

Sipwise NGCP Operations Manual

PRO/CARRIER

Sipwise GmbH <support@sipwise.com>

Table of Contents

1. Introduction . . . 1
   1.1. About this Handbook . . . 1
   1.2. What is the Sipwise C5 PRO/CARRIER? . . . 1
   1.3. The Advantages of the Sipwise C5 PRO/CARRIER . . . 1
   1.4. Who is the Sipwise C5 PRO/CARRIER for? . . . 2
   1.5. Getting Help . . . 2
2. Architecture . . . 3
   2.1. Platforms . . . 3
   2.2. Provisioning . . . 4
   2.3. API and Web Interface . . . 4
   2.4. SIP Signaling and Media Relay . . . 5
   2.5. MySQL Database . . . 11
   2.6. Redis Database . . . 12
   2.7. High Availability and Fail-Over . . . 13
   2.8. Scaling CARRIER beyond one Hardware Chassis . . . 14
   2.9. Scaling to a Geo-Redundant setup . . . 15
   2.10. Instances . . . 18
3. Deployment . . . 23
   3.1. Installation Prerequisites . . . 23
   3.2. Installation CARRIER . . . 26
   3.3. Installation Pro . . . 30
   3.4. What's happening when booting the Sipwise Deployment ISO . . . 38
4. Concept . . . 41
   4.1. Contacts . . . 41
   4.2. Resellers . . . 41
   4.3. SIP Domain . . . 42
   4.4. Contracts . . . 42
   4.5. Customers . . . 43
   4.6. Subscribers . . . 45
   4.7. SIP Peerings . . . 46
5. Kick-off . . . 48
   5.1. Creating a SIP Domain . . . 48
   5.2. Creating a Customer . . . 49
   5.3. Creating a Subscriber . . . 53
   5.4. Domain Preferences . . . 56
   5.5. Subscriber Preferences . . . 59
   5.6. Creating Peerings . . . 61
   5.7. Configuring Rewrite Rule Sets . . . 78
6. Billing . . . 95
   6.1. Billing Profiles . . . 95
   6.2. Peak Time Call Rating Modes . . . 101

6.3. Prepaid Accounting . . . 102
6.4. Fraud Detection and Locking . . . 102
6.5. Billing Customizations . . . 104
6.6. Notes on Billing and Call Rating . . . 132
6.7. Billing Data Export . . . 132
7. Features . . . 150
7.1. About the Admin Web Interface . . . 150
7.2. Managing System Administrators . . . 151
7.3. Access Control for SIP Calls . . . 156
7.4. Call Forwarding and Call Hunting . . . 166
7.5. Call Forking by Q value . . . 176
7.6. Local Number Porting . . . 177
7.7. P-Early-Media . . . 187
7.8. Emergency Mapping . . . 188
7.9. Emergency Priorization . . . 196
7.10. SIP Message Filtering . . . 202
7.11. SIP Trunking with SIPconnect . . . 208
7.12. Trusted Subscribers . . . 212
7.13. Peer Probing . . . 212
7.14. Fax Server . . . 216
7.15. Voicemail System . . . 218
7.16. Configuring Subscriber IVR Language . . . 225
7.17. Sound Sets . . . 226
7.18. Conference System . . . 233
7.19. Malicious Call Identification (MCID) . . . 237
7.20. Subscriber Profiles . . . 238
7.21. SIP Loop Detection . . . 240
7.22. Call-Through Application . . . 240
7.23. Calling Card Application . . . 244
7.24. Invoices and Invoice Templates . . . 248
7.25. Email Reports and Notifications . . . 263
7.26. The Vertical Service Code Interface . . . 270
7.27. XMPP and Instant Messaging . . . 273
7.28. Call Recording . . . 274
7.29. Media Transcoding and Transrating . . . 281
7.30. Announcement Before Call Setup . . . 288
7.31. Emulated Ringback Tone . . . 289
7.32. Store Recent Calls and Redial . . . 290
7.33. Time sets management . . . 292
7.34. Announcement To Callee . . . 305
7.35. Header Manipulations . . . 306
7.36. Phonebook . . . 313
7.37. Subscriber Location Mappings . . . 316

7.38. STIR/SHAKEN ... 322
7.39. Fileshare ... 327
7.40. Batch Provisioning ... 334
7.41. Call List Suppressions ... 356
7.42. Message Body Filtering ... 359
8. Extra Modules ... 360
8.1. Cloud PBX ... 360
8.2. Sipwise sip:phone App (SIP client) ... 467
8.3. Lawful Interception ... 487
8.4. 3rd Party Call Control ... 514
8.5. 3PCC functionality (TPCC), CSTA sessions and Websocket component ... 530
8.6. SMS (Short Message Service) on Sipwise C5 ... 551
9. Configuration Framework ... 564
9.1. Configuration templates ... 564
9.2. config.yml, constants.yml and network.yml files ... 569
9.3. ngcpcfg and its command line options ... 570
10. Network Configuration ... 573
10.1. General Structure ... 573
10.2. Advanced Network Configuration ... 577
11. Instances configuration ... 589
11.1. The network.yml structure ... 589
11.2. Instances operation ... 590
11.3. Interfaces ... 591
11.4. Connections between instances ... 593
11.5. Connections to databases ... 596
11.6. Disable default services ... 598
12. Software Upgrade ... 600
12.1. Release Notes ... 600
12.2. Overview ... 600
12.3. Planning a software upgrade ... 602
12.4. Pre-upgrade checks ... 602
12.5. Pre-upgrade steps ... 607
12.6. Upgrading Sipwise C5 CARRIER ... 609
12.7. Upgrading Sipwise C5 PRO ... 612
12.8. Post-upgrade steps ... 614
12.9. Applying the Latest Hotfixes ... 616
13. Backup, Recovery and Maintenance ... 618
13.1. Sipwise C5 Backup ... 618
13.2. Recovery ... 619
13.3. Reset Database ... 619
13.4. Synchronize database ... 619
13.5. Accounting Data (CDR) Cleanup ... 621
13.6. Managing packages ... 626

14. Security, Performance and Troubleshooting . . . 633
14.1. Sipwise SSH access to Sipwise C5 . . . 633
14.2. Firewalling . . . 633
14.3. Password management . . . 640
14.4. Remote 'root' logins via SSH . . . 641
14.5. 'sudo-io': logging input/output of commands run through 'sudo' . . . 642
14.6. SSL certificates . . . 643
14.7. Securing your Sipwise C5 against SIP attacks . . . 644
14.8. Topology Hiding . . . 646
14.9. System Requirements and Performance . . . 648
14.10. Troubleshooting . . . 650
14.11. Log file obfuscation . . . 656
14.12. NGCP Panel passwords encryption . . . 658
15. Monitoring and Alerting . . . 659
15.1. Internal Monitoring . . . 659
15.2. Statistics Dashboard . . . 660
15.3. External Monitoring Using SNMP . . . 661
16. Licenses . . . 667
16.1. What is Subject to Licensing? . . . 667
16.2. How Licensing Works . . . 667
16.3. How to Configure Licenses . . . 668
16.4. How to Monitor License Client . . . 668
17. Customer Self-Care . . . 670
17.1. Customer Self-Care User Interface (CSC UI) . . . 670
17.2. New (default) Vue.JS-based CSC UI . . . 670
17.3. Old (deprecated) Perl-based CSC UI . . . 670
17.4. The Customer Self-Care Web Interface . . . 670
17.5. The Voicemail Menu . . . 674
17.6. Company Hours . . . 675
18. REST API . . . 676
18.1. API Workflows for Customer and Subscriber Management . . . 676
18.2. API performance considerations . . . 683
Appendix . . . 684
Appendix A: Corosync/Pacemaker . . . 685
Appendix B: Basic Call Flows . . . 697
Appendix C: Configuration Overview . . . 703
Appendix D: MariaDB encryption . . . 757
Appendix E: Disk partitioning . . . 759
Appendix F: Faxserver Configuration . . . 762
Appendix G: Storage Node . . . 774
Appendix H: NGCP Internals . . . 775
Appendix I: Kamailio pv_headers module . . . 783
Appendix J: Extra Configuration Scenarios . . . 792

Appendix K: NGCP CLI helpers and tools 794
Appendix L: Generation of 181 Call Is Being Forwarded 801
Appendix M: Handling WebRTC Clients 802
Appendix N: Batch Provisioning Extras 803
Appendix O: CSTA 811
Appendix P: Instances 822

Chapter 1. Introduction

1.1. About this Handbook

This handbook describes the architecture and the operational steps to install, operate and modify the Sipwise C5 PRO/CARRIER.

In various chapters, it describes the system architecture, the installation and upgrade procedures and the initial configuration steps to get your first users online. It then dives into advanced preference configurations such as rewrite rules, call blocking, call forwarding, etc.

There is a description of the customer self-care interface, how to configure the billing system and how to provision the system via the API.

Finally, it describes the internal configuration framework, the network configuration and gives hints about tweaking the system for better security and performance.

1.2. What is the Sipwise C5 PRO/CARRIER?

Sipwise C5 (also known as NGCP - the Next Generation Communication Platform) is a SIP-based Open Source Class 5 VoIP soft-switch platform that allows you to provide rich telephony services. It offers a wide range of features (e.g. call forwarding, voicemail, conferencing etc.) that can be configured by end users in the self-care web interface. For operators, it offers a web-based administrative panel that allows them to configure subscribers, SIP peerings, billing profiles, and other entities. The administrative web panel also shows the real-time statistics for the whole system. For tight integration into existing infrastructures, Sipwise C5 provides a powerful REST API interface.

Sipwise C5 has three solutions that differ in call capacity and service redundancy: CARRIER, PRO and CE. The current handbook describes the PRO/CARRIER solution.

The Sipwise C5 CARRIER comes pre-installed on six or more servers in one or more Lenovo Flex System Enterprise Chassis, see Architecture. Apart from your product specific configuration, there is no initial configuration or installation to be done to get started.

The Sipwise C5 PRO can be pre-installed on two hardware servers or deployed in a customer virtual environment. Apart from your product specific configuration, there is no initial configuration or installation to be done to get started.

1.3. The Advantages of the Sipwise C5 PRO/CARRIER

Opposed to free VoIP software, Sipwise C5 is not a single application, but a complete software platform based on Debian GNU/Linux.

Using a highly modular design approach, Sipwise C5 leverages popular open-source software like MySQL, NGINX, Kamailio, SEMS, Asterisk, etc. as its core building blocks. These blocks are glued together using optimized and proven configurations and workflows and are complemented by functionality developed by Sipwise to provide fully-featured and easy-to-operate VoIP services.

The installed applications are managed by the Sipwise C5 Configuration Framework. This configuration framework makes it possible to change low-level system parameters in a single place, so Sipwise C5

Copyright © Sipwise GmbH - All rights reserved

administrators don't need to have any knowledge of dozens of different configuration files from different packages. This provides a bullet-proof way of operating, changing and tweaking an otherwise quite complex system.

Once configured, integrated web interfaces are provided for both end users and Sipwise C5 administrators. Provisioning and billing API allows companies to tightly integrate Sipwise C5 into existing OSS/BSS infrastructures to optimize workflows.

1.4. Who is the Sipwise C5 PRO/CARRIER for?

The Sipwise C5 PRO/CARRIER are specifically tailored to companies who want to provide fully-featured SIP-based VoIP service without having to go through the steep learning curve of SIP signalling. It integrates the different building blocks to make them work together in a reasonable way. The Sipwise C5 PRO/CARRIER is already deployed all around the world by all kinds of VoIP operators, using it as Class5 soft-switch, as Class4 termination platform or even as Session Border Controller with all kinds of access networks, like Cable, DSL, WiFi and Mobile networks.

1.5. Getting Help

1.5.1. Phone Support

Depending on your support contract, you are eligible to contact our Support Team by phone either during business hours or around the clock. Business hours refer to the CET/CEST time zone (Europe/Vienna). Please check your support contract to find out the type of support you've purchased.

Before calling our Support Team, please also open a ticket in our Ticket System and provide as much detail as you can for us to understand the problems, fix them and investigate the cause. Please provide the number of your newly created ticket when asked by our support personnel on the phone.

You can find phone numbers, Ticket System URL, and account information in your support contract. Please make this information available to the persons in your company maintaining Sipwise C5.

1.5.2. Ticket System

Depending on your support contract, you can create either a limited or an unlimited amount of support tickets on our Web-based Ticket System. Please provide as much information as possible when opening a ticket, especially the following:

- **WHAT** is affected (e.g. the whole system is unreachable, or customers can't register or place calls)
- **WHO** is affected (e.g. all customers, only parts of it, and **WHICH** parts - only customers in a particular domain or customers with specific devices, etc.)
- **WHEN** did the problem occur (time frames, or after the firmware of specific devices types have been updated, etc.)

Our Support Team will ask further questions via the Ticket System along the way of troubleshooting your issue. Please provide the information as soon as possible to solve your issue promptly.

Copyright © Sipwise GmbH - All rights reserved

Chapter 2. Architecture

2.1. Platforms

2.1.1. CARRIER Platform

The Sipwise C5 CARRIER platform is composed by a cluster of four different node types, which are all deployed in active/standby pairs:

- **Web-Servers** (web1a/web1b): Provide northbound interfaces (CSC, API) via HTTPS for provisioning
- **DB-Servers** (db1a/db1b): Provide the central persistent SQL data store for customer data, peering configuration, billing data etc.
- **Proxy-Servers** (proxy1a/proxy1b .. proxy4a/proxy4b): Provide the SIP and XMPP signalling engines, application servers and media relays to route Calls and IM/Presence and serve media to the endpoints.
- **Load-Balancers** (lb1a/lb1b): Provide a perimeter for SIP and XMPP signalling.

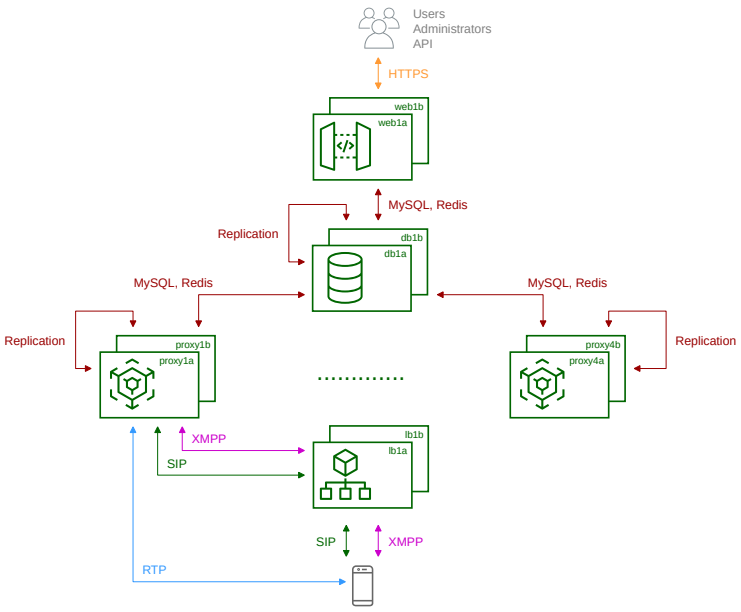


Figure 1. CARRIER Architecture Overview

The system is provisioned via the web servers on a central pair of db servers. Signalling is entering the system via the *lb* servers to a cluster of proxies, which in turn communicate directly (caching and shared data) and indirectly (static provisioning data replicated via master/slave) with the db servers. Each pair of proxy is capable of handling any subscriber, so subscribers are not bound to specific "home proxies". Once a call starts on a proxy pair, it is ensured that the full range of services is provided on that pair (voicemail, media, billing, ...) until call-teardown. Failures on an active *proxy* node cause a fail-over to the corresponding stand-by node within the *proxy* pair, taking over the full signalling and media without interruptions.

2.1.2. PRO Platform

The Sipwise C5 PRO platform consists of two identical appliances working in active/standby mode. The components of a node are outlined in the following figure:

Copyright © Sipwise GmbH - All rights reserved

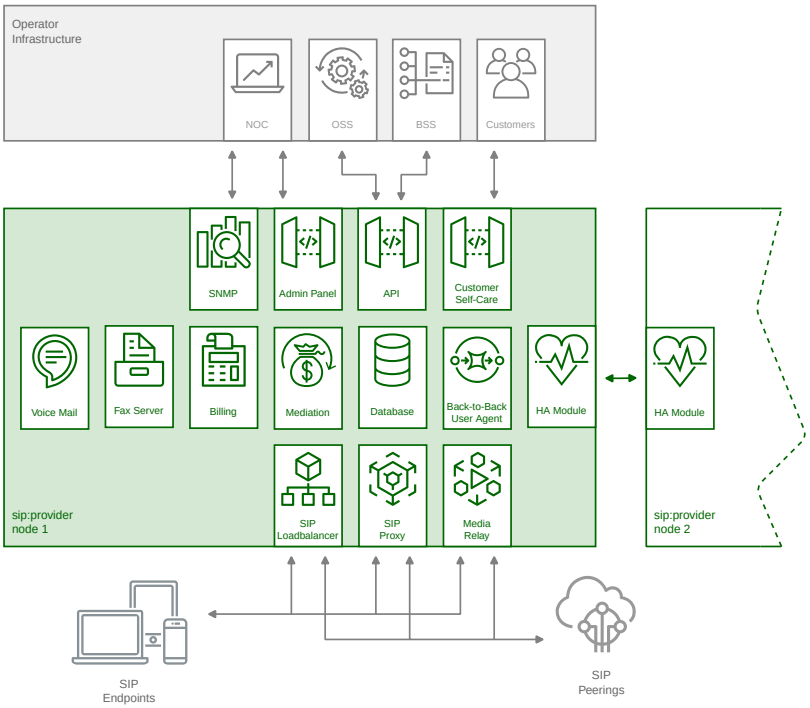


Figure 2. PRO Architecture Overview

The main building blocks of Sipwise C5 are:

- Provisioning
- SIP Signaling and Media Relay
- Mediation and Billing
- Monitoring and Alerting
- High Availability and Fail-Over

2.2. Provisioning

Any HTTPS traffic for provisioning (web interfaces, northbound APIs) but also for phone auto-provisioning enters the platform on the active web server. The web server runs an nginx instance acting as a reverse proxy for the ngcp-panel process, which in turn provides the provisioning functionality.

The web server is connected to the db server pair, which provides a persistent relational data store via MySQL and a high-performance system cache using Redis key-value store.

2.3. API and Web Interface

Copyright © Sipwise GmbH - All rights reserved

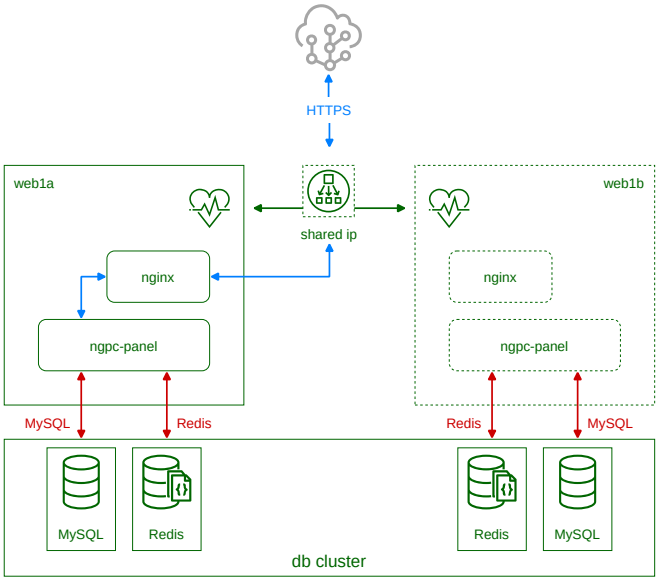


Figure 3. CARRIER Web Server Overview

The web server pair is an active/standby pair of nodes connected via an HA service (GCS/CRM). If one of the servers fail (by losing connection to the outside while the standby server is still connected, or caused by a hardware failure, or if it's down due to maintenance), the standby server takes over the shared IP address of the active node and continues serving the provisioning interface.

2.4. SIP Signaling and Media Relay

In SIP-based communication networks, it is important to understand that the signaling path (e.g. for call setup and tear-down) is completely independent of the media path. On the signaling path, the involved endpoints negotiate the call routing (which user calls which endpoint, and via which path - e.g. using SIP peerings or going through the PSTN - the call is established) as well as the media attributes (via which IPs/ports are media streams sent and which capabilities do these streams have - e.g. video using H.261 or Fax using T.38 or plain voice using G.711). Once the negotiation on signaling level is done, the endpoints start to send their media streams via the negotiated paths.

On a CARRIER any signalling traffic enters and leaves the system via load balancers, which act as a perimeter towards the customer devices and performs NAT handling, DoS and DDoS mitigation. New connections are routed to a random pair of proxy servers, which do the actual routing for SIP and XMPP. The proxy servers also engage media relays for voice and video streams, which bypass the load balancers and communicate directly with the customer devices for performance reasons.

The components involved in SIP and Media on the Sipwise C5 PRO/CARRIER are shown in the following figure:

Copyright © Sipwise GmbH - All rights reserved

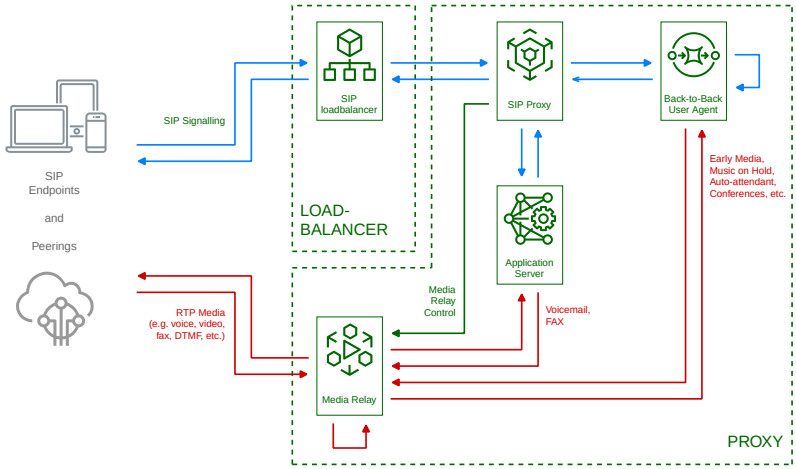


Figure 4. SIP and Media Relay Components

2.4.1. SIP Load-Balancer

The SIP load-balancer is a Kamailio instance acting as ingress and egress point for all SIP traffic to and from the system. It's a high-performance SIP proxy instance based on Kamailio and is responsible for sanity checks of inbound SIP traffic. It filters broken SIP messages, rejects loops and relay attempts and detects denial-of-service and brute-force attacks and gracefully handles them to protect the underlying SIP elements. It also performs the conversion of TLS to internal UDP and vice versa for secure signaling between endpoints and Sipwise C5, and does far-end NAT traversal in order to enable signaling through NAT devices.

The load-balancer is the only SIP element in the system which exposes a SIP interface to the public network. Its second leg binds in the switch-internal network to pass traffic from the public internet to the corresponding internal components.

The name load-balancer comes from the fact that when scaling out Sipwise C5 beyond one pair of servers, the load-balancer instance becomes its own physical node and then handles multiple pairs of proxies behind it.

On the public interface, the load-balancer listens on port 5060 for UDP and TCP, as well as on 5061 for TLS connections. On the internal interface, it speaks SIP via UDP on port 5060 to the other system components, and listens for XMLRPC connections on TCP port 5060, which can be used to control the daemon.

 Copyright © Sipwise GmbH - All rights reserved

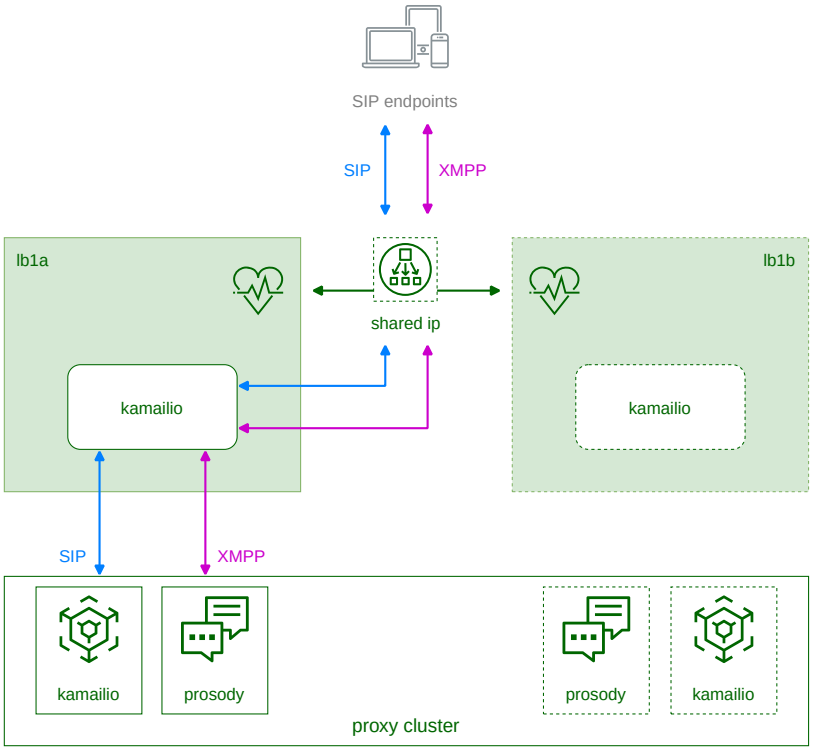


Figure 5. CARRIER Load Balancer Overview

A node in a load balancer pair runs two services besides the usual HA service.

One is a state-less instance of kamailio, providing an extremely fast relay of SIP messages. Kamailio takes care of converting TCP and TLS connections from the customer devices to UDP for internal communication towards proxies, and it performs far-end NAT traversal by inspecting the SIP messages and comparing it to the actual source address where packets have been received from, then modifying the SIP messages accordingly. If a SIP message is received by the load balancer, it distinguishes between new and ongoing SIP transactions by inspecting the To-Tags of a message, and it determines whether the message is part of an established dialog by inspecting the Route header. Sanity checks are performed on the headers to make sure the call flows adhere to certain rules for not being able to bypass any required element in the routing path. In-dialog messages are routed to the corresponding proxy servers according to the Route defined in the message. Messages initiating a new transaction and/or dialog (registrations, calls etc) are routed to a randomly selected proxy. The selection algorithm is based on a hash over the Call-ID of the message, so the same proxy sending a authentication challenge to an endpoint will receive the authenticated message again.

The second service running on a load balancer is *haproxy*, which is acting as load balancing instance for XMPP messages. The same way the SIP load balancer routes SIP messages to the corresponding proxy, the haproxy passes XMPP traffic on to the proxy maintaining a session with a subscriber, or randomly selects a proxy in case of a new connection while automatically failing over on timeouts.

Its config files reside in `/etc/ngcp-config/templates/etc/kamailio/lb/`, and changes to these files are applied by executing `ngcpcfg apply "my commit message"`.

Copyright © Sipwise GmbH - All rights reserved

TIP

The SIP load-balancer can be managed via the commands `ngcp-service start kamailio-lb`, `ngcp-service stop kamailio-lb` and `ngcp-service restart kamailio-lb`. Its status can be queried by executing `ngcp-service status kamailio-lb` or `ngcp-service summary | grep "kamailio-lb"`. Also `ngcp-kamctl lb` and `ngcp-kamcmd lb` are provided for querying kamailio functions, for example: `ngcp-kamcmd lb htable.dump ipban`. Execute the command: `ngcp-kamctl lb fifo system.listMethods` or `ngcp-kamcmd lb system.listMethods` to get the list of all available queries.

2.4.2. SIP Proxy/Registrar

The SIP proxy/registrar (or short *proxy*) is the work-horse of Sipwise C5. It's also a separate Kamailio instance running in the switch-internal network and is connected to the provisioning database via MySQL, authenticates the endpoints, handles their registrations on the system and does the call routing based on the provisioning data. It is also connected to no-sql backend (Redis) for processing speed purposes and for e.g. in this way manages ACC data, location records etc. For each call, the proxy looks up the provisioned features of both the calling and the called party (either subscriber or domain features if it's a local caller and/or callee, or peering features if it's from/to an external endpoint) and acts accordingly, e.g. by checking if the call is blocked, by placing call-forwards if applicable and by normalizing numbers into the appropriate format, depending on the source and destination of a call.

It also writes start- and stop-records for each call, which are then transformed into call detail records (CDR) by the mediation system.

If the endpoints indicate negotiation of one or more media streams, the proxy also interacts with the *Media Relay* to open, change and close port pairs for relaying media streams over Sipwise C5, which is especially important to traverse NAT.

The proxy listens on UDP port 5062 in the system-internal network. It cannot be reached directly from the outside, but only via the SIP load-balancer.

Its config files reside in `/etc/ngcp-config/templates/etc/kamailio/proxy/`, and changes to these files are applied by executing `ngcpcfg apply "my commit message"`.

TIP

The SIP proxy can be controlled via the commands `ngcp-service start kamailio-proxy`, `ngcp-service stop kamailio-proxy` and `ngcp-service restart kamailio-proxy`. Its status can be queried by executing `ngcp-service status kamailio-proxy` or `ngcp-service summary | grep "kamailio-proxy"`. Also `ngcp-kamctl proxy` and `ngcp-kamcmd proxy` are provided for querying kamailio functions, for example: `ngcp-kamctl proxy ul show`. Execute the command: `ngcp-kamctl proxy fifo system.listMethods` or `ngcp-kamcmd proxy system.listMethods` to get the list of all available queries.

2.4.3. SIP Back-to-Back User-Agent (B2BUA)

The SIP B2BUA (also called SBC within the system) decouples the first call-leg (calling party to Sipwise C5) from the second call-leg (Sipwise C5 to the called party).

The software part used for this element is a commercial version of SEMS, with the main difference to the open-source version that it includes a replication module to share its call states with the stand-by node.

 Copyright © Sipwise GmbH - All rights reserved

This element is typically optional in SIP systems, but it is always used for SIP calls (INVITE) that don't have Sipwise C5 as endpoint. It acts as application server for various scenarios (e.g. for feature provisioning via Vertical Service Codes and as Conferencing Server) and performs the B2BUA decoupling, topology hiding, caller information hiding, SIP header and Media feature filtering, outbound registration, outbound authentication, Prepaid accounting and call length limitation as well as Session Keep-Alive handler.

