv
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             UDP SIP SRV
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_udp_sip_srv
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             TCP SIPS SRV
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_tcp_sips_srv
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             SIP TLS SRV
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_sip_tls_srv
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             SIP RR SRV
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_sip_rr_srv
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             SIP MWI SRV
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_sip_mwi_srv
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             XMPP client SRV
        contact_groups                  admins
```

(continues on next page)

(continued from previous page)

```
        notifications_enabled           1
        check_command                   check_nrpe!check_xmpp_client_srv
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             XMPP server SRV
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_xmpp_server_srv
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             XMPP conference server SRV
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_xmpp_conf_srv
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             TCP Voicemail SRV
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_tcp_vm_srv
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             Check SIPXCONFIG
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_sipxconfig
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             Check SIPXCDR
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_sipxcdr
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             Check MySQL homer.db
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_homer
        }
```

(continues on next page)

(continued from previous page)

```
define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             MongoDB Connection Check
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_mongo_connect
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             MongoDB Long running ops
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_mongo_lag
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             MongoDB Operations Count
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_nrpe!check_mongo_ops
        }

define service{
        use                             generic-service
        host_name                       sipx1.example.org
        service_description             SSL Certificate Expiration
        contact_groups                  admins
        notifications_enabled           1
        check_command                   check_ssl_certificate
        }
```

The command definitions for all commands prefixed with check_nrpe should be defined on sipx1.example.org within /usr/local/nagios/etc/nrpe.cfg, for example:

```
# system checks
command[check_users]=/usr/local/nagios/libexec/check_users -w 5 -c 10
command[check_load]=/usr/local/nagios/libexec/check_load -w 15,10,5 -c 30,25,20
command[check_root]=/usr/local/nagios/libexec/check_disk -w 20% -c 10% -p /dev/mapper/
↪vg_root-lv_root
command[check_boot]=/usr/local/nagios/libexec/check_disk -w 20% -c 10% -p /dev/vda1
command[check_zombie_procs]=/usr/local/nagios/libexec/check_procs -w 5 -c 10 -s Z
command[check_total_procs]=/usr/local/nagios/libexec/check_procs -w 200 -c 250
command[check_swap]=/usr/local/nagios/libexec/check_swap -w 80% -c 50%
command[check_memory]=/usr/local/nagios/libexec/check_memory.pl

# sipx service checks
command[check_sipxconfig]=/usr/local/nagios/libexec/check_postgres.pl -db SIPXCONFIG -
↪-action connection
command[check_sipxcdr]=/usr/local/nagios/libexec/check_postgres.pl -db SIPXCDR --
↪action connection
```

(continues on next page)

(continued from previous page)

```
command[check_ui]=/usr/local/nagios/libexec/check_http -w5 -c 10 --ssl -H sipx1.
↪example.org -u /sipxconfig/app
command[check_sip_registration]=/usr/local/nagios/libexec/check_registrations.sh
command[check_ntp_time]=/usr/local/nagios/libexec/check_ntp_time -H sipx1.example.org␣
↪-w 0.5 -c 1
command[check_mongo_connect]=/usr/bin/python /usr/local/nagios/libexec/check_mongo -H␣
↪sipx1.example.org -A connect
command[check_mongo_ops]=/usr/bin/python /usr/local/nagios/libexec/check_mongo -H␣
↪sipx1.example.org -A count
command[check_mongo_lag]=/usr/bin/python /usr/local/nagios/libexec/check_mongo -H␣
↪sipx1.example.org -A long

# dns checks
command[check_tcp_sip_srv]=/usr/local/nagios/libexec/check_dns -H _sip._tcp.example.
↪org -s 127.0.0.1 -q SRV
command[check_udp_sip_srv]=/usr/local/nagios/libexec/check_dns -H _sip._udp.example.
↪org -s 127.0.0.1 -q SRV
command[check_tcp_sips_srv]=/usr/local/nagios/libexec/check_dns -H _sips._tcp.example.
↪org -s 127.0.0.1 -q SRV
command[check_sip_tls_srv]=/usr/local/nagios/libexec/check_dns -H _sip._tls.example.
↪org -s 127.0.0.1 -q SRV
command[check_sip_mwi_srv]=/usr/local/nagios/libexec/check_dns -H _sip._tcp.mwi.
↪example.org -s 127.0.0.1 -q SRV
command[check_sip_rr_srv]=/usr/local/nagios/libexec/check_dns -H _sip._tcp.rr.example.
↪org -s 127.0.0.1 -q SRV
command[check_tcp_vm_srv]=/usr/local/nagios/libexec/check_dns -H _sip._tcp.vm.example.
↪org -s 127.0.0.1 -q SRV
command[check_xmpp_server_srv]=/usr/local/nagios/libexec/check_dns -H _xmpp-server._
↪tcp.example.org -s 127.0.0.1 -q SRV
command[check_xmpp_client_srv]=/usr/local/nagios/libexec/check_dns -H _xmpp-client._
↪tcp.example.org -s 127.0.0.1 -q SRV
command[check_xmpp_conf_srv]=/usr/local/nagios/libexec/check_dns -H _xmpp-server._tcp.
↪conference.example.org -s 127.0.0.1 -q SRV
```

As there are checks that are executed server side, those need to be defined in /usr/local/nagios/etc/objects/commands.cfg on the Nagios server:

```
define command{
command_name check_tftp
command_line $USER1$/check_tftp --get $HOSTADDRESS$ 000000000000.cfg 7167
}

define command{
command_name check_jabber
command_line $USER1$/check_jabber -H $HOSTADDRESS$ --expect='xmlns="jabber:client"␣
↪from="example.org"'
}

define command {
command_name check_ssl_certificate
command_line $USER1$/check_ssl_certificate -H $HOSTADDRESS$ -c 3 -w 7
}
```

9.1.5 3rd Party Checks

check_jabber is used for XMPP checks. check_mongo is used for MongoDB checks. check_postgres is used for

PostgreSQL checks. For check_sip_registration I created a shell script that utilizes sipx-dbutil.

9.1.6 Additional Notes

- You may find some checks complain of missing utils.pm. If you do, check if the script is making any references to the nagios plugins directory. You may need to alter the path to /usr/local/nagios/libexec/.
- Be sure to inspect any firewalls between your Nagios server and the sipXcom/sipx servers prior to running your checks. Some services such as ssh are restrictive by default in the sipXcom/sipx firewall.
- It is possible to utilize sipsak to test against the SIP stack, however **be aware that by default sipxcom SIP security feature will will ban the source IP address of client using default sipsak User Agent string.**
- Try not to cause unnecessary stress or bandwidth consumption on the server with your service checks. Once a day is probably good enough for a check interval for some services such as the SSL certificate check. See the "External Command Check Interval" section here : http://nagios.sourceforge.net/docs/3_0/configmain.html.

9.2 Graylog

Graylog is open source log management/aggregation software. A fully supported Commercial/Enterprise version also exists. For this example I am using the open source version on a Debian 10 server.

9.2.1 Installation on Debian 10

Starting from a fresh Debian 10 minimal installation:

```
apt-get update && apt-get upgrade -y
apt-get install apt-transport-https openjdk-11-jre-headless uuid-runtime pwgen␣
↪dirmngr curl
apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 4B7C549A058F8B6B
echo "deb http://repo.mongodb.org/apt/debian buster/mongodb-org/4.2 main" | tee /etc/
↪apt/sources.list.d/mongodb-org-4.2.list
apt-get update && apt-get install mongodb-org -y
systemctl daemon-reload
systemctl enable mongod.service
systemctl restart mongod.service
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | apt-key add -
echo "deb https://artifacts.elastic.co/packages/oss-6.x/apt stable main" | tee -a /
↪etc/apt/sources.list.d/elastic-6.x.list
apt-get update && apt-get install elasticsearch-oss -y
echo "cluster.name: graylog" >> /etc/elasticsearch/elasticsearch.yml
echo "action.auto_create_index: false" >> /etc/elasticsearch/elasticsearch.yml
systemctl daemon-reload
systemctl enable elasticsearch.service
systemctl restart elasticsearch.service
wget https://packages.graylog2.org/repo/packages/graylog-3.1-repository_latest.deb
dpkg -i graylog-3.1-repository_latest.deb
apt-get update && apt-get install graylog-server -y
```

For admin password as password and hash edit /etc/graylog/server/server.conf and set:

```
echo "password_secret =␣
↪naln41C22HRxw3hy9mJ8bipFWBo1aewKFgtXDXp22dNjNJNqEtid6uC0476zIfX5iQ3mZuRp9y7h3XcNY63inPo6vJy7FuLP
↪"
echo "root_password_sha2 =␣
↪5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8"
echo "http_bind_address = 192.168.1.114:9000"
echo "http_publish_uri = http://192.168.1.114:9000"
systemctl enable graylog-server.service
systemctl start graylog-server.service
```

The Graylog webui should now be up on http://192.168.1.114:9000. Create a GELF UDP input using the default port 12201.

9.2.2 Fluentd on Graylog server

Fluent Bit is an open source and multi-platform Log Processor and Forwarder which allows you to collect data/logs from different sources, unify and send them to multiple destinations. It's fully compatible with Docker and Kubernetes environments. Fluent Bit is written in C and has a pluggable architecture supporting around 30 extensions.

For this example fluentd is running on the Graylog server. It is used to convert data received into the Graylog GELF format.

```
# fluentd on graylog
apt-get install sudo ntp ntpdate ntpstat ruby-gelf
curl -L https://toolbelt.treasuredata.com/sh/install-debian-buster-td-agent3.sh
systemctl daemon-reload
systemctl enable td-agent
td-agent-gem install gelf
cd /etc/td-agent/plugin
wget https://raw.githubusercontent.com/emsearcy/fluent-plugin-gelf/master/lib/fluent/
↪plugin/out_gelf.rb
cd ../
```

Append to /etc/td-agent/td-agent.conf:

```
<source>
    type syslog
    tag hostname_goes_here
</source>
<match *.*>
    type copy
    <store>
        type gelf
        host 0.0.0.0
        port 12201
        flush_interval 5s
    </store>
    <store>
        type stdout
    </store>
</match>
```

Restart the service and configure the service to start at boot with:

```
systemctl restart td-agent
systemctl enable td-agent
```

9.2.3 Fluentbit on the sipxcom server

For this example I am using fluentbit on the sipxcom server to ship logs to the fluentd instance of the Graylog server:

```
# fluentbit on sipx/uniteme centos7
cd /etc/yum.repos.d/
nano fluentbit.repo
```

Inside fluentbit.repo:

```
[fluentbit]
name = fluentbit
baseurl = http://packages.fluentbit.io/centos/7
gpgcheck=1
gpgkey=http://packages.fluentbit.io/fluentbit.key
enabled=1
```

Next update the packages:

```
yum update
yum install td-agent-bit -y
mv /etc/td-agent-bit/td-agent-bit.conf ~/td-agent-bit.conf.orig
nano /etc/td-agent-bit/td-agent-bit.conf
```

Inside td-agent-bit.conf:

```
[INPUT]
    Name cpu
    Tag  cpu.local
    Interval_Sec 1

[INPUT]
    Name mem
    Tag memory

[INPUT]
    Name disk
    Tag disk.local
    Interval_Sec 1

[INPUT]
    Name netif
    Tag netif.eth0
    Interval_Sec 1
    Interface eth0

[INPUT]
    Name health
    Tag health.proxy
    Host 192.168.2.14
    Port 5060
    Interval_Sec 60
    Alert true
    Add_Host true
    Add_Port true

[INPUT]
    Name health
```

(continues on next page)

(continued from previous page)

```
    Tag health.registrar
    Host 192.168.2.14
    Port 5070
    Interval_Sec 60
    Alert true
    Add_Host true
    Add_Port true

[INPUT]
    Name health
    Tag health.bridge
    Host 192.168.2.14
    Port 5090
    Interval_Sec 60
    Alert true
    Add_Host true
    Add_Port true

[INPUT]
    Name health
    Tag health.mongo
    Host 127.0.0.1
    Port 27017
    Interval_Sec 60
    Alert true
    Add_Host true
    Add_Port true

[INPUT]
    Name health
    Tag health.pgsql
    Host 127.0.0.1
    Port 5432
    Interval_Sec 60
    Alert true
    Add_Host true
    Add_Port true

[INPUT]
    Name health
    Tag health.dns
    Host 127.0.0.1
    Port 53
    Interval_Sec 60
    Alert true
    Add_Host true
    Add_Port true

[INPUT]
    Name tail
    Path /var/log/sipxpbx/proxy_stats.json
    Refresh_Interval 1
    Parser json

[OUTPUT]
    Name  forward
    Match *
```

(continues on next page)

(continued from previous page)

```
    Host 192.168.1.114
    Port 24224
```

And finally restart the service:

```
service td-agent-bit restart
```

You should now see Graylog reporting activity on the GELF input.

CHAPTER

TEN

MAINTENANCE

Sipxcom runs so well that it can be easy to become complacent on server maintenance. A common issue we see is lack of free disk space.

Warning: If the server runs out of free disk space all services will halt!

If the server has ran out of free disk space, free up some space by deleting files and reboot. The system should recover upon service startup. Backup archives and logs usually consume the most space. Both can be safely deleted.

10.1 Disk Maintenance and Backup Integrity

- All sipx logs are beneath /var/log/sipxpbx/. Mongo logs are beneath /var/log/mongodb/. Apache logs are beneath /var/log/httpd/. Postgresql logs are beneath /var/lib/pgsql/data/pg_log/. Rotated logs are suffixed with a date, or may end with a *.gz extension. Any rotated logs should be deleted periodically (monthly or yearly recommended) to conserve disk space, reduce snapshot size, etc.
- If using scheduled backups make certain the "Number of backups to keep" option is not set to unlimited. Backups can be very large. That option should be set to a very conservative level (5 or less recommended).
- Periodically verify that the scheduled backup archives are being created and saved correctly to a safe location.

10.2 Software updates

- Run a 'yum update -y' on a daily, weekly, or monthly schedule to keep the OS patched with the latest security updates. This won't upgrade sipxcom packages unless you've changed the sipxcom repo file beneath /etc/yum.repos.d/ to point to a different version.
- Check that your sipxcom version is the latest stable version at least once a year. The footer of the sipxcom webui should indicate what version it is. Always check the release notes of the new version for any critical notices prior to upgrading.
- Check annually that your phones are running the latest GA firmware for the model. Phone bugs are mitigated by firmware upgrades. Polycom has two firmware pages available, one for SoundPoint and SoundStation IP models and the other for VVX models.

CHAPTER

ELEVEN

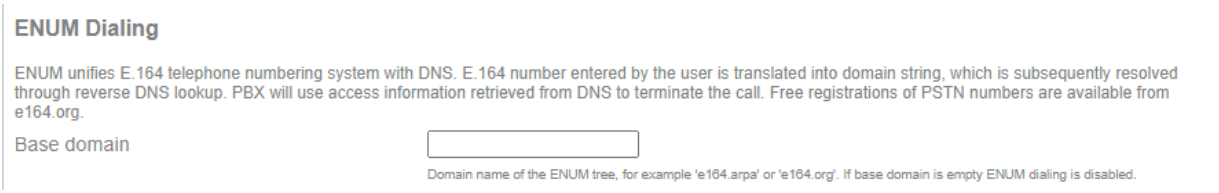
OTHER / HOW TO

11.1 Use blockchain DNS for ENUM / E.164 Records

The Emercoin blockchain can store DNS records that map a telephone number to a (SIP) domain name (ENUM / E.164). Emercoin named this service ENUMER.

Emercoin is a fork of Bitcoin. The Emercoin NVS (Name Value Storage) is very close to Namecoin, the first fork of Bitcoin. Emercoin is the only blockchain DNS we're aware of that supports NAPTR records. Please correct us if we're wrong.

On sipxcom, the ENUM Dialing settings can be found beneath System - Services - *SIP Registrar*. To use Emercoin ENUMER you only need to point the 'Base Domain' to a server running the Emercoin wallet. There used to be a public service (enum.enumer.org) you could point to, but that appears to be down at the time of this writing.



Note: The Base Domain should be input as the FQDN of the server running the Emercoin wallet rather than its IP. You may need to *create a custom A record* for it in your DNS zone.

The server running the Emercoin wallet should have the DNS service enabled and enum added:

```
EmerDNSallowed=$enum|.coin|.emc|.lib|.bazar      # add Allowed TLDs with ENUM
enumtrust=ver:enum
enumtollfree=@enum:tollfree
```

Note: The official signing authority (**ver:enum**) is Emercoin, but you should be able to create your own **ver** type record and point the **enumtrust** parameter to that. Otherwise you'll need Emercoin to verify and sign the record.

As Emercoin is a public blockchain you can use official explorers to view all enum records currently stored. The example below is the (officially signed) record for the eZuce main number.

Type	Name	Value	Registered At	Expires in
enum	enum:19782961005:0	SIG=ver:enum|IJor4wFXNVLABAJ+iCQvjlMQcC9+xJXAk8fQZLqLwjjuT3fVb1GVyS2iQh/TUnuefbM/ExCsKpkxPfs3mKtPwGQ= E2U+sip=100|10|!^.*$!sip:9782961005@uniteme.ezuce.com!	383972	1342097

11.2 Use SIP on a Raspberry Pi (BareSIP)

BareSIP is a portable and modular SIP User Agent with audio and video support. It is written almost completely in C.

BareSIP is one of few SIP user agents available for a Raspberry Pi.

It has a very impressive feature set! For example, it can use the RPi (CSI interface) camera as a video source. There is also a NoIR version of the RPi camera for low light situations.

Note: A USB camera will work much better than the CSI interface cameras. The CSI cameras require manual focus and probably won't give you as high of a fps rate as you would get using a USB camera.

The JACK Audio and Opus codec (mono or stereo) support are very handy when working with pro audio gear. For example, pair the RPi with a Pisound hat to terminate two wireless lavalier microphones at 48 kHz sample rate.

BareSIP is available within the Debian (and Raspbian/RaspiOS) 8, 9, and 10 repositories by default:

```
# sudo apt-cache search baresip
baresip - portable and modular SIP user-agent - metapackage
baresip-core - portable and modular SIP user-agent - core parts
baresip-ffmpeg - portable and modular SIP user-agent - FFmpeg codecs and formats
baresip-gstreamer - portable and modular SIP user-agent - GStreamer pipelines
baresip-gtk - portable and modular SIP user-agent - GTK+ front-end
baresip-x11 - portable and modular SIP user-agent - X11 features
```

To install baresip on Debian or RPi:

```
# sudo apt-get install baresip
```

After installation the configuration, SIP account, and speed dial (contacts) configuration files are beneath the ~/.baresip subdirectory. There are examples of these within the BareSIP documentation.

On the sipxcom side you only need to create a regular (not phantom) user to register as. Use the 'user ID' and 'SIP password' values as the **auth_user** and **auth_pass** account configuration value:

```
#    ;auth_user=username
#    ;auth_pass=password
```

Sipxcom uses TCP transport for phones by default. Configure Baresip to use TCP transport with:

```
#    ;transport=tcp
```

11.3 Build your own stratum 1 NTP server with Raspberry Pi

11.3.1 Shopping List

- gps hat - https://www.adafruit.com/product/2324
- antenna - https://www.adafruit.com/product/960
- sma adapter - https://www.adafruit.com/product/851
- battery - https://www.adafruit.com/product/380
- rpi3b - https://www.adafruit.com/product/3055

- case - https://www.adafruit.com/product/2258
- 5v 2.5a power adapter - https://www.adafruit.com/product/1995

11.3.2 Configuration

In /etc/ntp.conf:

```
enable kernel
enable pps
enable stats

driftfile /var/lib/ntp/ntp.drift

statistics loopstats peerstats clockstats
filegen loopstats file loopstats type day enable
filegen peerstats file peerstats type day enable
filegen clockstats file clockstats type day enable

# pps ref
server 127.127.28.2 iburst prefer
fudge 127.127.28.2 refid PPS

# gps shared mem
server 127.127.28.0 iburst
fudge 127.127.28.0 refid GPS

# gps peers
peer pi-ntp1.home.mattkeys.net iburst
peer pi-ntp3.home.mattkeys.net iburst
peer pi-ntp4.home.mattkeys.net iburst

server time.nist.gov

# backup pools
pool 0.us.pool.ntp.org iburst
pool 1.us.pool.ntp.org iburst
pool 2.us.pool.ntp.org iburst
pool 3.us.pool.ntp.org iburst

restrict -4 default kod notrap nomodify nopeer limited
restrict -6 default kod notrap nomodify nopeer limited
restrict 127.0.0.1
restrict ::1
restrict source notrap nomodify
```

In /etc/default/gpsd:

```
START_DAEMON="true"
USBAUTO="true"
DEVICES="/dev/serial0 /dev/pps0"
GPSD_OPTIONS="-n -G"
```

In /boot/config.txt append:

```
# enable GPS PPS
dtoverlay=pps-gpio,gpiopin=4
```

In /boot/cmdline.txt:

```
dwc_otg.lpm_enable=0 console=tty1 root=PARTUUID=6e172edd-02 rootfstype=ext4␣
↪elevator=deadline fsck.repair=yes rootwait quiet splash plymouth.ignore-serial-
↪consoles
```

I use this cron script (/etc/cron.custom/bouncegps.sh) to make certain gps has a lock before ntp starts:

```
/etc/init.d/ntp stop
/etc/init.d/gpsd stop
/usr/sbin/ntpdate 192.168.3.1
/etc/init.d/gpsd start
sleep 1m
/etc/init.d/ntp start
```

Don't forget to chmod +x it, then add it in the bottom of /etc/rc.local:

```
/etc/cron.custom/bouncegps.sh
exit 0
```

Reboot and you should have it choosing the pps reference within 10 minutes or so:

```
pi@pi-ntp2:~ $ ntpq -pn
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*127.127.28.2    .PPS.            0 l   56   64  377    0.000   -0.003   0.001
x127.127.28.0    .GPS.            0 l   55   64  377    0.000  -157.25   3.220
 0.us.pool.ntp.o .POOL.          16 p    -   64    0    0.000    0.000   0.001
 1.us.pool.ntp.o .POOL.          16 p    -   64    0    0.000    0.000   0.001
 2.us.pool.ntp.o .POOL.          16 p    -   64    0    0.000    0.000   0.001
 3.us.pool.ntp.o .POOL.          16 p    -   64    0    0.000    0.000   0.001
-192.168.3.199   192.168.3.123    2 s   39   64  376    0.839    0.055   0.050
+192.168.3.107   .PPS.            1 s   56   64  376    0.613   -0.007   0.012
+192.168.3.123   .PPS.            1 s   58   64  376    0.610   -0.006   0.011
-132.163.96.2    .NIST.           1 u    1   64  377   54.556    1.749   0.366
```

11.4 Interconnect two disparate sipxcom servers

You can connect disparate sipxcom servers (different SIP domains) by creating a SIP trunk between them. The SIP trunk operates similar to a phone registration, authenticating with user credentials when required (407 Proxy Authentication Required).

11.4.1 Example Scenario

Alice is employed by company that uses sipxcom as their PBX.

Registration to her employer sipxcom server is available over the public internet. Hopefully it is protected by a SBC. She can register any SIP phones at her house to the employer sipxcom server, and place calls without any problems.

That's great, but Alice personally owns all the phones at her house. She doesn't want her employer to manage the configuration and firmware of her phones, or allow the employer to use things like intercom on the phones inside her home. To ensure that, she would like to register all the phones in her home to **a sipxcom server running on her private network** rather than directly to her employer sipxcom server. This will allow Alice to remain in full control of the phones in her home.

Her home sipxcom server is not exposed to the public internet. It is protected by a good quality NAT router/firewall. She's certain it doesn't have any SIP ALGs enabled, and it has handy features like packet capture in the event she needs to troubleshoot the connection to her employer. She could also collect a sipxcom snapshot if needed.

As it is similar to a phone registration, port forwarding is not required on Alice's NAT firewall/router.

She has three SIP phones on her private network, one in each bedroom.

On her employer sipxcom server, Alice is configured as a normal user (not phantom) on extension 5568.

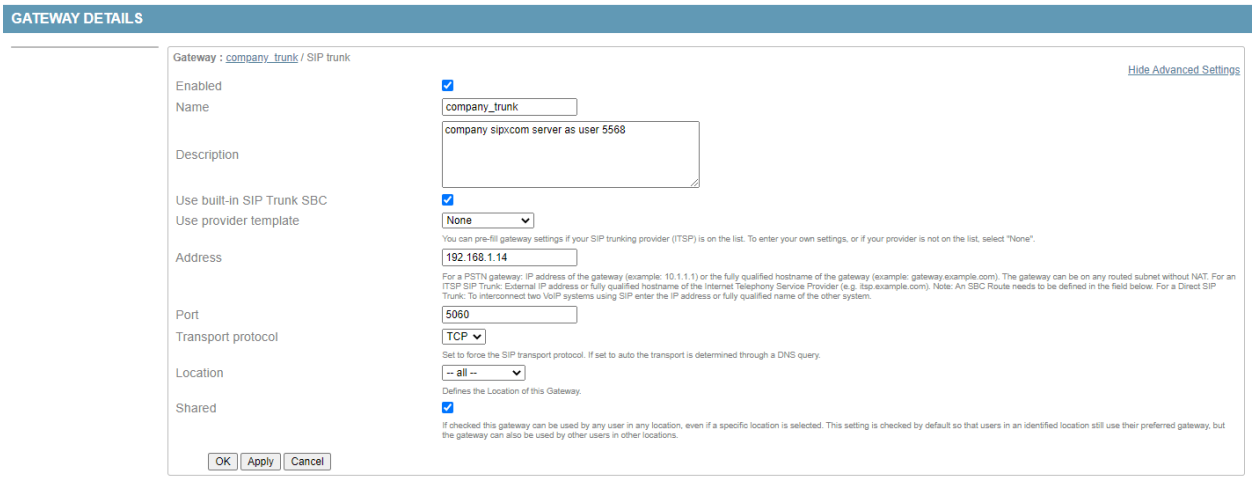
On her home sipxcom server, Alice has configured 3 normal users (not phantom). 200 for the master bedroom, 201 for the first guest bedroom, and 202 in the second guest bedroom. She has one phone assigned to each user. All three are successfully registered when she checks Diagnostics - Registrations. She can place calls between 200 to 201 and 202 without any problems, and they to her.

Alice wants all three phones (200, 201, 202) to ring at the same time when someone calls her extension at work, 5568. This should allow her to answer a incoming call while she is in any room.

11.4.2 Configuration

On her home sipxcom server, Alice logs in as superadmin and navigates to Devices - Gateways.

Next she clicks the 'Add new gateway' drop down and selects SIP Trunk.



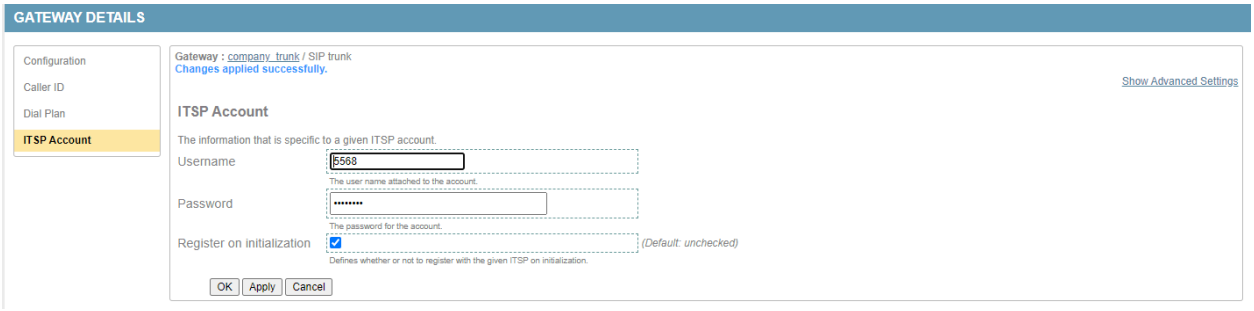
She enters her employer sipxcom server IP address or SIP domain name in the Address field.

Note: The DNS A record and SIP SRV records must be available if you specified by SIP domain.

She enters 5060 (just like a phone) in the Port field, and specifies TCP transport. She specified TCP transport because it is more reliable and doesn't have the size limitations that UDP has (1500 bytes).

Note: If you use the IP rather than SIP domain name, verify the employer sipxcom server has that IP listed in System - Settings - Domain under Domain Aliases. If there are multiple servers running proxy/reg, all the proxy/reg server IPs should be listed in the domain aliases as well.

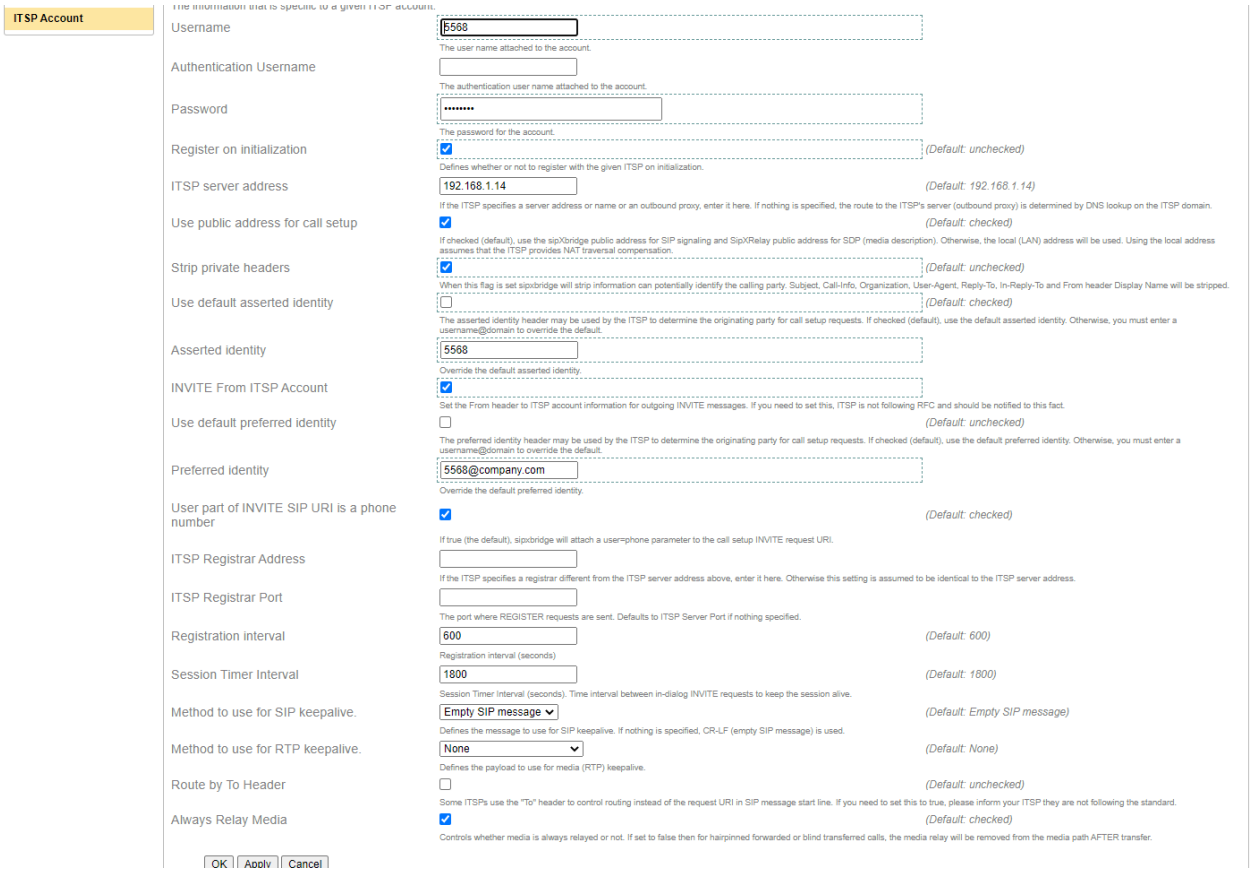
She clicks apply to save, then navigates to the "ITSP Account" tab to enter her company user id (5568) and SIP password cedentials the trunk should authenticate with.



She clicks **Apply** to save again.

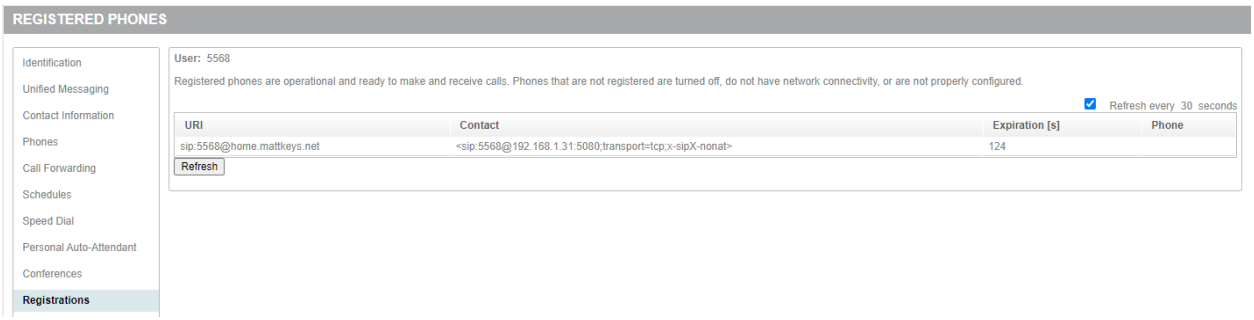
There are a few more settings under **Show Advanced Settings** she needs to tweak:

- She needs to check **Strip private headers** to remove any local user tags within the SIP messaging towards her employer sipxcom server.
- She needs to uncheck **Use default asserted identity** and set 5568 as the **Asserted Identity** for any SIP messaging to the employer sipxcom server.
- She needs to check **INVITE from ITSP Account**, so the **From:** will always be 5568 to her employer server. There will be no caller ID or user ID rewrites (the employer server should do that).
- She also sets the **Preferred identity** as 5568@company.com, where company.com is the employer SIP domain.

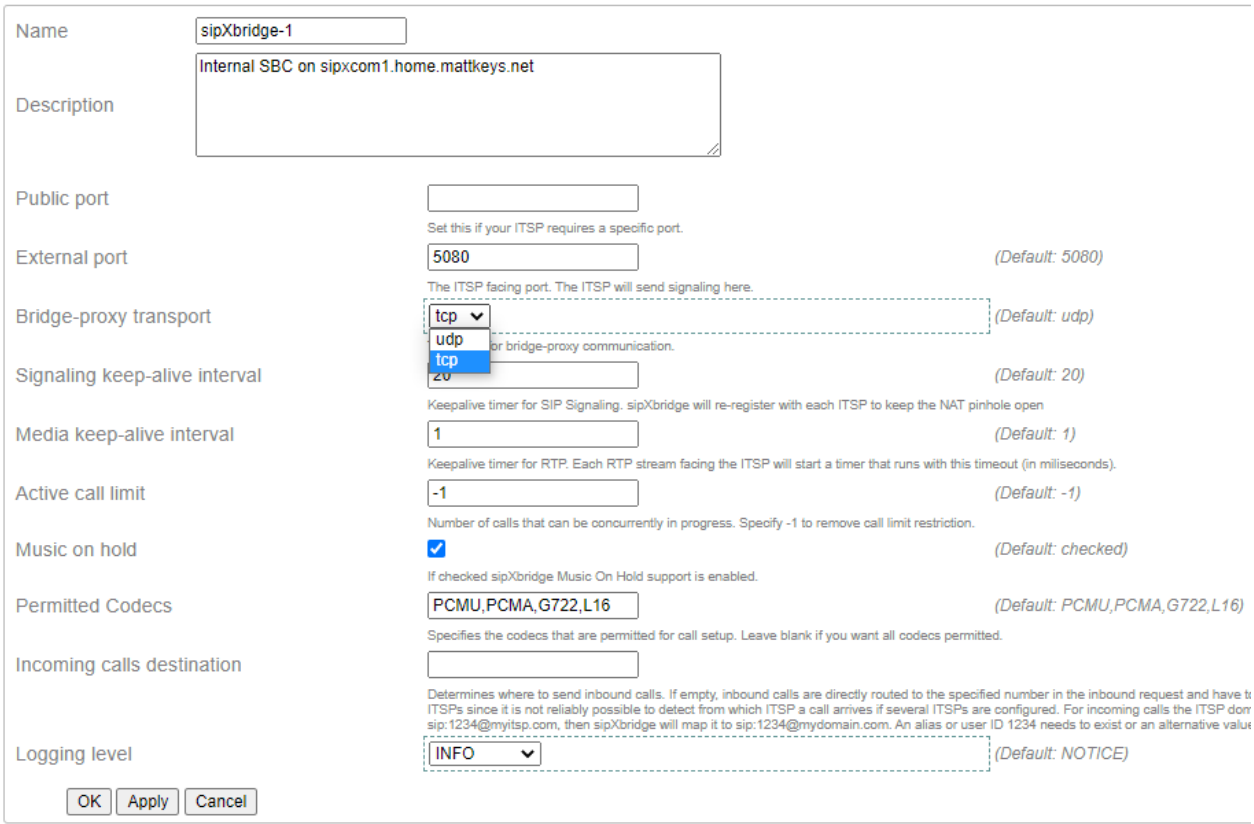


After clicking **Apply** to save, Alice then checks Diagnostics - SIP Trunk statistics on her sipxcom server to verify the SIP Trunk registration was successful against the employer sipxcom.

If Alice has access to the employer sipxcom server, she could also verify on the employer sipxcom beneath Users - 5568 - Registrations. The trunk registration should be listed, and there should be 'transport=tcp' specified in the **Contact** field of the registration.



Note: Don't forget to change system - services - SIP trunk - sipXbridge-1 - Bridge-proxy transport to TCP from default UDP on both sipxcom servers to keep transport consistent.

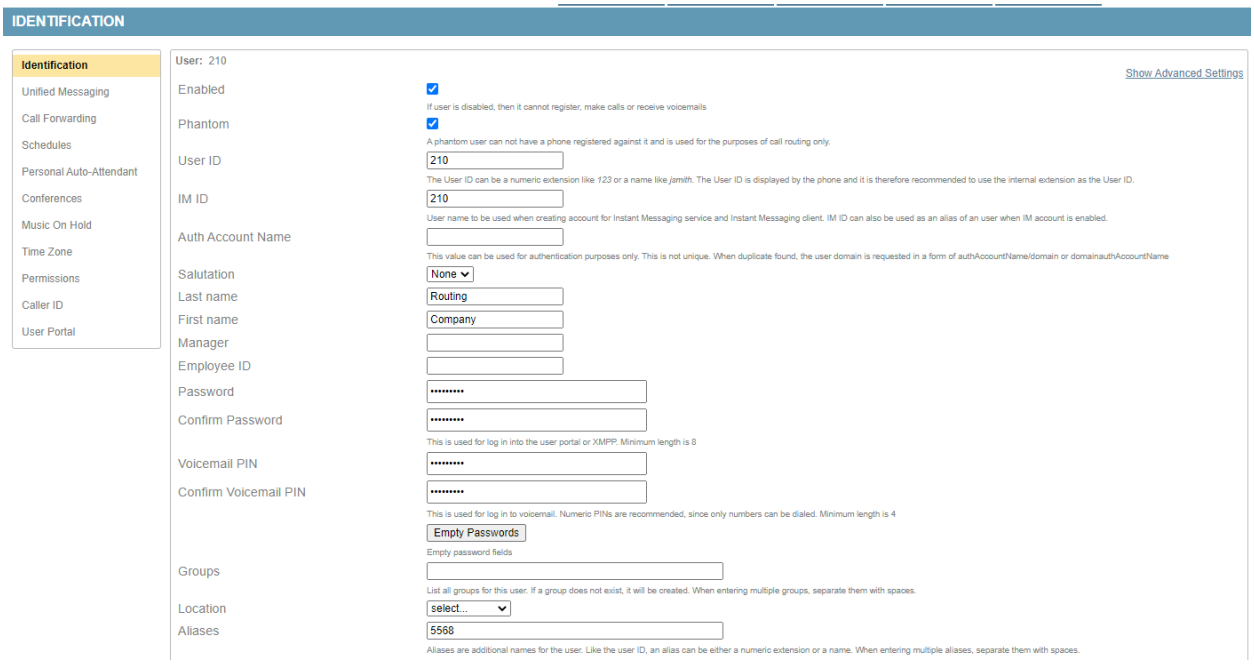


TCP transport is used by default for phone registrations to sipxcom (proxy/reg). This setting will help prevent any

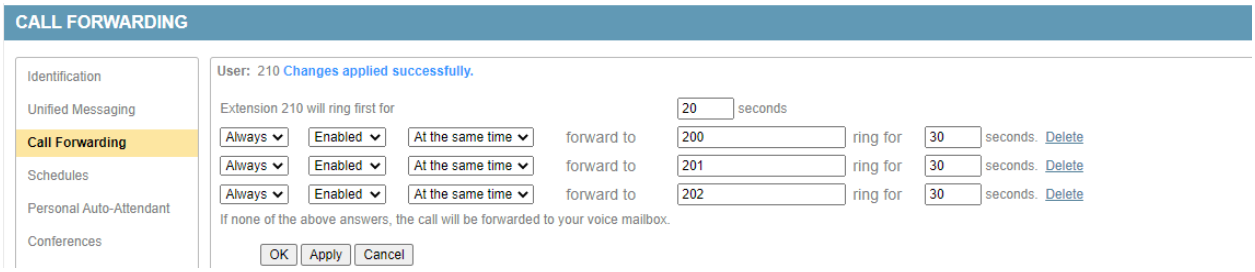
udp/tcp transport changes that could break signaling.

Any inbound INVITEs sent **from the employer sipxcom** will be **To: <sip:5568**, so Alice needs to terminate **5568** on her sipxcom server (similar to a *DID*). The SIP trunk connection alone does not do this.

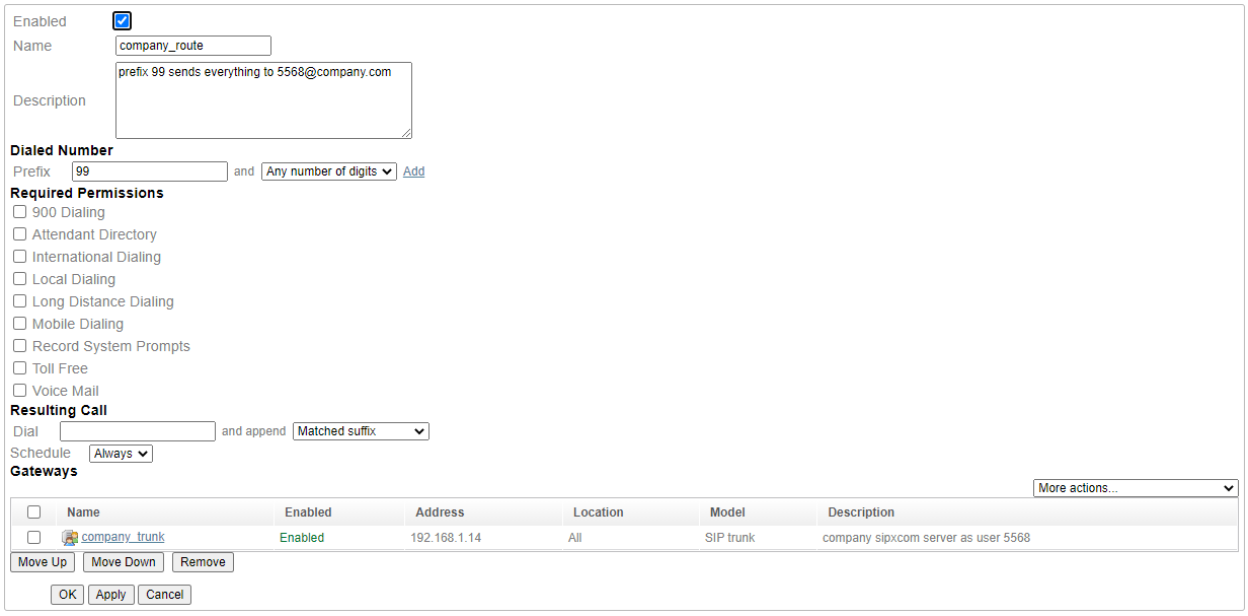
On her home sipxcom server, Alice creates a **phantom** user 210 with 5568 in the Alias field for this purpose.



Next Alice adds Call Forwards under phantom user 210 for 'at the same time' to 200, 201, and 202 – the three bedrooms. She tests this by asking a co-worker registered on the company server to dial 5568, which is successful. She might test again through the PSTN by calling the company with her mobile phone, then dialing her extension 5568 from the company AA.



The final piece is outbound dialing. To do that Alice needs to create a *dial plan* entry on her sipxcom server at home. She navigates to System - Dialing - Dial Plans, then clicks 'Add new rule' and selects a 'Custom' plan. She configures a prefix of 99 and any number of digits to dial the entire matched suffix through the SIP trunk.



After clicking **Apply** to save, Alice can test this by dialing the prefix of 99 and any extension on the company server. For example if a co-worker is at extension 5515 she could dial 995515. To test outbound to the PSTN through that trunk, she would dial it prefixed with 99 as well, like 994235551212.

In her test calls she should verify there is bidirectional audio after the call is established, and that the call remains established longer than 30 seconds.

CHAPTER

TWELVE

REST API REFERENCE

12.1 Overview

A number of APIs have been implemented to facilitate customization in the following areas:

- Phones
- Phone groups
- Gateways
- IVR
- DNS
- Message Waiting Indication (MWI)
- SIP Registrar
- Registrations
- Page groups
- Park orbits
- SIP Proxy
- My Buddy
- Shared Appearance Agent (SAA)
- REST service
- Schedules
- Dial Plan
- Call Detail Records (CDR)
- Servers
- The E911 functionality has been enhanced to provide user location of 911 callers.

12.1.1 About REST

REST (REpresentational State Transfer) represents a new approach to systems architecture and a lightweight alternative to web services. RESTlet is the framework used by sipxconfig for exposing the RESTful API. Through the REST API you can retrieve information about an instance or make configuration changes. Requests are implemented with standard HTTP methods:

- GET to read
- PUT to create
- POST to update
- DELETE to delete

Note: The REST API is served over HTTPS only to ensure data privacy.

12.1.2 REST base URL

The base URL for the REST API is usually:

```
https://host.domain/sipxconfig/rest/
```

There are some resource URIs beneath /api instead:

```
https://host.domain/sipxconfig/api/
```

12.1.3 Using REST with cURL

cURL is a open source linux command line application for transferring data with URLs. Below is an example of using curl to print the content of a phonebook named 'sales' to standard CSV output:

```
curl -k https://superadmin:password@host.domain/sipxconfig/rest/phonebook/sales
```

Another example of placing a call as a regular user:

```
curl -k -X PUT https://200:password@host.domain/sipxconfig/rest/call/{phonenumber}
```

The HTTP **PUT** to the service URL indicates sipxconfig should place a call to {phonenumber}. The call can only be placed with authorized user credentials. It works in the same was as the click-to-call functionality in the User Portal. The user phone will ring first, then when answered the system places a call to {phonenumber}, then connects those two calls together. The SIP signaling is similar to a consultative/attended transfer.

12.1.4 OpenFire APIs

Openfire is a cross platform realtime communication server project based on the XMPP (Jabber) protocol developed by Ignite Realtime. The OpenFire REST API documentation is available on their site.

12.2 Auto Attendant (AA)

12.2.1 View AA List

Resource URI: /rest/auto-attendant

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
autoattendant	The items displayed in the list
name	Auto attendant name
systemId	System ID
specialSelected	Determines whether the AA is active or not.

Specific Response Codes: N/A

HTTP Method: GET Retrieves the list of auto-attendants configured.

Example:

```
# curl -k -X GET https://superadmin:password@192.168.1.31/sipxconfig/rest/auto-
↪attendant/
<autoAttendants>
  <autoAttendant>
    <name>Operator</name>
    <systemId>operator</systemId>
    <specialSelected>false</specialSelected>
  </autoAttendant>
  <autoAttendant>
    <name>After hours</name>
    <systemId>afterhour</systemId>
    <specialSelected>false</specialSelected>
  </autoAttendant>
```

Unsupported HTTP Methods: POST, PUT, DELETE

12.2.2 View or modify AA special mode

Resource URI: /rest/auto-attendant/specialmode

Default Resource Properties The resource is represented by the following properties when the GET is performed:

Property	Description
specialMode	The status of the AA special mode. Displays **true** if the AA is on and **false** if the AA is off.

Specific Response Codes: N/A

HTTP Method: GET Displays if the auto attendant is activated or not.

Example:

```
# curl -k -X GET https://superadmin:password@192.168.1.31/sipxconfig/rest/auto-
↪attendant/specialmode
<specialAttendant>
  <specialMode>false</specialMode>
```

HTTP Method: PUT The status is set to true and the special mode is activated.

Example:

```
# curl -k -X PUT -H "Content-Type: application/json" -d '{"specialMode":"true"}' ␣
↪https://superadmin:password@192.168.1.31/sipxconfig/rest/auto-attendant/specialmode
```

HTTP Method: DELETE The status is set to false and the special mode is deactivated.

Example:

```
# curl -k -X DELETE -H "Content-Type: application/json" -d '{"specialMode":"true"}' ␣
↪https://superadmin:password@192.168.1.31/sipxconfig/rest/auto-attendant/specialmode
```

Unsupported HTTP Methods: POST

12.2.3 Setting an AA in special mode

Resource URI: /rest/auto-attendant/{attendant}/special

Default Resource Properties N/A

Specific Response Codes:

- Error 400 - when the {attendant} is not found on PUT or DELETE.
- Error 409 - when the special mode is true on DELETE.

HTTP Method: PUT The auto attendant is marked as special.

HTTP Method: DELETE Remove the attendant special mode.

Unsupported HTTP Method: GET, POST

12.2.4 Enable an AA

Resource URI: /rest/auto-attendant/livemode/{code}

Default Resource Properties N/A

Specific Response Codes: N/A

HTTP Method: PUT The auto attendant with the specified code is enabled. Note that hte code represents the phones extension.

HTTP Method: DELETE The auto attendant with the specified code is disabled. Note that hte code represents the phones extension.

Unsupported HTTP Method: GET, POST

12.3 Branch

12.3.1 View or modify branches

Resource URI: /rest/branch

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
totalResults	The total number of results.
currentPage	Number of the current page.
totalPages	The number of total pages.
resultPerPage	Number of results per page.
ID	Unique identification number of the branch.
name	Branch name
description	Short description provided by the user.
address	The complete address of the branch.
street	The name of the street.
city	The name of the city.
country	The name of the country.
state	The name of the state.
zip	The postal zip code.
officeDesignation	The mail stop field.
phoneNumber	The phone number of the branch.
faxNumber	The fax number of the branch.

Specific Response Codes: Error 400 - Wrong ID when updating the branch

HTTP Method: GET Retrieves a list of branches defined in the system.

Example:

```
# curl -k -X GET https://superadmin:password@192.168.1.31/sipxconfig/rest/branch
<branch>
  <metadata>
    <totalResults>1</totalResults>
    <currentPage>1</currentPage>
    <totalPages>1</totalPages>
    <resultsPerPage>1</resultsPerPage>
  </metadata>
  <branches>
    <branch>
      <id>1</id>
      <name>whitehouse</name>
      <description>location description field</description>
      <address>
        <id>1</id>
        <street>1600 Pennsylvania Avenue NW</street>
        <city>Washington</city>
        <country>US</country>
        <state>DC</state>
        <zip>20500</zip>
      </address>
    </branch>
  </branches>
</branch>
```

HTTP Method: PUT Adds a new branch. The ID is automatically generated and any value entered is ignored.

Example:

```
# curl -k -X PUT -H "Content-Type: application/json" -d '{"branch":{"name":
↪"libofcongress","description":"library of congress","address":{"street":"101␣
↪Independence Ave SE","city":"Washington","state":"DC","zip":"20540"}}}'  https://
↪superadmin:password@192.168.1.31/sipxconfig/rest/branch
```

(continues on next page)

(continued from previous page)

```
<?xml version="1.0" encoding="UTF-8"?><response><code>SUCCESS_CREATED</code><message>
↪Created</message><data><id>3</id></data></response>
```

Unsupported HTTP Method: DELETE

12.3.2 View or modify a branch ID

Resource URI: /rest/branch/{id}

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
branch	The branch information is the same as /branch

Specific Response Codes: Error 400 - wrong ID when updating the branch

HTTP Method: GET Retrieves information on the branch with the specified ID.

Example:

```
# curl -k -X GET https://superadmin:password@192.168.1.31/sipxconfig/rest/branch/1
<branch>
  <id>1</id>
  <name>whitehouse</name>
  <description>location description field</description>
  <address>
    <id>1</id>
    <street>1600 Pennsylvania Avenue NW</street>
    <city>Washington</city>
    <country>US</country>
    <state>DC</state>
    <zip>20500</zip>
    <officeDesignation>ovaloffice</officeDesignation>
  </address>
  <phoneNumber>4235551212</phoneNumber>
  <faxNumber>4235552323</faxNumber>
</branch>
```

HTTP Method: PUT Updates the branch with the specified ID. Uses the same XML as for creation.

HTTP Method: DELETE Removes branch with the specified ID.

Unsupported HTTP Method: POST

12.4 DNS

12.4.1 View DNS settings

Resource URI: /api/dns/settings

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
path	Path to the setting.
type	Setting type. Possible options are **string**, **boolean**, or **enum**.
options	Availble setting options.
value	The current selected option of the setting.
defaultValue	The default value of the setting.
label	The setting label.
description	Short description provided by the user.

Specific Response Codes: N/A

HTTP Method: GET Retrieves the settings for all DNS entries in the system.

Example:

```
# curl -k -X GET https://superadmin:password@192.168.1.31/sipxconfig/api/dns/settings
{"settings":[{"path":"named-config/forwarders/forwarder_0","type":"string","options
↪":null,"value":"192.168.1.31","defaultValue":null,"label":"Primary External DNS␣
↪server","description":"DNS server in your company or your ITSP. Can also be a␣
↪publicly available DNS server like 8.8.8.8."},{"path":"named-config/forwarders/
↪forwarder_1","type":"string","options":null,"value":null,"defaultValue":null,"label
↪":"Secondary External DNS server","description":"In the event the primary DNS␣
↪server is unavailable, system will use this server."},{"path":"named-config/
↪forwarders/forwarder_2","type":"string","options":null,"value":null,"defaultValue
↪":null,"label":"Additional External DNS server","description":null},{"path":"named-
↪config/forwarders/forwarder_3","type":"string","options":null,"value":null,
↪"defaultValue":null,"label":"Additional External DNS server","description":null},{
↪"path":"named-config/forwarders/forwarder_4","type":"string","options":null,"value
↪":null,"defaultValue":null,"label":"Additional External DNS server","description
↪":null},{"path":"acl/ips","type":"string","options":null,"value":"192.168.1.31,172.
↪16.0.0/12,192.168.0.0/16,10.0.0.0/8,127.0.0.0/8","defaultValue":"192.168.1.31,172.
↪16.0.0/12,192.168.0.0/16,10.0.0.0/8,127.0.0.0/8","label":"Allow Recursion ACL",
↪"description":"Groups of hosts (comma separated values of IP addresses or subnet)␣
↪allowed to make recursive queries on the nameserver. <br/>Leave empty for allowing␣
↪all hosts to perform recursive queries on the nameserver."},{"path":"sys/unmanaged",
↪"type":"boolean","options":null,"value":"0","defaultValue":"0","label":"Unmanaged␣
↪Service","description":"Company or ITSP DNS servers to resolve ALL names instead of␣
↪local DNS servers."},{"path":"sys/unmanaged_servers/unmanaged_0","type":"string",
↪"options":null,"value":null,"defaultValue":null,"label":"Primary Unmanaged DNS␣
↪server","description":"DNS server in your company or your ITSP. Can also be a␣
↪publicly available DNS server like 8.8.8.8."},{"path":"sys/unmanaged_servers/
↪unmanaged_1","type":"string","options":null,"value":null,"defaultValue":null,"label
↪":"Secondary Unmanaged DNS server","description":"In the event the primary DNS␣
↪server is unavailable, system will use this server."},{"path":"sys/unmanaged_
↪servers/unmanaged_2","type":"string","options":null,"value":null,"defaultValue
↪":null,"label":"Additional Unmanaged DNS server","description":null},{"path":"sys/
↪unmanaged_servers/unmanaged_3","type":"string","options":null,"value":null,
↪"defaultValue":null,"label":"Additional Unmanaged DNS server","description":null}]}
```

12.4.2 View DNS settings from path

Resource URI: /api/dns/settings/{settingPath}

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
setting	The dns setting related information is similar to the one described under /dns/settings.

Specific Response Codes: N/A

HTTP Method: GET Retrieves the DNS settings from the speicifed path.

Example:

```
# curl -k -X GET https://superadmin:password@192.168.1.31/sipxconfig/api/dns/settings/
↪named-config/forwarders
{"settings":[{"path":"named-config/forwarders/forwarder_0","type":"string","options
↪":null,"value":"192.168.1.31","defaultValue":null,"label":"Primary External DNS␣
↪server","description":"DNS server in your company or your ITSP. Can also be a␣
↪publicly available DNS server like 8.8.8.8."},{"path":"named-config/forwarders/
↪forwarder_1","type":"string","options":null,"value":null,"defaultValue":null,"label
↪":"Secondary External DNS server","description":"In the event the primary DNS␣
↪server is unavailable, system will use this server."},{"path":"named-config/
↪forwarders/forwarder_2","type":"string","options":null,"value":null,"defaultValue
↪":null,"label":"Additional External DNS server","description":null},{"path":"named-
↪config/forwarders/forwarder_3","type":"string","options":null,"value":null,
↪"defaultValue":null,"label":"Additional External DNS server","description":null},{
↪"path":"named-config/forwarders/forwarder_4","type":"string","options":null,"value
↪":null,"defaultValue":null,"label":"Additional External DNS server","description
↪":null}]}
```

HTTP Method: PUT Updates the settings of the DNS server from the specified path. PUT data is plain text.

HTTP Method: DELETE Deletes the settings of the DNS server from the specified path.

Unsupported HTTP Method: POST

12.4.3 View DNS Advisor results

Resource URI: /api/dns/advisor/server/{serverId}

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
Missing naptr records	List of the missing NAPTR records, if any.
Missing A records	List of missing A records, if any.

Specific Response Codes: N/A

HTTP Method: GET Checks the DNS settings and if the settings are correct, no result is returned. Otherwise it retrieves the missing configurations.

Example:

```
# curl -k -X GET https://superadmin:password@192.168.1.31/sipxconfig/api/dns/advisor/
↪server/1

;; Missing naptr records
home.mattkeys.net. IN NAPTR \d 0 "s" "SIP+D2U" "" _sip._udp
home.mattkeys.net. IN NAPTR 1 0 "s" "SIP+D2T" "" _sip._tcp
```

(continues on next page)

(continued from previous page)

```
;; Missing a records
sipxcom1.home.mattkeys.net  IN A 192.168.1.31
```

Unsupported HTTP Method: PUT, POST, DELETE

12.5 e911 (eZuce Uniteme only)

This feature and API resource only applies to eZuce Uniteme. There is a workaround for sipxcom to get a similar result.

Note: The workaround isn't valid when using sipXbridge / SIP trunks because the webui will prevent adding multiple trunks to the same provider (IP or FQDN).

If using unmanaged gateways the workaround is to force outbound caller ID on the gateways utilized in the emergency dial plan. You can create multiple unmanaged gateways that point to the same IP address. Branch configurations can then be used to specify the outbound gateway, and permissions can then be used to secure the dial plan entries.

For example, a building with four floors could be configured as four branches – floor1, floor2, floor3, and floor4. Four gateways pointing to the same IP would be created for those four floors – e911floor1, e911floor2, e911floor3, and e911floor4. On each gateway specify the respective branch, and do not configure the gateways as "shared". Next force the outbound caller ID on each gateway, which is the ELIN that corresponds to the floor. Finally configure users in their respective floor branch, and add those 4 gateways to the emergency dial plan.

Note: It's a good idea to have a shared gateway as the last option in the emergency dial plan gateway list as a failsafe / last resort path.

12.5.1 About e911

The Enhanced 911 (E911) functionality has been implemented for handling emergency situations. Administrators can perform the required set up in order for Uniteme and Unite users to be able to call the 911 number when needed. The functionality uses location based technology to pin point the location of 911 callers and connect them to the appropriate public resources.

The system to automatically associates a location with the origin of the call. This location may be a physical address or other geographic reference information such as X/Y GPS coordinates. In sipXcom administrators are able to define physical locations and link them to users. Physical locations have a DID/ELIN (Emergency Location Identification Number) that will be sent out to the 911 dispatcher. Based on the called ID sent operators will be able to dispatch emergency services directly to the user's location.

Note:

- E911 is a system used only in North America.
- Calls made to other emergency telephone numbers are not supported.

12.5.2 Using the e911 REST API

sipXcom also defines a REST API to perform CRUD operations on the Emergency Resource Location (ERL) table and also to link users to locations. This API may be used by third parties in order to update the ERL data in the PS-ALI database (Private Switch/Automatic Location Identification). It also helps administrators update in bulk the locations table and link users to locations. The following resources for the E911 API are only available for users with administration rights:

Emergency Resource Location (ERL)

- View list of ERLs
- Filter ERLs by ELIN
- Filter ERLs by user name
- Filter ERLs by user groups
- Filter ERLs by the number of assigned phones
- Update ERLs for one or multiple phones
- Update ERLs for one or multiple phone groups

Registrations

- View registrations for an IP
- View registrations for a Line/Extension

Phones

- View list of phones
- View list of phones changed since dd/mm/yy

12.5.3 Emergency Resource Location (ERL)

Resource URI: /rest/erls

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
elin	ELIN number.
location	Caller location.
addressInfo	Address details.
description	Optional description.

Specific Response Codes: N/A

HTTP Method: GET Returns a list with all the ERLs defined in the system.

Example:

```
# curl -k -X GET https://superadmin:password@192.168.1.14/sipxconfig/rest/erls

<?xml version="1.0" encoding="UTF-8"?><e911Locations><e911Location><location>123 Test␣
↪Street, Chattanooga, TN, 37412</location><elin>4235551212</elin><addressInfo>123␣
↪Test Street, Chattanooga, TN, 37412</addressInfo><description>test</description></
↪e911Location></e911Locations>
```

HTTP Method: PUT Save a list of ERLs.

Example:

```
bar
```

HTTP Method: DELETE Delete the ERL with the specified ELIN

Unsupported HTTP Method: POST

12.5.4 Filter ERLs by ELIN

Resource URI: /rest/erl/elin/{elin}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
elin	ELIN number.
location	Caller location.
addressInfo	Address details.
description	Optional description.

Specific Response Codes: N/A

HTTP Method: GET Returns the ERLs with the specified ELIN.

Example:

```
foo
```

HTTP Method: PUT Update the ERL with the specified ELIN

HTTP Method: DELETE Delete the ERL with the specified ELIN

Unsupported HTTP Method: POST

12.5.5 Filter ERLs by user name

Resource URI: /rest/erl/user/{username}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
location	The location
elin	The ELIN number.
addressInfo	Address details.
description	Optional description.

Specific Response Codes: N/A

HTTP Method: GET Returns the ERL linked to the user identified by username. Data is plain text and represents the ELIN of the ERL.

Example:

```
foo
```

HTTP Method: PUT Update the ERL of the user. PUT data is plain text and represents the ERL.

Example:

```
bar
```

HTTP Method: DELETE Set the user ERL to none.

Example:

```
foo
```

Unsupported HTTP Method: POST

12.5.6 Filter ERLs by user groups

Resource URI: /rest/erl/group/{groupName}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
location	The location
elin	The ELIN number.
addressInfo	Address details.
description	Optional description.

Specific Response Codes: N/A

HTTP Method: GET Returns the ERL identified with the user group.

Example:

```
foo
```

HTTP Method: PUT Updates the ERL of the user group.

Example:

```
bar
```

Unsupported HTTP Method: POST

12.5.7 Filter ERLs by the number of assigned phones

Resource URI: /rest/erl/phone/{serial_number}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
location	The location
elin	The ELIN number.
addressInfo	Address details.
description	Optional description.
serial	phone serial

Specific Response Codes: N/A

HTTP Method: GET Retrieves a list with the locations for the phone(s).

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST, DELETE

12.5.8 Update ERLs for one or multiple phones

Resource URI: /rest/erl/phones

Default Resource Properties: N/A

Specific Response Codes: Error 400 if wrong ELIN or serial is specified.

HTTP Method: PUT Update locations for one or more phones.

Example:

```
bar
```

Unsupported HTTP Method: GET, POST, DELETE

12.5.9 Update ERLs for one or multiple phone groups

Resource URI: /rest/erl/phonegroup/{groupName}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
location	The location
elin	The ELIN number.
addressInfo	Address details.
description	Optional description.

Specific Response Codes: N/A

HTTP Method: GET Retrieves locations for phone groups.

Example:

```
foo
```

HTTP Method: PUT Updates locations for phone groups.

Example:

```
bar
```

HTTP Method: DELETE Deletes location for phone groups.

Unsupported HTTP Method: POST

12.6 Gateways

12.6.1 View all gateways

Resource URI: /api/gateways

Default Resource Properties The resource is prepresented by the following properties when the GET request is performed.

Property	Description
id	Gateway unique identification number.
name	Gateway name.
description	Short description provided by the user.
model	The model of the gateway.
enabled	Displays **true** if enabled, **false** if it is disabled.
address	The gateway IP or FQDN address.
addressPort	The gateway port number.
outboundPort	The gateway outbound port number.
addressTransport	The transport protocol to use.
shared	Displays **true** if enabled, **false** if it is not shared.
useInternalBridge	Displays **true** if using sipxbridge, **false** if it is not.
anonymous	Displays **true** if caller ID is blocked, **false** if it is not.
ignoreUserInfo	Displays **true** if 'ignore user caller id' is enabled, **false** if it is not.
transformUserExtensions	Displays **true** if 'transform extension' is enabled, **false** if it is not.
keepDigits	Number of ext digits that are kept before adding the caller ID prefix.

Specific Response Codes: N/A

HTTP Method: GET Retrieves all the gateways defined.

Example:

```
# curl -k -X GET https://superadmin:password@192.168.1.31/sipxconfig/api/gateways
{"gateways":[{"id":1,"name":"my_siptrunk","serialNo":null,"deviceVersion":null,
↪"description":"DIDs 4235550000 through 4235551000","model":{"modelId":
↪"sipTrunkStandard","label":"SIP trunk","vendor":null,"versions":null},"enabled
↪":true,"address":"192.168.1.14","addressPort":5060,"outboundAddress":null,
↪"outboundPort":5060,"addressTransport":"tcp","prefix":null,"shared":true,
↪"useInternalBridge":true,"branch":null,"callerAliasInfo":{"defaultCallerAlias":null,
↪"anonymous":false,"ignoreUserInfo":false,"transformUserExtension":false,"addPrefix
↪":null,"keepDigits":0,"displayName":null,"urlParameters":null}}]}
```

12.6.2 Filter gateways by model

Resource URI: /api/gateways/models

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
modelId	Gateway model.
label	Gateway label.
vendor	Gateway model vendor.

Specific Response Codes: N/A

HTTP Method: GET Retrieves all gateway models available in the database.

Example:

```
# curl -k -X GET https://superadmin:password@192.168.1.31/sipxconfig/api/gateways/
↪models
{"models":[{"modelId":"acmeGatewayStandard","label":"Acme 1000","vendor":"acme",
↪"versions":null},{"modelId":"audiocodesMP1X4_4_FXO","label":"AudioCodes MP114 FXO",
↪"vendor":"AudioCodes","versions":["audiocodes6.0","audiocodes5.8","audiocodes5.6",
↪"audiocodes5.4","audiocodes5.2","audiocodes5.0"]},{"modelId":"audiocodesMP1X8_8_FXO
↪","label":"AudioCodes MP118 FXO","vendor":"AudioCodes","versions":["audiocodes6.0",
↪"audiocodes5.8","audiocodes5.6","audiocodes5.4","audiocodes5.2","audiocodes5.0"]},{
↪"modelId":"audiocodesMediant1000","label":"AudioCodes Mediant 1000 PRI","vendor":
↪"AudioCodes","versions":["audiocodes6.0","audiocodes5.8","audiocodes5.6",
↪"audiocodes5.4","audiocodes5.2","audiocodes5.0"]},{"modelId":"audiocodesMediant2000
↪","label":"AudioCodes Mediant 2000 PRI","vendor":"AudioCodes","versions":[
↪"audiocodes6.0","audiocodes5.8","audiocodes5.6","audiocodes5.4","audiocodes5.2",
↪"audiocodes5.0"]},{"modelId":"audiocodesMediant3000","label":"AudioCodes Mediant␣
↪3000 PRI","vendor":"AudioCodes","versions":["audiocodes6.0","audiocodes5.8",
↪"audiocodes5.6","audiocodes5.4","audiocodes5.2","audiocodes5.0"]},{"modelId":
↪"audiocodesMediantBRI","label":"AudioCodes Mediant 1000 BRI","vendor":"AudioCodes",
↪"versions":["audiocodes6.0","audiocodes5.8","audiocodes5.6","audiocodes5.4",
↪"audiocodes5.2","audiocodes5.0"]},{"modelId":"audiocodesTP260","label":"AudioCodes␣
↪TP260","vendor":"AudioCodes","versions":["audiocodes6.0","audiocodes5.8",
↪"audiocodes5.6","audiocodes5.4","audiocodes5.2","audiocodes5.0"]},{"modelId":
↪"genericGatewayStandard","label":"Unmanaged gateway","vendor":null,"versions":null},
↪{"modelId":"sipTrunkStandard","label":"SIP trunk","vendor":null,"versions":null}]}
```

Unsupported HTTP Method: PUT, POST, DELETE

12.6.3 View or modify gateway ID

Resource URI: /gateways/{gatewayId}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
gateway	The gateway related inforamtion is the same as the /gateways resource.

Specific Response Codes: N/A

HTTP Method: GET Retrieves information on the gateway with the specified ID.

Example:

```
# curl -k -X GET https://superadmin:password@192.168.1.31/sipxconfig/api/gateways/1
{"id":1,"name":"my_siptrunk","serialNo":null,"deviceVersion":null,"description":"DIDs␣
↪4235550000 through 4235551000","model":{"modelId":"sipTrunkStandard","label":"SIP␣
↪trunk","vendor":null,"versions":null},"enabled":true,"address":"192.168.1.14",
↪"addressPort":5060,"outboundAddress":null,"outboundPort":5060,"addressTransport":
↪"tcp","prefix":null,"shared":true,"useInternalBridge":true,"branch":null,
↪"callerAliasInfo":{"defaultCallerAlias":null,"anonymous":false,"ignoreUserInfo
↪":false,"transformUserExtension":false,"addPrefix":null,"keepDigits":0,"displayName
↪":null,"urlParameters":null}}
```

HTTP Method: PUT Updates the gateway with the specified ID. Uses the same XML as for creation.

Example:

```
foo
```

HTTP Method: POST Creates a new gateway with the specified ID.

Example:

```
bar
```

HTTP Method: DELETE Removes the gateway specified by ID.

Example:

```
foo
```

Unsupported HTTP Method: N/A

12.6.4 View all settings of a gateway ID

Resource URI: /api/gateways/{gatewayId}/settings

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
path	Path to the setting.
type	Setting type. Possible options are **string**, **boolean**, or **enum**.
options	Available setting options.
value	The current selected option of the setting.
label	Setting label.
description	Short description provided by the user.

Specific Response Codes: N/A

HTTP Method: GET Retrieves settings for the specified gateway ID.

Example:

```
# curl -k -X GET https://superadmin:password@192.168.1.31/sipxconfig/api/gateways/2/
↪settings
{"settings":[{"path":"itsp-account/user-name","type":"string","options":null,"value":
↪"5568","defaultValue":null,"label":null,"description":null},{"path":"itsp-account/
↪authentication-user-name","type":"string","options":null,"value":null,"defaultValue
↪":null,"label":null,"description":null},{"path":"itsp-account/password","type":
↪"string","options":null,"value":"password","defaultValue":null,"label":null,
↪"description":null},{"path":"itsp-account/register-on-initialization","type":
↪"boolean","options":null,"value":"true","defaultValue":"false","label":null,
↪"description":null},{"path":"itsp-account/itsp-proxy-address","type":"string",
↪"options":null,"value":"192.168.1.14","defaultValue":"192.168.1.14","label":null,
↪"description":null},{"path":"itsp-account/use-global-addressing","type":"boolean",
↪"options":null,"value":"true","defaultValue":"true","label":null,"description":null}
```

(continues on next page)

(continued from previous page)

Unsupported HTTP Method: PUT, POST, DELETE

12.6.5 View or modify a setting for a gateway ID

Resource URI: /api/gateways/{gatewayId}/settings/{path:.*}

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
gateway	The gateway related information is the same as the /gateways/{gatewayId} resource.

Specific Response Codes: N/A

HTTP Method: GET Retrieve the setting specified in the path for the gateway ID.

Example:

```
foo
```

HTTP Method: PUT Updates the setting specified in the path for the gateway ID. PUT data is plain text.

Example:

```
bar
```

HTTP Method: DELETE Deletes the setting specified in the path for the gateway ID.

Example:

```
foo
```

Unsupported HTTP Method: POST

12.6.6 View port settings for a gateway ID

Resource URI: /api/gateways/{gatewayId}/port/{portId}/settings

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
setting	The port setting related information is similar to the one described under /gateways/{gatewayId}/settings.

HTTP Method: GET View port settings for the gateway with the specified ID.

Example:

```
bar
```

HTTP Method: PUT Updates the port settings for the gateway with the specified ID. PUT data is plain text.

Example:

```
foo
```

HTTP Method: DELETE Deletes the port settings for the gateway with the specified ID.

Example:

```
bar
```

Unsupported HTTP Method: POST

12.6.7 View or modify port settings

Resource URI: /api/gateways/{gatewayId}/port/{portId}/settings/{path:.*}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
setting	The port setting related information is similar to the one described under /gateway/settings.

Specific Response Codes: N/A

HTTP Method: GET Retrieves the port settings of the gateway with the specified ID.

Example:

```
foo
```

HTTP Method: PUT Updates the settings of the port. PUT data is plain text.

Example:

```
bar
```

HTTP Method: DELETE Deletes the settings of the port.

Example:

```
foo
```

Unsupported HTTP Method: POST

12.7 IVRs

12.7.1 View IVR Settings

Resource URI: /ivr/settings

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
path	Path to the setting
type	The setting type. Possible options are **string**, **boolean**, or **enum**.
options	Available setting options.
value	The current selected option of the setting.
defaultValue	The default value of the setting.
label	The setting label.
description	The description provided by the user.

Specific Response Codes: N/A

HTTP Method: GET Retrieves all the IVR settings.

Example:

```
bar
```

HTTP Method: PUT Updates the settings of the gateway. PUT data is plain text.

Example:

```
foo
```

HTTP Method: DELETE Deletes the settings of the gateway.

Example:

```
bar
```

Unsupported HTTP Method: POST

12.8 Licensing (Uniteme only)

Resource URI: /api/license

Default Resource Properties N/A

Specific Response Codes: N/A

12.8.1 View license information

HTTP Method: GET Retrieves the current license file.

Example:

```
# curl -k -X GET https://superadmin:password@192.168.1.14/sipxconfig/api/license/
{"uid":"","domain":"home.mattkeys.net","version":"","expire":"03-Jan-2026","support":"
↪","licenseType":"Deprecated OpenUC Reach","users":-1,"mobileDevices":-1,"company":
↪"Matts Lab","contactEmail":"mkeys@email.domain","contactName":"","contactPhone":""}
```

12.8.2 Modify license UID

Set the universal ID of the license.

HTTP Method: POST Used to update the license UID

Example:

```
# curl -k -X POST https://superadmin:12345678@10.4.0.103/sipxconfig/api/license/
↪508316691110519
{"uid":"508316691110519","domain":"gabi.test","version":"19.08","expire":"31-Dec-2039
↪","support":"31-Dec-2039","licenseType":"Subscription","users":500,"mobileDevices
↪":500,"company":"developers","contactEmail":"martin.harcar@ezuce.com","contactName":
↪"martin","contactPhone":"454614465465"}
```

Unsupported HTTP Method: PUT, DELETE

12.9 Message Waiting Indication (MWI)

Resource URI: /mwi/settings

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
path	
type	
value	
defaultValue	
description	

Specific Response Codes: N/A

HTTP Method: GET Retrieves all MWI settings.

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST, DELETE

12.9.1 View or modify MWI settings

Resource URI: /mwi/settings/{settingPath}

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieves all MWI settings of the specified path.

Example:

```
bar
```

HTTP Method: PUT Updates the MWI settings of the specified path. PUT data is plain text.

Example:

```
foo
```

HTTP Method: DELETE Deletes the MWI setting from the specified path.

Example:

```
bar
```

12.10 Music On Hold (MOH)

12.10.1 View MOH settings

Resource URI: /moh/settings

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
dnsManager	Name of the DNS manager
dnsTestContext	

Specific Response Codes: N/A

HTTP Method: GET Retrieves all MOH settings.

Example:

```
# curl -k -X GET https://superadmin:password@192.168.1.31/sipxconfig/api/moh/settings
{"settings":[{"path":"moh-config/MOH_SOURCE","type":"enum","options":{"FILES_SRC":
↪"System Music Directory","NONE":"None","SOUNDCARD_SRC":"Sound Card"},"value":"FILES_
↪SRC","defaultValue":"FILES_SRC","label":"Music On Hold Source","description":
↪"Selects the source of the on hold music. If set to <em>System Music Directory</em>␣
↪the server will play all the music files from the system directory on a continuous␣
↪rotating basis. Setting it to <em>Sound Card</em> will stream audio from the local␣
↪sound card. <em>None</em> option will disable music on hold."}]}
```

Unsupported HTTP Method: PUT, POST, DELETE

12.10.2 View or modify MOH settings

Resource URI: /moh/settings/{settingPath}

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
path	The moh source
type	
options	Group of options defined for the moh source

Specific Response Codes: N/A

HTTP Method: GET Retrieves the MOH settings for the specified path.

Example:

```
# curl -k -X GET https://superadmin:password@192.168.1.31/sipxconfig/api/moh/settings/
↪moh-config/MOH_SOURCE
{"path":"moh-config/MOH_SOURCE","type":"enum","options":{"FILES_SRC":"System Music␣
↪Directory","NONE":"None","SOUNDCARD_SRC":"Sound Card"},"value":"FILES_SRC",
↪"defaultValue":"FILES_SRC","label":"Music On Hold Source","description":"Selects␣
↪the source of the on hold music. If set to <em>System Music Directory</em> the␣
↪server will play all the music files from the system directory on a continuous␣
↪rotating basis. Setting it to <em>Sound Card</em> will stream audio from the local␣
↪sound card. <em>None</em> option will disable music on hold."}
```

HTTP Method: PUT Updates the MOH settings of the specified path. PUT data is plain text.

Example:

```
bar
```

HTTP Method: DELETE Reverts the setting to the default value.

Unsupported HTTP Method: POST

12.10.3 View or upload MOH prompt files

Resource URI: /moh/prompts

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
dnsManager	Name of the DNS Manager
dnsTestContext	

Specific Response Codes: N/A

HTTP Method: GET Retrieves all MOH prompt files.

Example:

```
foo
```

HTTP Method: POST Uploads a new MOH prompt file.

Example:

```
bar
```

Unsupported HTTP Method: PUT, DELETE

12.10.4 Download MOH prompt files

Resource URI: /moh/prompts/{promptName}

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Downloads the MOH prompt based on the file name. Example: */moh/prompts/default.wav*.

Example:

```
foo
```

HTTP Method: DELETE Deletes the specified prompt message.

Unsupported HTTP Method: PUT, POST

12.10.5 Stream to MOH Prompt

Resource URI: /moh/prompts/{promptName}/stream

Default Resource Properties: N/A

Specific Response Codes: N/A

HTTP Method: GET Can be called by clients to stream prompts.

Example:

```
bar
```

Unsupported HTTP Method: PUT, POST, DELETE

12.11 My Buddy

12.11.1 View My Buddy Settings

Resource URI: /imbot/settings

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieves a list of all My Buddy settings defined.

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST, DELETE

12.11.2 View or modify My Buddy settings

Resource URI: /imbot/settings/{settingPath}

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieves My Buddy options for the setting from the specified path.

Example:

```
bar
```

HTTP Method: PUT Modifies My Buddy options for the specified path.

Example:

```
foo
```

Unsupported HTTP Method: POST

12.12 Page Groups

12.12.1 View or create page groups

Resource URI: /pagegroups

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
ID	Unique identification number of the page group
enabled	The status of the page group. Displays **true** if enabled, **false** if disabled.
GroupNumber	The group number of the page group.
timeout	The timeout value measured in seconds.
sound	Name of the file representing the sound to be played.
description	Short description provided by the user.

Specific Response Codes: N/A

HTTP Method: GET Retrieves all the page groups defined in the system.

Example:

```
bar
```

HTTP Method: POST Creates a new page group.

Example:

```
foo
```

Unsupported HTTP Method: PUT, DELETE

12.12.2 View or modify page groups by group ID

Resource URI: /pagegroups/{pagegroupId}

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
ID	Unique identification number of the page group.
enabled	The status of the page group. Displays **true** if enabled, **false** if disabled.
GroupNumber	The group number of the page group.
timeout	The timeout value measured in seconds.
sound	Name of the file representing the sound to be played.
description	Short description provided by the user.

Specific Response Codes: N/A

HTTP Method: GET Retrieves page groups with the specified ID.

Example:

```
foo
```

HTTP Method: PUT Modifies a page group with the specified ID.

Example:

```
bar
```

HTTP Method: DELETE Deletes the page group specified by ID.

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST

12.12.3 Manage page group services

Resource URI: /pagegroups/settings

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
path	The path of the page group
type	The type of the page group. Possible options are **string**, **boolean**, or **enum**.
value	The value of the field.
defaultValue	The default value of the field.
label	The label of the page group.

Specific Response Codes: N/A

HTTP Method: GET Retrieves options for the page groups in the system.

Example:

```
foo
```

HTTP Method: PUT Modifies options for the page groups in the system.

Example:

```
bar
```

HTTP Method: DELETE Deletes the paging group specified by path.

Unsupported HTTP Method: PUT, POST

12.12.4 View or create new prompt message

Resource URI: /pagegroups/prompts

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieves a list of page group prompts.

Example:

```
foo
```

HTTP Method: POST Uploads a new page group prompt message.

Unsupported HTTP Method: PUT, DELETE

12.12.5 Download page group prompts

Resource URI: /pagegroups/prompts/{promptName}

Default Resource Properties: N/A

Specific Response Codes: N/A

HTTP Method: GET Downloads prompt specified by file name.

Example:

```
bar
```

Unsupported HTTP Method: PUT, POST

12.12.6 Stream the page group prompt

Resource URI: /pagegroups/prompts/{promptName}/stream

Default Resource Properties: N/A

Specific Resource Codes: N/A

HTTP Method: GET Start the data stream of {promptName}.

Unsupported HTTP Method: PUT, POST, DELETE

12.13 Park Orbits

12.13.1 View park orbits

Resource URI: /orbits

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Downloads prompts based upon file name.

Example:

```
foo
```

HTTP Method: POST Uploads a park orbit prompt based upon file name.

Example:

```
bar
```

Unsupported HTTP Method: PUT, DELETE

12.13.2 View or modify park orbits

Resource URI: /orbits/{orbitId}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieves the park orbit settings of the specified ID.

Example:

```
foo
```

HTTP Method: PUT Modifies the park orbit settings of the specified ID.

Example:

```
bar
```

HTTP Method: DELETE Deletes the park orbit specified by ID.

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST

12.13.3 Manage park orbit options

Resource URI: /orbits/{orbitId}/settings

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieves service options for the specified park orbit ID.

Example:

```
bar
```

HTTP Method: PUT Modifies service options for the specified park orbit ID.

Example:

```
foo
```

HTTP Method: DELETE Deletes the options for the specified park orbit ID.

Example:

```
bar
```

Unsupported HTTP Method: PUT, POST

12.13.4 Manage the park orbit service

Resource URI: /orbits/{orbitId}/settings

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieves general service options.

Example:

```
foo
```

HTTP Method: PUT Modifies the general service options.

Example:

```
bar
```

HTTP Method: DELETE Deletes the service options.

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST

12.13.5 View or create new prompt message

Resource URI: /orbits/prompts

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieves a list of orbit prompts.

Example:

```
bar
```

HTTP Method: POST Uploads a new prompt message.

Unsupported HTTP Method: PUT, DELETE

12.13.6 Download park orbit prompts

Resource URI: /orbits/prompts/{promptName}

Default Resource Properties: N/A

Specific Response Codes: N/A

HTTP Method: GET Downloads the prompt of the specified file name.

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST

12.13.7 Stream the park orbit prompt

Resource URI: /orbits/prompts/{promptName}/stream

Default Resource Properties: N/A

Specific Response Codes: N/A

HTTP Method: GET

Example:

```
bar
```

Unsupported HTTP Method: PUT, POST, DELETE

12.14 Permissions

12.14.1 View or create permissions

Resource URI: /permission

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
totalResults	The total number of results.
currentPage	The page you are currently viewing.
totalPages	The number of total pages.
resultsPerPage	Number of permissions displayed on each page.
name	Name of the permission.
label	Alternative name of the description.
defaultValue	Default value. Displays **true** if enabled, **false** if disabled.
type	The type. Possible options are **string**, **boolean**, or **enum**.
builtIn	

Filtering Parameters:

Parameter	Description
page	Required. The requested page size.
pagesize	Required. The number of results to be displayed per page.
sortdir	Optional, forward or reverse
sortby	Optional, name or description

Specific Response Codes: N/A

HTTP Method: GET Retrieves a list of all permissions

Example:

```
foo
```

HTTP Method: PUT Adds a new permission. You must specify a body representing a new permission using the following template:

```
<permission>
<name>perm_7</name>
<label>apicreate</label>
<description>Created through the api</description>
<defaultValue>true</defaultValue>
</permission>
```

Note: Any new permissions created will not have any call permissions applied.

Unsupported HTTP Method: POST, DELETE

12.14.2 View or modify a permission ID

Resource URI: /permission/{name}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
permission	The permissions related information similar to /permission.

Specific Response Codes: Error 400 when {name} is not found or invalid

HTTP Method: GET Retrieves information on the permission specified by ID

Example:

```
foo
```

HTTP Method: PUT Updates permission with the specified ID. Uses the same XML as for creation.

Example:

```
bar
```

HTTP Method: DELETE Removes the permission specified by ID.

Example:

```
foo
```

Unsupported HTTP Method: POST

12.15 Phone

12.15.1 Send a phone profile

Example:

```
# curl -k -X PUT https://superadmin:password@192.168.1.31/sipxconfig/api/phones/
↪0004f280cc95/sendProfile/restart
```

12.15.2 Create a phone

Resource URI: /phones

Default Resource Properties

Property	Description
serialNumber	The serial number of the phone.
model	The phone model.
description	Description provided by the user.

Specific Response Codes: N/A

HTTP Method: GET Retrieves information about all phones.

Example:

```
foo
```

HTTP Method: POST Creates a new phone.

Example:

```
bar
```

Unsupported HTTP Methods: PUT, DELETE

12.15.3 Retrieve a phone profile

Resource URI: /phones/{serialNumber}/profile/{name}

Default Resource Properties: N/A

Specific Response Codes: Error 404 when {serialNumber} or {name} is not found.

HTTP Method: GET Retrieves the phone profile of the given serial number or filename

Unsupported HTTP Methods: POST, PUT, DELETE

12.15.4 View all phone models

Resource URI: /phones/models

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
modelId	Model unique ID
label	Model label

Filtering Parameters:

Parameter	Description
page	Required. The requested page size.
pagesize	Required. The number of results to be displayed per page.
sortdir	Optional. Forward or reverse.
sortby	Optional. Name or description.

Specific Response Codes: N/A

HTTP Method: GET Retrieves a list of all phone models.

Example:

```
foo
```

HTTP Method: PUT Updates the settings of the gateway. PUT data is plain text.

HTTP Method: DELETE Deletes the settings of the gateway.

Unsupported HTTP Method: POST

12.15.5 View or create a phone

Resource URI: /phones

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
ID	Phone unique identification number.
serialNo	The phone serial number.
description	Short description provided by the user.
label	Label of the phone model.
vendor	Vendor of the phone model.
lines	Lines assigned to the phone ID.
uri	The URI for the instance.
user	The user name.
userid	The user unique identification number.
displayName	The display name for the user.
password	The SIP password of the user.
registrationServer	The SIP registrar to use

Specific Repsonse Codes: N/A

HTTP Method: GET Retrieves all phones.

Example:

```
foo
```

HTTP Method: POST Creates a new phone.

Unsupported HTTP Method: PUT, DELETE

12.15.6 View or modify a phone

Resource URI: /phones/{phoneId}

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
phone	The phones information similar to /phones and /phones/models

Specific Response Codes: N/A

HTTP Method: GET Retrieve details for the phone with the specified phone ID or MAC address.

Example:

```
bar
```

HTTP Method: PUT Modify the phone with the specified phone ID or MAC address.

Example:

```
foo
```

HTTP Method: DELETE Delete the phone with the specified phone ID or MAC address.

Unsupported HTTP Method: POST

12.15.7 View or delete phones from groups

Resource URI: /phones/{phoneId}/groups

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
id	Group unique identification number.
name	The group name.
description	Description of the group provided by the user.
weight	

Specific Response Codes: N/A

HTTP Method: GET Retrieve the groups for the specified phone ID.

Example:

```
foo
```

HTTP Method: DELETE Deletes the groups for the specified phone ID.

Unsupported HTTP Method: POST, PUT

12.15.8 Delete or add phones in groups

Resource URI: /phones/{phoneId}/groups/{groupName}

Default Resource Properties: N/A

Specific Response Codes: N/A

HTTP Method: POST Add a phone in the specified group name.

Example:

```
bar
```

HTTP Method: DELETE Delete a phone from the specified group name.

Example:

```
foo
```

Unsupported HTTP Method: GET, PUT

12.15.9 View group settings

Resource URI: /phones/{phoneId}/settings/{settingPath}

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
path	Setting path
type	Setting type. Possible values are string or boolean.
value	
defaultValue	Default value.
label	Label setting.
description	Short description provided by the user.

Specific Response Codes: N/A

HTTP Method: GET Retrieves the settings from the group of the specified path.

Example:

```
foo
```

HTTP Method: PUT Update the setting or settings from the group of the specified path.

Example:

```
bar
```

HTTP Method: DELETE Delete the setting from the group of the specified path.

Example::

foo

Unsupported HTTP Method: POST

12.15.10 Delete or add lines to a phone ID

Resource URI: /phones/{phoneId}/lines

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
id	Line unique identification number
uri	The uri for the line.
user	The user name.
userId	User unique identification number
password	User password.
registrationServer	Name of SIP registrar server
registrationServerPort	The SIP registrar server port number

Specific Response Codes: N/A

HTTP Method: GET Retrieve the lines for the phone with the speicifed ID.

Example:

```
foo
```

HTTP Method: POST Add a new line for the phone with the specified ID.

Example:

```
bar
```

HTTP Method: DELETE Delete the setting from the group specified path.

Example:

```
foo
```

Unsupported HTTP Method: PUT

12.15.11 View or modify group settings

Resource URI: /phones/{phoneId}/lines/{lineId}/settings/{settingPath}

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
path	The setting path.
type	The setting type.
description	The description provided by the user.

Specific Response Codes: N/A

HTTP Method: GET Retrieve the group setting from the specified path.

Example:

```
bar
```

HTTP Method: PUT Modify the group setting from the specified path.

Example:

```
foo
```

HTTP Method: DELETE Delete the group setting from the specified path.

Example:

```
bar
```

Unuspported HTTP Method: POST

12.16 Phone Book

12.16.1 View a list of phone books

Resource URI: /phonebook

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
phonebooks	The phonebook name.

Specific Response Codes: N/A

HTTP Method: GET Retrieves a list of all phone books saved in the database.

Example:

```
foo
```

Unsupported HTTP Method: POST, PUT, DELETE

12.16.2 View phone book entries

Resource URI: /phonebook/{name}

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
firstName	First name of the phone book entry.
lastName	Last name of the phone book entry.
number	The number associated to the phone book entry.

Specific Response Codes: N/A

HTTP Method: GET Retrieves a list with all the phone book entries saved in the database.

Example:

```
bar
```

Unsupported HTTP Method: POST, PUT, DELETE

12.17 Phone groups

12.17.1 View or create phone groups

Resource URI: /phoneGroups

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
id	Group unique identification number.
name	The group name.
description	Description of the group provided by the user.
weight	

Specific Response Codes: N/A

HTTP Method: GET Retrieves all phone groups.

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST, DELETE

12.17.2 View or modify phone groups

Resource URI: /phoneGroups/{phoneGroupId}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieves the phone group with the specified ID.

Example:

```
bar
```

HTTP Method: PUT Updates the phone group. PUT data is plain text.

Example:

```
foo
```

HTTP Method: DELETE Deletes the phone group.

Example:

```
bar
```

Unsupported HTTP Method: POST

12.17.3 Move phone group up in ordering

Resource URI: /{groupId}/up

Default Resource Properties: N/A

Specific Response Codes: N/A

HTTP Method: PUT Move the phone group up. PUT data is plain text.

Example:

```
foo
```

Unsupported HTTP Method: GET, POST, DELETE

12.17.4 Move phone group down in ordering

Resource URI: /{groupId}/down

Default Resource Properties: N/A

Specific Response Codes: N/A

HTTP Method: PUT Move the phone group down. PUT data is plain text.

Example:

```
bar
```

Unsupported HTTP Method: GET, POST, DELETE

12.17.5 View settings for specific models in a phone group

Resource URI: /{groupId}/models

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
modelId	Model ID
label	Model label

Specific Response Codes: N/A

HTTP Method: GET Retrieves all phone models specified in the group.

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST, DELETE

12.17.6 View or modify all settings for a phone model in a phone group

Resource URI: /{groupId}/model/{modelName}/settings

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieves the phone group with the specified ID.

Example:

```
bar
```

HTTP Method: PUT Updates the phone group. PUT data is plain text.

Example:

```
foo
```

HTTP Method: DELETE Deletes the phone group.

Example:

```
bar
```

Unsupported HTTP Method: POST

12.17.7 View or modify one setting for a phone model in a phone group

Resource URI: /{groupId}/model/{modelName}/settings/{path:.*}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieves the setting in the specified path.

Example:

```
foo
```

HTTP Method: PUT Updates the setting in the specified path. PUT data is plain text.

Example:

```
bar
```

HTTP Method: DELETE Reverts the setting to the default value.

Example:

```
foo
```

Unsupported HTTP Method: POST

12.18 Proxy

12.18.1 View proxy settings

Resource URI: /proxy/settings

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieves a list of all proxy settings in the system.

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST, DELETE

12.18.2 View or modify proxy settings

Resource URI: /proxy/settings/{settingPath}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieves proxy settings from the specified path.

Example:

```
bar
```

HTTP Method: PUT Modifies proxy options for the setting from the specified path.

Example:

```
foo
```

HTTP Method: DELETE Deletes proxy options for the setting from the specified path.

Example:

```
bar
```

Unsupported HTTP Method: POST

12.19 Registrar

12.19.1 View registrar settings

Resource URI: /registrar/settings

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieves a list of all SIP registrar settings in the system.

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST, DELETE

12.19.2 View or modify registrar settings

Resource URI: /registrar/settings/{settingPath}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET

Example:

```
bar
```

HTTP Method: PUT Modifies SIP registrar options for the specified setting path.

Example:

```
foo
```

HTTP Method: DELETE Deletes SIP registrar options for the specified setting path.

Example:

```
bar
```

Unsupported HTTP Method: POST

12.20 Registrations

12.20.1 View all registrations

Resource URI: /registrations

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Filtering Parameters:

Parameter	Description
start	Required. The start date.
limit	Required. The max number of results to be displayed.

Specific Response Codes: N/A

HTTP Method: GET Retrieves all registrations in the system.

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST, DELETE

12.20.2 Filter registrations by users

Resource URI: /registrations/user/{userId}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
ID	The ID, name, or alias of the user.

Filtering Parameters:

Parameter	Description
start	Required. The start date.
limit	Required. The max number of results to be displayed.

Specific Response Codes: N/A

HTTP Method: GET Retrieves registrations for the specified user.

Example:

```
foo
```

HTTP Method: DELETE Removes registrations of the specified user.

Example:

```
bar
```

Unsupported HTTP Method: PUT, POST

12.20.3 Filter registrations by MAC address

Resource URI: /registrations/serialNo/{serialId}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
registrations	The number of registrations for the specified phone mac

Specific Response Codes: N/A

HTTP Method: GET Retrieves registrations for the specified MAC address.

Example:

```
foo
```

HTTP Method: DELETE Removes registrations for the specified MAC address.

Example:

```
bar
```

Unsupported HTTP Method: PUT, POST

12.20.4 Filter regsitrations by IPs

Resource URI: /registrations/ip/{ip}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieves registrations for the specified IP.

Example:

```
foo
```

HTTP Method: DELETE Removes registrations of the specified IP.

Example:

```
bar
```

Unsupported HTTP Method: PUT, POST

12.20.5 Filter registrations by Call-ID

Resource URI: /registrations/callId/{callId}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieves registrations of the specified Call-ID.

Example:

```
foo
```

HTTP Method: DELETE Remove registrations of the specified Call-ID.

Example:

```
bar
```

Unsupported HTTP Method: PUT, POST

12.20.6 Filter registrations by servers

Resource URI: /registrations/server/{serverId}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieves registrations of the server based on internal ID or FQDN.

Example:

```
foo
```

HTTP Method: DELETE Removes registrations for the specified server based on internal ID or FQDN.

Example:

```
bar
```

Unsupported HTTP Method: PUT, POST

12.21 REST server

12.21.1 View REST server settings

Resource URI: /restserver/settings

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieves a list of all REST server settings in the system.

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST, DELETE

12.21.2 View or modify REST server settings

Resource URI: /restserver/settings/{settingPath}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieves REST server options of the specified path.

Example:

```
foo
```

HTTP Method: PUT Modifies REST server options for the specified path.

Example:

```
bar
```

Unsupported HTTP Method: POST

12.22 Schedules

12.22.1 View all general schedules

Resource URI: /schedules/general

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieve all general schedules.

Example:

```
foo
```

Unsupported HTTP Method: POST, PUT, DELETE

12.22.2 View all schedules for a group ID

Resource URI: /schedules/group/{groupId}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieve schedules for the specified group ID.

Example:

```
bar
```

Unsupported HTTP Method: POST, PUT, DELETE

12.22.3 View all schedules for a user ID

Resource URI: /schedules/user/{userId}/all

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieve all schedules for the specified user ID.

Example:

```
foo
```

Unsupported HTTP Method: POST, PUT, DELETE

12.22.4 View personal schedules for a user ID

Resource URI: /schedules/user/{userId}/personal

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieve personal schedules for the specified user ID.

Example:

```
bar
```

HTTP Method: POST Create a personal schedule for the specified user ID.

Example:

```
foo
```

Unsupported HTTP Method: PUT, DELETE

12.22.5 View description for a schedule ID

Resource URI: /schedules/{scheduleId}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET View the description of the specified schedule ID.

Example:

```
bar
```

HTTP Method: PUT Update the description for the specified schedule ID.

Example:

```
foo
```

HTTP Method: DELETE Delete the description of the specified schedule ID.

Example:

```
bar
```

Unsupported HTTP Method: POST

12.22.6 Add or remove periods to a schedule ID

Resource URI: /schedules/{scheduleId}/period/{index}

Default Resource Properties: N/A

Specific Response Codes: N/A

HTTP Method: POST Create a schedule period for the specified schedule ID.

Example:

```
foo
```

HTTP Method: DELETE Remove a period for the specified schedule ID.

Example:

```
bar
```

Unsupported HTTP Method: GET, PUT

12.23 Shared Appearance Agent

12.23.1 View SAA settings

Resource URI: /saa/settings

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieves a list of all SAA settings.

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST, DELETE

12.23.2 View or modify SAA settings

Resource URI: /saa/settings/{settingPath}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description

Specific Response Codes: N/A

HTTP Method: GET Retrieves a list of all SAA settings for the specified path.

Example:

```
bar
```

HTTP Method: PUT Modifies SAA options for the specified path.

Example:

```
foo
```

HTTP Method: DELETE Deletes SAA options for the specified path.

Example:

```
bar
```

Unsupported HTTP Method: POST

12.24 Users

12.24.1 View avatar information

Resource URI: /avatar/{user}

Default Resource Properties: N/A

Specific Response Codes: N/A

HTTP Method: GET Retrieves avatar content for the specified user.

Example:

```
foo
```

Unsupported HTTP Method: POST, PUT, DELETE

12.24.2 View or modify users

Resource URI: /user

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
ID	Unique identification number of the user. If specified the branch property must be blank.
username	The user name used for authentication
lastName	Last name of the user
firstName	First name of the user
pin	User voicemail PIN
sipPassword	SIP password associated with the user
groups	The groups the user is a member of.
branch	Branch unique identification number. If specified the ID property must be blank.
alias	Any aliases associated with the user.

Filtering Parameters:

Parameter	Description
page	Required. The requested page size.
pagesize	Required. The number of results to be displayed per page.
sortdir	Required. Forward or reverse. If it is the only parameter used, it defaults to Name.
sortby	Required. Name or description.

Specific Response Codes: N/A

HTTP Method: GET Retrieves information on all users. Parameters to specify sorting are optional, but you should use both if you want sorting. If you only use the "sortdir" parameter, it defaults to "name".

Example:

```
foo
```

HTTP Method: PUT Adds a new user.

Example:

```
bar
```

Note:

- The ID is automatically generated. Any value entered will be ignored.
- If the PIN is empty the current PIN value will be preserved.
- The **branch**, **groups**, and **aliases** elements are optional.

Unsupported HTTP Method: POST, DELETE

12.24.3 View or modify a user ID

Resource URI: /user/{id}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
user	The user setting related information is the same as described under /user.

Specific Response Codes: N/A

HTTP Method: GET Retrieves information about the user specified by ID.

Example:

```
foo
```

HTTP Method: PUT Updates specified user ID. Uses the same XML as for creation. After an update the response data will contain an ID element with the id value of the item affected.

Example:

```
bar
```

HTTP Method: DELETE Removes the user specified by ID.

Unsupported HTTP Method: POST

12.24.4 View permissions for all users

Resource URI: /user-permission

Default Resource Properties: N/A

Filtering Parameters:

Parameter	Description
page	Required. The requested page size.
pagesize	Required. The number of results to be displayed per page.
sortdir	Optional, forward or reverse.
sortby	Optional, name or description.

Specific Response Codes: N/A

HTTP Method: GET Retrieves information on all users and their permission settings.

Example:

```
**Unsupported HTTP Method:** POST, PUT, DELETE
```

12.24.5 View or modify permissions for a user ID

Resource URI: /user-permission{id}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
id	The user ID. The value is automatically generated and any value is ignored.
lastName	The last name of the user.
firstName	The first name of the user.
permissions	List of permissions.
setting	List of settings.
name	Name of the setting.
value	Displays the status of the permission: **Enabled** or **Disabled**. It will be missing (empty) if the permission is set to the default and has never been changed.
defaultValue	The default value of **true** or **false**, for information only. It does not need to be provided and will be ignored. Not all permissions need to be updated at once. They can be listed individually or in subgroups.

Specific Response Codes: N/A

HTTP Method: GET Retrieves information on user with the specified id and its permissions.

Example:

```
foo
```

HTTP Method: PUT Sets permission values for a user.

Example:

```
bar
```

Unsupported HTTP Method: POST, DELETE

12.25 User Groups

12.25.1 View or modify user groups

Resource URI: /user-group

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
id	Unique identification number of the user group.
name	Name of the user group.

Filter Parameters:

Parameter	Description
page	Required. The requested page size.
pagesize	Required. The number of results to be displayed per page.
sortdir	Optional, forward or reverse.
sortby	Optional, name or description.

Specific Response Codes: N/A

HTTP Method: GET Retrieves information on all the user groups.

Example:

```
foo
```

HTTP Method: PUT Adds a new user group. The id is automatically generated any any value is ignored. The branch element is optional.

Example:

```
bar
```

Unsupported HTTP Method: POST, DELETE

12.25.2 View or modify a user group ID

Resource URI: /user-group/{id}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
totalResults	Number of total results.
currentPage	Number of the current page.
totalPages	Number of total pages.
resultsPerPage	Number of results per page.
group	Heading with details on the user group.
id	The group ID.
name	The group name.
description	Description of the group

Specific Response Codes: N/A

HTTP Method: GET Retrieves information on the user group specified by ID.

Example:

```
foo
```

HTTP Method: PUT Updates group with the specified ID. Uses the same XML as for creation.

Example:

```
bar
```

HTTP Method: DELETE Removes the user group specified by ID.

Example:

```
foo
```

Unsupported HTTP Method: POST

12.25.3 View or modify user group permissions

Resource URI: /user-group-permission

Default Resource Properties: N/A

Filter Parameters:

Parameter	Description
page	Required. The requested page size.
pagesize	Required. The number of results to be displayed per page.
sortdir	Optional, forward or reverse.
sortby	Optional, name or description.

Specific Response Codes: N/A

HTTP Method: GET Retrieves information on all user groups and their permission settings.

Unsupported HTTP Method: POST, PUT, DELETE

12.25.4 View or modify permissions for a group ID

Resource URI: /user-group-permission/{id}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
id	Group unique identification number.
name	Group name.
description	Description provided by the user.
name	Permission name.
value	Displays **true** if enabled, **false** if disabled.
defaultValue	The default value.

Filter Parameters:

Parameter	Description
page	Required. The requested page size.
pagesize	Required. The number of results to be displayed per page.
sortdir	Optional, forward or reverse.
sortby	Optional, name or description.

Specific Response Codes: Error 400 when {id} is invalid or not found.

HTTP Method: GET Retrieves information on the user group with the specified ID and its permissions.

Example:

```
foo
```

HTTP Method: PUT Sets permission values for a user group.

Note:

- The ID is automatically generated and any value will be ignored.
- The setting element must contain a value element.
- The value must be set to either enable or disable. The value element is blank if the permission is set to the default and has never been changed.
- The defaultValue is for information only and is read-only.
- Not all permissions need to be updated at once. THey can be listed individually or in subgroups.

Unsupported HTTP Method: POST, DELETE

12.26 User Services

12.26.1 Calls

Initiate Calls

Resource URI: /my/call/{to} or /call/{to}

Default Resource Properties: N/A

Specific Response Codes: Error 400 when {to} is not a valid SIP URI

HTTP Method: PUT PUT method requires a non empty body which is ignored. Supported as a GET for clients that do not handle PUT.

Example:

```
foo
```

Unsupported HTTP Method: GET, POST, DELETE

View or modify user call forwarding

Resource URI: /my/forward

Default Resource Properties: N/A

Specific Response Codes: N/A

HTTP Method: GET Retrieves user call forwarding settings without shcedules.

Example:

```
foo
```

HTTP Method: PUT Modifies user call forwarding settings without schedules.

Example:

```
bar
```

Unsupported HTTP Method: POST, DELETE

View or modify call forwarding (with schedules)

Resource URI: /my/callfwd

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
expiration	The time measured in seconds that the phone will ring
type	Options are 'if no response' (call is not forked) or 'at the same time' (call is forked)
enabled	Value is **true** if the forward is enabled, **false** if it is disabled.
number	The number or extension to forward to.

Specific Response Codes: N/A

HTTP Method: GET Retrieves user call forwarding scheme with schedules.

Example:

```
foo
```

HTTP Method: PUT Modifies user call forwarding scheme with schedules.

Example:

```
bar
```

Unsupported HTTP Method: POST, DELETE

View or modify call forwarding schedules

Resource URI: /my/callfwdschedule

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
expiration	The time measured in seconds that the phone will ring
type	Options are 'if no response' (call is not forked) or 'at the same time' (call is forked)
enabled	Value is **true** if the forward is enabled, **false** if it is disabled.
number	The number or extension to forward to.

Specific Response Codes: Error 422 when schedule save or update failed. Error 403 on PUT or DELETE and the schedule id not specified.

HTTP Method: GET Retrieves call forwarding schedules.

Example:

```
foo
```

HTTP Method: PUT Modifies call fowarding schedules.

Example:

```
bar
```

Unsupported HTTP Method: POST, DELETE

View or modify a schedule ID

Resource URI: /my/callfwdsched/{id}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
description	Short description of the schedule.
periods	The start and end dates of the period. The format is hours/minutes.
scheduleId	The ID of the schedule.
name	Alternative name of the schedule.

Specific Response Codes: N/A

HTTP Method: GET Retrieves call forwarding schedules.

Example:

```
foo
```

HTTP Method: PUT Updates existing schedule of the specified ID.

Example:

```
bar
```

HTTP Method: DELETE Deletes existing schedule of the specified ID.

Example:

```
foo
```

Unsupported HTTP Method: POST

View active calls

Resource URI: /my/activecdrs

Default Resource Parameters The resource is represented by the following properties when the GET request is performed:

Property	Description
from	The user part of a sip uri, or pstn number
from-aor	The entire From URI
to	The target user part of a sip uri, or pstn number
to-aor	The entire To URI
direction	Direction of the call. Value is either INCOMING or OUTBOUND
recipient	The user that answered the call.
internal	If the call was internal extension to extension, **true** if yes, **false** if not.
type	Field is read-only, for internal use
start-time	Start time of the call.
duration	Length of the call.

Specific Response Codes: N/A

HTTP Method: GET Retrieves user active calls in xml or json format.

Example:

```
foo
```

Unsupported HTTP Method: POST, PUT, DELETE

12.26.2 Voicemail

Change voicemail PIN

Resource URI: /my/voicemail/pin/{pin}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
description	Short description of the schedule
periods	The start and end dates of the period. The format is hours/minutes.
scheduleId	The ID of the schedule.
name	Alternative name of the schedule.

Specific Response Codes: Error 400 on PUT when the new {pin} cannot be saved.

HTTP Method: GET Retrieves call forwarding schedules.

Example:

```
foo
```

HTTP Method: PUT Updates existing schedule given {id}

Example:

```
bar
```

HTTP Method: DELETE Deletes existing schedule given {id}

Example:

```
foo
```

Unsupported HTTP Method: POST

View or modify voicemail settings

Resource URI: /my/vmprefs

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
voicemailPermission	Determines whether the recipient of the call has the permission to receive voicemail. Value is **true** if the user has permission and **false** if not permitted.
emailformat	Determines the email format type. Options are null (no email sent), full (detailed email sent), or medium.
altEmailFormat	Determines the email format type for the secondary email address. Options are null, full, or medium.
greeting	Voicemail prompt callers hear before leaving a message.
emailAttachType	Determines whether the email has an attachment. Value is **yes** or **no**.
emailIncludeAudioAttachment	Determines weather the voicemail message is attached to the email. Displays **true** or **false**.
email	

Specific Response Codes: N/A

HTTP Method: GET Retrieves call forwarding schedules.

Example:

```
foo
```

HTTP Method: PUT Saves user voicemail settings.

Example:

```
bar
```

Unsupported HTTP Method: POST, DELETE

View voicemail folder as a RSS feed

Resource URI: /my/feed/voicemail/{folder}

Default Resource Properties: N/A

Specific Response Codes: N/A

HTTP Method: GET The voicemail folder is presented as a RSS feed.

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST, DELETE

View or modify voicemail personal attendant

Resource URI: /my/voicemail/attendant

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
personalAttendantPermission	
language	
operator	
menu	
overrideLanguage	
depositVM	
playVMDefaultOptions	

Specific Response Codes: N/A

HTTP Method: GET Retrieves the personal attendant settings.

Example:

```
# curl -k -X GET https://200:11111111@localhost/sipxconfig/rest/my/voicemail/attendant
{"personalAttendantPermission":true,"operator":"419","overrideLanguage":true,"menu":{
↪"5":"255","7":"999"},"language":"en","depositVM":true,"forwardDeleteVM":true,
↪"playVMDefaultOptions":true}
```

HTTP Method: PUT Updates the personal attendant settings.

Example:

```
# curl -k -X PUT -H "Content-Type: application/json" -d '{"operator":"419",
↪"overrideLanguage":true,"menu":{"5":"255","7":"999"},"language":"en","depositVM
↪":true,"forwardDeleteVM":true,"playVMDefaultOptions":true}' https://
↪200:11111111@localhost/sipxconfig/rest/my/voicemail/attendant
```

Unsupported HTTP Method: POST, DELETE

Set operators' personal attendant settings

Resource URI: /my/voicemail/operator/{operator}

Default Resource Properties: N/A

Specific Response Codes: N/A

HTTP Method: PUT Sets personal attendant operator user given {operator} value.

Example:

```
foo
```

Unsupported HTTP Method: GET, POST, DELETE

Reset operators' personal attendant settings

Resource URI: /my/voicemail/operator

Default Resource Properties: N/A

Specific Response Codes: N/A

HTTP Method: PUT Resets personal attendant operator user given the {operator} value.

Example:

```
bar
```

Unsupported HTTP Method: GET, POST, DELETE

12.26.3 Phonebook

Export phone book

Resource URI /my/phonebook

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
entry	The phonebook entry
first-name	Entry first name
last-name	Entry last name
number	
contact-information	
homeAddress	
officeAddress	
imID	
imDisplayName	
avatar	The avatar URL

Specific Response Codes:: N/A

HTTP Method: GET Retrieves the phonebook

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST, DELETE

View phone book page by page

Resource URI: /my/pagedphonebook?start={start-row}&end={end-row}

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
size	Total entries number
filtered-size	Returned entries number
start-row	First returned entry number
end-row	Last returned entry number
show-on-phone	
google-domain	
entries	

Specific Response Codes: N/A

HTTP Method: GET Returnes users from start row to end row.

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST, DELETE

Private phone book entries

Resource URI: /my/phonebook/entry/{entryId}

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
pb	
internalID	
uid	
vcard	
username	

Specific Response Codes: Error 747 when entryid is duplicated during save (POST)

HTTP Method: GET Retrieves all the entries from the private phone book.

Example:

```
foo
```

HTTP Method: PUT Modifies entries in the private phone book.

Example:

```
bar
```

HTTP Method: POST Creates a new private phone book entry

Example:

```
foo
```

HTTP Method: DELETE Deletes the entry specified by ID from the private phone book.

Example:

```
bar
```

Unsupported HTTP Method: N/A

Search for phone book contacts

Resource URI: /my/search/phonebook?query={searchterm}

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
entry	The result.

Specific Response Codes: N/A

HTTP Method: GET Searches the phone book. Search term can be a value of any user field like firstname, lastname, extension, etc.

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST, DELETE

Create or delete a private phone book

Resource URI: /my/phonebookentry/{internalid}

Default Resource Properties: N/A

Specific Response Codes: Error 404 when {internalid} is not found.

HTTP Method: PUT Creates a private phone book from a vcard template.

Example:

```
bar
```

HTTP Method: DELETE Deletes a private phone book entry.

Example:

```
foo
```

Unsupported HTTP Method: POST, DELETE

View contacts on display

Resource URI: /my/phonebook/showContactsOnPhone/{value}

Default Resource Properties: N/A

Specific Response Codes: N/A

HTTP Method: PUT Marks the showContactsOnPhone flag true or false in the user private phonebook.

Example:

```
bar
```

Unsupported HTTP Method: GET, POST, DELETE

Import Google Contacts

Resource URI: /my/phonebook/googleImport

Default Resource Properties: N/A

Specific Response Codes:

- Error 743 on POST when there is a google authentication error.
- Error 744 on POST when there is a google service error.
- Error 745 on POST when there is a google transport error.

HTTP Method: POST Imports google contacts into user private phonebook.

Example:

```
foo
```

Unsupported HTTP Method: GET, PUT, DELETE

12.26.4 Preferences

View user contact information

Resource URI: /my/contact-information

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
jobTitle	
jobDept	
companyName	
homeAddress	
officeAddress	
branchAddress	
imID	
imDisplayName	
emailAddress	
useBranchAddress	
salutation	
twitterName	
linkedinName	
facebookName	
xingName	
timestamp	
enabled	
ldapManaged	true if the user should be modified when importing from ldap
firstName	
lastName	

Specific Response Codes: N/A

HTTP Method: GET Retrieves the etnries from the private phone book.

Example:

```
foo
```

HTTP Method: PUT Updates user contact details.

Example:

```
bar
```

Unsupported HTTP Method: POST, DELETE

View or modify IM preferences

Resource URI: /my/im/prefs

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
statusCallInfo	If true include the caller info in the busy status of the XMPP message.
otpMessage	The content of the message used as XMPP status when user is busy.
voicemailonDnd	If true, all calls received when Do Not Disturb is set through XMPP client are forwarded directly to voicemail.
statusPhonePresence	If true advertise the user's busy status in the XMPP status message.

Specific Response Codes: N/A

HTTP Method: GET Retrieves instant messaging preferences

Example:

```
foo
```

HTTP Method: PUT Saves the modified IM preferences.

Example:

```
bar
```

Unsupported HTTP Method: POST, DELETE

View or modify My Buddy preferences

Resource URI: /my/imbot/prefs

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
confExit	If true conference exit messages are sent to mybuddy
vmBegin	If true notification are sent to mybuddy when callers enter the voicemail box.
vmEnd	If true notifications are sent as the caller exits the voicemail box.
confEnter	If true conference entry messages are sent to mybuddy

Specific Response Codes: N/A

HTTP Method: GET Retrieves mybuddy preferences.

Example:

```
foo
```

HTTP Method: PUT Modifies mybuddy preferences

Example:

```
bar
```

Unsupported HTTP Method: POST, DELETE

View or modify speed dial preferences

Resource URI: /my/speeddial

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
updatePhones	
canSubscribeToPresence	
buttons	speeddial number, label, blf true or false
groupSpeedDial	true to inherit group speed dials

Specific Response Codes: N/A

HTTP Method: GET Retrieves the speed dial preferences.

Example:

```
foo
```

HTTP Method: PUT Saves the modified speed dial preferences.

Example:

```
bar
```

Unsupported HTTP Method: POST, DELETE

Activate active greeting

Resource URI: /my/mailbox/{user}/preferences/activegreeting/{greeting}

Default Resource Properties: N/A

Specific Response Codes: Plain text values should be none, standard, outofoffice, or extendedabsence. For other content the "none" greeting will be saved.

HTTP Method: PUT Sets active greeting setting for a specific user.

Example:

```
foo
```

Note: The greeting cannot be empty.

Unsupported HTTP Method: GET, POST, DELETE

12.26.5 Hunt Groups

Get all hunt groups

Example:

```
curl -v -k -X GET https://superadmin:password@192.168.1.31/sipxconfig/api/callgroups
```

Get all huntgroups that have extension starting with a prefix

Example:

```
curl -v -k -X GET https://superadmin:password@localhost/sipxconfig/api/callgroups/
↪prefix/33
```

Get call group given extension

Example:

```
curl -v -k -X GET https://superadmin:password@localhost/sipxconfig/api/callgroups/3399
```

Create a hunt group

Example:

```
curl -v -k -X POST -H "Content-Type: application/json" -d '{"name":"ppp1","extension":
↪"4444","description":"","enabled":true,"did":null,"ringBeans":[],
↪"fallbackDestination":null,"voicemailFallback":true,"userForward":true,"useFwdTimers
↪":false}' https://superadmin:password@localhost/sipxconfig/api/callgroups
```

Duplicate hunt group extension as new hunt group with a different extension

Example:

```
curl -v -k -X POST  https://superadmin:password@localhost/sipxconfig/api/callgroups/
↪3399/duplicate/55665
```

Rotate rings for hunt group

Example:

```
curl -v -k -X POST  https://superadmin:password@localhost/sipxconfig/api/callgroups/
↪3399/rotate/3
```

Update hunt group with extension

Example:

```
curl -v -k -X PUT -H "Content-Type: application/json" -d '{"name":"ppp1","extension":
↪"4444","description":"kkkkk","enabled":true,"did":null,"ringBeans":[],
↪"fallbackDestination":null,"voicemailFallback":true,"userForward":true,"useFwdTimers
↪":false}' https://superadmin:password@localhost/sipxconfig/api/callgroups/4444
```

Delete hunt group

Example:

```
curl -v -k -X DELETE https://superadmin:password@localhost/sipxconfig/api/callgroups/
↪3399
```

12.26.6 Conferences

Filter conferences for a user ID

Resource URI: /my/conferences

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
conferences	
enabled	
name	
description	
extension	
accessCode	

Specific Response Codes: N/A

HTTP Method: GET Returns a list of all conferences for a specific user.

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST, DELETE

View conference details

Resource URI: /my/conferencedetails/{confName}

Default Resource Properties The resource is represented by the following properties when the GET request is performed.

Property	Description
conference	
extension	
locked	
members	
id	
name	
imID	
uuid	
volumeIn	
energyLevel	
canHear	
canSpeak	

Specific Response Codes:

- Error 404 when {confName} is not found.
- Error 403 when authenticated user is not the owner of {confName}
- Error 406 when {confName} is found but not active (no participants)

HTTP Method: GET Returns conference details including participant details.

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST, DELETE

View conference settings for all users

Resource URI: /my/conferences

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
conference	
enabled	
description	
extension	
accessCode	

Specific Response Codes: N/A

HTTP Method: GET Gets user conference settings for all user owned conferences.

Example:

```
foo
```

Unsupported HTTP Method: POST, DELETE

View user conference details

Resource URI: /my/conferences/{name}

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
enabled	
named	
autoRecord	
quickStart	
video	
sendActiveVideoOnly	
maxMembers	
moh	
moderatedRoom	
publicRoom	

Specific Response Codes: N/A

HTTP Method: GET Retrieve user conference details of {name}

Example:

```
foo
```

HTTP Method: PUT Saves user conference details of {name}

Example:

```
bar
```

Unsupported HTTP Method: POST, DELETE

12.27 FreeSWITCH Conference Commands

12.27.1 About Conference Services

The Conference Web Services APIs allow the administrator to send commands to the FreeSWITCH platform.

12.27.2 Base URL

The base URL for the Conference web services is

```
https://username:password@host.domain/sipxconfig/rest/my/conference/{conference-name}
```

For the above URL you can use the /{command}&{arg 1}&{arg 2}... URL with the PUT HTTP method to send the desired commands and arguments to FreeSwitch.

Available FreeSWITCH commands and arguments

Command Name	Command Details	Usage
bgdial		<endpoint_module_name>/<destination> <callerid number> <callerid name>
deaf	Make a conference member deaf.	<[member_id|all]|last>
dial	Dial a destination via a specific endpoint.	<endpoint_module_name>/<destination> <callerid number> <callerid name>
dtmf	Send DTMF to any member of the conference.	<[member_id|all|last]> <digits>
energy	Adjusts the conference energy level for a specific member.	<member_id|all|last> [<newval>]
hup	Kick without the kick sound.	conference <confname> hup <[member_id|all|last]>
kick	Kicks a specific member form a conference.	<[member_id|all|last]>
list	Lists all or a specific conference members.	conference list [delim <string>]
lock	Lock a conference so no new members will be allowed to enter.	lock
mute	Mutes a specific member in a conference.	<[member_id|all]|last>
norecord	Remove recording for a specific conference.	<[filename|all]>
nopin	Removes a pin for a specific conference.	nopin
pin	Sets or changes a pin number for a specific conference. Note: if you set a conference pin and then issue a command like conference <confname> dial sofia/default/123456@softswitch, 123456 will not be challenged with a pin but he will just joins the conference named <confname>.	<pin#>
play	Play an audio file in a conference to all members or to a specific member. You can stop that same audio with the stop command below.	<file_path> [async|<member_id>]
record		<filename>
relate	Mute or Deaf a specific member to another member.	<member_id> <other_member_id> [nospeak|nohear|clear]
say	Write a message to all members in the conference.	<text>
saymember	Write a messaget to a specific member in a conference.	<member_id> <text>
stop	Stops any queued audio from playing.	<[current|all|async|last]> [<member_id>]
transfer	Transfer a member from one conference to another conference. To transfer a member to another extension use the api transfer command with the uuid of their session.	<conference_name> <member id> [...<member id>]
unmute	Unmute a specific member of a conference.	<[member_id|all]|last>
undeaf	Allow a specific member to hear the conference.	<[member_id|all]|last>
unlock	Unlock a conference so that new members can enter.	unlock
volume_in	Adjusts the input volume for a specific conference member.	<member_id|all|last> [<newval>]
volume_out	Adjust the output volume for a specific conference member.	<member_id|all|last>

Specific Response Codes:

- Error 404 when {confName} is not found.
- Error 403 when authenticated user is not the owner of {confName}
- Error 400 when no {command} is specified or the command is incorrect.

12.27.3 Dial Additional Information

If the caller ID values are not set, the variables set in the conference.conf.xml are used. Specifically, the value for caller-id-number is used for the number and the value for caller-id-name is used for the name. If the conference is dynamically created as a result of this API and the caller-id-number and caller-id-number is not provided in the API call then the number and name will be "00000000" and respectively "FreeSWITCH".

Example:

```
conference testconf dial {originate_timeout=30}sofia/default/1000@softswitch␣
↪1234567890 FreeSWITCH_Conference
```

The above API call will dial out of a conference named "testconf" to the user located at the specified endpoint with a 30 second timeout. The endpoint will see the call as coming from "FreeSWITCH_Conference" with a caller id of 1234567890.

Note: The values provided in the dial string overwrite the caller-id-number and caller-id-name variables provided at the end of the API call.

12.27.4 List Additional Information

The output generated by the system is named by default with the following format:

```
<conference name> (<member_count> member[s][locked]),
```

Where locked can represent either the locked or unlockes status of the conference.

The following items are a separated list in CSV format for each conference leg.

Item	Description
ID of participant	
Register string of participants	
UUID of participants call leg	
Caller ID Number	
Status	Options are 'hear' (mute/deaf), 'speak' (deaf/undeaf), 'talking' (sound energy), 'video' (video enabled), and 'floor' (member owns the floor).
VolumeIn	
VolumeOut	
EnergyLevel	

12.27.5 Related Additional Information

Conference Examples

Member 1 may now no longer speak to member 2, i.e. member 2 now cannot hear member 1

```
conference my_conf relate 1 2 nospeak:
```

Member 1 may now speak to member 2 again

```
conference my_conf relate 1 2 clear:
```

Member 1 now cannot hear member 2

```
conference my_conf relate 1 2 nohear:
```

Member 1 can now hear member 2 again

```
confernce my_conf relate 1 2 clear:
```

Command Examples

Lock a conference with name "WeeklyTeamConf"

```
# curl -k -X PUT https://200:123@localhost/sipxconfig/rest/my/conference/
↪WeeklyTeamConf/lock
```

Invite user in conference given username

```
# curl -k https://400:123@gerula-dev.buc.ro/sipxconfig/rest/my/conference/Conf400/
↪invite\&401
```

Invite user in conference given IM ID

```
# curl -k https://400:123@gerula-dev.buc.ro/sipxconfig/rest/my/conference/Conf400/
↪inviteim\&401im
```

Other examples

```
# curl -k https://400:123@gerula-dev.buc.ro/sipxconfig/rest/my/conference/Conf400/xml_
↪list

# curl -k https://400:123@gerula-dev.buc.ro/sipxconfig/rest/my/conference/Conf400/
↪kick\&all

# curl -k https://400:123@gerula-dev.buc.ro/sipxconfig/rest/my/conference/Conf400/
↪record

# curl -k https://400:123@gerula-dev.buc.ro/sipxconfig/rest/my/conference/Conf400/
↪record\&stop

# curl -k https://400:123@gerula-dev.buc.ro/sipxconfig/rest/my/conference/Conf400/
↪record\&status

# curl -k https://400:123@gerula-dev.buc.ro/sipxconfig/rest/my/conference/Conf400/
↪record\&duration
```

Sample PHP click to call code

```
<?php
$to="101";//Number to dial
$from="5001";//userid in sipx
$pass="1234";//sipx pin (NOT SIP password)
//replace sipx.gcgov.local with your sipx server
$url = "http://sipx.gcgov.local:6667/callcontroller/".$from."/".$to."?
↪isForwardingAllowed=true";
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_DIGEST);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_USERPWD, $from.":".$pass);
$result = curl_exec($ch);
curl_close($ch);
?>
```

Sample contact information

```
<contact-information>
<jobTitle>Data Entry Assistant</jobTitle>
<jobDept>Data Management Services</jobDept>
<companyName>Museum of Science</companyName>
<homeAddress>
<city>NY</city>
</homeAddress>
<officeAddress>
<street>1 Science Park</street>
<city>Boston</city>
<country>US</country>
<state>MA</state>
<zip>02114</zip>
</officeAddress>
<imId>myId</imId>
<emailAddress>john.doe@example.com</emailAddress>
<useBranchAddress>false</useBranchAddress>
<avatar>https://secure.gravatar.com/avatar/8eb1b522f60d11fa897de1dc6351b7e8?s=80&amp;
↪d=G</avatar>
<firstName>John</firstName>
<lastName>Doe</lastName>
</contact-information>
```

12.27.6 System

Change user portal password

Resource URI: /my/portal/password/{password}

Default Resource Properties: N/A

Specific Response Codes: Error 400 on PUT when {password} is not valid (or less than 8 characters, or null)

HTTP Method: PUT Change user portal password with {password}

Example:

```
foo
```

Unsupported HTTP Method: GET, POST, DELETE

View fax extensions and DID number

Resource URI: /my/faxprefs

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
extension	Extension number
did	DID number

Specific Response Codes: N/A

HTTP Method: GET Gets user fax extension and DID number.

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST, DELETE

View configuration servers' time

Resource URI: /my/time

Default Resource Properties: N/A

Specific Response Codes: N/A

HTTP Method: GET Retrieves the configuration server time.

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST, DELETE

View login details

Resource URI: /my/logindetails

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
login-details	Header for log in details
userName	The user name
imID	IM name
ldapImAuth	Determines if LDAP auth is enabled, true or false
sipPassword	The SIP password

Specific Response Codes: N/A

HTTP Method: GET Retrieves login detail.

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST, DELETE

View user details, password, and the servers' hostname

Resource URI: /my/faxprefs

Default Resource Properties The resource is represented by the following properties when the GET request is performed:

Property	Description
logindetails	Header for log in details
userName	The user name
imID	IM name
ldapImAuth	Determines if LDAP auth is enabled, true or false
sipPassword	The SIP password
pin	
im-location	
fqdn	The IM server FQDN

Unsupported HTTP Method: PUT, POST, DELETE

Keep session alive

Resource URI: /my/keepalive

Default Resource Properties: N/A

Specific Response Codes: N/A

HTTP Method: GET Meant to be periodically called by clients in order to keep their web session alive.

Example:

```
foo
```

Unsupported HTTP Method: PUT, POST, DELETE

CHAPTER

THIRTEEN

SOAP API REFERENCE

Note: Web services defined in this section for configuration of the system are all SOAP based services and require administrator privileges to be used.

13.1 About SOAP

The SOAP API enables administrators to perform a variety of functions offered by sipxconfig, but without the need to directly interacting with the sipxconfig webui.

The server utilizes the Apache Axis framework.

13.1.1 SOAP base URL

The base URL for the configuration API is the following:

```
https://host.domain/sipxconfig/services/
```

13.1.2 SOAP use case examples

Use case examples for the SOAP APIs might be:

- Integration of sipxcom functionality with your company Intranet site.
- Automate or script processes such as adding or importing users, updating phones, assigning phones to groups, etc.
- Customize the sipxconfig webui to suit your needs.

You can use SOAP with WDSL, which is a formal API definition, and generate bindings in your preferred programming language such as Python, Perl, Ruby, Java, and others. It is recommended to select a programming language with good SOAP client support.

Ruby: From the WSDL you can use the SOAP4R project to build client bindings.

Perl: Install SOAP for Perl with:

```
perl -MCPAN -e 'install SOAP::lite'
```

Command line: Use the following command:

```
java -jar $WsdlDocDir/wsdldoc.jar {color}
-title "sipXconfig SOAP API v3.2" {color}
-dir `pwd`"/ws-api-3.2" {color}
http://sipXcom.sipfoundry.org/rep/sipXcom/main/sipXconfig/web/src/org/sipfoundry/
↪sipxconfig/api/sipxconfig.wsdl
```

13.2 Administration Services

The following resources for the Configuration API are only available for users with administration rights.

Permissions

- Add permissions
- Find permissions
- Manage permissions

Call Groups

- Add call group
- Find call groups
- WSDL Call Group

Users

- Add users
- Find users
- Manage users

Phones

- Add phones
- Find phones
- Manage phones

Tests

- Reset

13.2.1 Permissions

The Permission Web Services supported are SOAP based services. These services use the Web Service Definition Language (WSDL) to define the interfaces supported.

URI

```
https://host.domain/sipxconfig/services/PermissionService
```

WSDL

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions targetNamespace="http://www.sipfoundry.org/2007/08/21/ConfigService
↪" xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="http://www.
↪sipfoundry.org/2007/08/21/ConfigService" xmlns:intf="http://www.sipfoundry.org/2007/
↪08/21/ConfigService" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap=
↪"http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema
↪">
- <!--
WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)
-->
<wsdl:types>
<schema targetNamespace="http://www.sipfoundry.org/2007/08/21/ConfigService" xmlns=
↪"http://www.w3.org/2001/XMLSchema">
<complexType name="Permission">
<sequence>
<element name="name" type="xsd:string" />
<element maxOccurs="1" minOccurs="0" name="label" nillable="true" type="xsd:string" />
<element maxOccurs="1" minOccurs="0" name="description" nillable="true" type=
↪"xsd:string" />
<element maxOccurs="1" minOccurs="0" name="defaultValue" nillable="true" type=
↪"xsd:boolean" />
<element maxOccurs="1" minOccurs="0" name="type" nillable="true" type="xsd:string" />
<element maxOccurs="1" minOccurs="0" name="builtIn" nillable="true" type="xsd:boolean
↪" />
</sequence>
</complexType>
<complexType name="AddPermission">
<sequence>
<element name="permission" type="impl:Permission" />
</sequence>
</complexType>
<element name="AddPermission" type="impl:AddPermission" />
<complexType name="PermissionSearch">
<sequence>
<element maxOccurs="1" minOccurs="0" name="byName" type="xsd:string" />
<element maxOccurs="1" minOccurs="0" name="byLabel" type="xsd:string" />
</sequence>
</complexType>
<complexType name="FindPermission">
<sequence>
<element name="search" type="impl:PermissionSearch" />
</sequence>
</complexType>
<element name="FindPermission" type="impl:FindPermission" />
<complexType name="ArrayOfPermission">
<sequence>
<element maxOccurs="unbounded" minOccurs="0" name="item" type="impl:Permission" />
</sequence>
</complexType>
<complexType name="FindPermissionResponse">
<sequence>
<element name="permissions" type="impl:ArrayOfPermission" />
</sequence>
</complexType>
<element name="FindPermissionResponse" type="impl:FindPermissionResponse" />
<complexType name="Property">
```

(continues on next page)

(continued from previous page)

```
<sequence>
<element name="property" type="xsd:string" />
<element name="value" nillable="true" type="xsd:string" />
</sequence>
</complexType>
<complexType name="ManagePermission">
<sequence>
<element name="search" type="impl:PermissionSearch" />
<element maxOccurs="unbounded" name="edit" type="impl:Property" />
<element maxOccurs="1" minOccurs="0" name="deletePermission" nillable="true" type=
↪"xsd:boolean" />
</sequence>
</complexType>
<element name="ManagePermission" type="impl:ManagePermission" />
</schema>
</wsdl:types>
<wsdl:message name="findPermissionRequest">
<wsdl:part element="impl:FindPermission" name="FindPermission" />
</wsdl:message>
<wsdl:message name="managePermissionRequest">
<wsdl:part element="impl:ManagePermission" name="ManagePermission" />
</wsdl:message>
<wsdl:message name="addPermissionRequest">
<wsdl:part element="impl:AddPermission" name="AddPermission" />
</wsdl:message>
<wsdl:message name="findPermissionResponse">
<wsdl:part element="impl:FindPermissionResponse" name="FindPermissionResponse" />
</wsdl:message>
<wsdl:message name="addPermissionResponse" />
<wsdl:message name="managePermissionResponse" />
<wsdl:portType name="PermissionService">
<wsdl:operation name="addPermission" parameterOrder="AddPermission">
<wsdl:input message="impl:addPermissionRequest" name="addPermissionRequest" />
<wsdl:output message="impl:addPermissionResponse" name="addPermissionResponse" />
</wsdl:operation>
<wsdl:operation name="findPermission" parameterOrder="FindPermission">
<wsdl:input message="impl:findPermissionRequest" name="findPermissionRequest" />
<wsdl:output message="impl:findPermissionResponse" name="findPermissionResponse" />
</wsdl:operation>
<wsdl:operation name="managePermission" parameterOrder="ManagePermission">
<wsdl:input message="impl:managePermissionRequest" name="managePermissionRequest" />
<wsdl:output message="impl:managePermissionResponse" name="managePermissionResponse" /
↪>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="PermissionServiceSoapBinding" type="impl:PermissionService">
<wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="addPermission">
<wsdlsoap:operation soapAction="" />
<wsdl:input name="addPermissionRequest">
<wsdlsoap:body use="literal" />
</wsdl:input>
<wsdl:output name="addPermissionResponse">
<wsdlsoap:body use="literal" />
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="findPermission">
```

(continues on next page)

(continued from previous page)

```
<wsdlsoap:operation soapAction="" />
<wsdl:input name="findPermissionRequest">
<wsdlsoap:body use="literal" />
</wsdl:input>
<wsdl:output name="findPermissionResponse">
<wsdlsoap:body use="literal" />
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="managePermission">
<wsdlsoap:operation soapAction="" />
<wsdl:input name="managePermissionRequest">
<wsdlsoap:body use="literal" />
</wsdl:input>
<wsdl:output name="managePermissionResponse">
<wsdlsoap:body use="literal" />
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="ConfigImplService">
<wsdl:port binding="impl:PermissionServiceSoapBinding" name="PermissionService">
<wsdlsoap:address location="https://47.134.206.174:8443/sipxconfig/services/
↪PermissionService" />
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Note: wsdlsoap:address location specified at the end of the WSDL will be specific to your system.

13.2.2 Add Permissions

Name: *addPermission*

Description: Add a custom call permission to the system.

Input Parameters:

Name	Value type	Required/Optional	Description	Editable/Read only
name	string	Required	The name of the permission to add. Even though it is a required parameter its value is ignored and an internal name is generated.	Editable
label	string	Optional	The label of the permission to add. Label is the name displayed in the webui.	Editable
description	string	Optional	Indicates if the permission is enabled (true) or disabled (false) by default for users.	Editable
defaultValue	boolean	Optional	Indicates if the permission is enabled (true) or disabled (false) by default for users.	Editable
type	string		The type of permission. Read only, any string on add will be ignored.	Read Only
builtIn	boolean		Indicates if the permission is builtin (true) or not (false). Read only, any string on add will be ignored.	Read Only

Output parameters: Empty response.

Example: Adding a call permission with label "Test three" whose default value is false:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con=
↪"http://www.sipfoundry.org/2007/08/21/ConfigService">
<soapenv:Header/>
<soapenv:Body>
<con:AddPermission>
<permission>
<name>Test3</name>
<label>Test Three</label>
<description>Third test permission</description>
<defaultValue>false</defaultValue>
</permission>
</con:AddPermission>
</soapenv:Body>
</soapenv:Envelope>
```

13.2.3 Find Permissions

Name: *findPermission*

Description: Search for a permission or permissions defined in the system.