Due to the fact that typical SIP proxies (like the load-balancer and proxy in Sipwise C5) do only interfere with the content of SIP messages where it's necessary for the SIP routing, but otherwise leave the message intact as received from the endpoints, whereas the B2BUA creates a new call leg with a new SIP message from scratch towards the called party, SIP message sizes are reduced significantly by the B2BUA. This helps to bring the message size under 1500 bytes (which is a typical default value for the MTU size) when it leaves Sipwise C5. That way, chances of packet fragmentation are quite low, which reduces the risk of running into issues with low-cost SOHO routers at customer sides, which typically have problems with UDP packet fragmentation.

The SIP B2BUA only binds to the system-internal network and listens on UDP port 5080 for SIP messages from the load-balancer or the proxy, on UDP port 5048 for control messages from the cli tool and on TCP port 8090 for XMLRPC connections to control the daemon.

In cases when B2B is engaged into processing the media (RTP/RTCP data), it uses this UDP ports range by default: 15000 - 19999.

Its configuration files reside in `/etc/ngcp-config/templates/etc/sems-b2b`, and changes to these files are applied by executing `ngcpcfg apply "my commit message"`.

TIP The SIP B2BUA can be controlled via the commands `ngcp-service start b2b`, `ngcp-service stop b2b` and `ngcp-service restart b2b`. Its status can be queried by executing `ngcp-service status b2b` or `ngcp-service summary | grep "b2b"`.

2.4.4. SIP App-Server

The SIP App-Server is an Asterisk instance used for voice applications like Voicemail and Reminder Calls. It is also used in the software-based Faxserver solution to transcode SIP and RTP into the IAX protocol and vice versa, in order to talk to the Software Fax Modems. Asterisk uses the MySQL database as a message spool for voicemail, so it doesn't directly access the file system for user data. The voicemail plugin is a slightly patched version based on Asterisk 16.2.1 to make Asterisk aware of Sipwise C5 internal UUIDs for each subscriber. That way a SIP subscriber can have multiple E164 phone numbers, but all of them terminate in the same voicebox.

The App-Server listens on the internal interface on UDP port 5070 for SIP messages and by default uses media ports in the range from UDP port 10000 to 14999.

The configuration files reside in `/etc/ngcp-config/templates/etc/asterisk`, and changes to these files are applied by executing `ngcpcfg apply "my commit message"`.

TIP The SIP App-Server can be controlled via the commands `ngcp-service start asterisk`, `ngcp-service stop asterisk` and `ngcp-service restart asterisk`. Its status can be queried by executing `ngcp-service status asterisk` or `ngcp-service summary | grep "asterisk"`.

Copyright © Sipwise GmbH - All rights reserved

2.4.5. Message Routing and Media Relay

The Media Relay (also called *rtpengine*) is a Kernel-based packet relay, which is controlled by the SIP proxy. For each media stream (e.g. a voice and/or video stream), it maintains a pair of ports in the range of port number 30000 to 44999. When the media streams are negotiated, *rtpengine* opens the ports in user-space and starts relaying the packets to the addresses announced by the endpoints. If packets arrive from different source addresses than announced in the SDP body of the SIP message (e.g. in case of NAT), the source address is implicitly changed to the address the packets are received from. Once the call is established and the rtpengine has received media packets from both endpoints for this call, the media stream is pushed into the kernel and is then handled by a custom Sipwise iptables module to increase the throughput of the system and to reduce the latency of media packets.

The rtpengine internally listens on UDP port 12222 for control messages from the SIP proxy. For each media stream, it opens two pairs of UDP ports on the public interface in the range of 30000 and 40000 per default, one pair on even port numbers for the media data, and one pair on the next odd port numbers for metadata, e.g. RTCP in case of RTP streams. Each endpoint communicates with one dedicated port per media stream (opposed to some implementations which use one pair for both endpoints) to avoid issues in determining where to send a packet to. The rtpengine also sets the QoS/ToS/DSCP field of each IP packet it sends to a configured value, 184 (0xB8, *expedited forwarding*) by default.

The kernel-internal part of the rtpengine is facilitated through an *iptables* module having the target name `RTPENGINE`. If any additional firewall or packet filtering rules are installed, it is imperative that this rule remains untouched and stays in place. Otherwise, if the rule is removed from iptables, the kernel will not be able to forward the media packets and forwarding will fall back to the user-space daemon. The packets will still be forwarded normally, but performance will be much worse under those circumstances, which will be especially noticeable when a lot of media streams are active concurrently. See the section on Firewalling for more information.

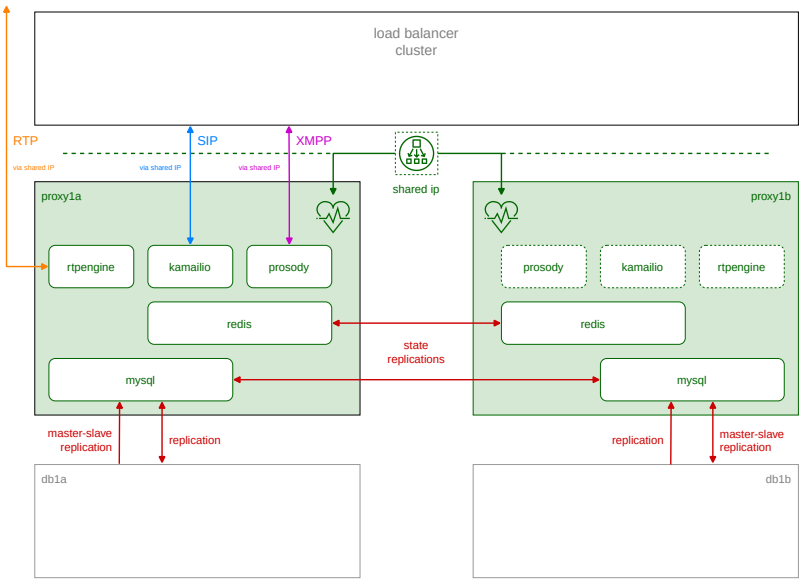


Figure 6. CARRIER Proxy Server Overview

Proxy servers also come in pairs, and by default there are four pairs of proxies in a standard Sipwise C5 CARRIER setup.

The proxies are responsible for doing the actual SIP routing and media handling and the XMPP presence and chat message deliveries. Each proxy pair can handle any subscriber on the overall

 Copyright © Sipwise GmbH - All rights reserved

system, compared to the concept of "home proxies" in other architectures. The advantage of this approach is that the overall system can be scaled extremely easily by adding more proxy pairs without having to redistribute subscribers.

Once a load balancer sends a new message to a proxy, the SIP transaction and/or dialog gets anchored to this proxy. That way it is ensured that a call starting on a proxy is also ended on the same proxy. Hence, the full range of feature handling like media relay, voicemail, fax, billing and rating is performed on this proxy. So, there is no a central point for various tasks, potentially leading to a non-scalable bottleneck. Due to the anchoring, proxies come in pairs and replicate all internal state information to the standby node via Redis. In case of fail-over, the full signalling and media are moved to the standby node without interruption.

The complete static subscriber information like authentication credentials, number mappings, feature settings etc. are replicated from the db cluster down to the local MySQL instance of the proxies. The ratio of db read requests of static subscriber data versus reading and writing volatile and shared data is around 15:1, and this approach moves the majority of the static read operations from the central db cluster to the local proxy db.

Volatile and shared information needed by all proxies in the cluster is read from and written to the db cluster. This mainly includes SIP registration information and XMPP connection information.

Billing and rating is also performed locally on the proxies, and only completed CDRs (rated or unrated depending on whether rating is enabled) are transferred to the central db cluster for consumption via the northbound interfaces.

For SIP, the relevant instances on a proxy are kamailio acting as a stateful proxy for SIP registration and call routing, sems acting as a back-to-back user-agent for prepaid billing and application server, rtpengine as media relay and RTP/SRTP transcoder, and asterisk as voicemail server. XMPP is handled by an instance of prosody, and several billing processes mediate start and stop records into CDRs and rate them according to the relevant billing profiles.

The rtpengine configuration file is `/etc/ngcp-config/templates/etc/default/ngcp-rtpengine-daemon`, and changes to this file are applied by executing `ngcpcfg apply "my commit message"`. The UDP port range can be configured via the `config.yml` file under the section `rtpengine`. The QoS/ToS value can be changed via the key `qos.tos_rtp`.

TIP The Media Relay can be controlled via the commands `ngcp-service start rtpengine`, `ngcp-service stop rtpengine` and `ngcp-serivce restart rtpengine`. Its status can be queried by executing `ngcp-service status rtpengine"` or `ngcp-service summary | grep "rtpengine"`.

2.5. MySQL Database

The MySQL database consists of a pair of active/standby MySQL servers. They run a MySQL master/master replication with replication integrity checks to ensure data consistency and redundancy.

The MySQL servers on both physical nodes synchronize via the row-based master/master replication. In theory, any of the two servers in the pair can be used to write data to the database, however, in practice the shared IP address is used towards clients accessing the service, hence only the active MySQL server will receive the write requests and replicate them to the standby one.

Copyright © Sipwise GmbH - All rights reserved

2.5.1. Provisioning Database (CARRIER-only)

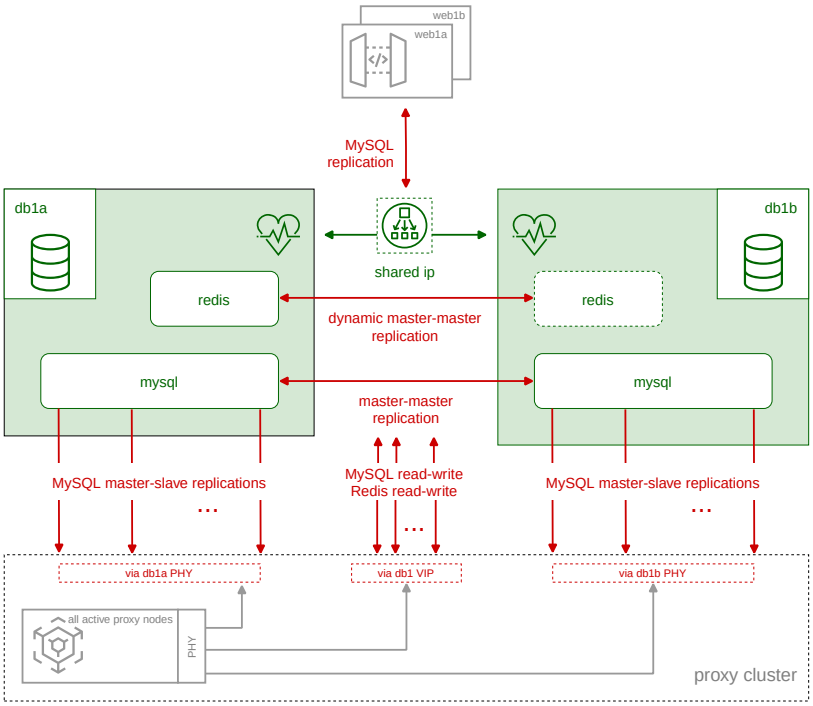


Figure 7. CARRIER DB Server Overview

The db server pair is another active/standby pair with automatic fail-over. Nodes in the pair are running a MySQL master/master replication with replication integrity checks to ensure data redundancy and safety. Any changes via provisioning interfaces are stored in the MySQL cluster. The second service is a Redis master/slave replication with automatic master propagation on fail-over. This Redis cluster is used as a high-performance volatile system cache for various components which need to share state information across nodes.

2.5.2. Persistent MySQL Database (CARRIER-only)

The MySQL instances on the *db* nodes synchronize via row-based master/master replication. In theory, any of the two servers in the pair can be used to write data to the database, however in practice a shared IP is used towards clients accessing the service, so only one node will receive the write requests. This is done to ensure transparent and instant convergence of the db cluster on fail-over for the clients.

On top of that, the first node of the db pair also acts as a master in a master/slave replication towards all *proxy* nodes in the system. That way, proxies can access read-only provisioning data directly from their local databases, resulting in reduced latency and significant off-loading of read queries on the central db cluster.

2.6. Redis Database

The Redis database is used as a high-performance key/value storage for global system datashared across proxies. This includes calls information and concurrent calls counters for customers and subscribers, etc..

The active-standby replication ensures that the data is immediately copied from the active node to the standby one. As all sensitive call information is held in the shared storage, Sipwise C5 makes it possible

 Copyright © Sipwise GmbH - All rights reserved

to switch the operational state from active to standby on one physical node and from standby to active on the other node without any call interruptions. Your subscribers will never notice that their calls being established on one physical server, were successfully moved to another one and successfully completed there.

On a CARRIER a Redis master/slave setup is used to provide a high-performance key/value storage for global system data shared across proxies. This includes concurrent call counters for customers and subscribers, as a subscriber could place two simultaneous calls via two different proxy pairs.

2.7. High Availability and Fail-Over

2.7.1. Overview

The two servers of a complete Sipwise C5 system form a pair, a simple cluster with two nodes. Their names are fixed as `sp1` and `sp2`, however neither of them is inherently a *first* or a *second*. They're both equal and identical and either can be the active node of the cluster at any time. Only one node is always ever active, the other one is in standby mode and does not perform any active functions.

High availability is achieved through constant communication between the two nodes and constant state replication from the active node to the standby one. Whenever the standby node detects that the other node has become unresponsive, has gone offline and has failed in any other way, it will proceed with taking over all resources and becoming the active node, with all operations resuming where the failed node has left off. Through that, the system will remain fully operational and service disruption will be minimal.

When the failed node comes back to life, it will become the new standby node, replicate everything that has changed in the meantime from the new active node, and then the cluster will be back in fully highly available state.

TIP The login banner at the SSH shell provides information about whether the local system is currently the active one or the standby one. See Administration for other ways to differentiate between the active and the standby node.

2.7.2. Nomenclature and Alternatives

The HA architecture consists of two components: the Group Communication System, also known as *GCS*, and the Cluster Resource Manager, also known as *CRM*. Sipwise C5 supports two alternatives for these components:

1. Corosync/Pacemaker: This is the newer and more modern software and the successor of Heartbeat version 2. It splits the HA framework into its two components, with Corosync providing the GCS service and Pacemaker providing the CRM service. It provides several additional features over Heartbeat version 2, and is the default for new Sipwise C5 installations. See the Corosync/Pacemaker chapter for detailed information.
2. Heartbeat version 2: This is the older and more basic software which provided both GCS and CRM services. It is now obsolete and not available anymore, and systems that use it should have been migrated away before upgrading to this release.

Copyright © Sipwise GmbH - All rights reserved

2.7.3. Core Concepts and Configuration

The direct Ethernet crosslink between the two nodes provides the main mechanism of HA communication between them. All state replication happens over this link. Additionally, the GCS service uses this link to communicate with the other node to see if it's still alive and active. A break in this link will therefore result in a *split brain* scenario, with either node trying to become the active one. This is to be avoided at all costs.

The `config.yml` file allows specification of a list of *ping nodes* under the key `ha.pingnodes`, which are used by the CRM service to determine if local network communications are healthy. Both servers will then constantly compare the number of locally reachable ping nodes with each other, and if the standby server is able to reach more of them, then it will become the active one.

The main resource that the CRM service manages is the shared service IP address. Each node has its own static IP address configured on its first Ethernet interface (`neth0`), which is done outside of the Sipwise C5 configuration framework (i.e. in the Debian-specific config file `/etc/network/interfaces`). The shared service IP is specified in `network.yml` at the key `hosts.sp1|sp2.neth0.shared_ip`. The CRM service will configure it as a secondary IP address on the first Ethernet interface (`neth0:0`) on the active node and will deconfigure it on the standby node. Thus, all network communications with this IP address will always go only to the currently active node.

2.7.4. Administration

The current status of the local Sipwise C5 node can be determined using the `ngcp-check-active` shell command. This command produces no output, but returns an exit status of *0* for the active node and *1* for the standby node. A more complete shell command to produce visible output could be: `ngcp-check-active -v`

To force a currently active node into standby mode, use the command `ngcp-make-standby`. For the opposite effect, use the command `ngcp-make-active`. This will also always affect the state of the other node, as the system automatically makes sure that always only one node is active at a time.

2.8. Scaling CARRIER beyond one Hardware Chassis

If Sipwise C5 CARRIER is scaled beyond 250,000 subscribers and therefore exceeds one chassis, a second chassis is put into place. This chassis provides another two web servers, two db servers, two load balancers and 8 proxies, doubling the capacity of the system.

2.8.1. Scaling the DB cluster

The DB cluster is the only node type which requires a notable change on the architecture.

DB01a/b nodes have master<->master replication for High-Availability

DB01prx01a + DB01prx01b are masterslave replication for read/write scale (write to remote/shared db01, read from local prx DB).

Separate hot and cold data. Hot in Redis for low IO. Cold in MariaDB.

Separate huge data (e.g. voicemail, voisniff data) to separate 'storage' DB node.

With such setup the central db01 pair can handle all the planned and unexpected DB load without the

 Copyright © Sipwise GmbH - All rights reserved

significant hardware resource usage. DB01a and DB01b can be located in different Geo-locations for High-Availability (low latency link is required for replications).

Further DB nodes scalability can be achieved using Geo-redundant setup. Please contact Sipwise sales team for more details here.

2.8.2. Scaling the proxy cluster

New proxy nodes replicate via master/slave from the *db* nodes in the chassis as usual. Since the db cluster holds all provisioning information of all subscribers, the *proxy* nodes join the cluster transparently and will start serving subscribers as soon as all services on a new proxy are reachable from the load balancers.

2.8.3. Scaling the load balancers

Load balancers start serving subscribers as soon as they are made visible to the subscribers. This could either be done via DNS round-robin, but the better approach is to configure a DNS SRV record, which allows for more fine-grained control like weighting load-balancer pairs and allowing fail-over from one pair to another on the client side.

The load balancers use the Path extension of SIP to make sure during SIP registration that calls targeted to a subscriber are routed via the same load balancer pair which the subscriber used during registration for proper traversal of symmetric NAT at the customer premise.

A SIP or XMPP request reaching a load balancer can be routed to any available proxy in the whole system, or only to proxies belonging to the same chassis as the load balancer, depending on the system configuration.

2.8.4. Scaling the web servers

New web server pairs are made available to web clients via DNS round-robin. Any pair of web servers can be used to read or write provisioning information via the web interfaces or the API.

2.9. Scaling to a Geo-Redundant setup

A basic Geo-Redundant configuration can be achieved by simply deploying all the sp1 nodes (A nodes in case of CARRIER) into a location and the sp2 nodes (B nodes in case of CARRIER) into another one. The locations have to be connected by a reliable and low latency layer 2 link.

This setup has many advantages and gives the possibility to have full business continuity in case one of the locations goes completely down but it has also some drawbacks:

- in case one site goes down, the remaining site is in an 'unstable' state due to missing HA nodes.
- only one location is active at a time. Therefore it is not possible to take advantage of all the benefits of a geo-located system. This can be improved by activating services as 'instances' as described here.
- in case the connection between the two systems goes down, a split brain scenario will happen causing an instability of the whole system.

A new alternative approach consists of the deployment of two fully operational PRO or Carrier systems, from now on referred to as 'cluster', connected by the aforementioned reliable and low latency layer 2

Copyright © Sipwise GmbH - All rights reserved

link.

Compared to the previous setup, this configuration has the following advantages:

- single management interface access (API and Web interface) for both systems
- geo-locate the SIP/RTP connections and traffic in order to always connect endpoints to the nearest cluster (it requires dedicated network configurations)
- fallback of the endpoint connections to the other cluster node
- communications between subscribers registered on different clusters remains internal
- in case one cluster goes down, the other continues to work in standard mode with a local HA
- in case the interconnection between clusters down, the two systems continue to work independently. After the connection is re-established, a dedicated DB resynchronization will happen.

IMPORTANT

In this architecture the interconnection link between the two clusters will be used not only for database replica and internal synchronization but also for internal SIP/RTP traffic. Due to that, it is important that the link has very low latency and high throughput.

The final setup looks like:

Copyright © Sipwise GmbH - All rights reserved

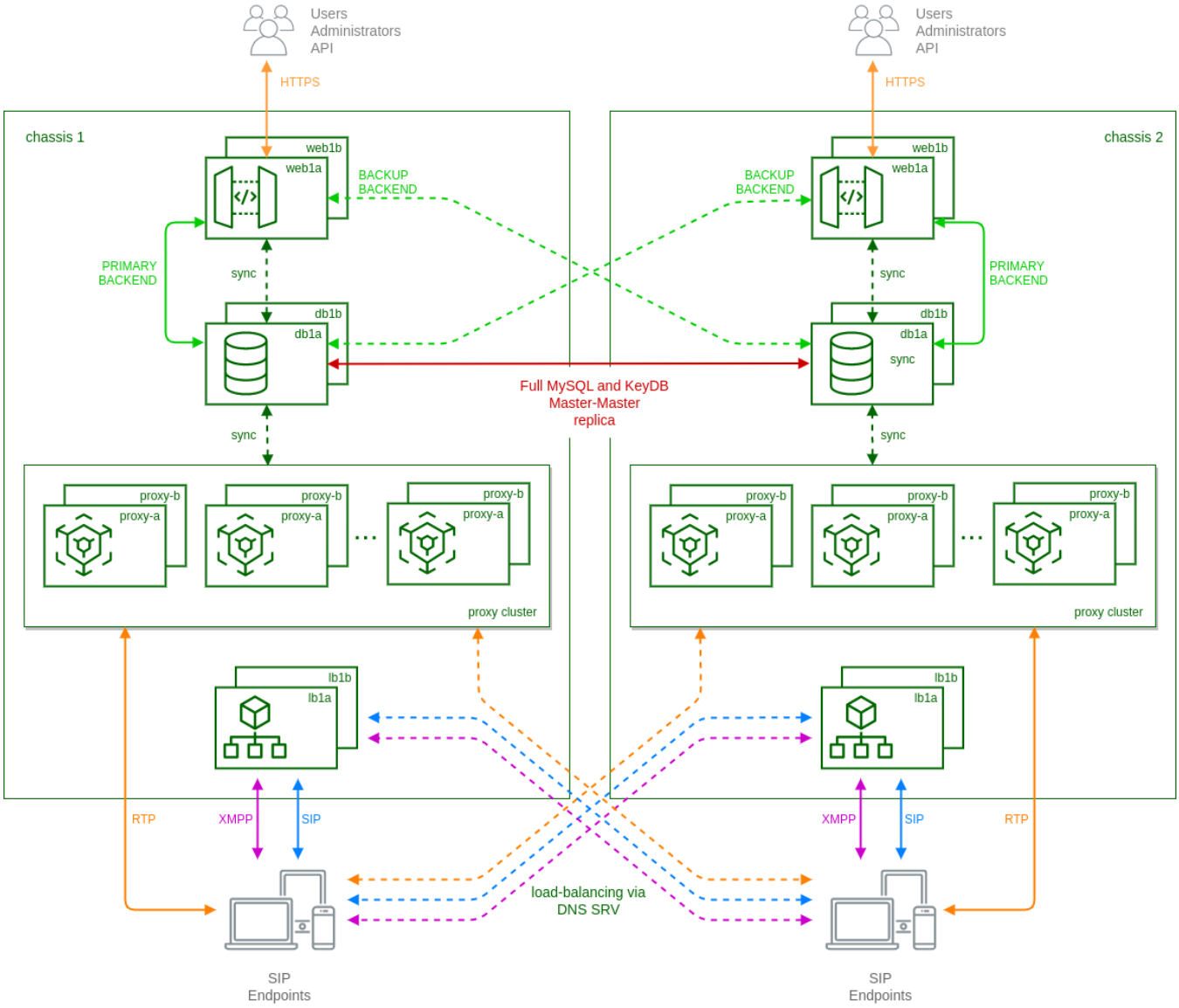


Figure 8. Geo-Redundant Carrier Overview

In particular:

- each 'web' node has a built-in mechanism based on ha-proxy to select which 'db' node to use as backend. By default they always try to use the local node as primary connection and switch to the remote node in case of missing local connection or faulty local replica status.
- mysql and keydb databases running on 'db' nodes are in a full master-master replica setup. This is to ensure the presence of same data (provisioning, locations, CDRs, etc.) on both clusters.
- each cluster uses local 'lb' and 'prx' nodes as in standard carrier architecture, except when the called subscriber is registered on the other cluster. In these cases local 'prx' node can directly contact the remote 'lb' to route the calls to the final destination.
- using DNS-SRV records, endpoints are usually registered on the nearest cluster. In case of failure of the local connection, then DNS-SRV is responsible to route new registrations to remote cluster.
- ngcpcfg framework enables the operator to manage the whole system from one cluster node.

For any additional details on the Geo-Redundant setup and how to configure it, please contact Sipwise sales.

Copyright © Sipwise GmbH - All rights reserved

2.10. Instances

The Sipwise C5 instance is the basic building block to operate Sipwise C5 in a new way (using active-active concepts).

In summary, it is kind of the micro-services architecture for Sipwise C5. The main idea is to isolate each Sipwise C5 service, assign a dedicated floating IP address to it and allow it to work on a specific node/location.

An instance is defined as <service>:<label>@<location>.

Table 1. Instances Terms

Term	Scope	Description
<service>	Mandatory	NGCP/systemd service from /etc/ngcp-service/nsservices.yml.
<label>	Mandatory	defines an instance, hardcoded at the moment
<location>	Mandatory	Target node/type/cluster_set to start a service.

The migration to the instances concept is in a stage of active improvement, however starting with mr10.5.1, it is already possible to active it and get its benefits. At the moment only the following services support creation of instances:

Table 2. Instances Supported

Service name	Label name	Status
kamailio-lb	lb	full support
kamailio-proxy	proxy	full support
sems-b2b	b2b	full support
asterisk	voicemail	with small limitations: reminder service is not working, fax service is not working

There is no real limit to the number of instances which can be created for each service type (lb, proxy, sems-b2b etc.). Generally, a limitation is an amount of resources on the OS used for deployment.

In order to configure and enable instances on the Sipwise C5, please read this chapter: Instances configuration.

2.10.1. Active/Active and Active/Stand-by

Main difference between approaches

The main difference lays in the scope of the high-availability and the redundancy level, which can be provided by both of the clustering approaches.

 Copyright © Sipwise GmbH - All rights reserved

It's important to mention that Sipwise C5 by default (and in most cases) uses two sided clustering, which means: there are two locations where services can be deployed. That's why the Active/Stand-by and Active/Active clustering approaches perfectly fit onto that.

Of course, it is still possible to set up a cluster with instances on the Carrier grade system as well and this assumes that there can be more than two locations for each type of the service (LBs, PRXs, DBs etc.).

Active/Stand-by

The Active/Stand-by clustering approach has been settled in the VoIP/SIP area for quite a long time, as well as in other telephony related areas not particularly related to the SIP and H.323 protocols. This approach used to be a good fit for that period of time, when technologies, protocols and libraries involved in the implementation of different cluster methods weren't that much developed, extended and well tested in production. And it still can be a good fit for certain setups of customers/companies.

The main idea of the Active/Stand-by approach is to provide to a telephony setup a sort of redundancy, when one active side goes down for any reason (IP network issues, hardware issues in a datacenter, processes go down on the active side for any reason etc.), and bring up all the same services, as well as a shared IP address on the Stand-by side, in order to provide telephony based services back as soon as possible after the failure on the previously active side (master).

Such an approach of course has a list of negative aspects, which make it less efficient in comparison to the Active/Active setups:

- **Resources consumption** - stand-by side does consume resources, even though does not process any SIP calls
- **Split-brain** - under certain circumstances there is the possibility to fall into the split-brain scenario
- **Failover detection time** - the clustering tools/logic must be smart enough to detect as fast as possible the failure on any of the cluster layers (IP connectivity, services failure etc.)
- **Failover migration time** - the failover process still takes certain time to bring all services up on the stand-by side, from hundreds of milliseconds up to 5-10 seconds (and sometimes even more)
- **Services flapping** - such setup can in certain cases create a 'flapping' case, when the Active side migrates back and forth, because the usual Master for any reason appears/disappears frequently

However, it has of course a list of advantages:

- **No load-balancing superstructure** - there is no need to solve an obstacle with the load-balancing as a superstructure, because there is only one point obtaining IP traffic
- **Simpler debugging** - this solution is much simpler to debug
- **Less complicated database clustering** - no need to solve SQL/No-SQL replication obstacles, such as one when two sides perform write operations simultaneously (Master/Master replication)

Active/Active

The Active/Active clustering approach is already something not really new, and has been present for a while on the IP telephony market.

There are a number of important advantages given by the Active/Active approach:

Copyright © Sipwise GmbH - All rights reserved

- **Processing efficiency** - doubled processing capabilites, when both sides of the cluster are engaged into calls processing
- **Better failover** - in case of the failover, only half of subscribers/calls currently being processed need to be migrated.
- **Advanced maintenance** - there is the possibility to switch the load-balancing to one side of the Active/Active cluster and to do a maintenance on the other, without an interruption of services

A list of disadvantages:

- **Load-balancing** - there is a need to decide how the IP traffic (SIP calls) will be balanced between two Active sides, which can be based on different approaches (DNS SRV/NAPTR, transport based balancing, a separate load-balancing component)
- **Debug** - the debugging of this setup is a bit more complicated. The Active/Active approach with instances has more things to configure, control and to maintain.
- **Database clustering** - the Active/Active approach has a Master/Master database replication for both SQL and NoSQL, which makes the support of the database backend a bit more complicated as well.

However, even though the Active/Active approach looks much better in comparison to the Active/Stand-by one, it still shares a list of inherent difficulties with it:

- **Split-brain** - in case both of the sites decide that the remote site is down, they both will undertake failover on themselves and hence there will be Active/Active x 2 setup, which will heavily affect systems in production.
- **Failover detection time** - in the same way as with Active/Stand-by, clustering tools must be very accurate and quick to migrate services quickly in case of failure on one of the sites
- **Failover migration time** - in the same way as with Active/Stand-by, failover takes time to migrate that half of subscribers/calls being affected at the moment

NOTE it's important to understand that in the scope of Sipwise C5 the LB component is an inseparable internal component facing IP traffic, and as such Sipwise C5 cannot function without it, because it is not a general load-balancing solution which works separately in front of the Active/Active service to balance traffic between two active sites.

Active/Active approach implementation in Sipwise C5

There is a list of requirements which are to be fulfilled to deploy the Active/Active approach:

- there must be a stable IP interconnection between sites
- there must be a sufficient number of public/private IP addresses reserved, see information below
- additionally, it is **recommended** to have a load-balancing solution present in front of two active sites, which will dispatch IP traffic

A list of the IP addresses needed in order to deploy Active/Active based on a PRO system:

- x2 public IP addresses for cluster management purposes (optional)
- x2 public IP addresses for LB floating IPs, external traffic

 Copyright © Sipwise GmbH - All rights reserved

- x2 private IP addresses for LB floating IPs, internal system traffic
- x2 private IP addresses for Proxy floating IPs, internal system traffic
- x2 private IP addresses for Sems-b2b floating IPs, internal system traffic
- x2 private IP addresses to bind non-instantiated default services, such as database, ngcp-panel and other (optional)

NOTE 127.0.0.0/8 cannot be used to bind instances on.

NOTE The IP configuration will differ on Carrier grade setups, for details please get in contact with the Sipwise Operations team.

As already mentioned, each of the services will have to take a default location it prefers. Under normal network conditions, when there are no issues, services will be settled on their locations and interact with each other in the manner defined by the instances connections.

It is however recommended that services which must constantly work with each other are colocated. For example in the PRO setup: instances LB-1, Proxy-1 and SEMS-1 should be configured to run on location 1 by default, and instances LB-2, Proxy-2 and SEMS-2 to run on location 2.

IMPORTANT Each type of service requires a specific kind of connection to the SQL/NoSQL backend: Instances Connections to Databases

The main clustering tools used in Sipwise C5 are:

- **Corosync** - a transport mechanism for the cluster, which builds up an interconnection between all the cluster nodes, and ensures to carry the cluster data securely (encrypted)
- **Pacemaker** - the brain, logic of the cluster. It provides all the algorithms and controls the instances via the systemd supervisor.
- **Custom developed part** - there is a list of things which were internally developed, to improve the clustering capabilities and implement the instances concept

There is no need to debug/configure Pacemaker/Corosync separately, because it is maintained by the network.yml 'instances' section. So everything what is to be implemented in the Active/Active cluster must be configured in the network.yml 'instances' section, and any direct interaction with pacemaker, for example via crms or its configurations, can damage or negatively affect the cluster.

IMPORTANT If there are doubts about how to properly deploy the Active/Active approach over the currently existing setup, in other words to upgrade the system from Active/Stand-by to Active/Active approach, please contact the Sipwise Operations team to get assistance.

Switch off normal services

By default instances are running concurrently with the standard services/daemons.

Such an approach has an advantage, it makes it possible for a system administrator to perform a smooth migration to the instances architecture. Also it's worth mentioning that a migration to instances for all default services is not mandatory. It is possible, for example, to migrate only the kamailio-lb service, while all the rest of services can be kept running in a standard manner.

Copyright © Sipwise GmbH - All rights reserved

When the most important/required steps of the configuration are done and all those migrated standard services are not doing any significant work, they can be safely disabled, for that see Disable default services.

2.10.2. Limitations

The instances concept is still partially experimental and it will be improved version by version.

As mentioned before, some features might not be supported 100% for services migrated to instances (i.e. 'reminder' or 'faxserver' for asterisk).

An additional upgrade might be required in the future for those systems that have been migrated to the instances architecture, since it is still under a stage of improvement.

2.10.3. Example

Please check Instance Appendix for a full example of the 'network.yml' file of a PRO system with instances defined.

Copyright © Sipwise GmbH - All rights reserved

Chapter 3. Deployment

This chapter provides a step by step instruction on how to set up a Sipwise C5 CARRIER/Pro nodes from scratch.

3.1. Installation Prerequisites

3.1.1. KVM

Use an USB Keyboard and any Monitor with VGA connector (male three-row 15-pin DE-15). A Mouse is not required.

3.1.2. Install Medium

The install CD provides the ability to easily install Sipwise C5 CE/PRO/Carrier, including automatic partitioning and installation of the underlying Debian system.

Burn the ISO to a CD with an application of your choice, or preferably put the ISO on a USB stick (all data will be wiped from there):

```
% dd if=sip_provider_mr10.5.7.iso of=/dev/sdX
```

IMPORTANT Do **not** specify a partition (like /dev/sdb1), but only the disk itself (like /dev/sdb) since the ISO already provides a partition table.

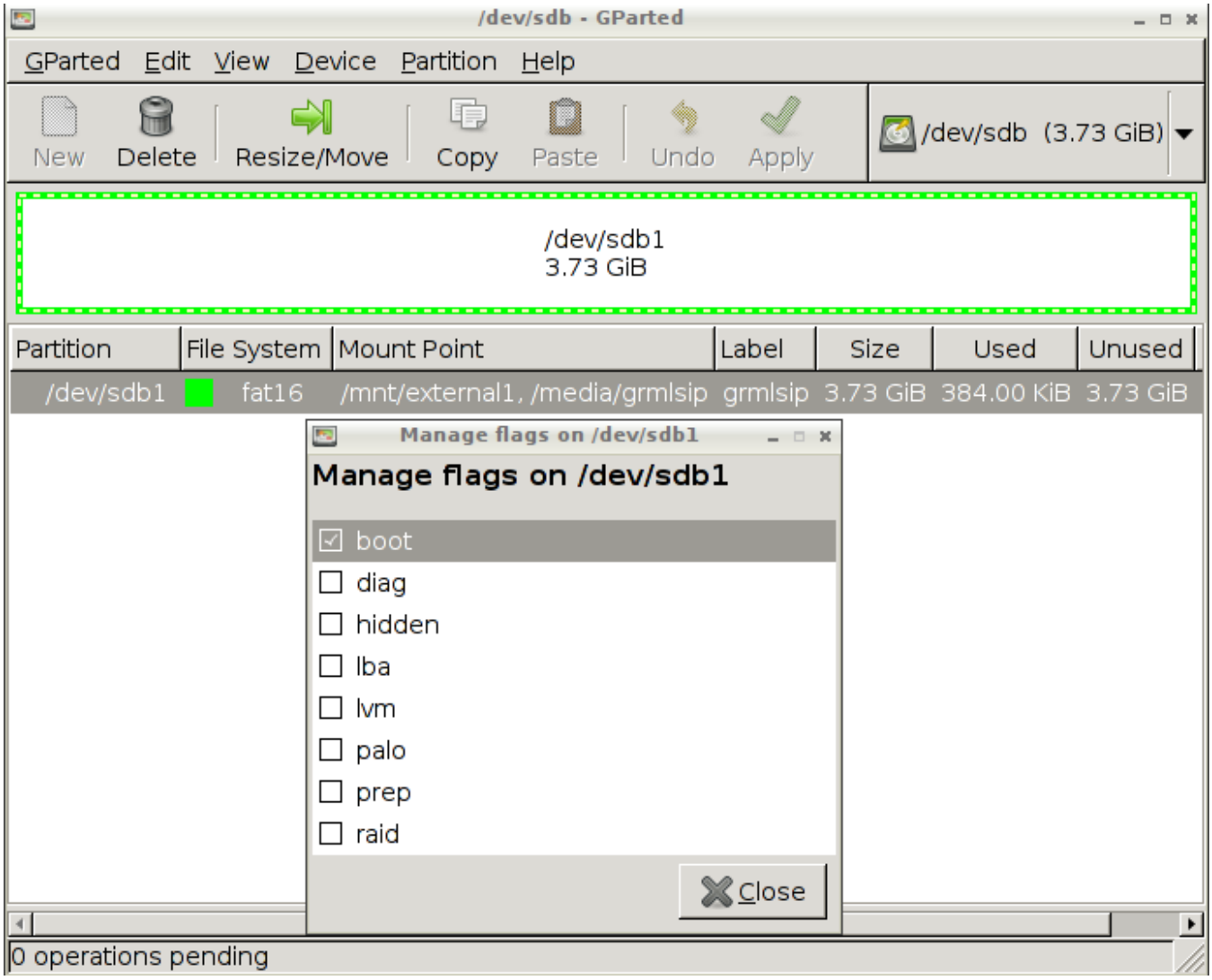
NOTE When dd-ing the ISO to a device the system is **NOT** (U)EFI capable. To be able to boot the device using (U)EFI you need to set it up using grml2usb, see the following section for further details.

Instructions for setting up (U)EFI capable USB device using grml2usb

Install grml2usb (see grml.org/grml2usb) or boot a Grml/Grml-Sipwise ISO to be able to use its grml2usb.

Create a FAT16 partition on the USB pen and enable the boot flag for the partition. The GParted tool is known to work fine:

Copyright © Sipwise GmbH - All rights reserved



Then invoke as root or through sudo (adjust /dev/sdX accordingly for your USB device):

```
# grml2usb sip_provider_mr10.5.7.iso /dev/sdX1
```

That's it.

This is how (U)EFI boot looks like:

Copyright © Sipwise GmbH - All rights reserved



whereas that's how BIOS boot looks like:



Copyright © Sipwise GmbH - All rights reserved

Instructions for setting up (U)EFI capable USB device on Mac OS

Invoke the Disk Utility (in german locales: "Festplatten-Dienstprogramm"), select the USB stick. Choose the partitioning tab, choose "1 Partition" in Volume Schema, name it "SIPWISE" and choose MS-DOS (FAT) as file system. Then "Apply" the settings.

Double-click on the Grml-Sipwise ISO and switch to a terminal, invoke "diskutil" to identify the device name of the ISO (being /dev/disk2s1 in the command line below).

Finally mount the first partition of the ISO and copy the files to the USB device, like:

```
% mkdir /Volumes/GRML
% sudo mount_cd9660 /dev/disk2s1 /Volumes/GRML
% cp -a /Volumes/GRML/* /Volumes/SIPWISE/
% diskutil unmount /Volumes/SIPWISE/
% sudo umount /Volumes/GRML
% sudo umount /Volumes/GRML
```

The resulting USB Stick should be bootable in (U)EFI mode now, meaning it should also boot on recent Mac OS systems (press the Option/Alt key there during boot and choose "EFI Boot" then).

3.1.3. Network

The setup routine needs both access to Sipwise mirror of public Debian repositories (https://debian.sipwise.com), to the Sipwise repositories (https://deb.sipwise.com) and Sipwise license server (https://license.sipwise.com).

3.2. Installation CARRIER

This section describes how to install vanilla carrier installation.

3.2.1. CARRIER Hardware

Sipwise C5 CARRIER starts with a minimum deployment of 50.000 subscribers, requiring one chassis with two web servers, two db servers, two load balancers and two proxies. A fully deployed Sipwise C5 CARRIER for 200.000 subscribers fills the chassis up with 14 servers, containing two web servers, two db servers, two load balancers and 8 proxies.

3.2.2. Power supply

Connect at least 2 power cords to chassis power supplies.

3.2.3. Initial chassis configuration

Connect Patch cords to Active CMM node on the backside of each chassis and connect them to the switch.

3.2.4. Chassis IP-management setup

By default chassis will try to obtain address using DHCP protocol, if not:

Copyright © Sipwise GmbH - All rights reserved

- connect using laptop to CMM try to use IP from label on CMM
- setup IP address

Change CMM password to at least 12-chars string generated using Linux *pwgen* as an example.

Change Password for switch modules (IO).

TIP Remember to download for CRM FRU numbers: Status Table View Export to CSV Also it will be useful to copy SN numbers to that CSV file.

Login to CMM and Copy Mac Address of first adapter of Each node through:

- Chassis Management Reports Mac Address and copy Mac1 Column

On each compute node check and configure:

- Raid setup: for deployments we are using RAID1
- Select Legacy boot mode (Boot Legacy mode)

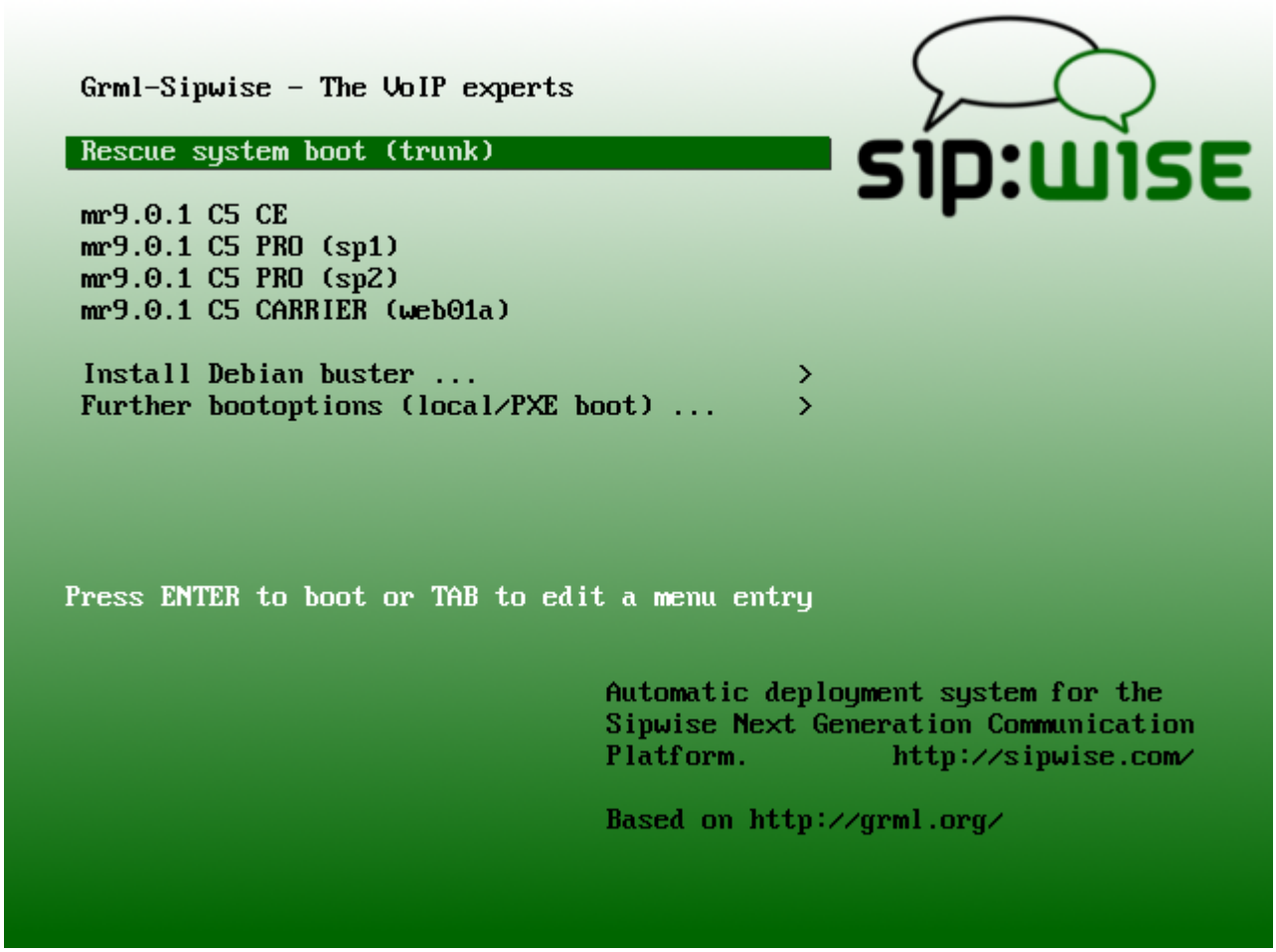
3.2.5. Bootstrapping first node

First node you have to install is: **web01a** All another nodes are using web01a as PXE boot server and source of Debian-packages.

Insert the Install Medium (when using USB or CD), reboot the server, and when prompted in the top right corner, press *F11* to access the Boot menu. Choose *SATA Optical Drive* when using a CD, or *Hard drive C:Removable XXX* when using a USB stick.

You will be presented with the Sipwise installer bootsplash:

Copyright © Sipwise GmbH - All rights reserved



Navigate down to mr10.5.7 CARRIER (web01a) and press <enter> to start the automatic installation process. This will take quite some time (depending on the network connection speed and the installation medium).

Once the Debian base system is installed and the ngcp-installer is being executed, you can login via ssh to *root@<ip>* (password is *sipwise* as specified by the ssh boot option) and watch Sipwise C5 installation process by executing *tail -f /mnt/var/log/ngcp-installer.log -f /tmp/deployment-installer-debug.log*). After the installation has been finished and when prompted on the terminal to do so, press *r* to reboot. Make sure the Install Medium is ejected in order to boot into the newly installed system. Once up, you can login with user *root* and password *sipwise*.

Then you need to run the initial configuration for the first node:

WARNING It is strongly recommended to run ngcp-initial-configuration within terminal multiplexer like screen.

```
screen -S ngcp
ngcp-initial-configuration
```

3.2.6. Network Configuration

For successful bootstrapping all other nodes you have to correctly fill in the network.yml. You can edit network.yml in your favorite text editor. If needed, add missing sections (for prx0Ya and prx0Yb)

 Copyright © Sipwise GmbH - All rights reserved

according to the low-level design architecture doc. Important to write down into 'ha_int' section mac address of first adapter (MAC1), only network adapters with particular Mac-address, which are listed in network.yml are able to boot over PXE and be provisioned by API. Then you should apply your configuration changes, Use the following commands:

```
ngcpcfg apply "Initial network configuration"
ngcpcfg push --shared-only
```

3.2.7. Deployment of the rest nodes in chassis

Power ON web01b from CMM web GUI (by default it will try to boot over PXE) and wait until web01b is deployed and reboots completely.

WARNING It is strongly recommended to run ngcp-initial-configuration within terminal multiplexer like screen.

Run:

```
screen -S ngcp
ngcp-initial-configuration --join
```

When it is finished you can deploy all other A-nodes even in parallel. After it you can deploy rest B-nodes.

All nodes but web01a should be configured with the '--join' option after the reboot:

WARNING It is strongly recommended to run ngcp-initial-configuration within terminal multiplexer like screen.

```
screen -S ngcp
ngcp-initial-configuration --join
```

If your web01b is in another chassis you should follow procedure described below:

Disconnect patch-cord cable from EXT1 port of SM1 which was connected to our office switch. Connect cable between the two Switch Modules (EXT1 on SM1 on first chassis and EXT1 on SM1 on second chassis). Power ON web01a from CMM web GUI and wait until web01a boots. Login to CMM on second chassis and start Node for **web01b** (by default it will try to boot over PXE). Wait until web01b is deployed and reboots completely. Begin deployment for all nodes in A-chassis (turn them on even in parallel). After process of deployment A-nodes completes you can deploy rest nodes in B-chassis.

3.2.8. Checking install:

SSH on web01a and run the following command:

Copyright © Sipwise GmbH - All rights reserved

```
ngcp-status --all
```

On prx-nodes there are two mysql instances running which handles the following replications: sp1<sp2 (port 3306) and db01localhost (port 3308). Check the replication of tables between DB-node and PRX-node with ngcp-mysql-replication-check:

```
root@prx01a:~# ngcp-mysql-replication-check -a -v
[prx01a] Replication slave is running on localhost:3306 from 'sp2'. No
replication errors.
[prx01a] Replication slave is running on 127.0.0.1:3308 from 'db01a'. No
replication errors.
[prx01a] Replication slave is running on 127.0.0.1:3308 from 'db01b'. No
replication errors.
```

3.3. Installation Pro

The two nodes are installed one after the other by performing the following steps. It can be bare-hardware installation, virualised one (VMware, Proxmox) or Cloud (Google Cloud).

3.3.1. PRO Hardware

This section is valid for bare-hardware installation only.

Hardware Specifications

Sipwise provides Sipwise C5 platform fully pre-installed on two Lenovo ThinkSystem SR250 servers. Their most important characteristics are:

- 1x 6C 12T E-2246G CPU @ 3.60GHz
- 64 GB RAM (DDR4 ECC)
- 2x 480Gb Lenovo branded SATA SSD
- 1x 4Port intel i350 add-on network card

Hardware Prerequisites

In order to put Sipwise C5 into operation, you need to rack-mount it into 19" racks.

You will find the following equipment in the box:

- 2 servers
- 2 pairs of rails to rack-mount the servers
- 2 cable management arms

You will additionally need the following parts as they are not part of the distribution:

- 4 power cables

Copyright © Sipwise GmbH - All rights reserved

NOTE The exact type required depends on the location of installation, e.g. there are various forms of power outlets in different countries.

- At least 2 CAT5 cables to connect the servers to the access switches for external communication
- 1 CAT5 cable to directly connect the two servers for internal communication

Rack-Mount Installation

Install the two servers into the rack (either into a single one or into two geographically distributed ones).

The rails shipped with the servers fit into standard 4-Post 19" racks. If they do not fit, please consult your rack vendor to get proper rails.

The following figure shows the mounted rails:

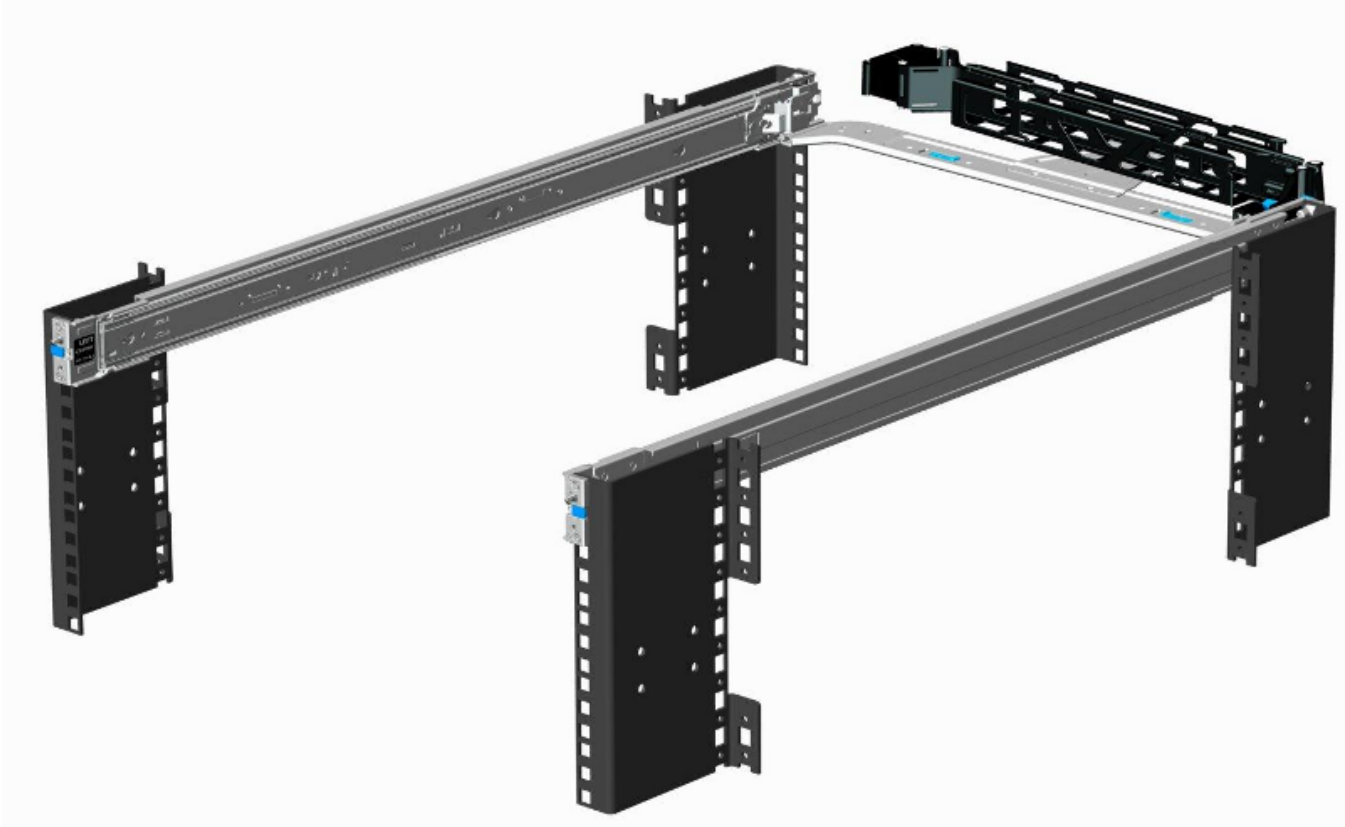


Figure 9. Rack-mounted Rails

Power Supply Cabling

Each server has two redundant Power Supply Units (PSU). Connect one PSU to your normal power circuit and the other one to an Uninterruptible Power Supply Unit (UPS) to gain the maximum protection against power failures.

The cabling should look like in the following picture to prevent accidental power cuts:

Copyright © Sipwise GmbH - All rights reserved



Figure 10. Proper PSU Cabling

Use a 230V/16A power supply and connect a power cord with a C13 jack to each of the two servers, using any of the two power supply units (PSU) in the back of the servers.

Network Cabling

Internal Communication

The *high availability (HA)* feature of Sipwise C5 requires that a direct Ethernet connection between the servers is established. One of the network interfaces must be dedicated to this functionality.

External Communication

Remaining network interfaces may be used to make the servers publicly available for communication services (SIP, messaging, etc.) and also for their management and maintenance.

Internal Communication

Patch a cross-link with a straight CAT5 cable between the two servers by connecting the cable to the network interface assigned to the HA component by Sipwise. The direct cross cable is applied for maximum availability because this connection is used by the servers to communicate with each other internally.

IMPORTANT We strongly suggest against using a switch in between the servers for this internal interface. Using a switch is acceptable only if there is no another way to connect the two ports (e.g. if you configure a geographically distributed installation).

NOTE In case you are using a switch for cross-link make sure to enable *portfast* mode on Cisco switches. The thing is that STP puts the port into learning mode for 90 seconds, after it comes up for the first time. During this learning phase, the link is technically up, but no traffic passes through, so the GCS service will detect the other node as dead during boot. The *portfast* mode tells the switch to skip the learning phase and go to forwarding state right away: `spanning-tree portfast [trunk]`.

Copyright © Sipwise GmbH - All rights reserved

External Communication

For both servers, depending on the network configuration, connect one or more straight CAT5 cables to the ports on the servers network cards and plug them into the corresponding switch ports. Information about proper ports of the servers to be used for this purpose are provided by Sipwise.

3.3.2. Initial BIOS Configuration

Power on both servers, and when prompted on the top right corner, press *F2* to access the BIOS menu.

Automatic Power-On Setting:

Navigate to *System SecurityAC Power Recovery* and change the setting to *On* by pressing the <*right*> key. This will cause the server to immediately boot, as soon as it's connected to the power supply, which helps to increase availability (e.g. after an accidental shutdown, it can be remotely powered on again by power-cycling both PSU simultaneously via an IP-PDU).

Go back with <esc> until prompted for *Save changes and exit*, and choose that option.

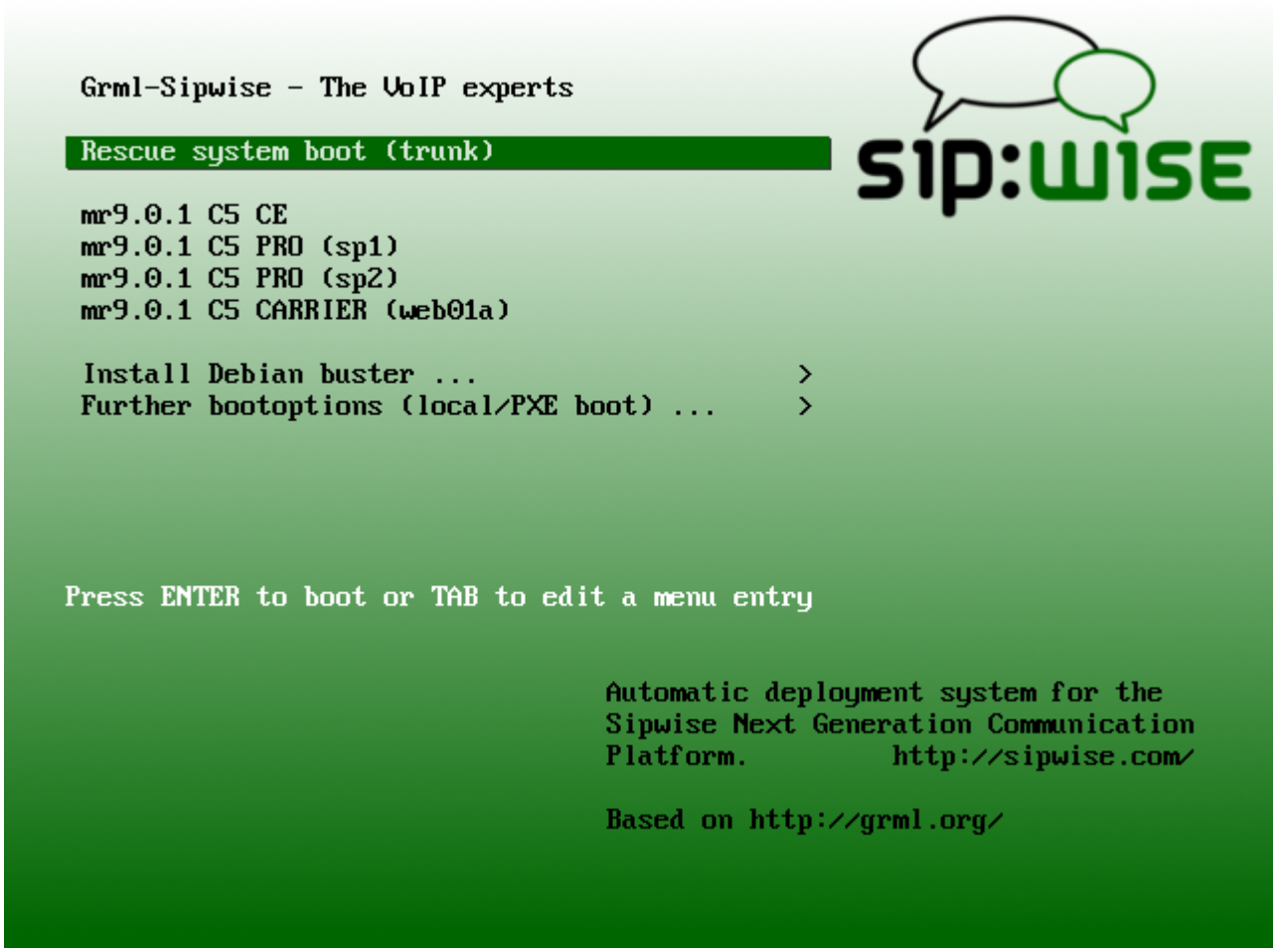
3.3.3. Bootstrapping first node

First node you have to install is: **sp1 sp2** node is using **sp1** as PXE boot server and source of Debian-packages.

Insert the Install Medium (when using USB or CD), reboot the server, and when prompted in the top right corner, press *F11* to access the Boot menu. Choose *SATA Optical Drive* when using a CD, or *Hard drive C:Removable XXX* when using a USB stick.

You will be presented with the Sipwise installer bootsplash:

Copyright © Sipwise GmbH - All rights reserved



Navigate down to mr10.5.7 PRO (sp1) and press <enter> to start the automatic installation process. This will take quite some time (depending on the network connection speed and the installation medium).