Input Parameters:

Name	Value type	Required/Optional	Description	Editable/Read only
byName	string	Optional	Indicates a name for permissions by their name. May be null.	Editable
byLabel	string	Optional	Indicates a search for permissions by their label. May be null.	Editable

Output Parameters: Array of items representing the permissions found in the search.

Name	Value Type	Description
name	string	The name of the permission.
label	string	The value representing the label of the permission.
description	string	Describes the permission.
defaultValue		Boolean indicating if the permission is enabled (true) or disabled (false) by default for users.
builtIn	boolean	Indicates if the permission is builtin (true) or not (false).

Example: Search to find all permissions defined in the system.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con=
↪"http://www.sipfoundry.org/2007/08/21/ConfigService">
<soapenv:Header/>
<soapenv:Body>
<con:FindPermission>
</con:FindPermission>
</soapenv:Body>
</soapenv:Envelope>
```

Example: Search to find permission with label "test three".

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con=
↪"http://www.sipfoundry.org/2007/08/21/ConfigService">
<soapenv:Header/>
<soapenv:Body>
<con:FindPermission>
<search>
<byLabel>Test Three</byLabel>
</search>
</con:FindPermission>
</soapenv:Body>
</soapenv:Envelope>
```

13.2.4 Manage Permissions

Name: *managePermission*

Description: Manage (update or delete) existing permissions defined in the system. Only permissions which are not built into the system can be edited or deleted.

Input Parameters:

Name	Value type	Required/Optional	Description	Editable/Read only
byName	string	Optional	String used to indicate a search for permissions by their name. May be null. Name is internally generated value and may not be useful for searching.	Editable
byLabel	string	Optional	Indicates a search for permissions by their label. May be null.	Editable
property			Name of the permission field to edit.	
value			Value to use for the permission field being edited.	
deletePermission	boolean	Optional	Indicating to delete (true) a permission.	

Output Parameters: Empty response.

Example:

```
foo
```

13.3 Call Groups

The call group web services are SOAP based services. These services use the Web Service Definition Language (WSDL) to define the interfaces supported. A call group service deals with information related to hunt groups. Any information queried or added in one of the implemented services are mapped to a hunt group in the configuratio database.

URI

```
https://host.domain/sipxconfig/services/CallGroupService
```

WSDL

```
<wsdl:definitions targetNamespace="http://www.sipfoundry.org/2007/08/21/ConfigService
↪" xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="http://www.
↪sipfoundry.org/2007/08/21/ConfigService" xmlns:intf="http://www.sipfoundry.org/2007/
↪08/21/ConfigService" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap=
↪"http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema
↪">
<!--
WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)
-->
<wsdl:types>
<schema targetNamespace="http://www.sipfoundry.org/2007/08/21/ConfigService" xmlns=
↪"http://www.w3.org/2001/XMLSchema">
<complexType name="UserRing">
<sequence>
<element name="expiration" type="xsd:int" />
<element name="type" type="xsd:string" />
<element name="position" type="xsd:int" />
<element name="userName" type="xsd:string" />
</sequence>
</complexType>
<complexType name="CallGroup">
<sequence>
<element name="name" type="xsd:string" />
<element maxOccurs="1" minOccurs="0" name="extension" nillable="true" type="xsd:string
↪" />
<element maxOccurs="1" minOccurs="0" name="description" nillable="true" type=
↪"xsd:string" />
<element maxOccurs="1" minOccurs="0" name="enabled" nillable="true" type="xsd:boolean
↪" />
<element maxOccurs="unbounded" minOccurs="0" name="rings" nillable="true" type=
↪"impl:UserRing" />
</sequence>
</complexType>
<complexType name="AddCallGroup">
<sequence>
<element name="callGroup" type="impl:CallGroup" />
</sequence>
</complexType>
<element name="AddCallGroup" type="impl:AddCallGroup" />
<complexType name="ArrayOfCallGroup">
<sequence>
<element maxOccurs="unbounded" minOccurs="0" name="item" type="impl:CallGroup" />
</sequence>
</complexType>
<complexType name="GetCallGroupsResponse">
<sequence>
<element name="callGroups" type="impl:ArrayOfCallGroup" />
</sequence>
</complexType>
<element name="GetCallGroupsResponse" type="impl:GetCallGroupsResponse" />
</schema>
</wsdl:types>
```

(continues on next page)

(continued from previous page)

```
<wsdl:message name="getCallGroupsResponse">
<wsdl:part element="impl:GetCallGroupsResponse" name="GetCallGroupsResponse" />
</wsdl:message>
<wsdl:message name="getCallGroupsRequest" />
<wsdl:message name="addCallGroupResponse" />
<wsdl:message name="addCallGroupRequest">
<wsdl:part element="impl:AddCallGroup" name="AddCallGroup" />
</wsdl:message>
<wsdl:portType name="CallGroupService">
<wsdl:operation name="addCallGroup" parameterOrder="AddCallGroup">
<wsdl:input message="impl:addCallGroupRequest" name="addCallGroupRequest" />
<wsdl:output message="impl:addCallGroupResponse" name="addCallGroupResponse" />
</wsdl:operation>
<wsdl:operation name="getCallGroups">
<wsdl:input message="impl:getCallGroupsRequest" name="getCallGroupsRequest" />
<wsdl:output message="impl:getCallGroupsResponse" name="getCallGroupsResponse" />
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="CallGroupServiceSoapBinding" type="impl:CallGroupService">
<wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="addCallGroup">
<wsdlsoap:operation soapAction="" />
<wsdl:input name="addCallGroupRequest">
<wsdlsoap:body use="literal" />
</wsdl:input>
<wsdl:output name="addCallGroupResponse">
<wsdlsoap:body use="literal" />
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="getCallGroups">
<wsdlsoap:operation soapAction="" />
<wsdl:input name="getCallGroupsRequest">
<wsdlsoap:body use="literal" />
</wsdl:input>
<wsdl:output name="getCallGroupsResponse">
<wsdlsoap:body use="literal" />
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="ConfigImplService">
<wsdl:port binding="impl:CallGroupServiceSoapBinding" name="CallGroupService">
<wsdlsoap:address location="https://47.134.206.174:8443/sipxconfig/services/
↪CallGroupService" />
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

13.3.1 Add Call Groups

Name	Value type	Required/Optional	Description	Editable/Read only
name	string	Required	String representing the name of the hunt group to add.	Editable
extension	string	Optional	The extension to be associated with the hunt group.	Editable
description	string	Optional	Describes the hunt group.	Editable
enabled	boolean	Optional	Describing the members of the hunt group, their position in the group, time (in seconds) to ring the user.	0 or more repetitions.
expiration			Time in seconds to present the call to the user.	
type	string		The ring sequence. Can be "delayed" or "immediate". Delay is used to build a sequential type hunt group and immediate a broadcast type hunt group. A mix can be used.	
position			The unique position (starting at 0) of the user in the group.	
username	string		The extension of the user.	

Output Parameters: Empty response.

Example: Add a new hunt group "TestGroup2" that is enabled, dialable at extension 556 and contains 4 members (212, 215, 211, and 221).

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con=
↪"http://www.sipfoundry.org/2007/08/21/ConfigService">
<soapenv:Header/>
<soapenv:Body>
<con:AddCallGroup>
<callGroup>
<name>TestGroup2</name>
<!-Optional:->
<extension>556</extension>
<!-Optional:->
<description>Sample SOAP created Hunt Group</description>
<!-Optional:->
<enabled>true</enabled>
<!-Zero or more repetitions:->
<rings>
<expiration>10</expiration>
<type>delayed</type>
<position>0</position>
<userName>212</userName>
```

(continues on next page)

(continued from previous page)

```
</rings>
<rings>
<expiration>10</expiration>
<type>immediate</type>
<position>1</position>
<userName>215</userName>
</rings>
<rings>
<expiration>15</expiration>
<type>immediate</type>
<position>2</position>
<userName>211</userName>
</rings>
<rings>
<expiration>15</expiration>
<type>delayed</type>
<position>3</position>
<userName>221</userName>
</rings>
</callGroup>
</con:AddCallGroup>
</soapenv:Body>
</soapenv:Envelope>
```

13.3.2 Get Call Groups

Name: *getCallGroups*

Description: Query the hunt groups defined in the system.

Input Parameters: None

Output Parameters: Array of items representing the permissions found in the search.

Name	Value Type	Description
name	string	The name of the hunt group.
extension	string	Representing the extension associated with the hunt group.
description	string	Describes the hunt group.
enabled	boolean	Indicates if the hunt group is enabled (true) or disabled (false).
rings	array	The members of the hunt group, their position, time (in seconds) to ring the user (0 or more repetitions).
expiration		Time in seconds to present the call to user.
type	string	The ring sequence. Can be 'delayed' or 'immediate'. Delayed is sequential, immediate is broadcast.
position		The unique position (starting at 0) of the user in the group.
username	string	The extension of the user.

Example: Query the hunt groups defined in the system.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Header/>
<soapenv:Body/>
</soapenv:Envelope>
```

13.4 Users

The user web services are SOAP based services. These services use the Web Service Definition Language (WSDL) to define the interfaces supported.

URI

```
https://host.domain/sipxconfig/services/UserService
```

WSDL

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions targetNamespace="http://www.sipfoundry.org/2007/08/21/ConfigService
↪" xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="http://www.
↪sipfoundry.org/2007/08/21/ConfigService" xmlns:intf="http://www.sipfoundry.org/2007/
↪08/21/ConfigService" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap=
↪"http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema
↪">
- <!--
WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)
-->
<wsdl:types>
<schema targetNamespace="http://www.sipfoundry.org/2007/08/21/ConfigService" xmlns=
↪"http://www.w3.org/2001/XMLSchema">
<complexType name="User">
<sequence>
<element name="userName" type="xsd:string" />
<element name="pintoken" nillable="true" type="xsd:string" />
<element name="lastName" nillable="true" type="xsd:string" />
<element name="firstName" nillable="true" type="xsd:string" />
<element name="sipPassword" nillable="true" type="xsd:string" />
<element maxOccurs="unbounded" minOccurs="0" name="aliases" nillable="true" type=
↪"xsd:string" />
<element name="emailAddress" nillable="true" type="xsd:string" />
<element maxOccurs="unbounded" minOccurs="0" name="groups" nillable="true" type=
↪"xsd:string" />
<element maxOccurs="unbounded" minOccurs="0" name="permissions" nillable="true" type=
↪"xsd:string" />
<element maxOccurs="1" name="branchName" nillable="true" type="xsd:string" />
</sequence>
</complexType>
<complexType name="AddUser">
<sequence>
<element name="user" type="impl:User" />
<element name="pin" type="xsd:string" />
</sequence>
</complexType>
<element name="AddUser" type="impl:AddUser" />
<complexType name="UserSearch">
```

(continues on next page)

(continued from previous page)

```
<sequence>
<element maxOccurs="1" minOccurs="0" name="byUserName" type="xsd:string" />
<element maxOccurs="1" minOccurs="0" name="byFuzzyUserNameOrAlias" type="xsd:string" /
↪>
<element maxOccurs="1" minOccurs="0" name="byGroup" type="xsd:string" />
</sequence>
</complexType>
<complexType name="FindUser">
<sequence>
<element name="search" type="impl:UserSearch" />
</sequence>
</complexType>
<element name="FindUser" type="impl:FindUser" />
<complexType name="ArrayOfUser">
<sequence>
<element maxOccurs="unbounded" minOccurs="0" name="item" type="impl:User" />
</sequence>
</complexType>
<complexType name="FindUserResponse">
<sequence>
<element name="users" type="impl:ArrayOfUser" />
</sequence>
</complexType>
<element name="FindUserResponse" type="impl:FindUserResponse" />
<complexType name="Property">
<sequence>
<element name="property" type="xsd:string" />
<element name="value" nillable="true" type="xsd:string" />
</sequence>
</complexType>
<complexType name="ManageUser">
<sequence>
<element name="search" type="impl:UserSearch" />
<element maxOccurs="unbounded" name="edit" type="impl:Property" />
<element maxOccurs="1" minOccurs="0" name="deleteUser" nillable="true" type=
↪"xsd:boolean" />
<element maxOccurs="1" minOccurs="0" name="addGroup" nillable="true" type="xsd:string
↪" />
<element maxOccurs="1" minOccurs="0" name="removeGroup" nillable="true" type=
↪"xsd:string" />
<element maxOccurs="1" minOccurs="0" name="updateGroup" nillable="true" type=
↪"xsd:string" />
</sequence>
</complexType>
<element name="ManageUser" type="impl:ManageUser" />
</schema>
</wsdl:types>
<wsdl:message name="findUserRequest">
<wsdl:part element="impl:FindUser" name="FindUser" />
</wsdl:message>
<wsdl:message name="addUserRequest">
<wsdl:part element="impl:AddUser" name="AddUser" />
</wsdl:message>
<wsdl:message name="manageUserResponse" />
<wsdl:message name="addUserResponse" />
<wsdl:message name="manageUserRequest">
<wsdl:part element="impl:ManageUser" name="ManageUser" />
```

(continues on next page)

(continued from previous page)

```
</wsdl:message>
<wsdl:message name="findUserResponse">
<wsdl:part element="impl:FindUserResponse" name="FindUserResponse" />
</wsdl:message>
<wsdl:portType name="UserService">
<wsdl:operation name="addUser" parameterOrder="AddUser">
<wsdl:input message="impl:addUserRequest" name="addUserRequest" />
<wsdl:output message="impl:addUserResponse" name="addUserResponse" />
</wsdl:operation>
<wsdl:operation name="findUser" parameterOrder="FindUser">
<wsdl:input message="impl:findUserRequest" name="findUserRequest" />
<wsdl:output message="impl:findUserResponse" name="findUserResponse" />
</wsdl:operation>
<wsdl:operation name="manageUser" parameterOrder="ManageUser">
<wsdl:input message="impl:manageUserRequest" name="manageUserRequest" />
<wsdl:output message="impl:manageUserResponse" name="manageUserResponse" />
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="UserServiceSoapBinding" type="impl:UserService">
<wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="addUser">
<wsdlsoap:operation soapAction="" />
<wsdl:input name="addUserRequest">
<wsdlsoap:body use="literal" />
</wsdl:input>
<wsdl:output name="addUserResponse">
<wsdlsoap:body use="literal" />
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="findUser">
<wsdlsoap:operation soapAction="" />
<wsdl:input name="findUserRequest">
<wsdlsoap:body use="literal" />
</wsdl:input>
<wsdl:output name="findUserResponse">
<wsdlsoap:body use="literal" />
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="manageUser">
<wsdlsoap:operation soapAction="" />
<wsdl:input name="manageUserRequest">
<wsdlsoap:body use="literal" />
</wsdl:input>
<wsdl:output name="manageUserResponse">
<wsdlsoap:body use="literal" />
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="ConfigImplService">
<wsdl:port binding="impl:UserServiceSoapBinding" name="UserService">
<wsdlsoap:address location="https://47.134.206.174:8443/sipxconfig/services/
↪UserService" />
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Note: wsdlsoap:address location specified at the end of the WSDL will be specific to your system.

13.4.1 Add Users

Name: *addUser*

Description: Add a new user to the system.

Input Parameters:

Name	Value type	Required/Optional	Description	Editable/Read only
userName	string	Required	The name of the user to add.	Editable
pinToken	string	optional	Internally generated token that is an encrypted version of the voicemail pin. This should not be specified as it will be internally generated.	Read Only
lastName	string	Optional	The last name of the user.	Editable
firstName	string	Optional	The first name of the user.	Editable
sipPassword	string	Optional	The SIP password for the user.	
aliases	array		Array of strings, each representing membership in defined groups.	
permissions	array		Array of strings, each representing a permission name that is granted to the user. Permissions can be general or call permissions. See *findPermission*	
pin	string	Required	The PIN for the user.	Editable
branchName	string	Optional	User branch	Editable

Output Parameters: Empty response.

Example: Add a new user 223 to the system with sip password 4567, pin 1234, along with various permissions and group memberships in the branch Berlin.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con=
↪"http://www.sipfoundry.org/2007/08/21/ConfigService">
<soapenv:Header/>
<soapenv:Body>
<con:AddUser>
<user>
<userName>223</userName>
<lastName>Einstein</lastName>
<firstName>Albert</firstName>
<sipPassword>4567</sipPassword>
<emailAddress>albertE@yahoo.com</emailAddress>
<branchName>Berlin</branchName>
<groups>Managers</groups>
<permissions>FreeswitchVoicemailServer</permissions>
<permissions>InternationalDialing</permissions>
```

(continues on next page)

(continued from previous page)

```
<permissions>LocalDialing</permissions>
<permissions>LongDistanceDialing</permissions>
<permissions>Mobile</permissions>
<permissions>TollFree</permissions>
<permissions>Voicemail</permissions>
<permissions>music-on-hold</permissions>
<permissions>perm_8</permissions>
<permissions>personal-auto-attendant</permissions>
<permissions>tui-change-pin</permissions>
</user>
<pin>1234</pin>
</con:AddUser>
</soapenv:Body>
</soapenv:Envelope>
```

13.4.2 Find Users

Name: *findUser*

Description: Find defined user(s) in the system.

Input Parameters: Either null for a listing of all users, or one of the following optional parameters.

Name	Value Type	Required/Optional	Description	Editable/Read Only
byUserName	string	Optional	The name of the user to find.	Editable
byFuzzyUserNameOrAlias	string	Optional	A partial user name or alias to search for, a type of wildcard search.	Editable
byGroup	string	Optional	The users which are members of a particular defined group.	Editable

Output parameters: An array of 0 or more of the following.

Name	Value Type	Description
userName	string	The name of the user to add.
pinToken	string	Internally generated token that is an encrypted version of the pin.
lastName	string	The last name of the user.
firstName	string	The first name of the user.
sipPassword	string	The SIP password for the user.
aliases	array	Array of strings, each representing a user alias.
emailAddress	string	The email address of the user.
groups	array	Array of strings, each representing membership in defined groups.
pin	string	The PIN for the user.
branchName	string	Users branch.

Example: Find all defined users in the system.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con=
↪"http://www.sipfoundry.org/2007/08/21/ConfigService">
<soapenv:Header/>
<soapenv:Body>
```

(continues on next page)

(continued from previous page)

```
<con:FindUser>
</con:FindUser>
</soapenv:Body>
</soapenv:Envelope>
```

Example: Find all users that are members of the group "Managers".

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con=
↪"http://www.sipfoundry.org/2007/08/21/ConfigService">
<soapenv:Header/>
<soapenv:Body>
<con:FindUser>
<search>
<byGroup>Managers</byGroup>
</search>
</con:FindUser>
</soapenv:Body>
</soapenv:Envelope>
```

Example: Find the user 223.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con=
↪"http://www.sipfoundry.org/2007/08/21/ConfigService">
<soapenv:Header/>
<soapenv:Body>
<con:FindUser>
<search>
<byUserName>223</byUserName>
</search>
</con:FindUser>
</soapenv:Body>
</soapenv:Envelope>
```

Example: Find users whose userName begins with 22.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con=
↪"http://www.sipfoundry.org/2007/08/21/ConfigService">
<soapenv:Header/>
<soapenv:Body>
<con:FindUser>
<search>
<byFuzzyUserNameOrAlias>22</byFuzzyUserNameOrAlias>
</search>
</con:FindUser>
</soapenv:Body>
</soapenv:Envelope>
```

13.4.3 Manage Users

Name: *manageUser*

Description: Manage (update or delete) users defined in the system.

Input Paramters: Either null to list all, or one of the following optional parameters.

Name	Value Type	Required/Optional	Description	Editable/Read Only
byUserName	string	Optional	The name of the user to find.	Editable
byFuzzyUserNameOrAlias	string	Optional	A partial username or alias to search for. A type of wildcard search.	Editable
byGroup	string	Optional	The users which are members of a particular defined group.	Editable
property	string	Optional	Name of the user field to edit.	Editable
value	string	Optional	Value to use for the user field being edited.	Editable
deleteUser	boolean	Optional	Indicates to delete (true) user(s). Dependent upon search results.	Editable
addGroup	string	Optional	The name of the group to add the user(s) to. Dependent upon search results.	Editable
removeGroup	string	Optional	The name of the group to remove the user(s) from. Dependent upon search results.	Editable
updateBranch	string	Optional	The name of the branch to update the user(s) to.	Editable

Output Parameters: Empty response

Example: Remove all users in the system beginning with 22 from group Managers

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con=
↪"http://www.sipfoundry.org/2007/08/21/ConfigService">
<soapenv:Header/>
<soapenv:Body>
<con:ManageUser>
<search>
<byFuzzyUserNameOrAlias>22</byFuzzyUserNameOrAlias>
</search>
<removeGroup>Managers</removeGroup>
</con:ManageUser>
</soapenv:Body>
</soapenv:Envelope>
```

Example: Add user with username 211 to the group Managers

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con=
↪"http://www.sipfoundry.org/2007/08/21/ConfigService">
<soapenv:Header/>
<soapenv:Body>
<con:ManageUser>
<byUserName>211</byUserName>
</search>
<addGroup>Managers</addGroup>
</con:ManageUser>
</soapenv:Body>
</soapenv:Envelope>
```

Example: Change all users from group Managers to have a *lastName* of "SuperDog" and *firstName* of "I am"

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con=
↪"http://www.sipfoundry.org/2007/08/21/ConfigService">
<soapenv:Header/>
<soapenv:Body>
```

(continues on next page)

(continued from previous page)

```
<con:ManageUser>
<search>
<!-Optional:->
<byGroup>Managers</byGroup>
</search>
<!-1 or more repetitions:->
<edit>
<!-You may enter the following 2 items in any order->
<property>lastName</property>
<value>SuperDog</value>
</edit>
<edit>
<!-You may enter the following 2 items in any order->
<property>firstName</property>
<value>I am</value>
</edit>
</con:ManageUser>
</soapenv:Body>
</soapenv:Envelope>
```

Example: Update user with username 211 to the branch Berlin

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con=
↪"http://www.sipfoundry.org/2007/08/21/ConfigService">
<soapenv:Header/>
<soapenv:Body>
<con:ManageUser>
<byUserName>211</byUserName>
</search>
<updateBranch>Berlin</updateBranch>
</con:ManageUser>
</soapenv:Body>
</soapenv:Envelope>
```

13.5 Park Orbits

The park orbit web services are SOAP based services. These services use the Web Service Definition Language (WSDL) to define the interfaces supported.