Once the Debian base system is installed and the ngcp-installer is being executed, you can login via ssh to *root@<ip>* (password is *sipwise* as specified by the ssh boot option) and watch Sipwise C5 installation process by executing *tail -f /mnt/var/log/ngcp-installer.log -f /tmp/deployment-installer-debug.log*). After the installation has finished and when prompted on the terminal to do so, press *r* to reboot. Make sure the Install Medium is ejected in order to boot into the newly installed system. Once up, you can login with user *root* and password *sipwise*.

Then you need to run the initial configuration for the first node:

WARNING It is strongly recommended to run ngcp-initial-configuration within terminal multiplexer like screen.

```
screen -S ngcp
ngcp-initial-configuration
```

3.3.4. Network Configuration

The Sipwise C5 PRO pair uses *eth0* for the public interface, and *eth1* for a small, dedicated internal network on the cross-link for the GCS, replication and synchronization. Both of them are configured automatically at install time.

Copyright © Sipwise GmbH - All rights reserved

For the service to be used for SIP, RTP, HTTP etc. you need to configure a floating IP in the same network as you have configured on *eth0*. Put this IP address into the *shared_ip* array of the interface which contains the *ext* types in */etc/ngcp-config/network.yml*.

```
ngcp-network --set-interface=eth0 --shared-ip=1.2.3.5
```

Once done, execute *ngcpcfg apply "added shared ip"*, which will restart (beside others) the GCM/CRM processes, which in turn will configure a virtual interface *eth0:0* with your floating IP.

After the configuration you can proceed to the second node.

IMPORTANT You need to keep sp1 running and connected to the network in order to set up sp2 correctly, because the install procedure automatically synchronizes various configurations between the two nodes during the setup phase.

Setting up PRO sp2

Power on the server, insert the Install Medium (when using USB or CD), and when prompted in the top right corner, press *F11* to access the Boot menu. Choose *SATA Optical Drive* when using a CD, or *Hard drive C:Removable XXX* when using a USB stick.

When finished, press <enter> to start the automatic installation process.

Again you can watch Sipwise C5 installer once the Debian base system is installed and the ngcp-installer is being executed by ssh into *root@<ip>* (password is *sipwise* as specified by the ssh boot option) and executing *tail -f /mnt/var/log/ngcp-installer.log*.

After the installation has finished and when prompted on the terminal to do so, press *r* to reboot. Make sure the Install Medium is ejected in order to boot into the newly installed system. Once up, you can login with user *root* and password *sipwise*.

Then you need to run the initial configuration for the second node:

WARNING It is strongly recommended to run ngcp-initial-configuration within terminal multiplexer like screen.

```
screen -S ngcp
ngcp-initial-configuration --join
```

3.3.5. Verify running Cluster configuration

After both sp1 and sp2 have been set up and are rebooted into the freshly installed system, one node should show *cluster node active* when logging into the machine, the other node should either have the default message of the day, or *cluster node inactive*.

Check the running processes on both nodes by executing *ngcp-service summary* on both of them.

Output of the active node:

Copyright © Sipwise GmbH - All rights reserved

```
# ngcp-service summary
Ok Service                          Managed    Started   Status
-- -------------------------------- ---------- --------- ----------
   approx                           managed    on-boot   active
   asterisk                         managed    by-ha     active
   coturn                           unmanaged  by-ha     inactive
   corosync                         managed    on-boot   active
   dhcp                             unmanaged  by-ha     inactive
   exim                             managed    on-boot   active
   glusterfsd                       managed    on-boot   active
   grafana                          managed    on-boot   active
   haproxy                          unmanaged  on-boot   inactive
   kamailio-lb                      managed    by-ha     active
   kamailio-proxy                   managed    by-ha     active
   kannel-bearerbox                 unmanaged  by-ha     inactive
   kannel-smsbox                    unmanaged  by-ha     inactive
   monit                            managed    on-boot   active
   mysql                            managed    on-boot   active
   mysql_cluster                    unmanaged  on-boot   inactive
   ngcp-eaddress                    unmanaged  on-boot   inactive
   ngcp-faxserver                   managed    by-ha     active
   ngcp-license-client              managed    on-boot   active
   ngcp-lnpd                        unmanaged  on-boot   inactive
   ngcp-logfs                       managed    on-boot   active
   ngcp-mediator                    managed    by-ha     active
   ngcp-panel                       managed    on-boot   active
   ngcp-pushd                       unmanaged  by-ha     inactive
   ngcp-rate-o-mat                  managed    by-ha     active
   ngcp-snmp-agent                  managed    on-boot   active
   ngcp-voisniff                    managed    by-ha     active
   ngcp-websocket                   unmanaged  by-ha     inactive
   ngcp-witnessd                    managed    on-boot   active
   ngcpcfg-api                      managed    on-boot   active
   nginx                            managed    on-boot   active
   ntpd                             unmanaged  on-boot   inactive
   openvpn                          unmanaged  on-boot   inactive
   openvpn-vip                      unmanaged  by-ha     inactive
   pacemaker                        managed    on-boot   active
   prosody                          managed    by-ha     active
   redis                            managed    on-boot   active
   redis-master                     managed    by-ha     active
   rtpengine                        managed    by-ha     active
   rtpengine-recording              unmanaged  by-ha     inactive
   rtpengine-recording-nfs-mount    unmanaged  on-boot   inactive
   sems                             managed    by-ha     active
   sems-b2b                         unmanaged  by-ha     inactive
   slapd                            unmanaged  by-ha     inactive
   snmpd                            managed    on-boot   active
   snmptrapd                        unmanaged  on-boot   inactive
   ssh                              managed    on-boot   active
   syslog                           managed    on-boot   active
   systemd-timesyncd                managed    on-boot   active
```


Copyright © Sipwise GmbH - All rights reserved

Output of the standby node:

```
# ngcp-service summary
Ok Service                          Managed    Started   Status
-- -------------------------------- ---------- --------- ---------
   approx                           managed    on-boot   active
   asterisk                         managed    by-ha     inactive
   corosync                         managed    on-boot   active
   coturn                           unmanaged  by-ha     inactive
   dhcp                             managed    by-ha     inactive
   exim                             managed    on-boot   active
   glusterfsd                       managed    on-boot   active
   grafana                          managed    on-boot   active
   haproxy                          unmanaged  on-boot   inactive
   kamailio-lb                      managed    by-ha     inactive
   kamailio-proxy                   managed    by-ha     inactive
   kannel-bearerbox                 unmanaged  by-ha     inactive
   kannel-smsbox                    unmanaged  by-ha     inactive
   monit                            managed    on-boot   active
   mysql                            managed    on-boot   active
   mysql_cluster                    unmanaged  on-boot   inactive
   ngcp-eaddress                    unmanaged  on-boot   inactive
   ngcp-faxserver                   managed    by-ha     inactive
   ngcp-license-client              managed    on-boot   active
   ngcp-lnpd                        unmanaged  on-boot   inactive
   ngcp-logfs                       managed    on-boot   active
   ngcp-mediator                    managed    by-ha     inactive
   ngcp-panel                       managed    on-boot   active
   ngcp-pushd                       unmanaged  by-ha     inactive
   ngcp-rate-o-mat                  managed    by-ha     inactive
   ngcp-snmp-agent                  managed    on-boot   active
   ngcp-voisniff                    managed    by-ha     inactive
   ngcp-websocket                   unmanaged  by-ha     inactive
   ngcp-witnessd                    managed    on-boot   active
   ngcpcfg-api                      managed    on-boot   active
   nginx                            managed    on-boot   active
   ntpd                             unmanaged  on-boot   inactive
   openvpn                          unmanaged  on-boot   inactive
   openvpn-vip                      unmanaged  by-ha     inactive
   pacemaker                        managed    on-boot   active
   prosody                          managed    by-ha     inactive
   redis                            managed    on-boot   active
   redis-master                     managed    by-ha     inactive
   rtpengine                        managed    by-ha     inactive
   rtpengine-recording              unmanaged  by-ha     inactive
   rtpengine-recording-nfs-mount    unmanaged  on-boot   inactive
   sems                             managed    by-ha     inactive
   sems-b2b                         unmanaged  by-ha     inactive
   slapd                            unmanaged  by-ha     inactive
   snmpd                            managed    on-boot   active
   snmptrapd                        unmanaged  on-boot   inactive
```

Copyright © Sipwise GmbH - All rights reserved

```
ssh                             managed    on-boot   active
syslog                          managed    on-boot   active
systemd-timesyncd               managed    on-boot   active
```

If your output matches the output above, you're fine.

Also double-check if the replication is up and running by executing *mysql* and do the query *show slave status\G*, which should NOT report any error on both nodes.

3.3.6. Synchronizing configuration changes

Between *sp1* and *sp2* there is a shared glusterfs storage, which holds the configuration files. If you change any config option on the active server and apply it using *ngcpcfg apply "my commit message"*, then execute *ngcpcfg push* to propagate your changes to the second node.

TIP Note that *ngcpcfg apply "my commit message"* is implicitly executed on the other node if you push configurations to it.

3.4. What's happening when booting the Sipwise Deployment ISO

What happens when booting the Sipwise Deployment ISO is roughly:

- Grml ISO boots up
- The Grml system checks for the netscript boot option (specified in the kernel command line which is visible at the boot splash when pressing <*TAB*>)
- The URL provided as argument to the netscript= boot option will be downloaded and executed
- The netscript provides all the steps that need to be executed depending on which boot options have been specified (see the following section for more information)

3.4.1. Deployment stages

If installing Debian or Debian plus ngcp this happens during deployment:

- boot options get evaluated
- if installing a PRO system and usb0 exists IPMI is configured with IP address 169.254.1.102
- checking for some known disks (based on a whitelist) to not cause any data loss by accident, exits if trying to install on an unknown disk type
- starting ssh server for remote access
- partition disk and set up partitions
- make basic software selection using /etc/debootstrap/packages
- run grml-debootstrap to install Debian base system
- adjusting /etc/hosts of target system
- adjusting /etc/udev/rules.d/70-persistent-net.rules of target system if installing a virtual PRO system

 Copyright © Sipwise GmbH - All rights reserved

If installing ngcp the following takes place:

- downloading according ngcp-installer version
- executing ngcp-installer with settings as specified by boot options (see the following section for further information)
- build ngcp-rtpengine kernel module (as this can't be done automatically inside a chroot where kernel version of deployment system doesn't necessarily match the kernel version of the installed system)
- stop any running processes of the ngcp system that have been started during installation
- copy generated log files to installed system
- adjust /etc/hosts, /etc/hostname and /etc/network/interfaces of the installed system
- kill any remaining ngcp system processes
- ask for rebooting/halting the system

3.4.2. Important boot options for the Sipwise Deployment ISO

- arch=i386 - install a 32bit Debian system instead of defaulting to 64bit system (use if so only for non-ngcp installations!)
- debianrelease=… - use specified argument as Debian release (instead of defaulting to *bullseye*)
- dns=… - use specified argument (IP address) as name server
- ip=$IP::$GATEWAY:$NETMASK:$HOSTNAME:$DEVICE:off - use static IP address configuration instead of defaulting to DHCP
- ngcpce - install CE flavour of ngcp
- ngcphostname=… - use specified argument as hostname for the ngcp system
- ngcpinstvers=… - use specified argument as ngcp-installer version (e.g. *0.7.3*) instead of defaulting to latest version
- ngcpsp1 - install PRO flavour of ngcp, system 1 (sp1)
- ngcpsp2 - install PRO flavour of ngcp, system 2 (sp2)
- ngcpppa=… - Use Sipwise PPA repository during installation
- nodhcp - disable DHCP (only needed if no network configuration should take place)
- nongcp - do not install any ngcp flavour but only a base Debian system
- puppetenv=… - install puppet software in target system and enable specified argument as its environment (note that the target system's hostname must be known to puppet's manifest)
- ssh=… - start SSH server with specified password for users *root* and *grml*

3.4.3. Logfiles

The settings assigned to the ngcp installer are available at */tmp/installer-settings.txt*.

The log file of the Debian installation process is available at */tmp/grml-debootstrap.log*.

The log file of the deployment process is available at */tmp/deployment-installer-debug.log*.

Copyright © Sipwise GmbH - All rights reserved

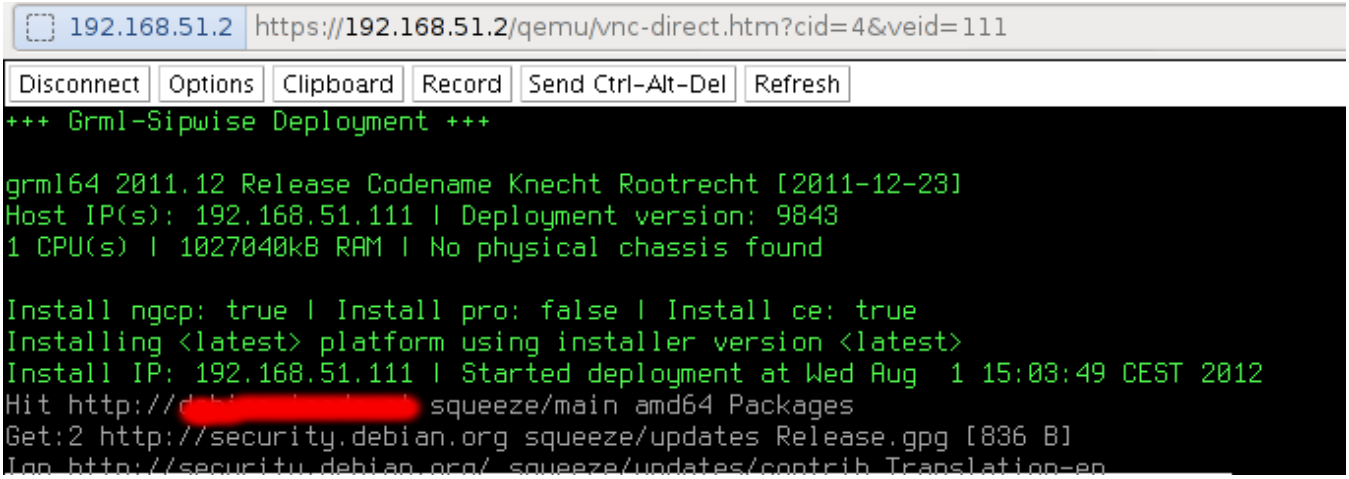
The log files of the ngcp installer are available at */mnt/var/log/ngcp-installer.log* as long as the installation is still running or if the installation fails. Once the installation process has been completed successfully the file system of the target system is not mounted any longer so you can't access */mnt/var/log/ngcp-installer…* any longer by default. But the log files are also available inside */var/log/* of the installed system so you can access them even after rebooting into the installed system. You can access the log files from the deployment ISO by executing:

```
root@spce ~ # Start lvm2
root@spce ~ # mount /dev/mapper/ngcp-root /mnt
root@spce ~ # ls -1 /mnt/var/log/{grml,ngcp}*.log
/mnt/var/log/grml-debootstrap.log
/mnt/var/log/deployment-installer-debug.log
/mnt/var/log/ngcp-installer.log
```

3.4.4. Debugging the deployment process

By default the deployment script enables an SSH server with password *sipwise* for users *root* and *grml* so you can login already while deployment is still running (or if it fails).

At the top of the system screen is a logo splash where you should find the IP address of the system so you can ssh to it:



If you think the problem might be inside the deployment script (which is available as */tmp/netscript.grml* on the system) itself enable the "debugmode" boot option to run it under "set -x" and enable timestamp tracing. The deployment process log is available at */tmp/deployment-installer-debug.log*.

3.4.5. Custom Installation

The installation and configuration steps are separated. First, the hard disk drive or SSD is partitioned, Debian is bootstrapped and NGCP packages are installed. Then the server is rebooted. Afterwards, it is necessary to run the ngcp-initial-configuration tool.

After the reboot you can modify the /etc/ngcp-installer/config_deploy.inc file to tune the configuration process.

 Copyright © Sipwise GmbH - All rights reserved

Chapter 4. Concept

4.1. Contacts

A contact contains information such as the name, the postal and email addresses, and others. A contact's main purpose is to identify entities (resellers, customers, peers and subscribers) it is associated with.

A person or an organization may represent a few entities and it is handy to create a corresponding organization's contact beforehand and use it repeatedly when creating new entities. In this case we suggest populating the **External #** field to distinguish between customers associated with the same contact.

Reseller	Contact	Customer	External #
Default	Rylic Longstaff	DTS	0007
		Morning Times	0008
TelephOne	Clare Fenn	Lantern Co	—
	Ike Leonard	City Bank	—

Note that the only required contact field is **email**. For contacts associated with customers, it will be used for sending invoices and notifications such as password reset, new subscriber creation and others. A contact for a subscriber is created automatically but only if you specify an email address for this subscriber. It is mainly used to send notification messages, e.g. in case of a password reset.

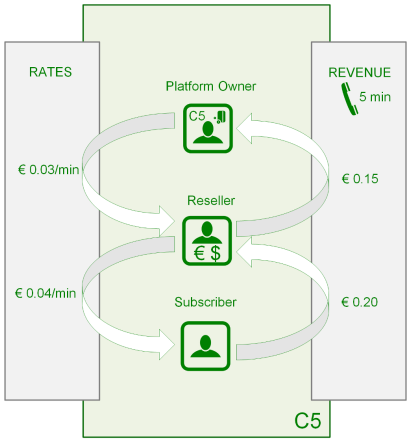
4.2. Resellers

The reseller model allows you to expand your presence in the market by including virtual operators in the sales chain. A virtual operator can be a company without its own VoIP platform and even without a technical background, but with sales presence in a market. You define such a company as a reseller in the platform: grant limited access to the administrative web interface (the reseller administrator will only see his own customers, domains and billing profiles) and define wholesale rates for this reseller. Then, the reseller is free to operate under its own brand, make up its retail rates, establish the customer base and resell your services to its customers. The reseller's profit is a margin between the wholesale and retail rates.

Let us consider an example:

- You operate in Munich and provide residential and business services.
- A company Cheap Call that has a strong presence in Frankfurt offers to resell your services under its own brand in this city.
- You define wholesale rates for Cheap Call, such as calls to Argentina at €0,03.
- Cheap Call defines its retail price and offers calls to Argentina at €0,04.
- When one of Cheap Call's subscribers makes a 5-minute call to Argentina, this subscriber will be charged €0,20.
- You will get €0,15 revenue and Cheap Call's profit will be €0,20 – €0,15 = €0,05.

Copyright © Sipwise GmbH - All rights reserved



A reseller usually uses dedicated IP addresses or SIP domain names to provide services. Also, a reseller can rebrand the self-care web interface for its customers and select languages per SIP domain that allows the reseller to operate even in multiple countries.

4.3. SIP Domain

A SIP domain represents an external Internet address where your subscribers register their SIP phones to make calls or send messages. The SIP domain also contains particular default configuration for all the subscribers registered with this SIP domain. A SIP domain can be a regular FQDN (e.g. sip.yourdomain.com) or a NAPTR/SRV record. Using IP addresses for SIP domains in production is **strongly discouraged**.

4.3.1. Additional SIP Domains

You can create as many SIP domains as required to satisfy your networking or marketing requirements, e.g.:

- A dedicated SIP domain is *suggested* per CloudPBX customer.
- A separate SIP domain may be dedicated to every whitelabel reseller.
- Multiple SIP domains may be used to provide services in different countries or regions.
- Multiple SIP domains may be used to brand your own services.

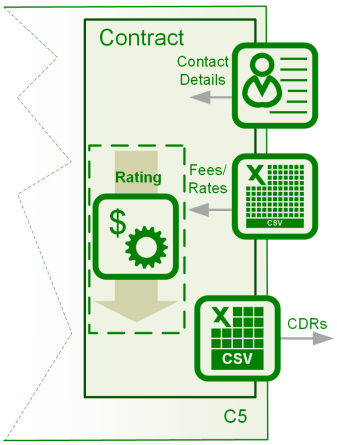
Domain	Purpose
sip.yourdomain.com	Your own domain for retail customers
sip.enterprise.com	Your big customer with Cloud PBX
sip.reseller.com	Your white-label reseller
sip.yourdomain.de	Your domain for providing a new service in another country

4.4. Contracts

A contract is a combination of a *contact* and a *billing profile*, hence it represents a business contract for your resellers and peering partners.

Copyright © Sipwise GmbH - All rights reserved

Contracts can be created in advance on the *Reseller and Peering Contracts* page, or immediately during creation of a peer or a reseller.



Note that the *customer* entity (described below) is a special type of the contract. A customer entity has an email and an invoice templates in addition to a contact and a billing profile.

4.5. Customers

A customer is a physical or legal entity whom you provide the VoIP service with and send invoices to. Here are the main features of a customer:

- Contains the contact and legal information. For example, an address or an email address for invoicing.
- Associated with a billing profile (to define fees per destination) and tracks the balance (used mostly for post-paid customers).
- Contains a certain number of subscribers who actually use the service and whose calls appear in the customer's list of CDRs.
- Provides some default parameters for all its subscribers. For example, voice prompts and call restriction.

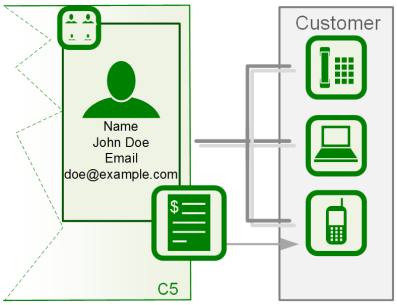
Here are two common examples of the customer model:

4.5.1. Residential and SOHO customers

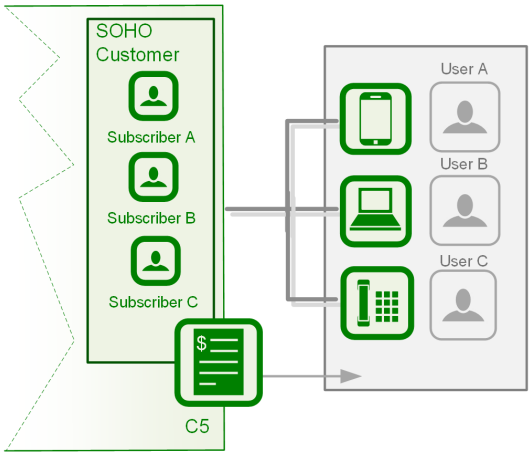
With this service you provide your residential and SOHO customers with one or multiple numbers and offer the service on a post-paid basis.

For a residential customer you usually create one *customer* entity with one *subscriber* under it. A residential customer can register multiple devices with the same number thus having a convenient Viber or Skype-like service: any device can be used to make a call and all of them will ring simultaneously when there is an incoming call. At the end of the billing period, you send an invoice to the customer.

Copyright © Sipwise GmbH - All rights reserved

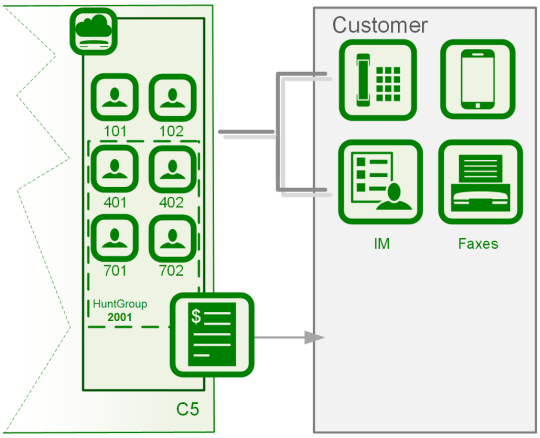


For SOHO customers you usually create multiple subscribers under the same customer and assign every subscriber a dedicated number to allow users make and receive calls. A common invoice will contain calls of all the subscribers.



4.5.2. Business customers with the Cloud PBX service

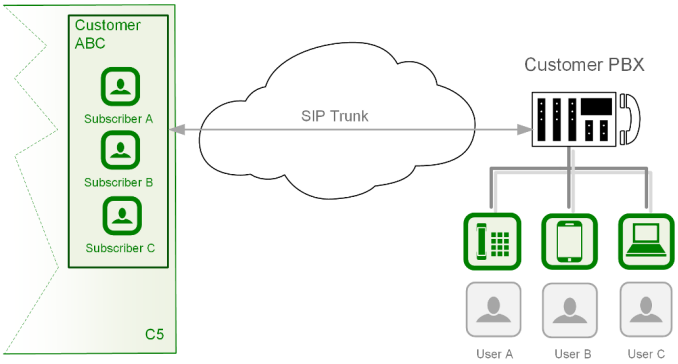
In this case you create a Customer and all the required entities under it to reflect the company's structure: subscribers, extensions, hunt groups, auto-attendant menus, etc.



4.5.3. SIP Trunking

If a customer PBX can register itself with Sipwise C5, you create a regular subscriber for it and configure a standard username/password authentication. Multiple PBX users can then send and receive calls.

Copyright © Sipwise GmbH - All rights reserved



Legacy PBX devices that are not capable of passing the *challenge*-based authentication can be authenticated by the IP address. Optionally, every user of such a PBX can be authenticated separately by the FROM header and the IP address. For more details, refer to the Trusted Sources section.

4.5.4. Mobile subscribers

The pre-paid model works perfectly for mobile application users. In this case you generally create a single subscriber under a customer.

4.5.5. Pre-paid subscribers who use your calling cards

In this case you will most likely create a single subscriber under a customer, although multiple subscribers would work as well. In the latter case, they will share and top-up the common balance. Notice that the *customer* entity itself does not contain any technical configuration for the VoIP service authentication and instead contains other entities called *subscribers*, which do.

4.6. Subscribers

Every subscriber represents a SIP line or a SIP trunk. For example, in the residential services a subscriber entity is dedicated to every user. In the SIP trunking scenario, a subscriber can be used to authenticate all VoIP traffic from the remote PBX device.

In the following table logical subscriber types and their purpose are described.

Service	Subscriber Type	Purpose	Features
Residential	Regular subscriber	A regular VoIP service	Requires a DID number to receive calls from outside of your network
Enterprise (CloudPBX)	Pilot subscriber	A base number for the enterprise customer; Lists all extra numbers (aliases)	Configures the rest of customer subscribers in its self-care web interface
	Extension	Extra numbers (DIDs, “implicit” extensions) for the enterprise customer	Can be dialed in different ways; The number configuration builds on top of the Pilot subscriber
	PBX Group	Forwards incoming calls to multiple extensions	Ringing policy defines in which order the extensions will ring

Copyright © Sipwise GmbH - All rights reserved

Service	Subscriber Type	Purpose	Features
SIP Trunk	Digest authentication	Dynamically registers a remote IP PBX device	Handles multiple users behind the IP PBX device
	IP authentication	IP authentication of legacy IP PBX devices incapable of registering with the platform	Might require Trusted Subscriber and Trusted Source configuration
Prepaid	Regular subscriber with prepaid billing profile	Authorization of services based on customer balance; Disconnection of calls on "zero balance"	Vouchers and Balance Top-Up, Profile Packages

TIP Subscriber **Aliases** can provide Extra DIDs or extension numbers to a subscriber.

4.7. SIP Peerings

A SIP peering is your interconnection with the external VoIP or PSTN network. Usually, a VoIP service provider has at least a few termination partners to offer its subscribers calls to virtually any landline and mobile destination.

SIP peerings also enable incoming calls to your platform. For example, if you rent a pool of DID numbers from a SIP peer and offer them to your residential and business customers.

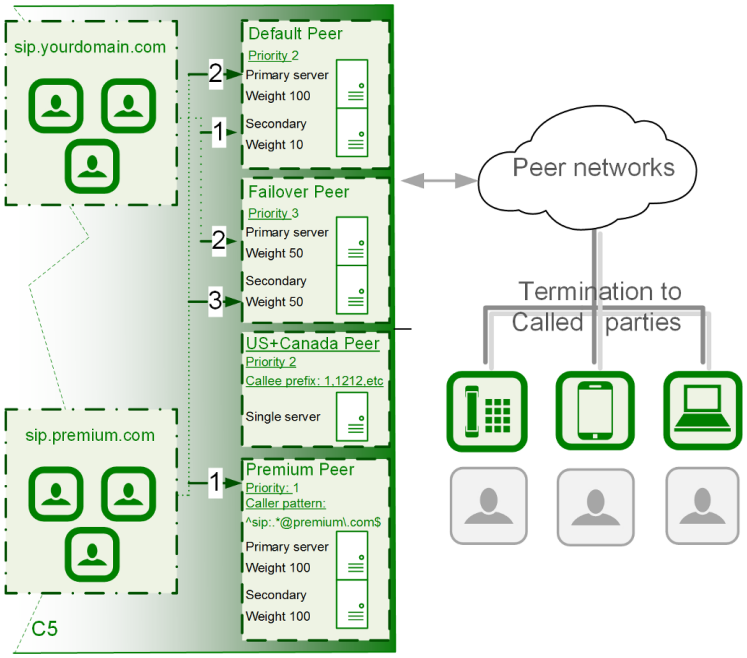
An interconnection with your termination partners and DID number providers can include multiple servers and enable both outbound and inbound calls, hence such a configuration is called a *SIP peering group*. You configure at least one SIP peering group for every partner and the main principle here is that all servers in a group terminate calls to the same set of listed destinations.

Any SIP peering group is associated with a *contract* for reconciliation and billing purposes and includes two main technical configurations:

- Peering Servers Represent connections to/from your SIP peering's network. The parameters include an IP address and/or a hostname of the remote part. For outbound calls, this is the destination address where to send calls to and for inbound calls it is an IP authorization of the remote server.
- Outbound/Inbound Peering Rules Outbound rules define through which SIP peering group a call from a specific subscriber will be sent for termination to a specific destination.

The example below shows four SIP peering groups with different priorities, callee prefixes (actual destinations offered by this SIP peering) and callee / called patterns (fine-tuning which callee request URIs and caller URIs are allowed through this SIP peering group).

 Copyright © Sipwise GmbH - All rights reserved



The figure shows how calls from premium subscribers can in the first place be routed through a dedicated SIP peering group unavailable to regular subscribers.

See the Routing Order Selection section for details about call routing.

Inbound rules allow filtering out incoming INVITE requests arriving from the corresponding SIP peering servers.

Copyright © Sipwise GmbH - All rights reserved

Chapter 5. Kick-off

A basic VoIP service configuration is fast, easy and straight-forward. Provided that your network and required DNS records have been preconfigured, the configuration of a VoIP service can be done purely via the administrative web interface. The configuration mainly includes the following steps:

- Reseller creation (optional)
- SIP domain configuration
- Customer creation
- Subscribers provisioning

Let us assume you are using the *1.2.3.4* IP address with an associated *sip.yourdomain.com* domain to provision VoIP services. This allows you to provide an easy-to-remember domain name instead of the IP address as the proxy server. Also, your subscribers' URIs will look like *1234567@sip.yourdomain.com*.

TIP Using an IP address instead of an associated FQDN (domain name) for a SIP domain is not suggested as it could add extra administrative work if you decide to relocate your servers to another datacenter or change IP addresses.

Go to the *Administrative Web Panel* (*Admin Panel*) running on *https://<ip>:1443/* and follow the steps below. The default web panel user and password are *administrator*, if you have not already changed it.

5.1. Creating a SIP Domain

A SIP domain is a connection point for your subscribers. The SIP domain also contains specific default configuration for all its subscribers.

TIP Thoroughly plan your domain names policy in advance and take into account that: 1) the name of a SIP domain cannot be changed after creating it in the administrative web panel; 2) subscribers cannot be moved from one domain to another and must be recreated.

To create a SIP domain, follow these steps:

1. Firstly, configure an FQDN on your DNS server for it.

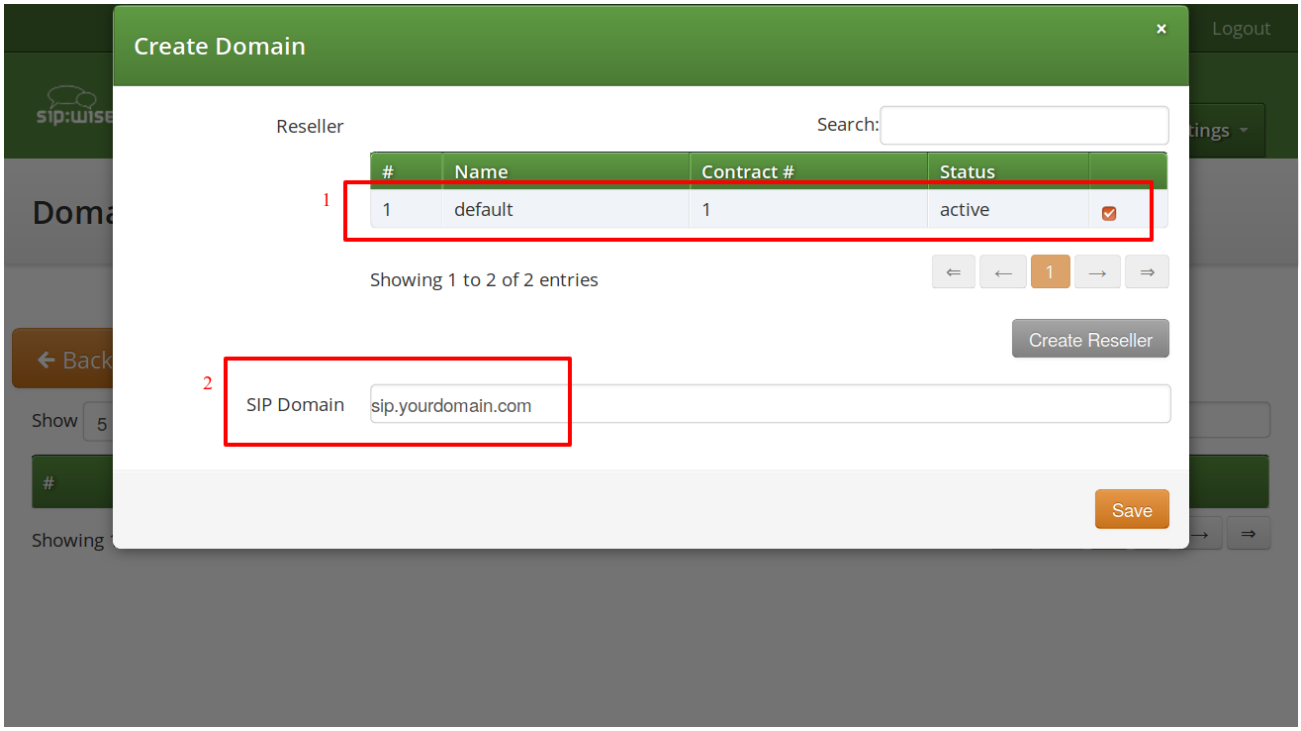
   The domain name must point to the physical IP address you are going to use for providing the VoIP service. A good approach is to create an SRV record:

   ```
   SIP via UDP on port 5060
   SIP via TCP on port 5060
   SIP via TCP/TLS on port 5061
   ```

2. Create a new SIP domain in the administrative web panel.

   Go to the *Domains* page and create a new SIP Domain using the FQDN created above.

 Copyright © Sipwise GmbH - All rights reserved



Select a *Reseller* who will own the subscribers in this SIP domain. Use the *default* virtual reseller if you provide services directly. Enter your SIP domain name and press *Save*.

3. Adjust the new SIP domain's preferences if necessary.

You can create multiple SIP domains reusing the existing IP address or adding a new one. Extra SIP domains are required e.g. if you would like to host a virtual operator on your platform, create separate domains for providing services in different countries or offer a new service.

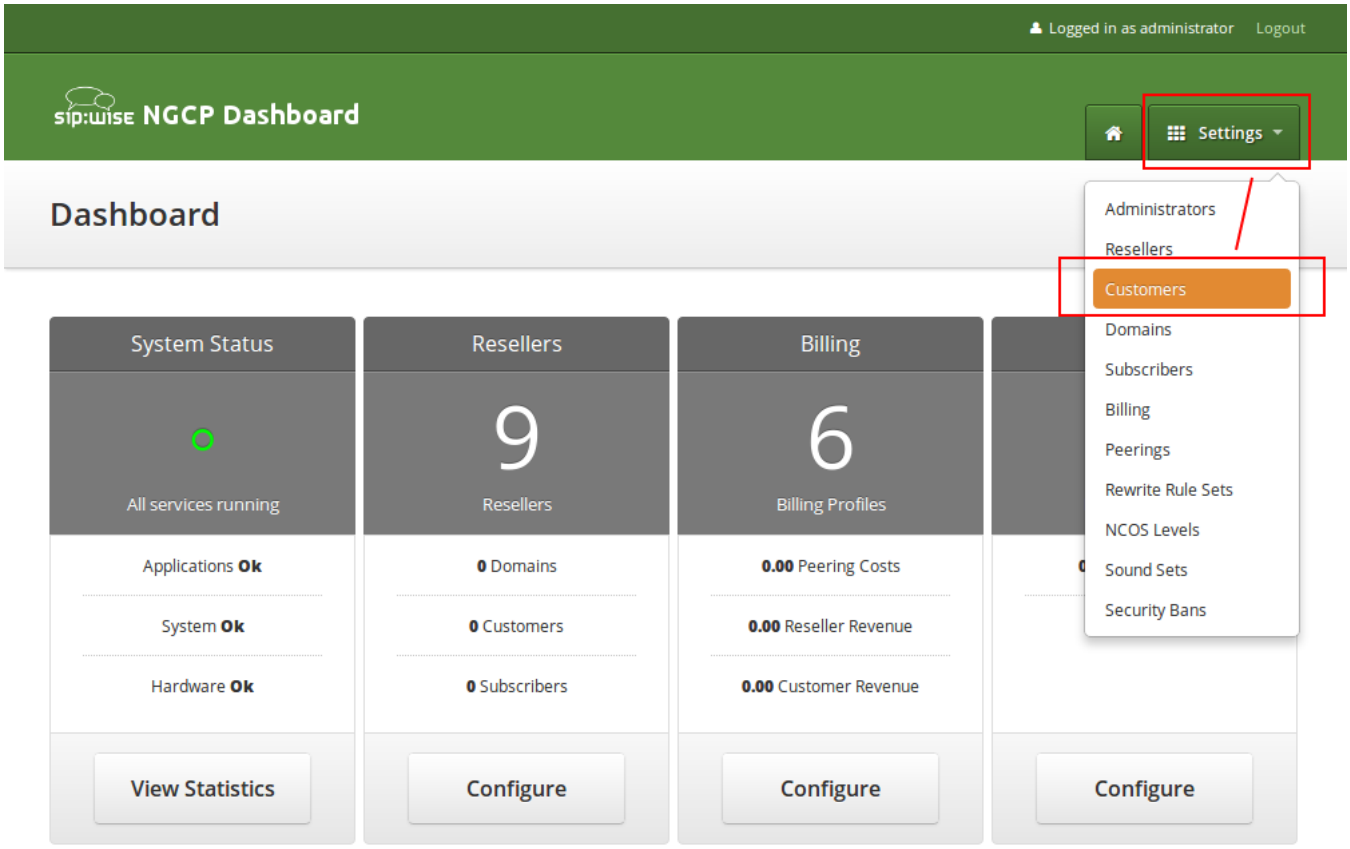
5.2. Creating a Customer

A Customer is a special type of contract acting as legal and billing information container for SIP subscribers. A customer can have one or more SIP subscriber entities that represent SIP lines.

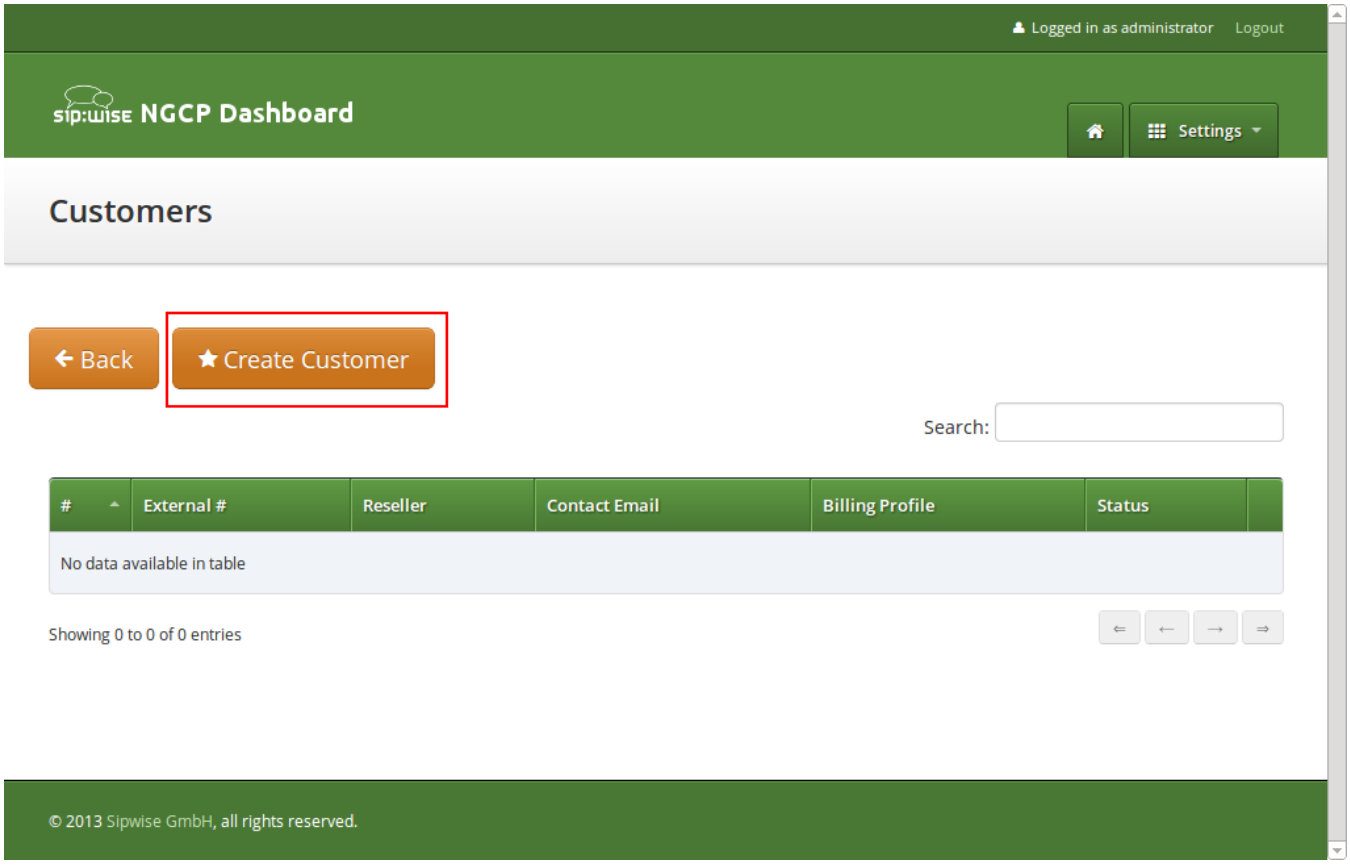
TIP For correct billing, notification and invoicing, create a customer with a single SIP subscriber for the residential service (as it normally has only one telephone line) and a customer with multiple SIP subscribers to provide a service to a company with many telephone lines.

To create a Customer, go to *SettingsCustomers*.

Copyright © Sipwise GmbH - All rights reserved



Click on *Create Customer*.



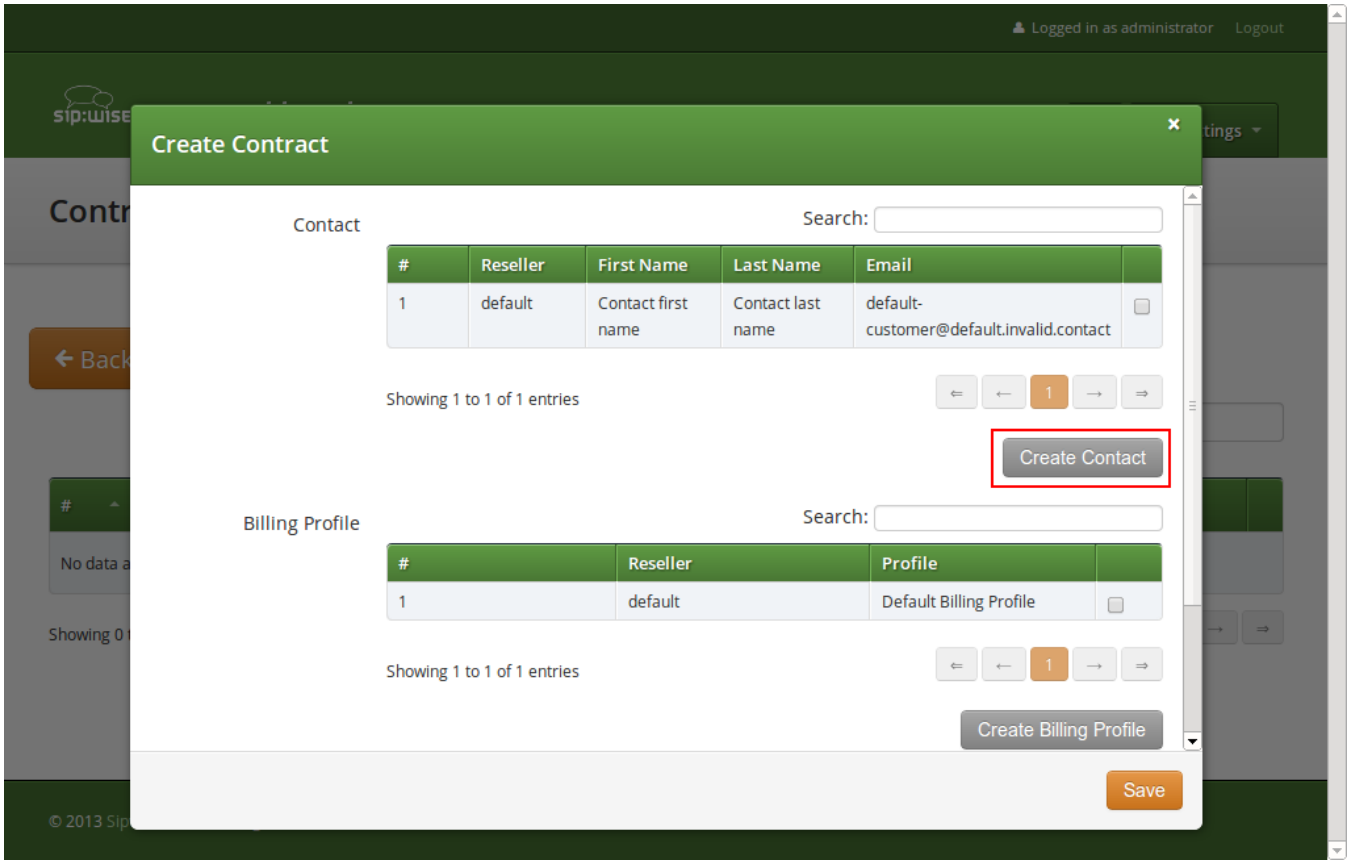
Each *Customer* has a *Contact*—a container for the personal and legal information that identifies a private

 Copyright © Sipwise GmbH - All rights reserved

or corporate customer.

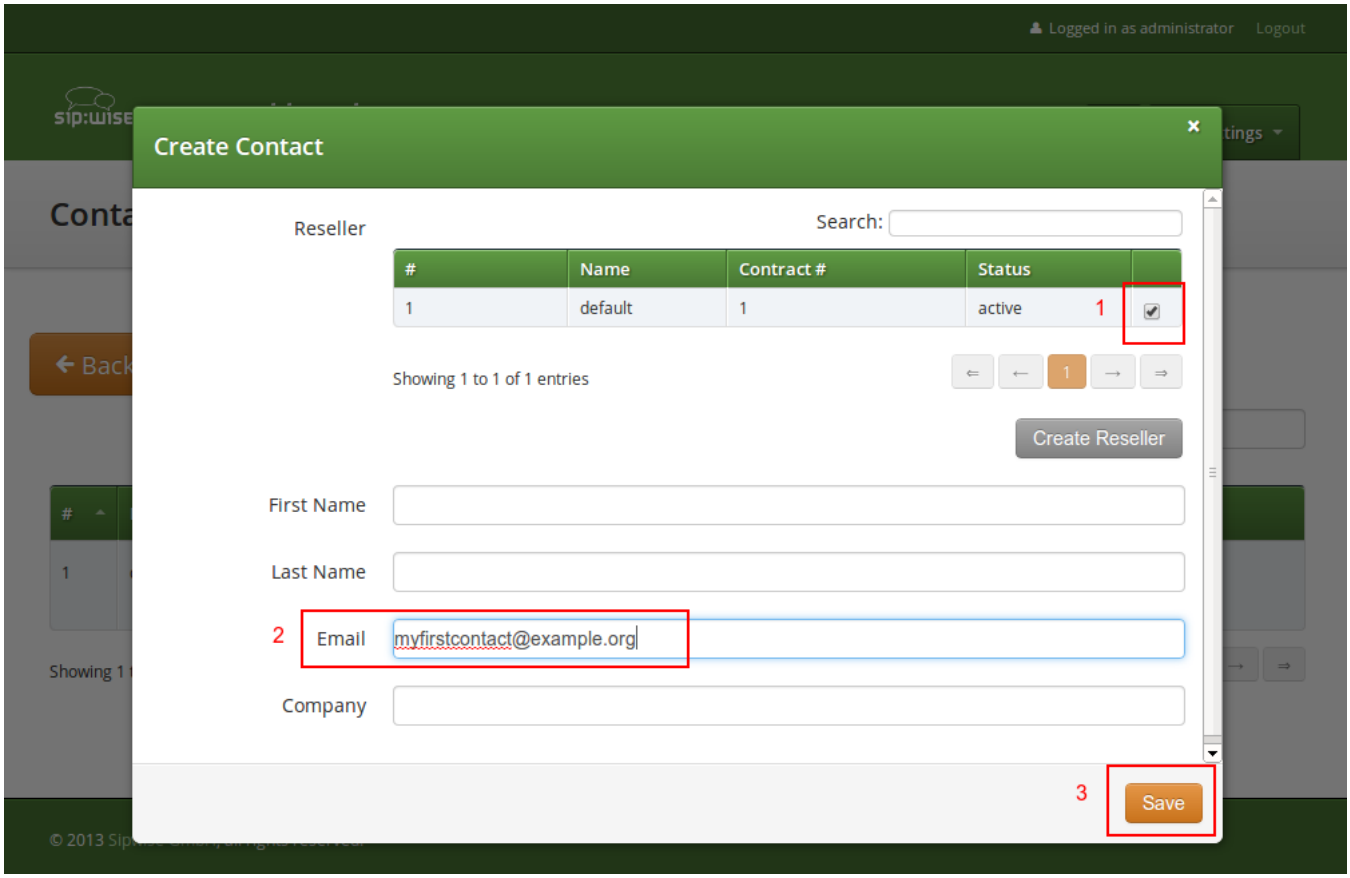
TIP Create a dedicated *Contact* for every *Customer* as it contains specific data e.g. name, address and IBAN that identifies this customer.

Click on *Create Contact* to create a new *Contact*.



Select the required *Reseller* and enter the contact details (at least an *Email* is required), then press *Save*.

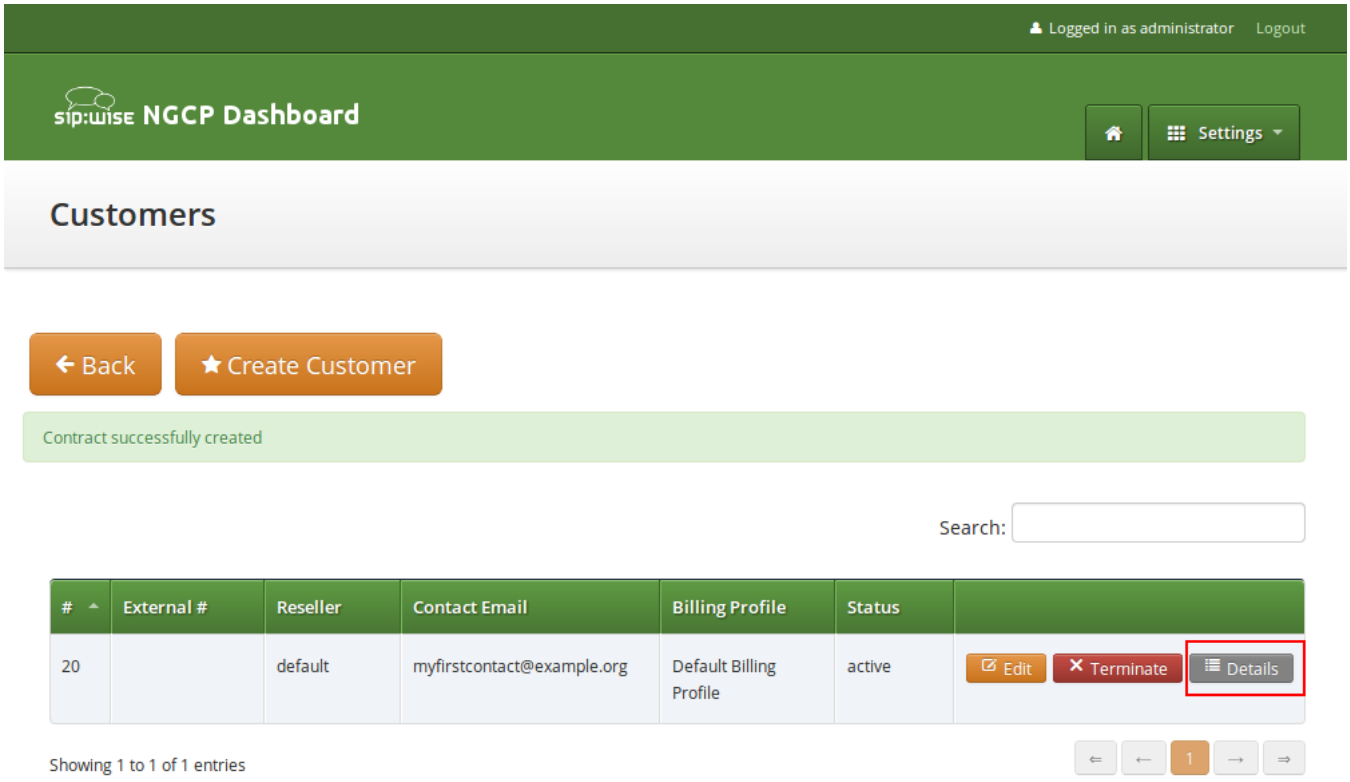
Copyright © Sipwise GmbH - All rights reserved



You will be redirected back to the *Customer* form. The newly created *Contact* is selected by default now, so only select a *Billing Profile* and press *Save*.

You will now see your first *Customer* in the list. Hover over the customer and click *Details* to make extra configuration if necessary.

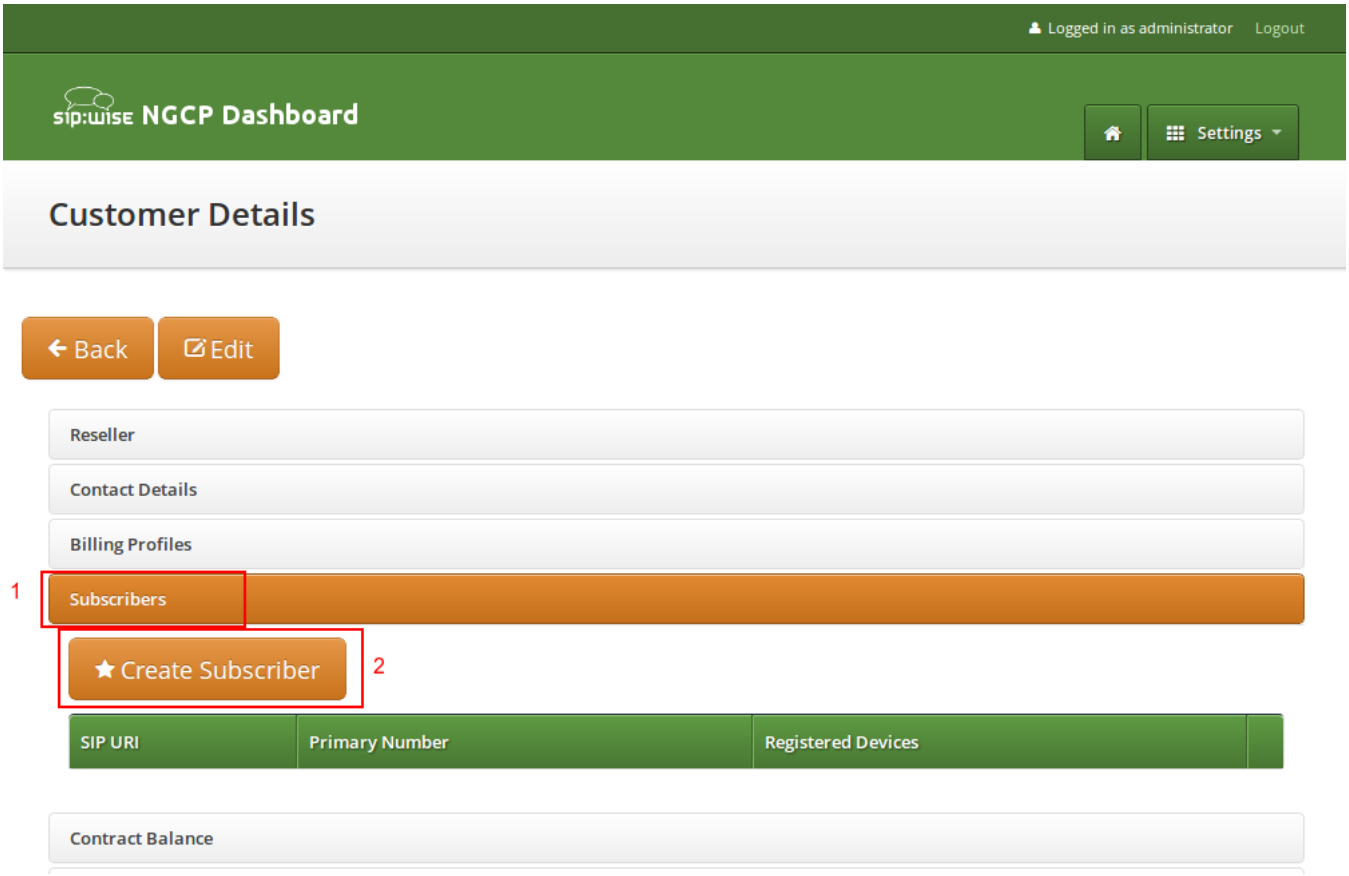
Copyright © Sipwise GmbH - All rights reserved



5.3. Creating a Subscriber

In your *Customer* details view, click on the *Subscribers* row, then click *Create Subscriber*.

Copyright © Sipwise GmbH - All rights reserved



Select a *SIP Domain* created earlier and specify required and optional parameters:

- **Domain**: The domain part of the SIP URI for your subscriber.
- **E164 Number**: This is the telephone number mapped to the subscriber, separated into *Country Code (CC)*, *Area Code (AC)* and *Subscriber Number (SN)*. For the first tests, you can set an imaginary number here and change it later when you get number blocks assigned by your PSTN interconnect partner. So in our example, we'll use *43* as CC, *99* as AC and *1001* as SN to form the imaginary number *+43 99 1001*.

TIP This number can actually be used to place calls between local subscribers, even if you don't have any PSTN interconnection. This comes in handy if you use phones instead of soft-clients for your tests. The format in which this number can be dialled, so the subscriber is reached is defined in Configuring Rewrite Rule Sets.

IMPORTANT Sipwise C5 allows a single subscriber to have multiple E.164 numbers to be used as aliases for receiving incoming calls. Also, Sipwise C5 supports so-called "implicit" extensions. If a subscriber has phone number 012345, but somebody calls 012345100, then NGCP first tries to send the call to number 012345100 (even though the user is registered as 012345). If Sipwise C5 then receives the 404 - Not Found response, it falls back to 012345 (the user-part with which the callee is registered).