URI

```
https://host.domain:8443/sipxconfig/services/ParkOrbitService
```

WSDL

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions targetNamespace="http://www.sipfoundry.org/2007/08/21/ConfigService
↪" xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="http://www.
↪sipfoundry.org/2007/08/21/ConfigService" xmlns:intf="http://www.sipfoundry.org/2007/
↪08/21/ConfigService" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap=
↪"http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema
↪">
- <!--
WSDL created by Apache Axis version: 1.4
```

(continues on next page)

(continued from previous page)

```
Built on Apr 22, 2006 (06:55:48 PDT)
-->
<wsdl:types>
<schema targetNamespace="http://www.sipfoundry.org/2007/08/21/ConfigService" xmlns=
↪"http://www.w3.org/2001/XMLSchema">
<complexType name="ParkOrbit">
<sequence>
<element name="name" type="xsd:string" />
<element maxOccurs="1" minOccurs="0" name="extension" nillable="true" type="xsd:string
↪" />
<element maxOccurs="1" minOccurs="0" name="description" nillable="true" type=
↪"xsd:string" />
<element maxOccurs="1" minOccurs="0" name="enabled" nillable="true" type="xsd:boolean
↪" />
<element maxOccurs="1" minOccurs="0" name="music" nillable="true" type="xsd:string" />
</sequence>
</complexType>
<complexType name="AddParkOrbit">
<sequence>
<element name="parkOrbit" type="impl:ParkOrbit" />
</sequence>
</complexType>
<element name="AddParkOrbit" type="impl:AddParkOrbit" />
<complexType name="ArrayOfParkOrbit">
<sequence>
<element maxOccurs="unbounded" minOccurs="0" name="item" type="impl:ParkOrbit" />
</sequence>
</complexType>
<complexType name="GetParkOrbitsResponse">
<sequence>
<element name="parkOrbits" type="impl:ArrayOfParkOrbit" />
</sequence>
</complexType>
<element name="GetParkOrbitsResponse" type="impl:GetParkOrbitsResponse" />
</schema>
</wsdl:types>
<wsdl:message name="addParkOrbitRequest">
<wsdl:part element="impl:AddParkOrbit" name="AddParkOrbit" />
</wsdl:message>
<wsdl:message name="addParkOrbitResponse" />
<wsdl:message name="getParkOrbitsResponse">
<wsdl:part element="impl:GetParkOrbitsResponse" name="GetParkOrbitsResponse" />
</wsdl:message>
<wsdl:message name="getParkOrbitsRequest" />
<wsdl:portType name="ParkOrbitService">
<wsdl:operation name="addParkOrbit" parameterOrder="AddParkOrbit">
<wsdl:input message="impl:addParkOrbitRequest" name="addParkOrbitRequest" />
<wsdl:output message="impl:addParkOrbitResponse" name="addParkOrbitResponse" />
</wsdl:operation>
<wsdl:operation name="getParkOrbits">
<wsdl:input message="impl:getParkOrbitsRequest" name="getParkOrbitsRequest" />
<wsdl:output message="impl:getParkOrbitsResponse" name="getParkOrbitsResponse" />
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="ParkOrbitServiceSoapBinding" type="impl:ParkOrbitService">
<wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="addParkOrbit">
```

(continues on next page)

(continued from previous page)

```
<wsdlsoap:operation soapAction="" />
<wsdl:input name="addParkOrbitRequest">
<wsdlsoap:body use="literal" />
</wsdl:input>
<wsdl:output name="addParkOrbitResponse">
<wsdlsoap:body use="literal" />
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="getParkOrbits">
<wsdlsoap:operation soapAction="" />
<wsdl:input name="getParkOrbitsRequest">
<wsdlsoap:body use="literal" />
</wsdl:input>
<wsdl:output name="getParkOrbitsResponse">
<wsdlsoap:body use="literal" />
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="ConfigImplService">
<wsdl:port binding="impl:ParkOrbitServiceSoapBinding" name="ParkOrbitService">
<wsdlsoap:address location="https://47.134.206.174:8443/sipxconfig/services/
↪ParkOrbitService" />
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

13.5.1 Add Park Orbits

Name: *addParkOrbit*

Description: Add a new call park orbit to the system.

Input Parameters: Either null to list all users, or provide one of the following optional parameters.

Name	Value Type	Required/Optional	Description	Editable/Read Only
name	string	Required	The name of the call park orbit to add.	Editable
extension	string	Optional	Dialable extension to be used.	Editable
description	string	Optional	Description of the call park orbit.	Editable
enabled	boolean	Optional	Indicates if the call park is enabled (true) or disabled (false)	Editable
music	path	Optional	Path to a wav file to play as background music for parked calls.	Editable

Note: wav files must be in the appropriate format of RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 8000 Hz

Output parameters: Empty response.

Example: Add a new call park orbit with the name ParkSales at extension 46 and is enabled.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con=
↪"http://www.sipfoundry.org/2007/08/21/ConfigService">
<soapenv:Header/>
<soapenv:Body>
<con:AddParkOrbit>
<parkOrbit>
<name>ParkSales</name>
<!-Optional:->
<extension>46</extension>
<!-Optional:->
<description>Sales calls park orbit</description>
<!-Optional:->
<enabled>true</enabled>
</parkOrbit>
</con:AddParkOrbit>
</soapenv:Body>
</soapenv:Envelope>
```

13.5.2 Get Park Orbits

Name: *getParkOrbits*

Description: Queries information on all call park orbits defined in the system.

Input Parameters: None

Output Parameters:

Name	Value Type	Description
name	string	The name of the call park orbit.
extension	string	The dialable extension.
description	string	A description of the call park orbit.
enabled	boolean	Indicates if the call park orbit is enabled (true) or disabled (false).
music		Path to a wav file to play as background music for parked calls.

Note: wav files must be in the appropriate format of RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 8000 Hz

Example: Query the call park orbits defined in the system.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Header/>
<soapenv:Body/>
</soapenv:Envelope>
```

13.6 Phones

The phone web services are SOAP based services. These services use the Web Service Definition Language (WSDL) to define the interfaces supported.

URI

```
https://host.domain/sipxconfig/services/PhoneService
```

WSDL

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions targetNamespace="http://www.sipfoundry.org/2007/08/21/ConfigService
↪" xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="http://www.
↪sipfoundry.org/2007/08/21/ConfigService" xmlns:intf="http://www.sipfoundry.org/2007/
↪08/21/ConfigService" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap=
↪"http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema
↪">
- <!--
WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)
-->
<wsdl:types>
<schema targetNamespace="http://www.sipfoundry.org/2007/08/21/ConfigService" xmlns=
↪"http://www.w3.org/2001/XMLSchema">
<complexType name="Line">
<sequence>
<element name="userId" type="xsd:string" />
<element name="uri" type="xsd:string" />
</sequence>
</complexType>
<complexType name="Phone">
<sequence>
<element name="serialNumber" type="xsd:string" />
<element name="modelId" type="xsd:string" />
<element maxOccurs="1" minOccurs="0" name="description" nillable="true" type=
↪"xsd:string" />
<element maxOccurs="unbounded" minOccurs="0" name="groups" nillable="true" type=
↪"xsd:string" />
<element maxOccurs="unbounded" minOccurs="0" name="lines" nillable="true" type=
↪"impl:Line" />
<element maxOccurs="1" minOccurs="0" name="deviceVersion" nillable="true" type=
↪"xsd:string" />
</sequence>
</complexType>
<complexType name="AddPhone">
<sequence>
<element name="phone" type="impl:Phone" />
</sequence>
</complexType>
<element name="AddPhone" type="impl:AddPhone" />
<complexType name="PhoneSearch">
<sequence>
<element maxOccurs="1" minOccurs="0" name="bySerialNumber" type="xsd:string" />
<element maxOccurs="1" minOccurs="0" name="byGroup" type="xsd:string" />
</sequence>
</complexType>
<complexType name="FindPhone">
<sequence>
<element name="search" type="impl:PhoneSearch" />
</sequence>
</complexType>
<element name="FindPhone" type="impl:FindPhone" />
<complexType name="ArrayOfPhone">
```

(continues on next page)

(continued from previous page)

```
<sequence>
<element maxOccurs="unbounded" minOccurs="0" name="item" type="impl:Phone" />
</sequence>
</complexType>
<complexType name="FindPhoneResponse">
<sequence>
<element name="phones" type="impl:ArrayOfPhone" />
</sequence>
</complexType>
<element name="FindPhoneResponse" type="impl:FindPhoneResponse" />
<complexType name="Property">
<sequence>
<element name="property" type="xsd:string" />
<element name="value" nillable="true" type="xsd:string" />
</sequence>
</complexType>
<complexType name="AddExternalLine">
<sequence>
<element name="userId" type="xsd:string" />
<element maxOccurs="1" minOccurs="0" name="displayName" type="xsd:string" />
<element maxOccurs="1" minOccurs="0" name="password" type="xsd:string" />
<element name="registrationServer" type="xsd:string" />
<element maxOccurs="1" minOccurs="0" name="voiceMail" type="xsd:string" />
</sequence>
</complexType>
<complexType name="ManagePhone">
<sequence>
<element name="search" type="impl:PhoneSearch" />
<element maxOccurs="unbounded" name="edit" type="impl:Property" />
<element maxOccurs="1" minOccurs="0" name="deletePhone" nillable="true" type=
↪"xsd:boolean" />
<element maxOccurs="1" minOccurs="0" name="addGroup" nillable="true" type="xsd:string
↪" />
<element maxOccurs="1" minOccurs="0" name="removeGroup" nillable="true" type=
↪"xsd:string" />
<element maxOccurs="1" minOccurs="0" name="addLine" nillable="true" type="impl:Line" /
↪>
<element maxOccurs="1" minOccurs="0" name="addExternalLine" nillable="true" type=
↪"impl:AddExternalLine" />
<element maxOccurs="1" minOccurs="0" name="removeLineByUserId" nillable="true" type=
↪"xsd:string" />
<element maxOccurs="1" minOccurs="0" name="removeLineByUri" nillable="true" type=
↪"xsd:string" />
<element maxOccurs="1" minOccurs="0" name="generateProfiles" nillable="true" type=
↪"xsd:boolean" />
<element maxOccurs="1" minOccurs="0" name="restart" nillable="true" type="xsd:boolean
↪" />
</sequence>
</complexType>
<element name="ManagePhone" type="impl:ManagePhone" />
</schema>
</wsdl:types>
<wsdl:message name="findPhoneRequest">
<wsdl:part element="impl:FindPhone" name="FindPhone" />
</wsdl:message>
<wsdl:message name="managePhoneResponse" />
<wsdl:message name="addPhoneResponse" />
```

(continues on next page)

(continued from previous page)

```
<wsdl:message name="managePhoneRequest">
<wsdl:part element="impl:ManagePhone" name="ManagePhone" />
</wsdl:message>
<wsdl:message name="addPhoneRequest">
<wsdl:part element="impl:AddPhone" name="AddPhone" />
</wsdl:message>
<wsdl:message name="findPhoneResponse">
<wsdl:part element="impl:FindPhoneResponse" name="FindPhoneResponse" />
</wsdl:message>
<wsdl:portType name="PhoneService">
<wsdl:operation name="addPhone" parameterOrder="AddPhone">
<wsdl:input message="impl:addPhoneRequest" name="addPhoneRequest" />
<wsdl:output message="impl:addPhoneResponse" name="addPhoneResponse" />
</wsdl:operation>
<wsdl:operation name="findPhone" parameterOrder="FindPhone">
<wsdl:input message="impl:findPhoneRequest" name="findPhoneRequest" />
<wsdl:output message="impl:findPhoneResponse" name="findPhoneResponse" />
</wsdl:operation>
<wsdl:operation name="managePhone" parameterOrder="ManagePhone">
<wsdl:input message="impl:managePhoneRequest" name="managePhoneRequest" />
<wsdl:output message="impl:managePhoneResponse" name="managePhoneResponse" />
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="PhoneServiceSoapBinding" type="impl:PhoneService">
<wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="addPhone">
<wsdlsoap:operation soapAction="" />
<wsdl:input name="addPhoneRequest">
<wsdlsoap:body use="literal" />
</wsdl:input>
<wsdl:output name="addPhoneResponse">
<wsdlsoap:body use="literal" />
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="findPhone">
<wsdlsoap:operation soapAction="" />
<wsdl:input name="findPhoneRequest">
<wsdlsoap:body use="literal" />
</wsdl:input>
<wsdl:output name="findPhoneResponse">
<wsdlsoap:body use="literal" />
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="managePhone">
<wsdlsoap:operation soapAction="" />
<wsdl:input name="managePhoneRequest">
<wsdlsoap:body use="literal" />
</wsdl:input>
<wsdl:output name="managePhoneResponse">
<wsdlsoap:body use="literal" />
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="ConfigImplService">
<wsdl:port binding="impl:PhoneServiceSoapBinding" name="PhoneService">
<wsdlsoap:address location="https://47.134.206.174/sipxconfig/services/PhoneService" /
↪>
```

(continues on next page)

(continued from previous page)

```
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Note: wsdlsoap:address location specified at the end of the WSDL will be specific to your system.

13.6.1 Add Phones

Name: *addPhone*

Description: Add a new phone to the system.

Input Parameters:

Name	Value Type	Required/Optional	Description	Editable/Read Only
serialNumber	string	Required	The MAC address of the phone.	Editable
modelId	string	Required	A supported model ID.	Editable
description	string	Optional	A description of the phone.	Editable
groups	string	Optional	The phone group(s) the new phone will be a member of.	Editable
lines	string	Optional	String representing assigned lines to the phone.	Editable
deviceVersion	string	Optional	The version of the phone.	Editable

Output Parameters: Empty response.

Example: Add a new polycom spip 321 phone to the system, assign line 221, and add to phone group FirstPhoneGroup.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con=
↪"http://www.sipfoundry.org/2007/08/21/ConfigService">
<soapenv:Header/>
<soapenv:Body>
<con:AddPhone>
<phone>
<serialNumber>000000000002</serialNumber>
<modelId>polycom321</modelId>
<!-Optional:->
<description>SOAP added phone</description>
<!-Zero or more repetitions:->
<groups>FirstPhoneGroup</groups>
<!-Zero or more repetitions:->
<lines>
<userId>221</userId>
<uri>221@Uniteme.ezuce.com</uri>
</lines>
<!-Optional:->
</phone>
</con:AddPhone>
</soapenv:Body>
</soapenv:Envelope>
```

List of supported phones:

```
aastra53i
aastra55i
aastra57i
aastra560m
aastra sip ip 53i
audiocodesMP112_FXS
audiocodesMP114_FXS
audiocodesMP118_FXS
audiocodesMP124_FXS
avaya-1210
avaya-1220
avaya-1230
bria
ciscoplus7911G
ciscoplus7941G
ciscoplus7945G
ciscoplus7961G
ciscoplus7965G
ciscoplus7970G
ciscoplus7975G
cisco7960
cisco7940
cisco7912
cisco7905
cisco18x
clearone
gtekAq10x
gtekHl20x
gtekVt20x
gsPhoneBt100
gsPhoneBt200
gsPhoneGxp2020
gsPhoneGxp2010
gsPhoneGxp2000
gsPhoneGxp1200
gsPhoneGxv3000
gsFxsGxw4004
gsFxsGxw4008
gsHt286
gsHt386
gsHt486
gsHt488
gsHt496
hitachi3000
hitachi5000
hitachi5000A
ipDialog
isphone
karelIP116
karelIP112
karelIP111
karelNT32I
karelNT42I
linksys901
linksys921
linksys922
```

(continues on next page)

(continued from previous page)

```
linksys941
linksys942
linksys962
linksys2102
linksys3102
linksys8000
SPA501G
SPA502G
SPA504G
SPA508G
SPA509G
SPA525G
mitel
nortel11xx
nortel1535
lip6804
lip6812
lip6830
polycom321
polycom320
polycom330
polycom331
polycom335
polycom430
polycom450
polycom550
polycom560
polycom650
polycom670
polycomVVX1500
polycom5000
polycom6000
polycom7000
snom300
snom320
snom360
snom370
snomM3
unidatawpu7700
```

13.6.2 Find Phones

Name: *findPhone*

Description: Find defined phone(s) in the system.

Input Parameters: Either null for a listing of all, or provide one of the following optional parameters.

Name	Value Type	Required/Optional	Description	Editable/Read Only
bySerialNumber	string	Required	The MAC address of the phone to find	Editable
byGroup	string	Optional	The phones which are members of the specified group.	Editable

Output Parameters: An array of 0 or more of the following.

Name	Value Type	Description
serialNumber	string	The MAC address of the phone.
extension	string	The dialable extension
description	string	The description of the phone

Example: List all defined phones.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con=
↪"http://www.sipfoundry.org/2007/08/21/ConfigService">
<soapenv:Header/>
<soapenv:Body>
<con:FindPhone>
</con:FindPhone>
</soapenv:Body>
</soapenv:Envelope>
```

Example: Find all phones that are members of the phone group FirstPhoneGroup.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con=
↪"http://www.sipfoundry.org/2007/08/21/ConfigService">
<soapenv:Header/>
<soapenv:Body>
<con:FindPhone>
<search>
<byGroup>FirstPhoneGroup</byGroup>
</search>
</con:FindPhone>
</soapenv:Body>
</soapenv:Envelope>
```

Example: Find all phones with the MAC address of 000000000001

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con=
↪"http://www.sipfoundry.org/2007/08/21/ConfigService">
<soapenv:Header/>
<soapenv:Body>
<con:FindPhone>
<search>
<bySerialNumber>000000000001</bySerialNumber>
</search>
</con:FindPhone>
</soapenv:Body>
</soapenv:Envelope>
```

13.6.3 Manage Phones

Name: *managePhone*

Description: Update or delete phones defined in the system.

Input Parameters: Either null for a listing of all, or provide one of the following optional parameters.

Name	Value Type	Required/Optional	Description	Editable/Read Only
bySerialNumber	string	Required	Search for a phone by MAC address. May be null.	Editable
byGroup	string	Optional	Phones which are members of a specified phone group.	Editable
property			The name of the phone field to edit.	
value			Value to use for the phone field being edited.	
deletePhone	boolean	Optional	Indicates to delete (true) the phone(s). Dependent upon search results.	
addGroup	string	Optional	The phone group to add the phone(s) to. Dependent upon search results.	
removeGroup	string	Optional	The phone group to remove the phone(s) from. Dependent upon search results.	
addLine	string	Optional	userid and uri to add to the phone(s). Dependent upon search results.	
addExternalLine	string	Optional	userid, dispalyname, password, registrationserver and voicemail of the external line to add to the phone(s). Dependent upon search results.	
userId	string	Required	The user portion of the SIP URI and default value for authorization.	
displayName	string	Optional	The display name to use for the userId.	
password	string	Optional	The password for the userid.	
registrationServer	string	Required	The domain the userid should register to.	
voicemail	string	Optional	The voicemail extension for the userid.	
removeLineByUserId	string	Optional	The userid of the line to remove from the phone(s). Dependent upon search results.	
removeLineByUri	string	Optional	The uri of the line to remove from the phone(s). Dependent upon search results.	
restart	boolean	Optional	Indicates to restart (true) the phone(s). Dependent upon search results.	

Output Parameters: Empty Response

Example: Delete all phones which are a part of the group FirstPhoneGroup.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con=
↪"http://www.sipfoundry.org/2007/08/21/ConfigService">
<soapenv:Header/>
<soapenv:Body>
<con:ManagePhone>
<search>
<byGroup>FirstPhoneGroup</byGroup>
</search>
<deletePhone>true</deletePhone>
</con:ManagePhone>
</soapenv:Body>
</soapenv:Envelope>
```

Example: Add line with userid 221 to the phones in FirstPhoneGroup, generate the profiles for the phones and restart them.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con=
↪"http://www.sipfoundry.org/2007/08/21/ConfigService">
<soapenv:Header/>
<soapenv:Body>
<con:ManagePhone>
<search>
<byGroup>FirstPhoneGroup</byGroup>
</search>
<addLine>
<userId>221</userId>
<uri>221@Uniteme.ezuce.com</uri>
</addLine>
<generateProfiles>true</generateProfiles>
<restart>true</restart>
</con:ManagePhone>
</soapenv:Body>
</soapenv:Envelope>
```

Example: Remove all the phones in FirstPhoneGroup.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con=
↪"http://www.sipfoundry.org/2007/08/21/ConfigService">
<soapenv:Header/>
<soapenv:Body>
<con:ManagePhone>
<search>
<byGroup>FirstPhoneGroup</byGroup>
</search>
<removeGroup>FirstPhoneGroup</removeGroup>
</con:ManagePhone>
</soapenv:Body>
</soapenv:Envelope>
```

13.7 Test

The test web services are SOAP based services. These services use the Web Service Definition Language (WSDL) to define the interfaces supported.

URI

```
https://host.domain/sipxconfig/services/TestService
```

WSDL

```
<wsdl:definitions targetNamespace="http://www.sipfoundry.org/2007/08/21/ConfigService
↪" xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="http://www.
↪sipfoundry.org/2007/08/21/ConfigService" xmlns:intf="http://www.sipfoundry.org/2007/
↪08/21/ConfigService" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap=
↪"http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema
↪">
- <!--
WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)
-->
<wsdl:types>
```

(continues on next page)

(continued from previous page)

```
<schema targetNamespace="http://www.sipfoundry.org/2007/08/21/ConfigService" xmlns=
↪"http://www.w3.org/2001/XMLSchema">
<complexType name="ResetServices">
<sequence>
<element maxOccurs="1" minOccurs="0" name="callGroup" nillable="true" type=
↪"xsd:boolean" />
<element maxOccurs="1" minOccurs="0" name="parkOrbit" nillable="true" type=
↪"xsd:boolean" />
<element maxOccurs="1" minOccurs="0" name="permission" nillable="true" type=
↪"xsd:boolean" />
<element maxOccurs="1" minOccurs="0" name="phone" nillable="true" type="xsd:boolean" /
↪>
<element maxOccurs="1" minOccurs="0" name="user" nillable="true" type="xsd:boolean" />
<element maxOccurs="1" minOccurs="0" name="superAdmin" nillable="true" type=
↪"xsd:boolean" />
</sequence>
</complexType>
<element name="ResetServices" type="impl:ResetServices" />
</schema>
</wsdl:types>
<wsdl:message name="resetServicesRequest">
<wsdl:part element="impl:ResetServices" name="ResetServices" />
</wsdl:message>
<wsdl:message name="resetServicesResponse" />
<wsdl:portType name="TestService">
<wsdl:operation name="resetServices" parameterOrder="ResetServices">
<wsdl:input message="impl:resetServicesRequest" name="resetServicesRequest" />
<wsdl:output message="impl:resetServicesResponse" name="resetServicesResponse" />
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="TestServiceSoapBinding" type="impl:TestService">
<wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="resetServices">
<wsdlsoap:operation soapAction="" />
<wsdl:input name="resetServicesRequest">
<wsdlsoap:body use="literal" />
</wsdl:input>
<wsdl:output name="resetServicesResponse">
<wsdlsoap:body use="literal" />
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="ConfigImplService">
<wsdl:port binding="impl:TestServiceSoapBinding" name="TestService">
<wsdlsoap:address location="https://47.134.206.174:8443/sipxconfig/services/
↪TestService" />
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

13.7.1 Reset services

Name: *resetServices*

Description: Resets (deletes) the data associated with one or more web services.

Warning: This is an extremely dangerous service as it could permanently delete large amounts of configuration data. Use extreme caution!

Input Parameters:

Name	Value Type	Required/Optional	Description	Editable/Read Only
callGroup	boolean	Optional	Indicates to delete (true) all callGroup (hunt group) data.	Editable
parkOrbit	boolean	Optional	Indicates to delete (true) all call park orbits.	Editable
permission	boolean	Optional	Indicates to delete (true) all non-system defined permissions.	Editable
phone	boolean	Optional	Indicates to delete (true) all defined phones.	Editable
user	boolean	Optional	Indicates to delete (true) all defined users except superadmin.	Editable
superadmin	boolean	Optional	Indicates to delete (true) all superadmin data except for the PIN.	Editable

Output Parameters: Empty response.

Example: Remove all hunt groups, park orbits, and permissions defined.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:con=
↪"http://www.sipfoundry.org/2007/08/21/ConfigService">
<soapenv:Header/>
<soapenv:Body>
<con:ResetServices>
<!-Optional:->
<callGroup>true</callGroup>
<!-Optional:->
<parkOrbit>true</parkOrbit>
<!-Optional:->
<permission>true</permission>
</con:ResetServices>
</soapenv:Body>
</soapenv:Envelope>
```

CHAPTER

FOURTEEN

INDICES AND TABLES

- genindex
- modindex
- search

INDEX