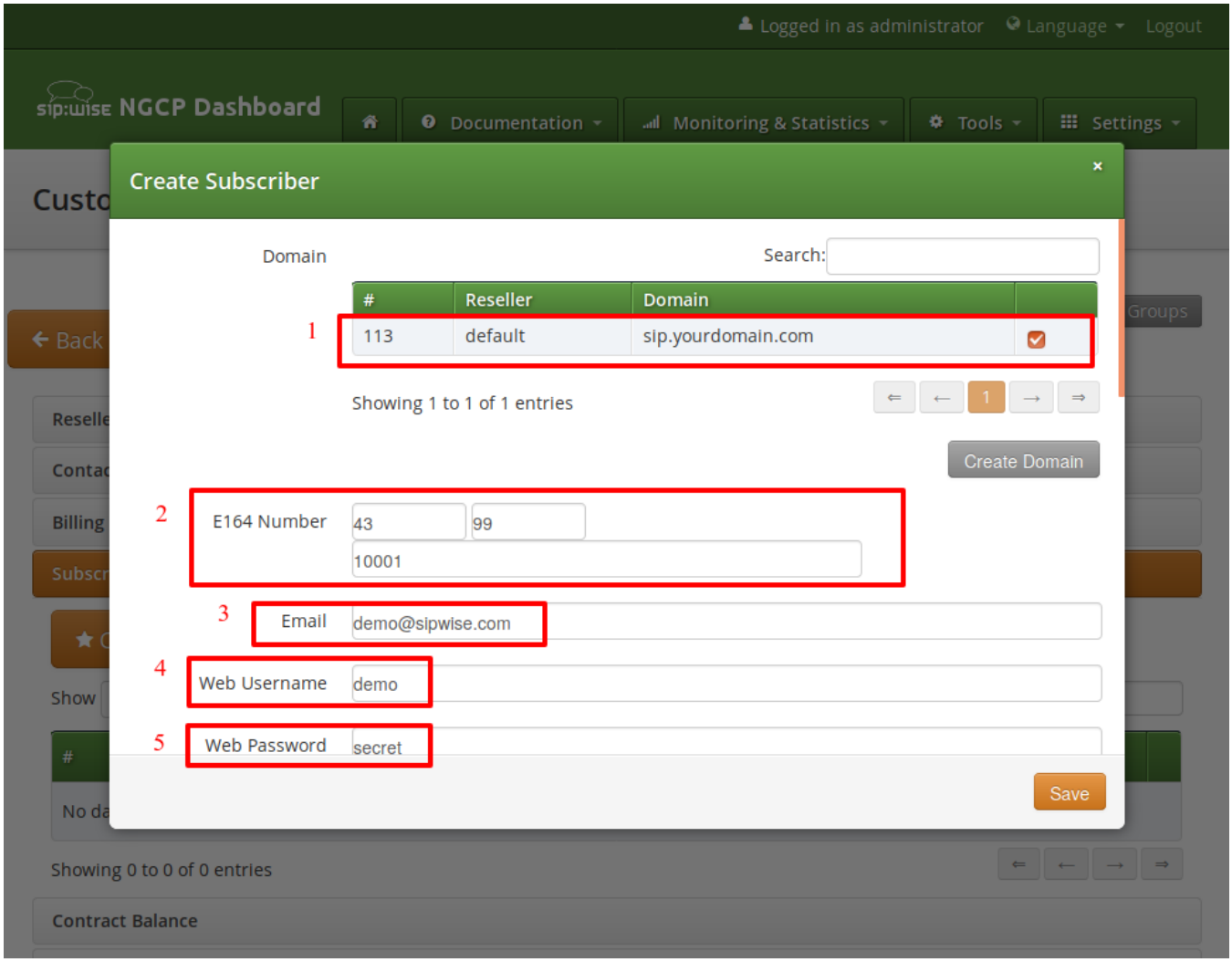
- **Email**: An email address for sending service-related notifications to.
- **Web Username**: This is the user part of the username the subscriber may use to log into her *Customer Self Care Interface*. The user part will be automatically suffixed by the SIP domain you choose for the **SIP URI**. Usually, the web username is identical to the **SIP URI**, but you may choose a

 Copyright © Sipwise GmbH - All rights reserved

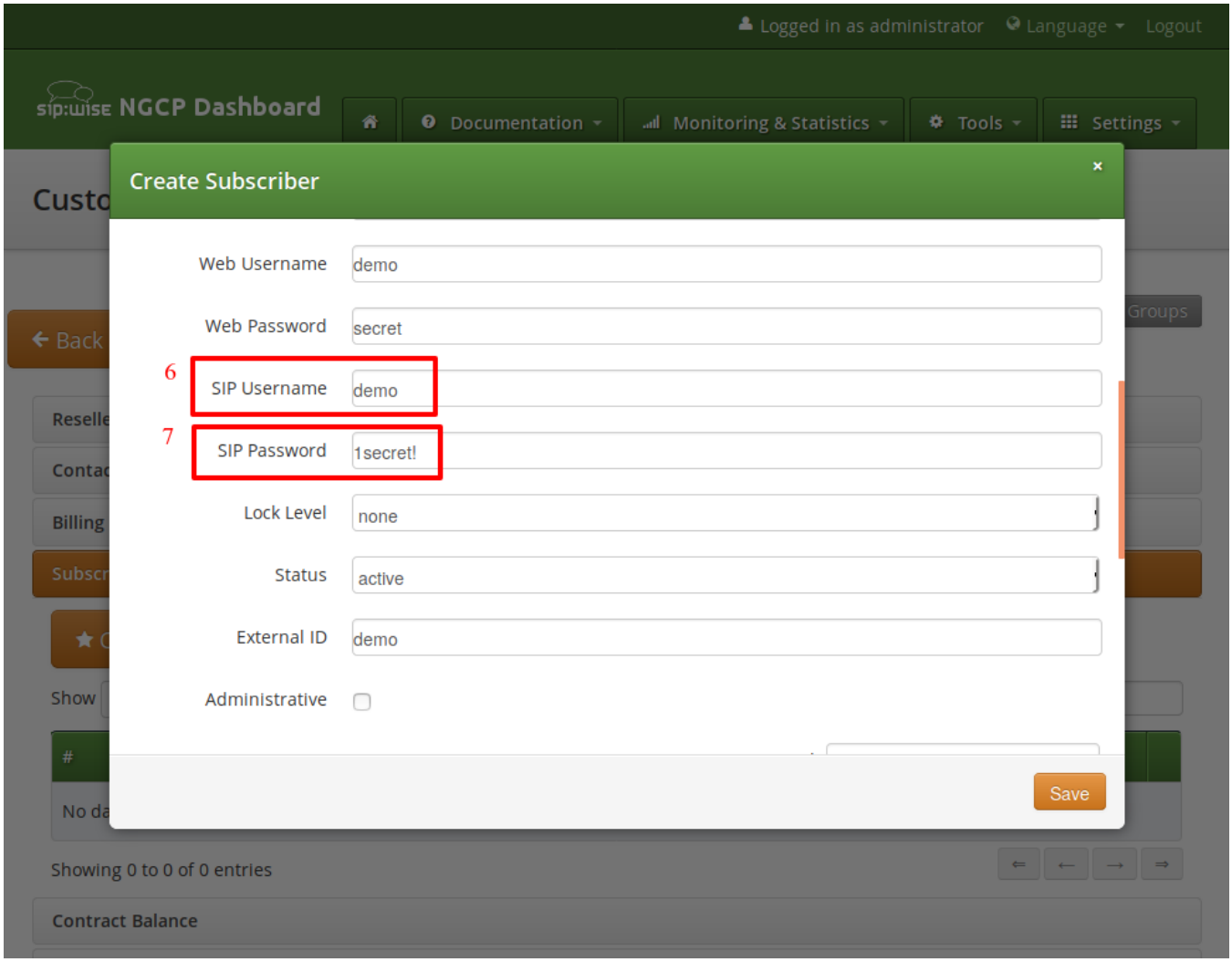
different naming schema.

CAUTION The web username needs to be unique. The system will return a fault if you try to use the same web username twice.

- **Web Password**: This is the password for the subscriber to log into her *Customer Self Care Interface*. It must be at least 6 characters long.
- **SIP Username**: The user part of the SIP URI for your subscriber.
- **SIP Password**: The password of your subscriber to authenticate on the SIP proxy. It must be at least 6 characters long.
- **Status**: You can lock a subscriber here, but for creating one, you will most certainly want to use the *active* status.
- **External ID**: You can provision an arbitrary string here (e.g. an ID of a 3rd party provisioning/billing system).
- **Administrative**: If you have multiple subscribers in one account and set this option for one of them, this subscriber can administrate other subscribers via the *Customer Self Care Interface*.



Copyright © Sipwise GmbH - All rights reserved



Repeat the creation of *Customers* and *Subscribers* for all your test accounts. You should have at least 3 subscribers to test the functionality of the NGCP.

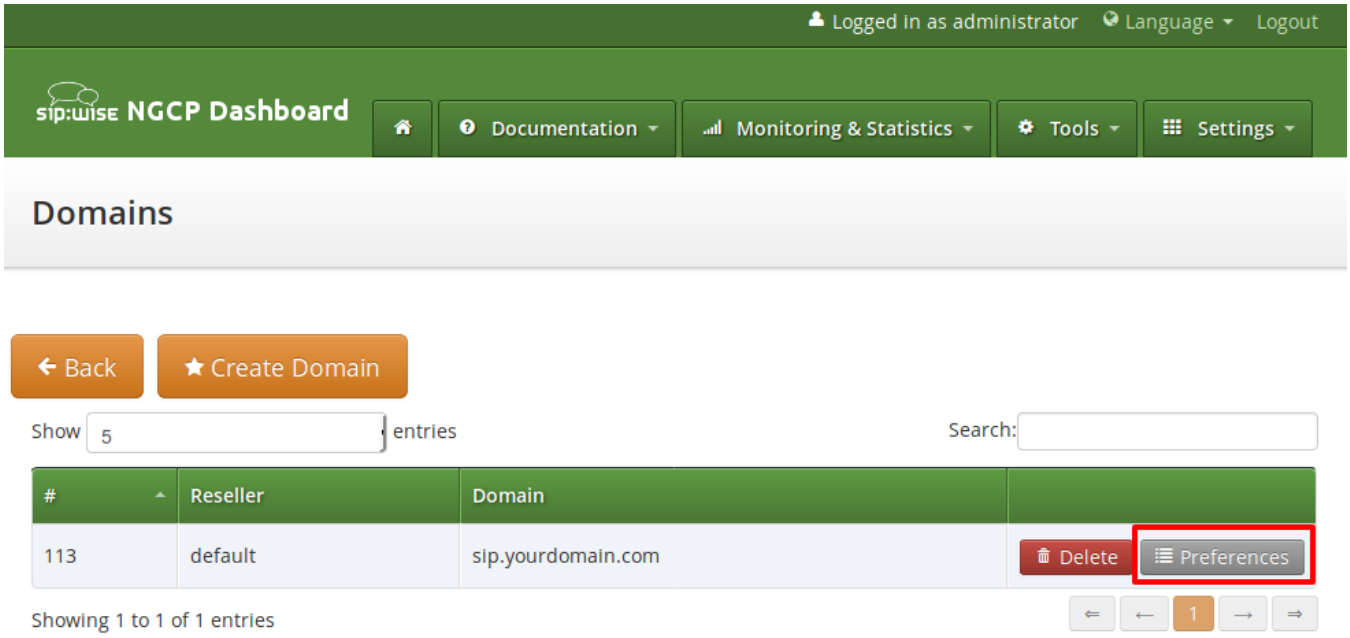
TIP At this point, you're able to register your subscribers to Sipwise C5 and place calls between these subscribers.

You should now revise the *Domain* and *Subscriber* Preferences.

5.4. Domain Preferences

The *Domain Preferences* are the default settings for *Subscriber Preferences*, so you should set proper values there if you don't want to configure each subscriber separately. You can later override these settings in the *Subscriber Preferences* if particular subscribers need special settings. To configure your *Domain Preferences*, go to *SettingsDomains* and click on the *Preferences* button of the domain you want to configure.

Copyright © Sipwise GmbH - All rights reserved



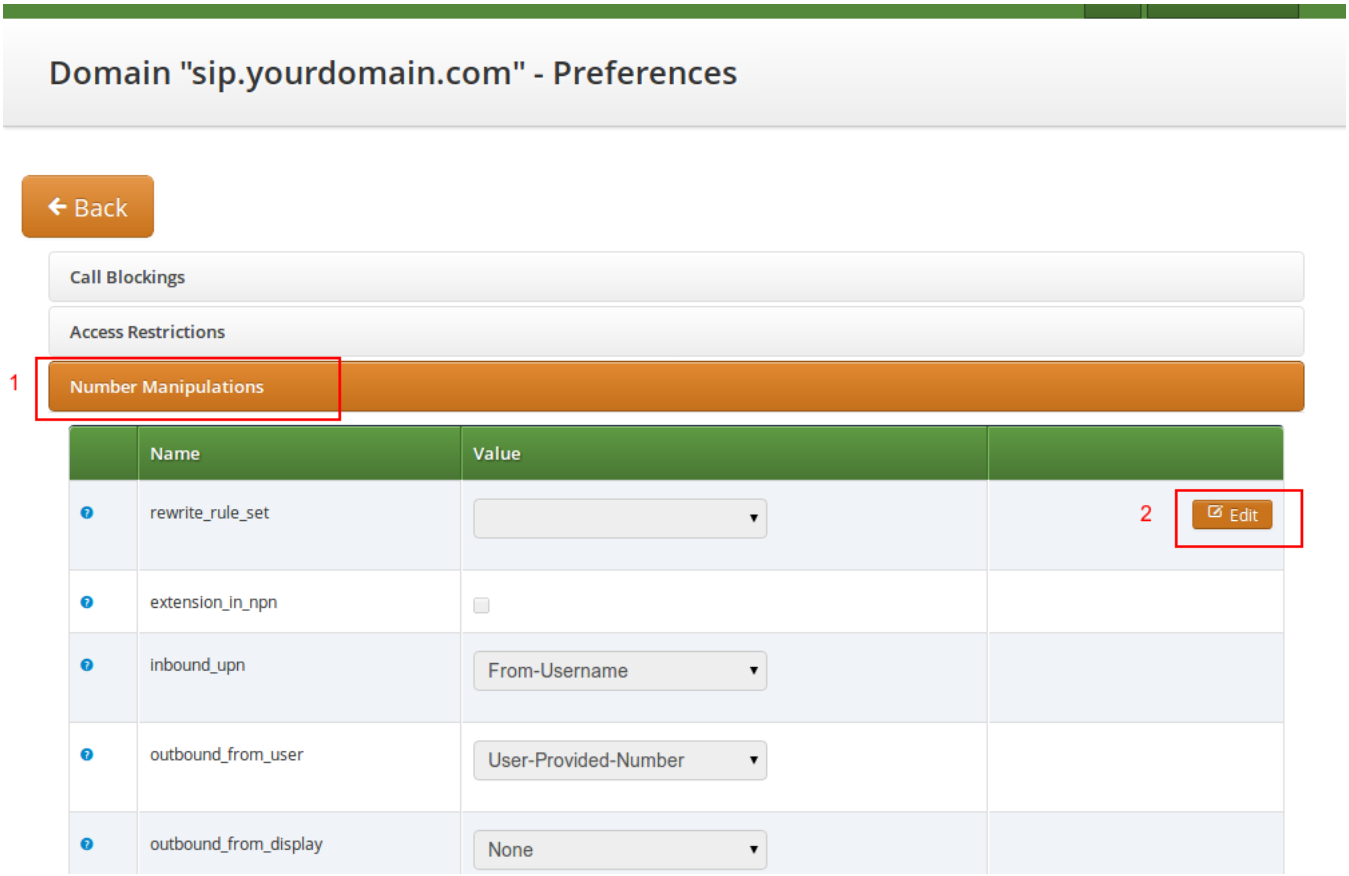
The most important settings are in the *Number Manipulations* group.

Here you can configure the following:

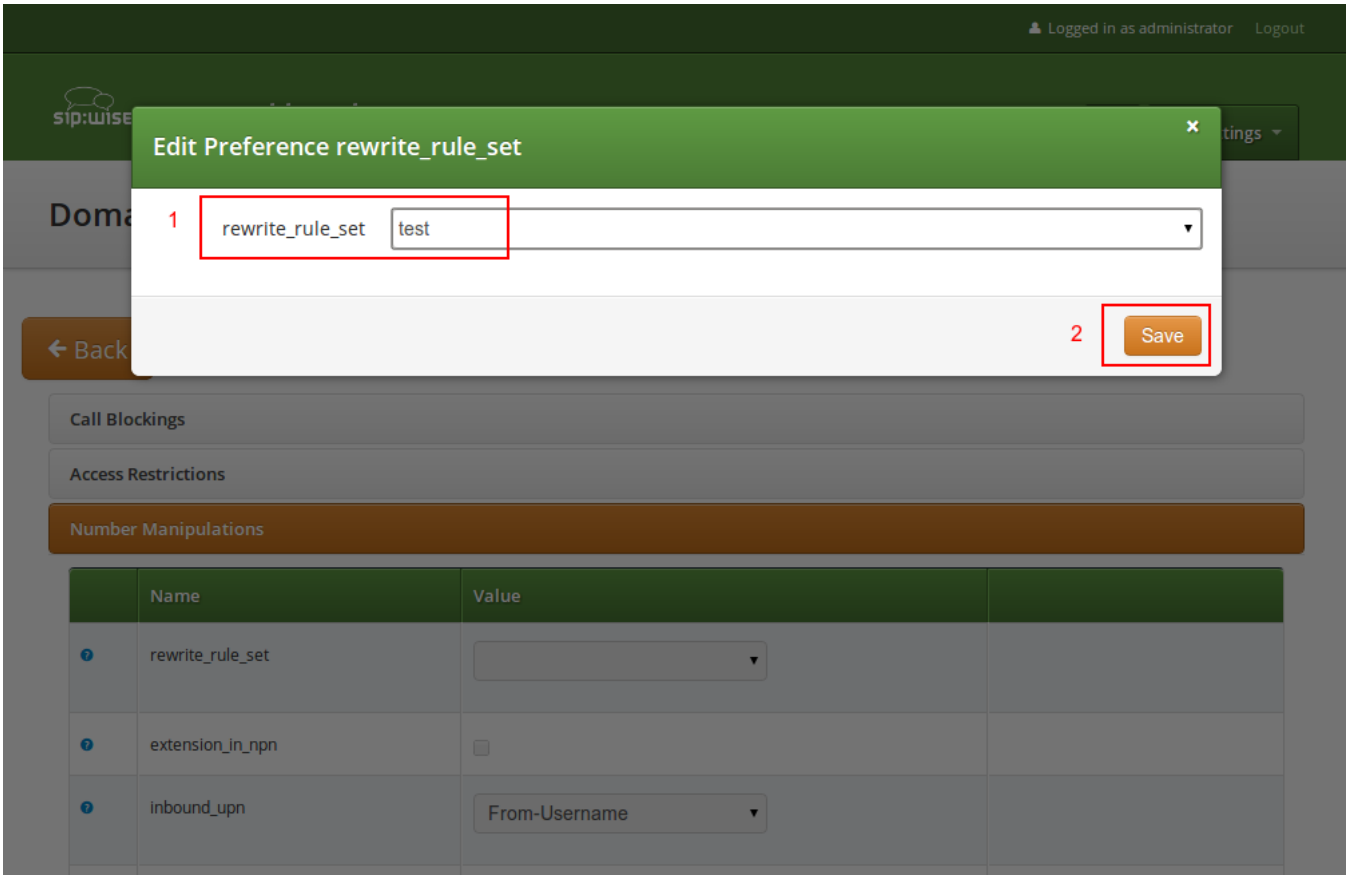
- for incoming calls - which SIP message headers to take numbers from
- for outgoing calls - where in the SIP messages to put certain numbers to
- for both - how these numbers are normalized to E164 format and vice versa

To assign a *Rewrite Rule Set* to a *Domain*, create a set first as described in Configuring Rewrite Rule Sets, then assign it to the domain by editing the *rewrite_rule_set* preference.

Copyright © Sipwise GmbH - All rights reserved



Select the *Rewrite Rule Set* and press *Save*.



Then, select the field you want the *User Provided Number* to be taken from for inbound INVITE

Copyright © Sipwise GmbH - All rights reserved

messages. Usually the *From-Username* should be fine, but you can also take it from the *Display-Name* of the From-Header, and other options are available as well.

5.5. Subscriber Preferences

You can override the *Domain Preferences* on a subscriber basis as well. Also, there are *Subscriber Preferences* which don't have a default value in the *Domain Preferences*.

To configure your *Subscriber*, go to *SettingsSubscribers* and click *Details* on the row of your subscriber. There, click on the *Preferences* button on top.

You want to look into the *Number Manipulations* and *Access Restrictions* options in particular, which control what is used as user-provided and network-provided calling numbers.

- For outgoing calls, you may define multiple numbers or patterns to control what a subscriber is allowed to send as user-provided calling numbers using the *allowed_clis* preference.
- If *allowed_clis* does not match the number sent by the subscriber, then the number configured in *cli* (the network-provided number) preference will be used as user-provided calling number instead.
- You can override any user-provided number coming from the subscriber using the *user_cli* preference.

NOTE Subscribers preference *allowed_clis* will be synchronized with subscribers primary number and aliases if *ossbssprovisioningauto_allow_cli* is set to **1** in */etc/ngcp-config/config.yml*.

NOTE Subscribers preference *cli* will be synchronized with subscribers primary number if *ossbssprovisioningauto_sync_cli* is set to **yes** in */etc/ngcp-config/config.yml*.

5.5.1. Subscriber authentication for outbound calls

There are cases when Sipwise C5 should pass the authentication process for a subscriber. In other words to pass the authentication process of an outbound call from behalf of the subscriber entity (configuration object).

Suppose there is Sipwise C5 and some other Class 5 system (can be just another Sipwise C5). You have recently migrated part of subscribers from another Class 5 system to Sipwise C5. But, you still have SIP peerings (with ITSPs) at that system, and you would like that recently migrated subscribers are still able to terminate calls via that another Class 5 system.

This is when the 'Remote Authentication' parameters start helping you. The call flow in this scenario will be: *Sipwise C5another Class 5 systemSIP peering*

And that system (another Class 5 system) will of course treat a call coming to it from Sipwise C5, as if that would be a direct call from the subscriber (indeed it's not).

This is when you need to be capable of the authentication and Sipwise C5 gives you this possibility.

You will need to go to subscriber's preferences and to know specific credentials to be used for that, in order to pass the authorization. To configure this setting, open the *Remote Authentication* tab and edit the following four preferences:

Copyright © Sipwise GmbH - All rights reserved

- **peer_auth_user:** <username for peer auth>
- **peer_auth_pass:** <password for peer auth>
- **peer_auth_realm:** <domain for peer auth>
- **peer_auth_hf_user:** <username parameter for Authorization hf>

NOTE 'peer_auth_hf_user' preference is optional and can be skipped. It allows you to set a specific username for the Proxy-Authorization header.

As soon as you define those parameters, a call from behalf of the subscriber, which will be terminated at another system, can be successfully authenticated.

5.5.2. Subscriber authentication for an outbound registration

This is approximately the same use case as for the 'Subscriber authentication for outbound calls' section, but this time for the registration process. Sipwise C5 can register at remote system, hence placing the location record on behalf of the subscriber.

NOTE Location record which will be saved at a remote system, will contain the contact of Sipwise C5 Load-Balancer, not the contact of the end subscriber.

The reason why you might need Sipwise C5 to register at remote system from behalf of the subscriber:

- you want to receive calls from remote system to your subscriber, as if your subscriber would receive this directly;
- that remote system doesn't accept a call sent from Sipwise C5 from behalf of the subscriber, without a registration beforehand;

NOTE This registration process will be completely independent from the end subscriber, and will be only triggered and controlled by Sipwise C5.

This is when the 'Remote Authentication' parameters help you again. You need to go to subscriber's preferences, open the *Remote Authentication* tab and now additionally enable outbound registration:

- **peer_auth_register:** *True*

Now in common with the preferences you defined previously in the 'Subscriber authentication for outbound calls' section, the registration process will start using given credentials.

IMPORTANT Remember, the subscriber's preference 'peer_auth_hf_user' affects invitation scenarios, as well as registration scenarios. Hence in case you set it, the username during digest, will be swapped.

5.5.3. Subscriber's preference for media in terms of IP Family (IPv4/IPv6)

It is possible to define which IP Family is preferred by a particular subscriber for sending media (RTP/RTCP), using the following preference:

*NAT and Media Flow Control*ipv46_for_rtpproxy, which can be set to:

 Copyright © Sipwise GmbH - All rights reserved

- **Force IPv4** - the subscriber prefers to send the media flow via IPv4
- **Force IPv6** - the subscriber prefers to send the media flow via IPv6
- **Auto-detect** - auto detection is enabled and depending on the IP Family used during the registration, the preference for media will be picked out accordingly
- **use domain default** - preference is inherited from the domain values

Both the domain and subscriber preferences contain the *NAT and Media Flow Control* section, which gives an access to the *ipv46_for_rtpproxy* preference.

TIP It is not necessarily, but is however recommended to set *ipv46_for_rtpproxyForce IPv6* for those subscribers which use IPv6 to send media (RTP/RTCP).

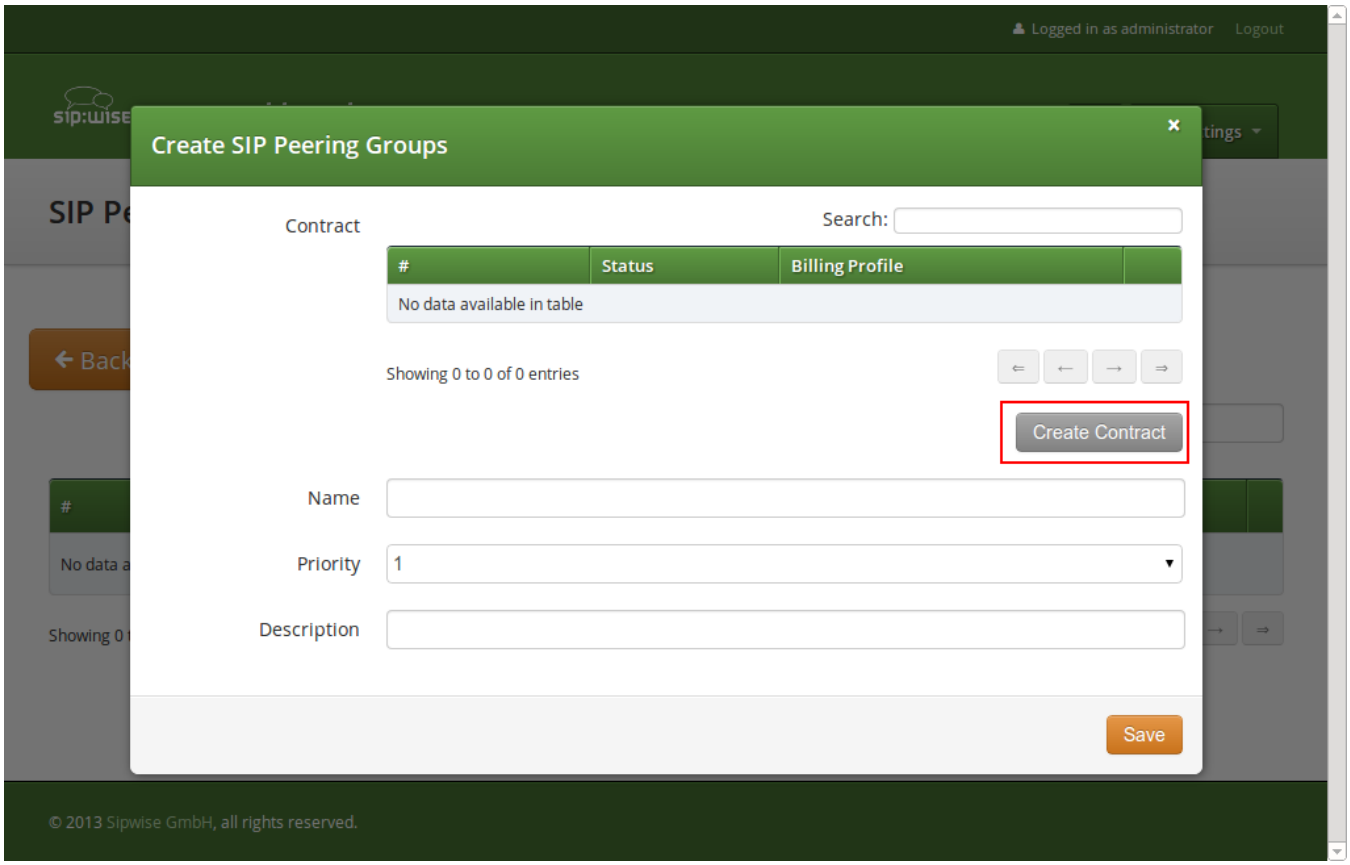
5.6. Creating Peerings

If you want to terminate calls at or allow calls from 3rd party systems (e.g. PSTN gateways, SIP trunks), you need to create SIP peerings for that. To do so, go to *SettingsPeerings*. There you can add peering groups, and for each peering group add peering servers and rules controlling which calls are routed over these groups. Every peering group needs a peering contract for correct interconnection billing.

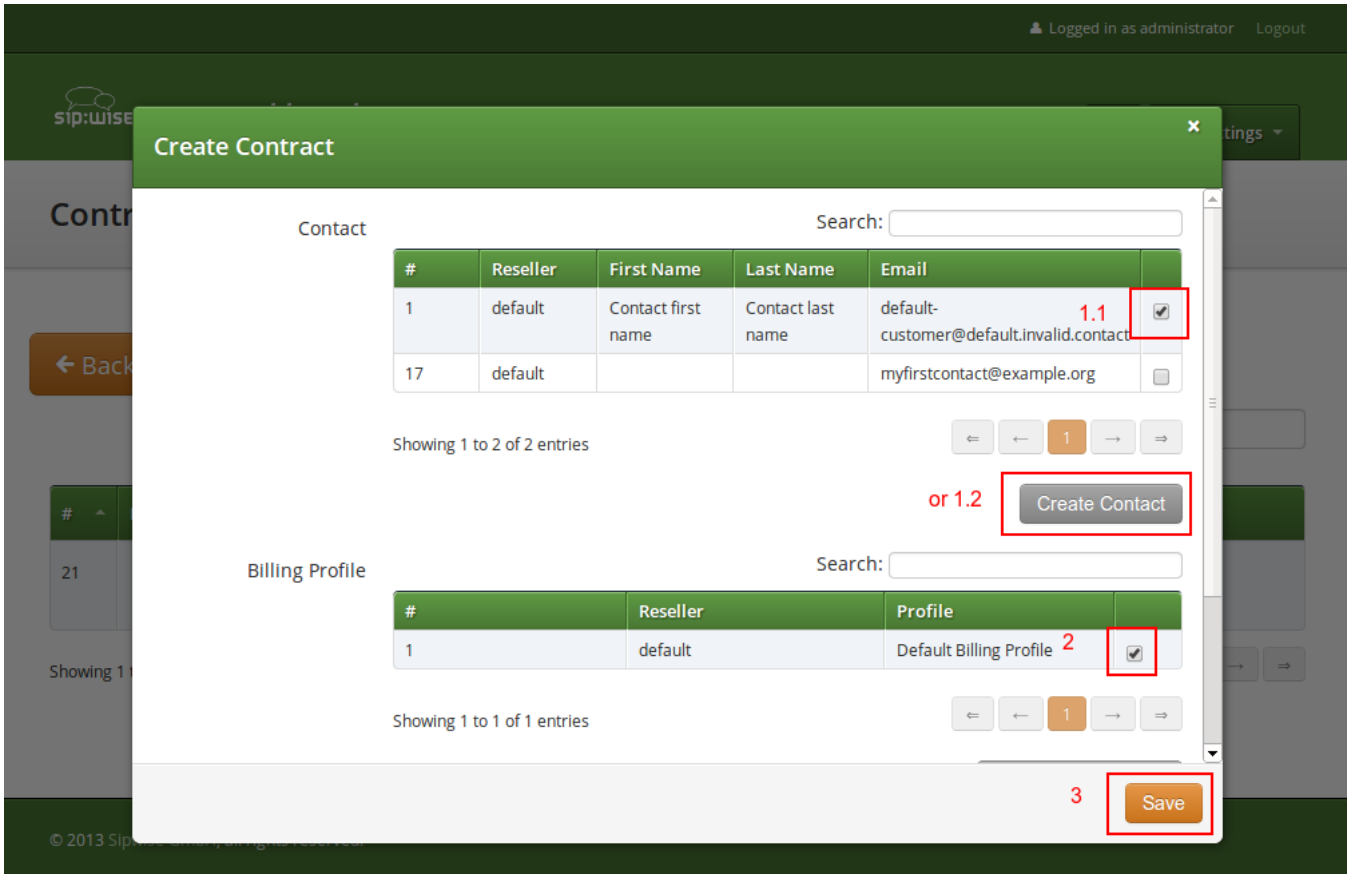
5.6.1. Creating Peering Groups

Click on *Create Peering Group* to create a new group.

In order to create a group, you must select a peering contract. You will most likely want to create one contract per peering group.



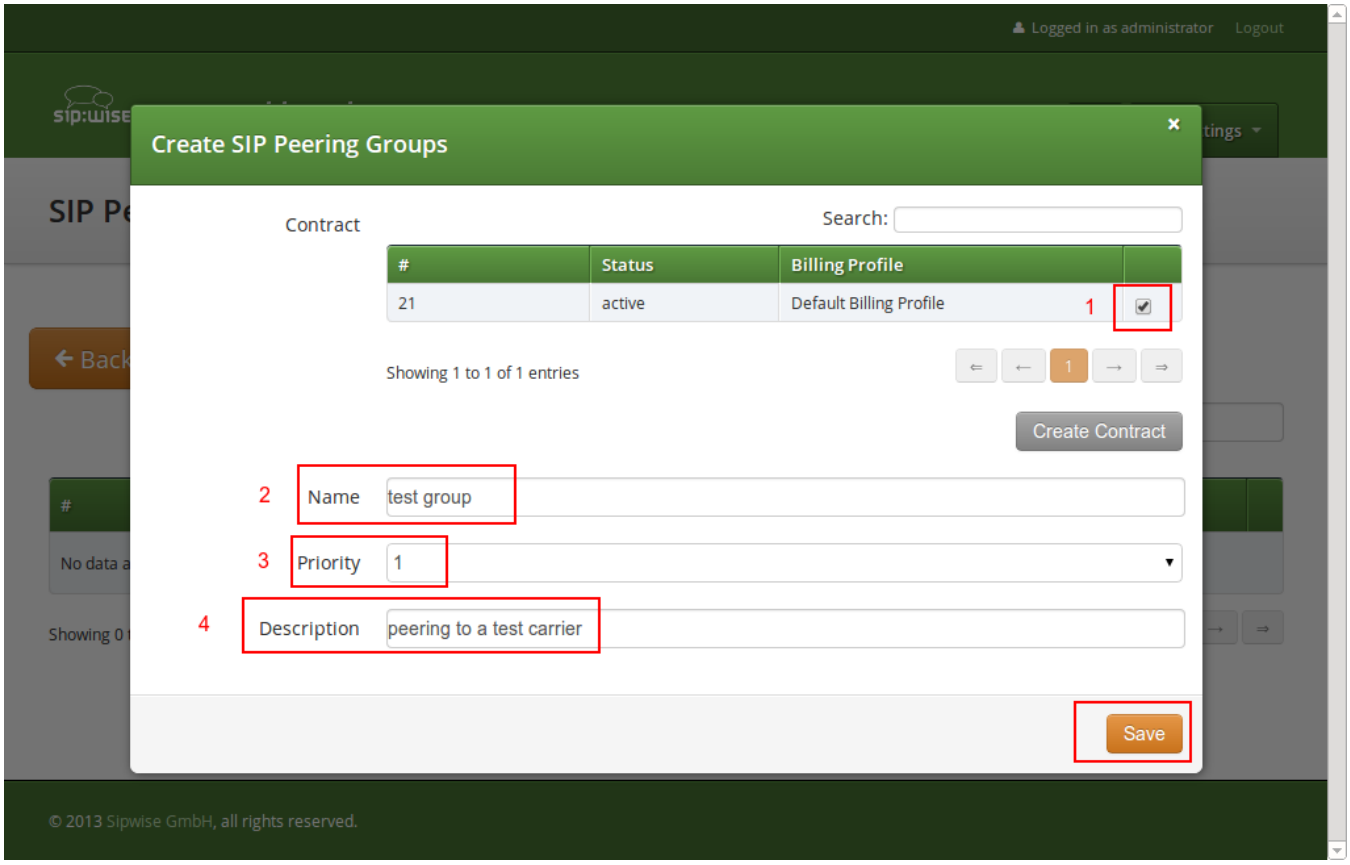
Click on *Create Contract* create a *Contact*, then select a *Billing Profile*.



Click *Save* on the *Contacts* form, and you will get redirected back to the form for creating the actual *Peering Group*. Put a name, priority and description there, for example:

- **Peering Contract:** select the id of the contract created before
- **Name:** test group
- **Priority:** 1
- **Description:** peering to a test carrier

Copyright © Sipwise GmbH - All rights reserved



The *Priority* option defines which *Peering Group* to favor (Priority **1** gives the highest precedence) if two peering groups have peering rules matching an outbound call. *Peering Rules* are described below.

Then click *Save* to create the group.

5.6.2. Creating Peering Servers

In the group created before, you need to add peering servers to route calls to and receive calls from. To do so, click on *Details* on the row of your new group in your peering group list.

To add your first *Peering Server*, click on the *Create Peering Server* button.

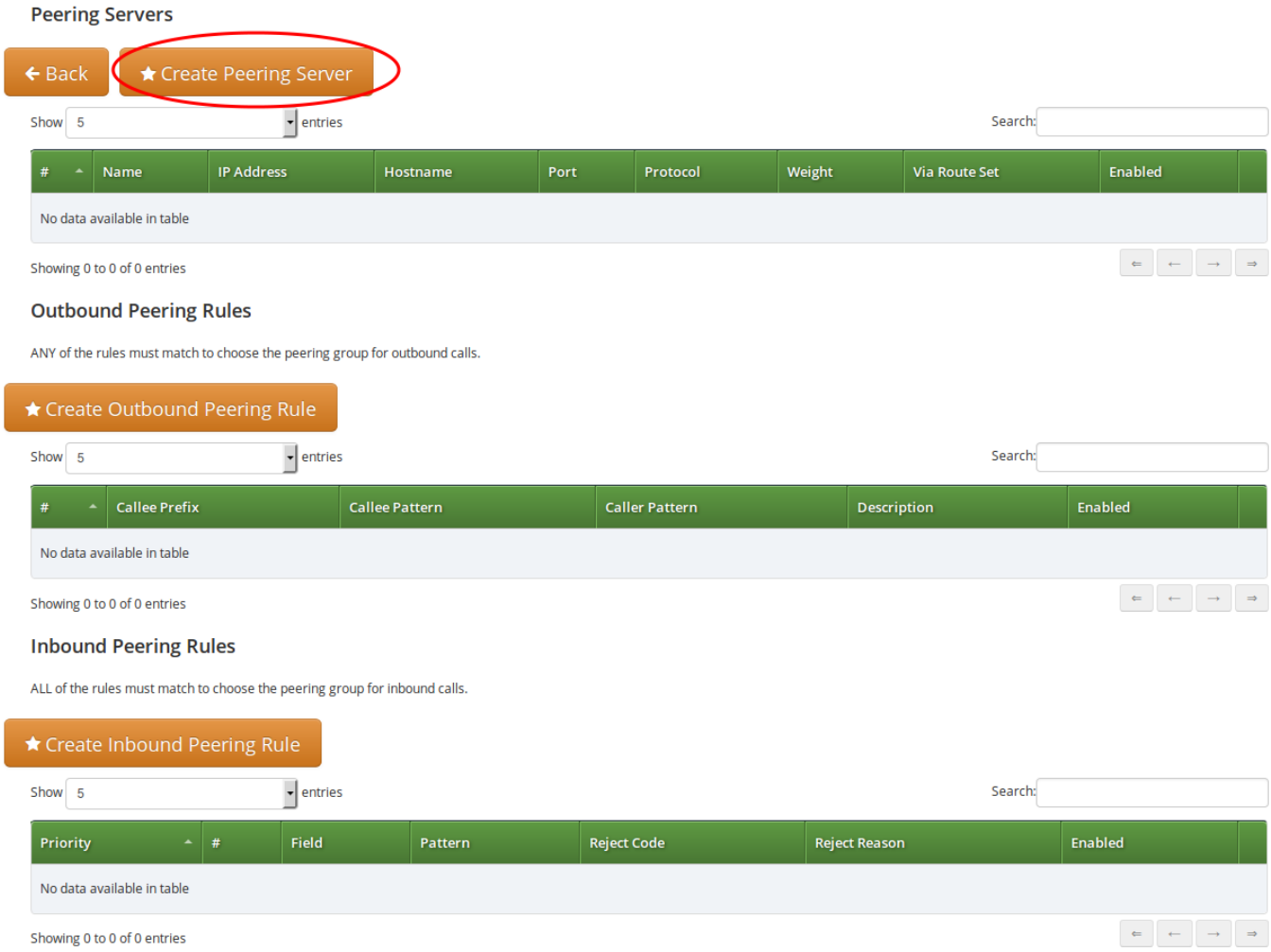


Figure 11. Create Peering Server

In this example, we will create a peering server with IP *2.3.4.5* and port *5060*:

- **Name**: test-gw-1
- **IP Address:** 2.3.4.5
- **Hostname:** leave empty
- **Port:** 5060
- **Protocol:** UDP
- **Weight:** 1
- **Via Route:** None
- **Enable Probing:** enable it, if remote side supports SIP OPTIONS ping

 Copyright © Sipwise GmbH - All rights reserved

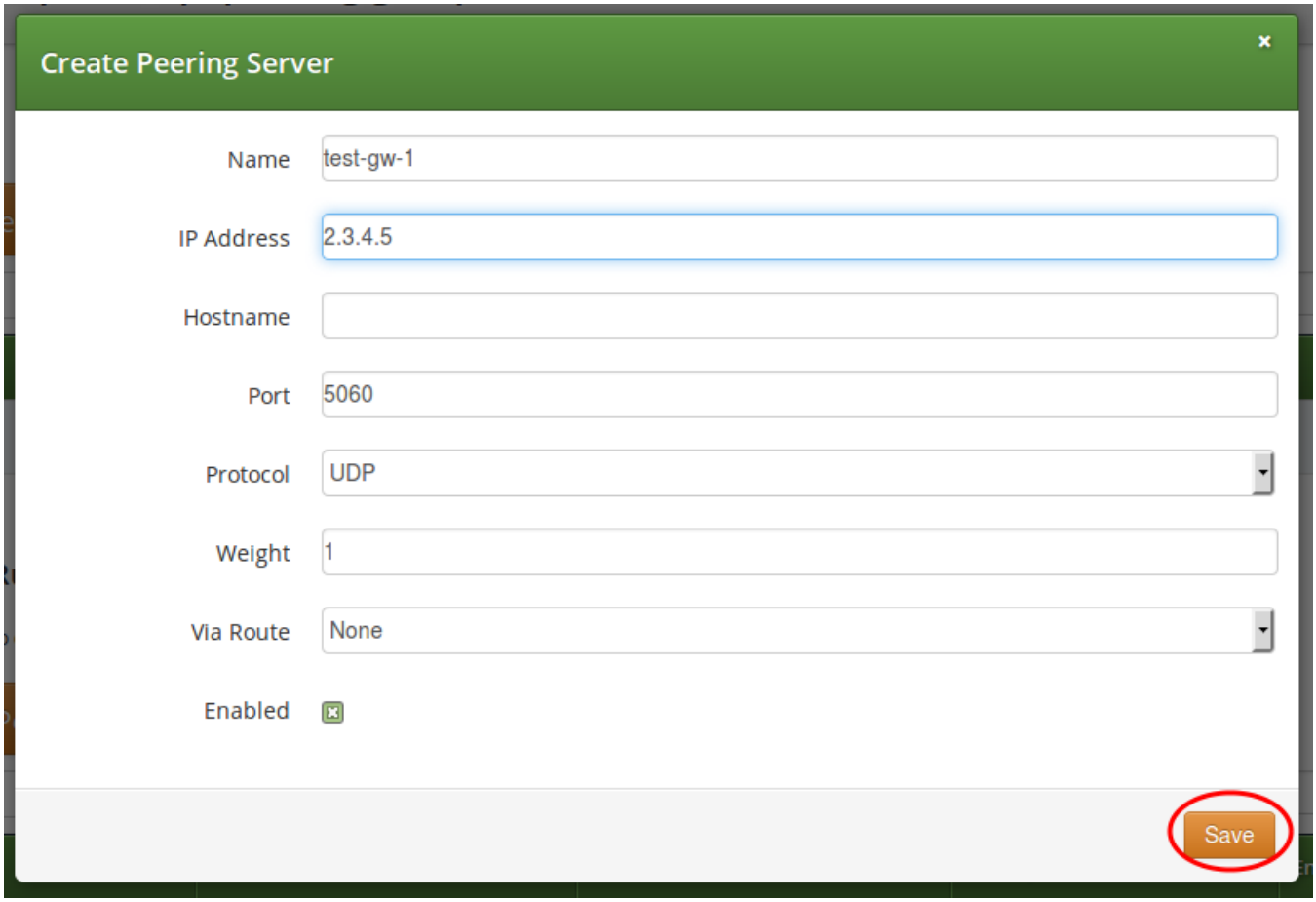


Figure 12. Peering Server Properties

Click *Save* to create the peering server.

TIP The *hostname* field for a peering server is optional. Usually, the IP address of the peer is used as the **domain** part of the Request URI. Fill in this field if a peer requires a particular hostname instead of the IP address. The IP address must always be given though as it is used for the selection of the inbound peer. By default outbound requests will always be sent to the specified IP address, no matter what you put into the *hostname* field. If you want to send the request using the DNS resolution of the configured *hostname*, disregarding in that way the IP, you have to enable outbound_hostname_resolution option in peer preferences.

TIP If you want to add a peering server with an IPv6 address, enter the address without surrounding square brackets into the *IP Address* column, e.g. *::1*.

You can force an additional hop (e.g. via an external SBC) towards the peering server by using the *Via Route* option. The available options you can select there are defined in /etc/ngcp-config/config.yml, where you can add an array of SIP URIs in kamailiolbexternal_sbc like this:

Copyright © Sipwise GmbH - All rights reserved

```
kamailio:
  lb:
    external_sbc:
      - sip:192.168.0.1:5060
      - sip:192.168.0.2:5060
```

Execute ngcpcfg apply "added external sbc gateways", then edit your peering server and select the hop from the *Via Route* selection.

Once a peering server has been created, this server can already send calls to the system.

NOTE Requests coming from a SIP peering are matched not only by the IP address and a transport protocol, but also using the source port of a message. This means, if your SIP peering server created at Sipwise C5 has the 'Port' value set to '5060', then it's expected that messages (requests) coming from this SIP peering, will have the source port '5060'. This however applies only to the UDP transport based connections (TCP and TLS are matched only using an IP address and a transport protocol).

Outbound Peering Rules

IMPORTANT To be able to send outbound calls towards the servers in the *Peering Group*, you also need to define *Outbound Peering Rules*. They specify which source and destination numbers are going to be terminated over this group. To create a rule, click the *Create Outbound Peering Rule* button.

Copyright © Sipwise GmbH - All rights reserved

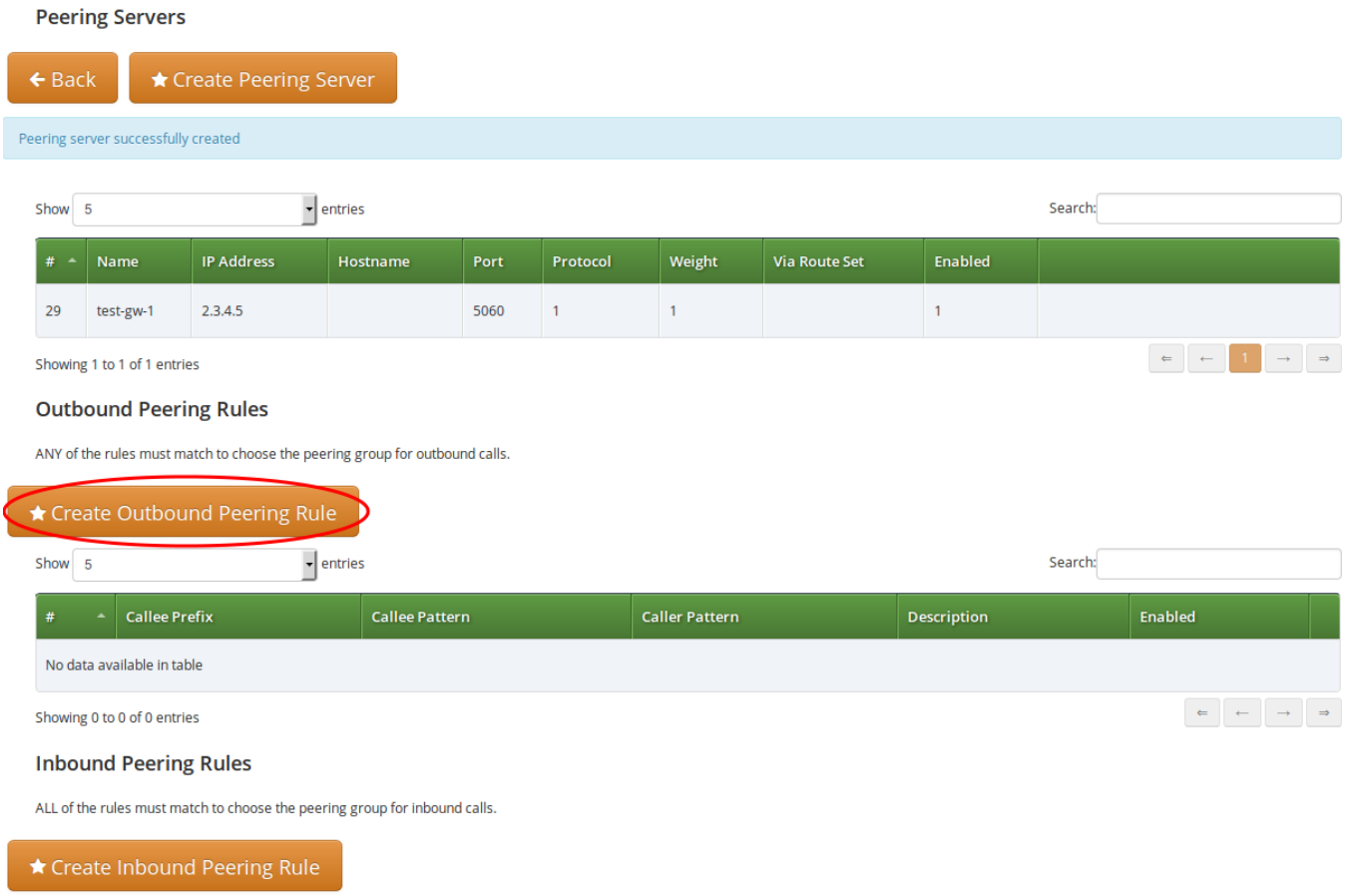


Figure 13. Create Outbound Peering Rule

Since the previously created peering group will be the only one in our example, we have to add a default rule to route *all* calls via this group. To do so, create a new peering rule with the following values:

- **Callee Prefix:** leave empty
- **Callee Pattern:** leave empty
- **Caller Pattern:** leave empty
- **Description:** Default Rule
- **Stopper:** leave empty

Copyright © Sipwise GmbH - All rights reserved

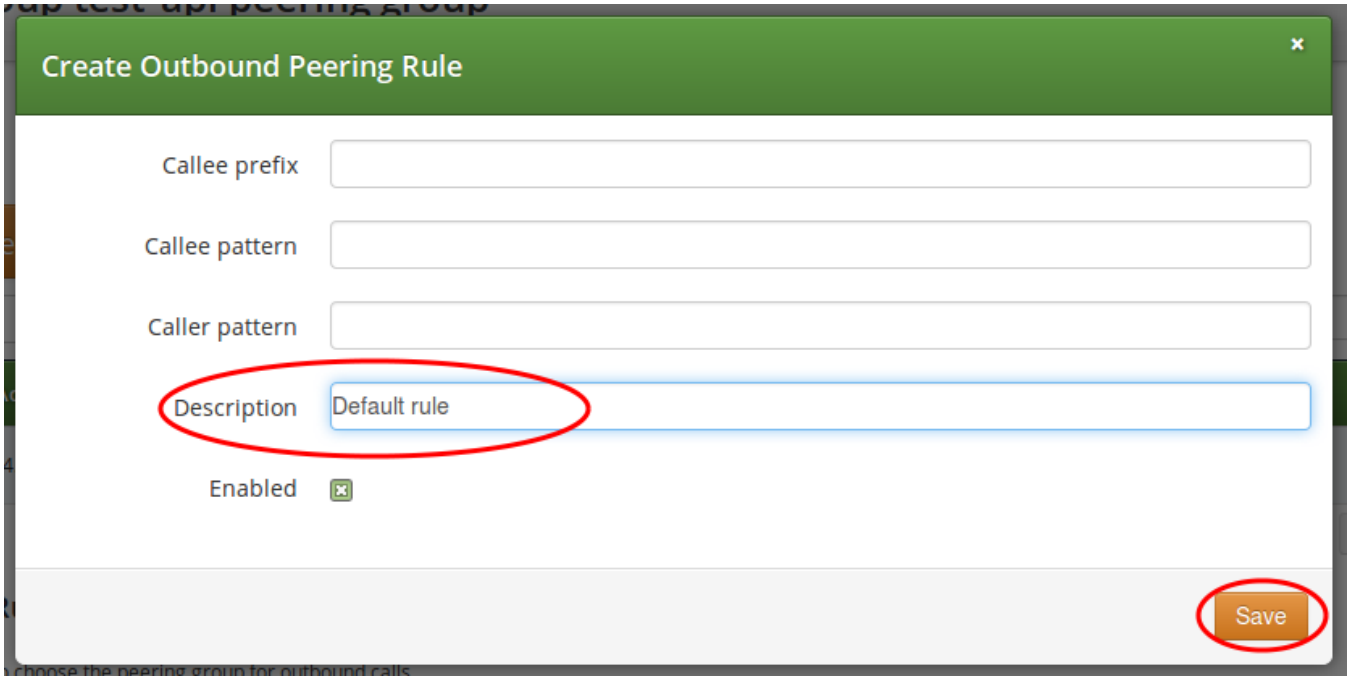


Figure 14. Outbound Peering Rule Properties

Then click *Save* to add the rule to your group.

TIP In contrast to the callee/caller pattern, the callee prefix has a regular alphanumeric string and can not contain any regular expression.

TIP If you set the caller or callee rules to refine what is routed via this peer, enter all phone numbers in full E.164 format, that is <*cc*><*ac*><*sn*>.

TIP The *Caller Pattern* field covers the whole URI including the subscriber domain, so you can only allow certain domains over this peer by putting for example @example\.com into this field.

Inbound Peering Rules

Sipwise C5 supports filtering SIP INVITE requests sent by SIP peers. The system administrator may define one or more matching rules for SIP URIs that are present in the headers of SIP INVITE requests, and select which SIP header (or part of the header) must match the pattern declared in the rule.

If the incoming SIP INVITE message has the proper headers, Sipwise C5 will accept and further process the request. If the message does not match the rule it will be rejected.

CAUTION An incoming SIP INVITE message must match **all the inbound peering rules** so that Sipwise C5 does not reject the request.

In order to **create an inbound peering rule** you have to select a peering group, press *Details* and then press *Create Inbound Peering Rule* button.

 Copyright © Sipwise GmbH - All rights reserved

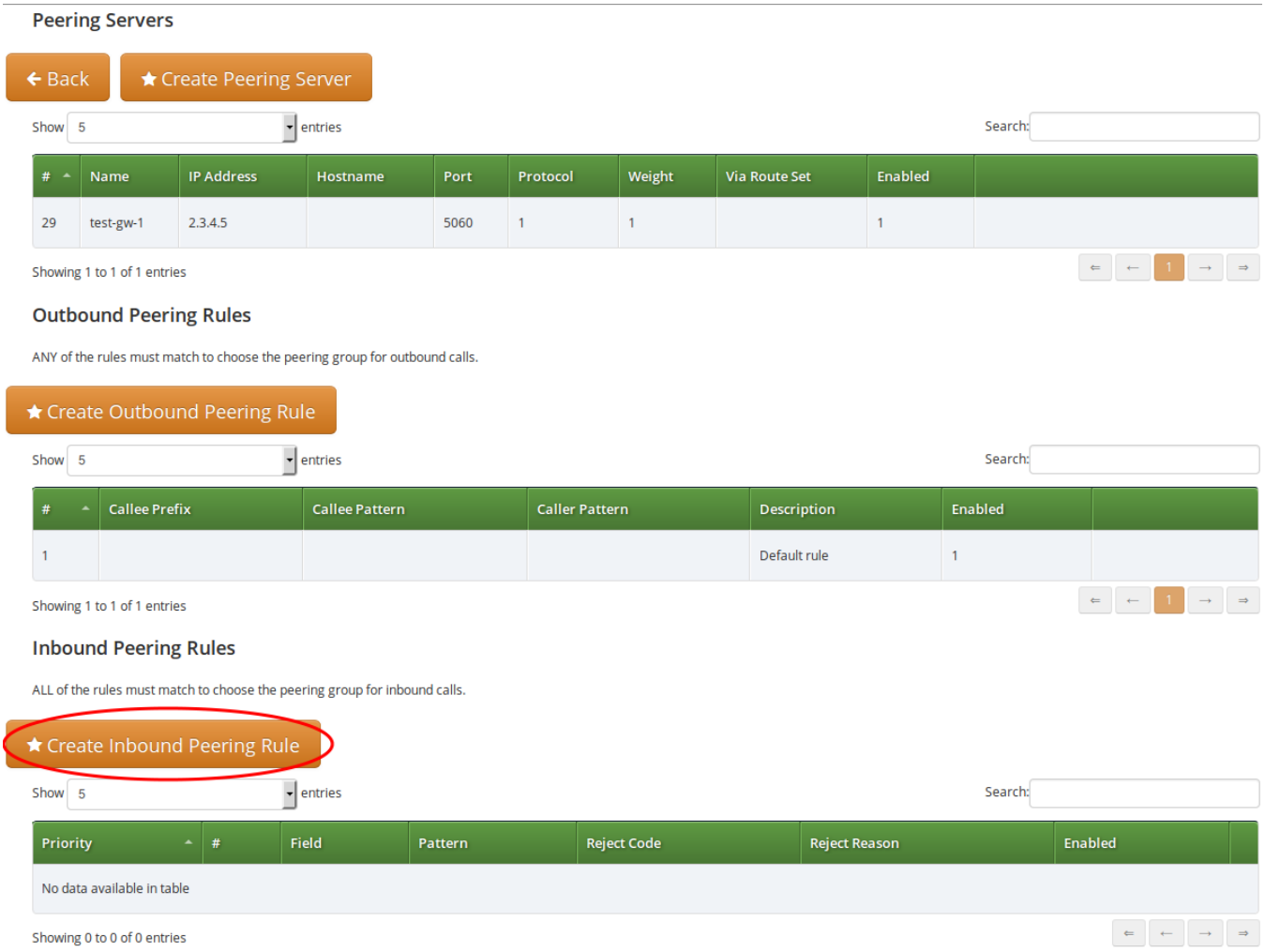


Figure 15. Create Inbound Peering Rule

An inbound peering rule has the following **properties**:

Copyright © Sipwise GmbH - All rights reserved

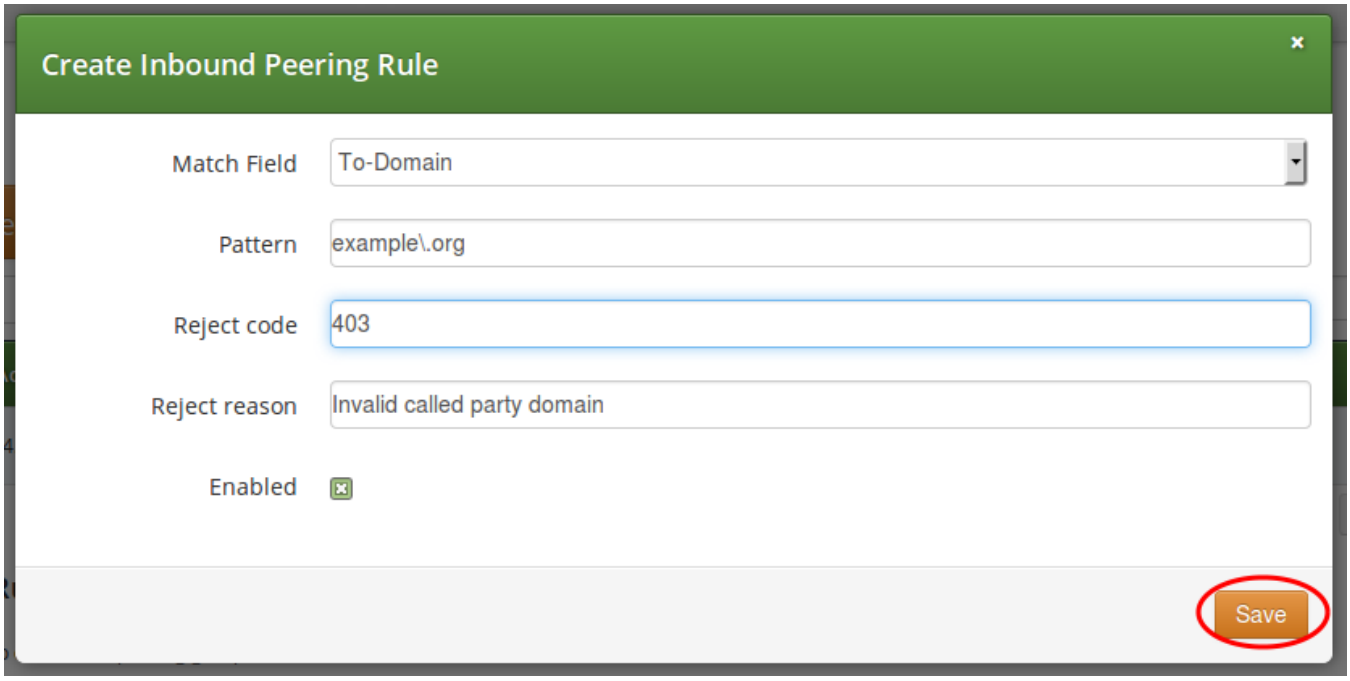


Figure 16. Inbound Peering Rule Properties

- Match Field: select which header and which part of that header in a SIP INVITE message will be checked for matching the pattern
- Pattern: a POSIX regular expression that defines the accepted value of a header; example: `^sip:.+@example\.org$`—this will match a SIP URI that contains "example.org" in the domain part
- Reject code: optional; a SIP status code that will be sent as a response to an INVITE request that does not match the pattern; example: 403
- Reject reason: optional; an arbitrary text that will be included in the SIP response sent with the *reject code*
- Enabled: a flag to enable / disable the particular inbound peering rule

NOTE Both of the properties Reject code and Reject reason must be left empty if a peering server (i.e. a specific IP address) is part of more peering groups. Such a configuration is useful when an incoming SIP INVITE request needs to be treated differently in the affected peering groups, based on its content, and that's why if the INVITE message only partly matches an inbound peering rule it should not be rejected.

TIP Inbound peering rules support POSIX regular expressions, that are different from PCRE regular expressions. So, for instance, an expression like `^3910[0-9]{5}$` can be written as `^3910\d{5}$` in PCRE and `^3910{5}$` in POSIX. The kind of regexp used depends on the underlying technology that uses that expression. Since ranges such as [0-9] are always correct, we suggest using that syntax.

When all settings for a peering group are done the details of the group look like:

 Copyright © Sipwise GmbH - All rights reserved

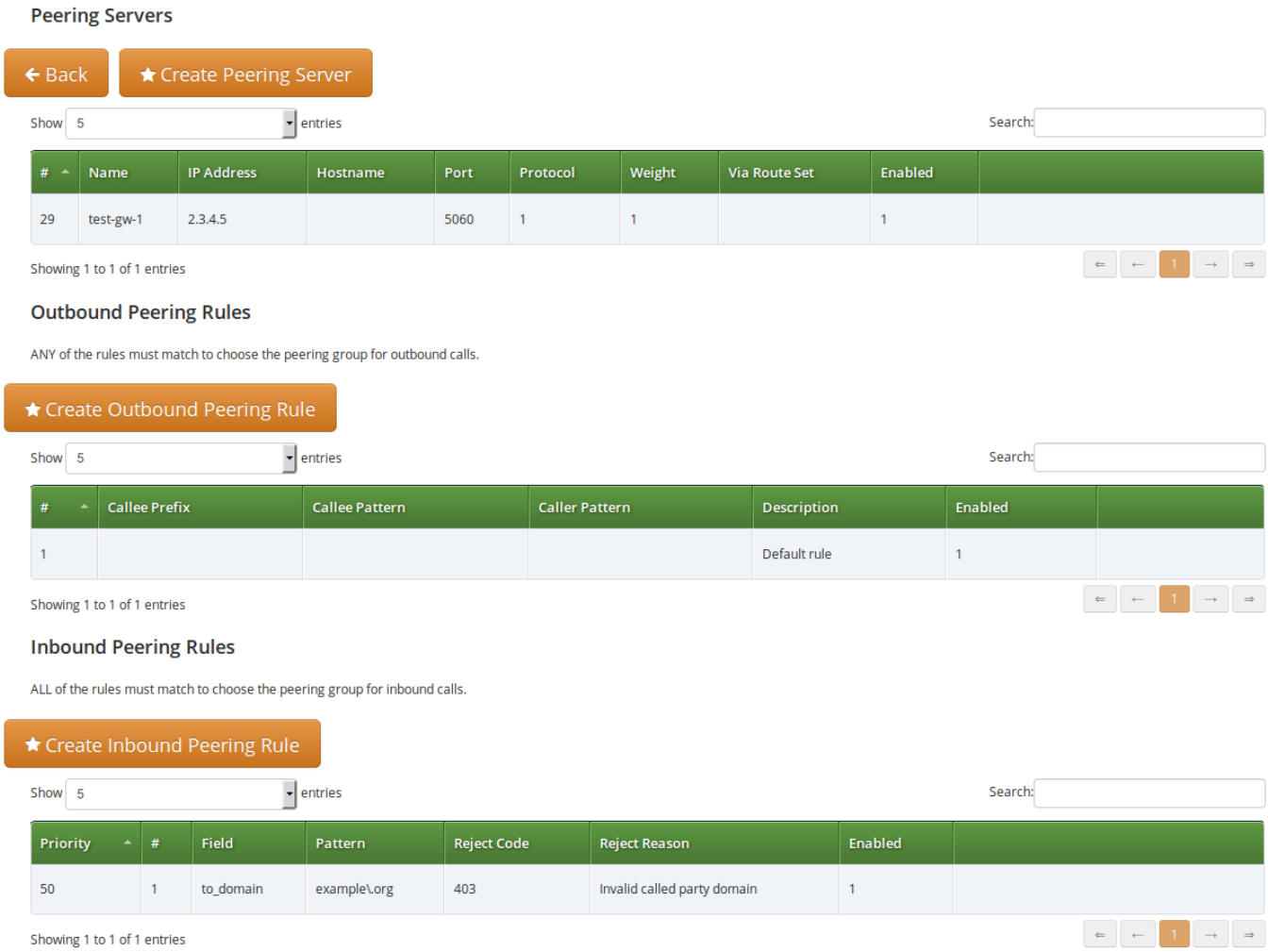


Figure 17. Peering Servers Overview

Routing Order Selection

The selection of peering groups and peering servers for outgoing calls is done in the following way:

1. All peering groups that meet the following criteria configured in the outbound peering rule are added to the list of routes for a particular call:

   Callee's username matches *callee prefix*

   Callee's URI matches *callee pattern*

   Caller's URI matches *caller pattern*

2. When all matching peering groups are selected, they are ordered by *callee prefix* according to the **longest match basis** (sometimes referred to as the **longest pattern match** or **maximum pattern length match**). One or more peering group with longest *callee prefix* match will be given first positions on the list of routes.

3. Peering groups with the same *callee prefix* length are further ordered by ***Priority***. Peering group(s) with the higher priorities will occupy higher positions.

IMPORTANT	Priority **1** gives the *highest* precedence to the corresponding peering group. Hence, a lower priority value will put the peering group higher in the list of routes (compared to other peering groups with the same *callee prefix* length).

Copyright © Sipwise GmbH - All rights reserved

Priority can be selected from **1** (highest) to **9** (lowest).

4. All peering servers in the peering group with the highest priority (e.g. priority **1**) are tried one-by-one starting from the highest server weight. Peering groups with lower priorities or with shorter *callee prefix* will be used only for fail-over.

   The ***weight*** of the peering servers in the selected peering group will influence the order in which the servers within the group will be tried for routing the outbound call. The weight of a server can be set in the range from **1** to **127**.

IMPORTANT	Opposite to the peering group priority, a peering server with a higher weight value has a *higher* precedence, but the server weight rather sets a probability than a strict order. E.g. although a peering server with weight **127** has the highest chance to be the first in the list of routes, another server with a lower weight (e.g. **100**) sometimes will be selected first.

In order to find out this probability knowing the weights of peering servers, use the following script:

Copyright © Sipwise GmbH - All rights reserved

```
#!/usr/bin/perl

#This script can be used to find out actual probabilities
#that correspond to a list of peering weights.

$num_args = $#ARGV + 1;
if ($num_args < 1) {
    print "Usage: lcr_weight_test.pl <list of weights (integers 1-
254)>\n";
    exit 0;
}

my $iters = 10000;
my @rands;

for (my $i=1; $i <= $iters; $i++) {
    my %elem;
    for (my $j=0; $j < $num_args; $j++) {
        my $random = int(rand(2000000000));
        $elem{"$j"} = $ARGV[$j] * $random;
    }
    push(@rands, \%elem);
}

my @counts;
for (my $j=0; $j < $num_args; $j++) {
    $counts["$j"] = 0;
}

foreach my $rand (@rands) {
    my $higher = 0;
    my $higher_key = 0;
    foreach $key (keys %{$rand}) {
        if ($rand->{$key} > $higher) {
            $higher = $rand->{$key};
            $higher_key = $key;
        }
    }
    $counts[$higher_key]++;
}

for (my $j=0; $j < $num_args; $j++) {
    my $prob = $counts[$j]/$iters;
    print "Peer with weight $ARGV[$j] has probability $prob \n";
}
```

Let us say you have 2 peering servers, one with weight 1 and another with weight 2. At the end—running the script as below—you will have the following traffic distribution:

Copyright © Sipwise GmbH - All rights reserved

```
# lcr_weight_test.pl 1 2

 Peer with weight 1 has probability 0.2522
 Peer with weight 2 has probability 0.7478
```

If a peering server replies with SIP codes 408, 500 or 503, or if a peering server doesn't respond at all, the next peering server in the current peering group is tried as a fallback. All the servers within the group are tried one after another until the call succeeds. If no more servers are left in the current peering group, the next group which matches the outbound peering rules is used.

NOTE The Sipwise C5 may use a slightly different approach in selecting the appropriate peering server if the *peer probing* feature is enabled. See the details in Peer Probing of the handbook.

Least Cost Routing (LCR) Configuration

The default call routing uses statically configured peering group priorities to decide where to send the calls. This solution is useful when you have an external SBC that makes all the routing decisions and is described in the Routing Order Selection section. The Sipwise C5 also allows you routing calls to the cheapest SIP peers saving your termination cost.

To enable LCR routing, do the following:

- Upload the billing fees provided by your peers to the corresponding peering billing profiles
- Enable the LCR module in config.yml (kamailio.proxy.perform_peer_lcr: yes)

When the LCR routing is enabled, the selection of peering groups would be the following:

1. All peering groups that meet the following criteria configured in the outbound peering rule are added to the list of routes for a particular call (for pure LCR you might want to omit these filters leaving them blank):

   Callee's username matches *callee prefix*

   Callee's URI matches *callee pattern*

   Caller's URI matches *caller pattern*
2. When all matching peering groups are selected, the longest matching *callee prefix* is selected from each of them. And the peering groups are *temporary* ordered according to the longest matching prefix and priority.
3. Then, the LCR module re-orders the peering groups starting from the lowest termination cost to the highest (ignoring the prefix length and peering group priorities).
4. The platform will first route the call to the servers of the first peering group in this list. If no peering server can terminate the call, the call would fail-over to the second peering group from the list and so on.

NOTE The peering servers in every peering group are sorted and tried according to their weight as described in the previous section.

Let us consider a short example. There are two peering groups (PG1 and PG2) that can deliver calls to

Copyright © Sipwise GmbH - All rights reserved

New York (e.g. 12121234567) and they have the following rates:

Peering Group	Prefix	Cost	Description
PG1	1	0.02	USA & Canada
PG2	1	0.05	USA & Canada
	1212	0.03	New York, USA

PG1 has only one rate that matches the dialed number, so that it will be taken into account, PG2 has two rates and the longest will be selected. The call will be routed to PG1 servers first as it has a cheaper price and can fail-over to PG2 servers.

The Sipwise C5 LCR feature together with the codec filtering, media transcoding, header manipulations, SIP, and RTP encryption and other SBC features make an external SBC unnecessary. This simplifies your VoIP network and cuts deployment and operation costs.

5.6.3. Authenticating and Registering against Peering Servers

Proxy-Authentication for outbound calls

If a peering server requires Sipwise C5 to authenticate for outbound calls (by sending a 407 as response to an INVITE), then you have to configure the authentication details in the *Preferences* view of your peer host.

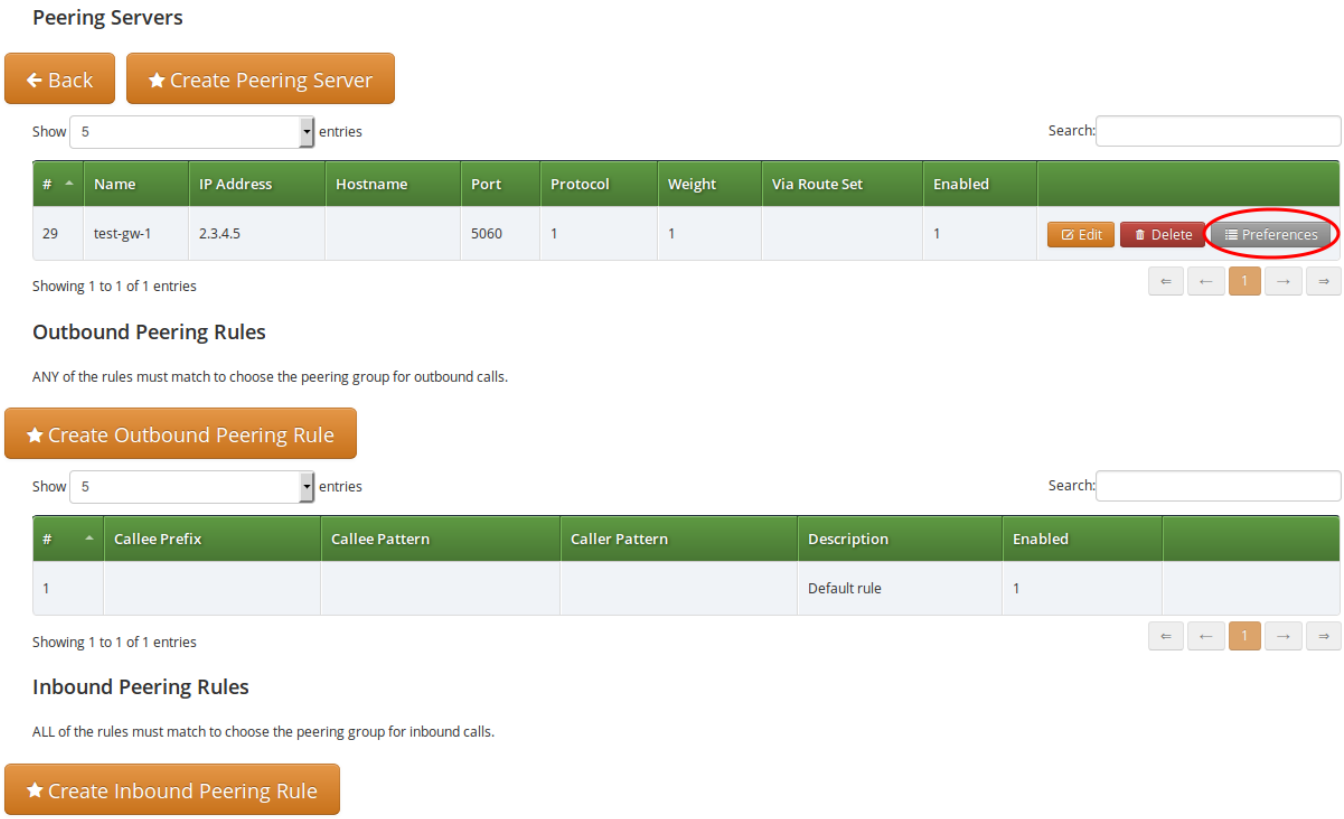


Figure 18. Select Peering Server Preferences

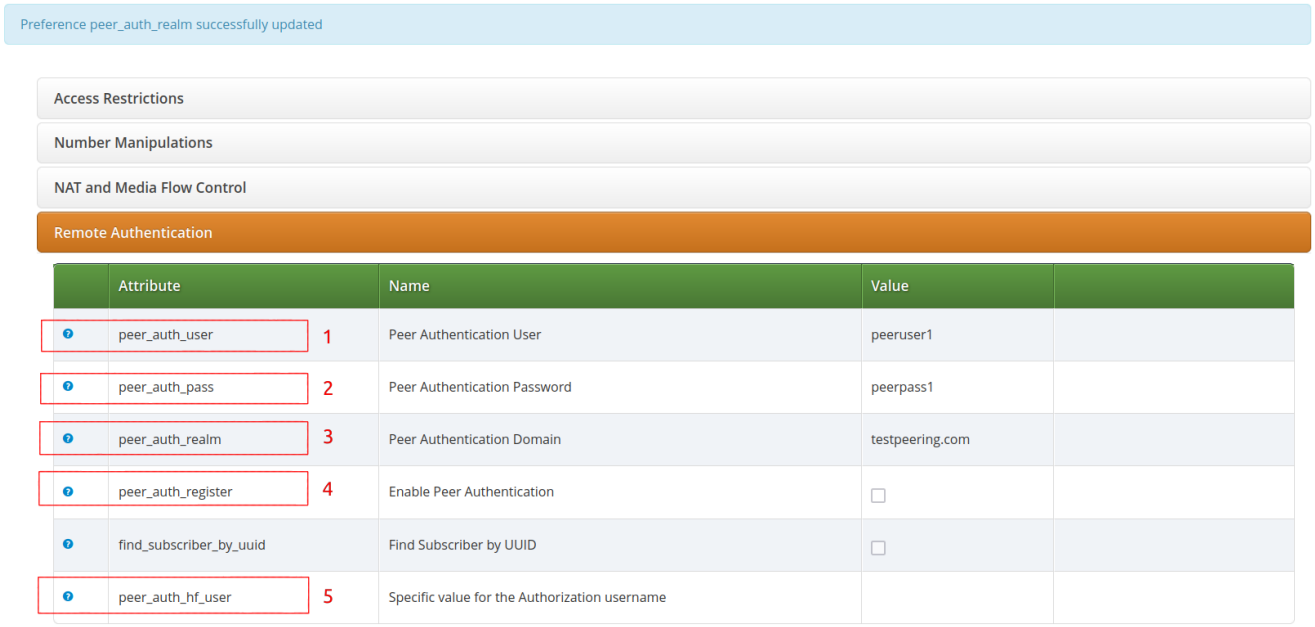
To configure this setting, open the *Remote Authentication* tab and edit the following four preferences:

Copyright © Sipwise GmbH - All rights reserved

- **peer_auth_user:** <username for peer auth>
- **peer_auth_pass:** <password for peer auth>
- **peer_auth_realm:** <domain for peer auth>
- **peer_auth_register:** <enable or disable an outbound registration for the peering>
- **peer_auth_hf_user:** <username parameter for Authorization hf>

NOTE 'peer_auth_hf_user' preference is optional and can be skipped. It allows you to set a specific username for the Proxy-Authorization header.

IMPORTANT Before the 10.1 version Sipwise C5 used to only have the */etc/ngcp-config/templates/etc/sems-b2b/etc/reg_agent.conf.tt2* configuration file to provide outbound registrations for SIP peering(s). Beginning from 10.1 you can choose, whether you want to initiate an outbound registration for your SIP peering using the web administration interface or using the */etc/ngcp-config/templates/etc/sems-b2b/etc/reg_agent.conf.tt2* configuration file. The recommended way however is to create registrations using the web administration interface. Keep in mind, in case you want to use the web administration interface for that, then you have to remove similar configuration entries from the */etc/ngcp-config/templates/etc/sems-b2b/etc/reg_agent.conf.tt2*, so that the web administration interface does not duplicate the same registration information (this can lead to several registration sessions created for the same SIP peering host, which is not desired).



 Copyright © Sipwise GmbH - All rights reserved

IMPORTANT

If you do NOT authenticate against a peer host, then the caller CLI is put into the From and P-Asserted-Identity headers, e.g. `"+4312345" <sip:+4312345@your-domain.com>`. If you DO authenticate, then the From header is `"+4312345" <sip:your_peer_auth_user@your_peer_auth_realm>` (the CLI is in the Display field, the peer_auth_user in the From username and the peer_auth_realm in the From domain), and the P-Asserted-Identity header is as usual like `<sip:+4312345@your-domain.com>`. So for presenting the correct CLI in *CLIP no screening* scenarios, your peering provider needs to extract the correct user either from the From Display-Name or from the P-Asserted-Identity URI-User.

TIP

If **peer_auth_realm** is set, the system may overwrite the Request-URI with the peer_auth_realm value of the peer when sending the call to that peer or peer_auth_realm value of the subscriber when sending a call to the subscriber. Since this is rarely a desired behavior, it is disabled by default starting with Sipwise C5 release 3.2. If you need the replacement, you should set *set_ruri_to_peer_auth_realm: 'yes'* in */etc/ngcp-config/config.yml*.

Registering at a Peering Server

A registration process for the SIP peering is quite simple, but since currently this functionality is present both in the web administration interface and in the */etc/ngcp-config/templates/etc/sems-b2b/etc/reg_agent.conf.tt2* (the last one is an inheritance and will be deprecated in the future), it's worth to dedicate a separate section with an additional description. So that it remains clear for the user of the Sipwise C5 how to properly handle that.

The first and a recommended way to do that - is the web administration interface. It's simple as that, you need to tick a check-box 'peer_auth_register', and if previously all needed authentication data has been provided (user, password and realm) the registration process will start right away.

The second way is a manual edition of the dedicated configuration file.

Create a new file */etc/ngcp-config/templates/etc/sems-b2b/etc/reg_agent.conf.customtt.tt2* as a copy of */etc/ngcp-config/templates/etc/sems-b2b/etc/reg_agent.conf.tt2*. Configure your peering servers with the corresponding credentials in the new created file */etc/ngcp-config/templates/etc/sems-b2b/etc/reg_agent.conf.customtt.tt2*, then execute *ngcpcfg apply "added upstream credentials"*.

IMPORTANT

Be aware that this will force SEMS to restart, which will drop running conference calls.

NOTE

Sipwise C5 supports outbound registrations via UDP, TCP and TLS. Please see */etc/ngcp-config/templates/etc/sems-b2b/etc/reg_agent.conf.tt2* to see examples how to enable it using needed transport.

Copyright © Sipwise GmbH - All rights reserved

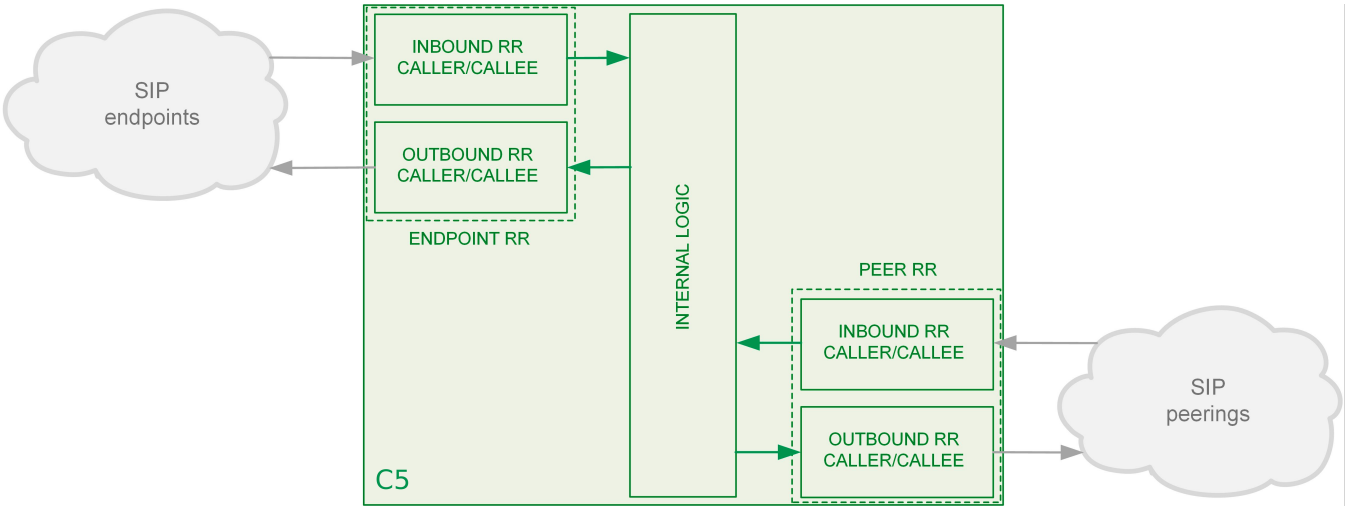
NOTE By default TCP/IP stack implementation in Linux kernel uses so called ephemeral ports for TCP transport, when it comes to originating a brand new TCP session towards remote side. This is usually in the range of 32768 – 60999. This also applies to TLS. However Sipwise C5 can and will always reuse already existing TCP sessions with subscribers, in order to send them out-of-dialog requests (for e.g. INVITE). It works so, because subscribers which register at Sipwise C5, initiate and constantly support a TCP session with Sipwise C5 (either with TCP keepalive mechanisms, or constantly sending new re-registrations or/and OPTIONS).

TIP You can force the Load-Balancer to use a fixed port for sending outbound registrations from your Sipwise C5 platform, by enabling option 'tcp_reuse_port' (config.yml kamailio.lb.tcp_reuse_port: yes). This will force the Load-Balancer to use the same socket descriptor(s), for establishing outbound sessions, as used for listening (this only relates to TCP and TLS). With UDP you can by default initiate sessions from Sipwise C5 using a constant port (usually 5060). Remember enabling 'tcp_reuse_port' will force all sessions (not only REGISTER) initiated from behalf of Sipwise C5 be established over local port engaged for listening (TCP or TLS).

TIP There is a possibility to define a specific value for the "username" parameter of the Authorization header, in case you want to have another username for the Digest process, than the one used in From/To headers. In order to do that, you have to define the option 'auth_user' for a desired registration entity. It's being defined separately for each registration entity.

5.7. Configuring Rewrite Rule Sets

On the NGCP, every phone number is treated in E.164 format *<country code><area code><subscriber number>*. Rewrite Rule Sets is a flexible tool to translate the caller and callee numbers to the proper format before the routing lookup and after the routing lookup separately. The created Rewrite Rule Sets can be assigned to the domains, subscribers and peers as a preference. Here below you can see how the Rewrite Rules are used by the system:



As from the image above, following the arrows, you will have an idea about which type of Rewrite Rules are applied during a call. In general:

 Copyright © Sipwise GmbH - All rights reserved

- Call from local subscriber A to local subscriber B: Inbound RR from local Domain/Subscriber A and Outbound Rewrite Rules from local Domain/Subscriber B.
- Call from local subscriber A to the peer: Inbound RR from local Domain/Subscriber A and Outbound Rewrite Rules from the peer.
- Call from peer to local subscriber B: Inbound RR from the Peer and Outbound Rewrite Rules from local Domain/Subscriber B.

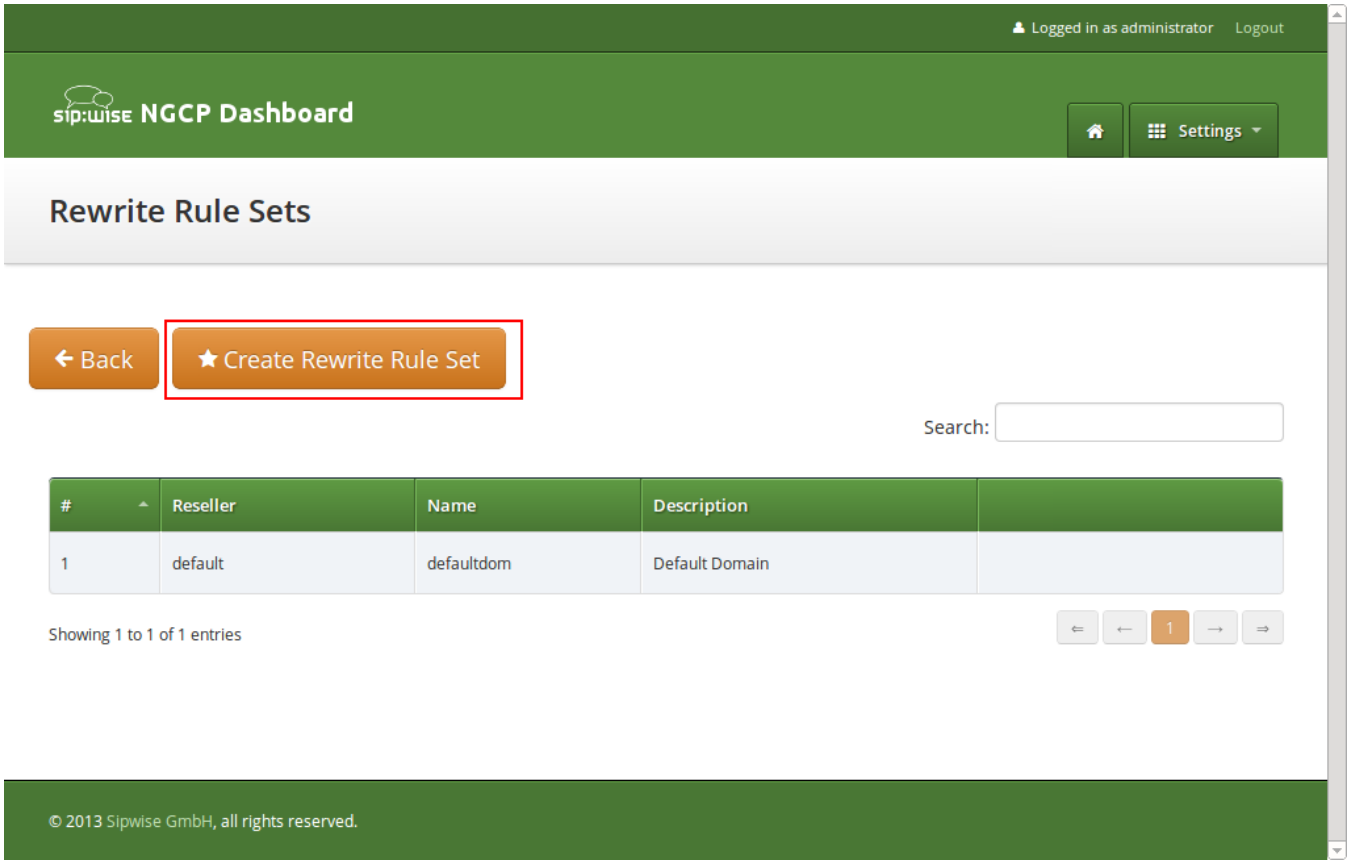
You would normally begin with creating a Rewrite Rule Set for your SIP domains. This is used to control what an end user can dial for outbound calls, and what is displayed as the calling party on inbound calls. The subscribers within a domain inherit Rewrite Rule Sets of that domain, unless this is overridden by a subscriber Rewrite Rule Set preference.

You can use several special variables in the Rewrite Rules, below you can find a list of them. Some examples of how to use them are also provided in the following sections:

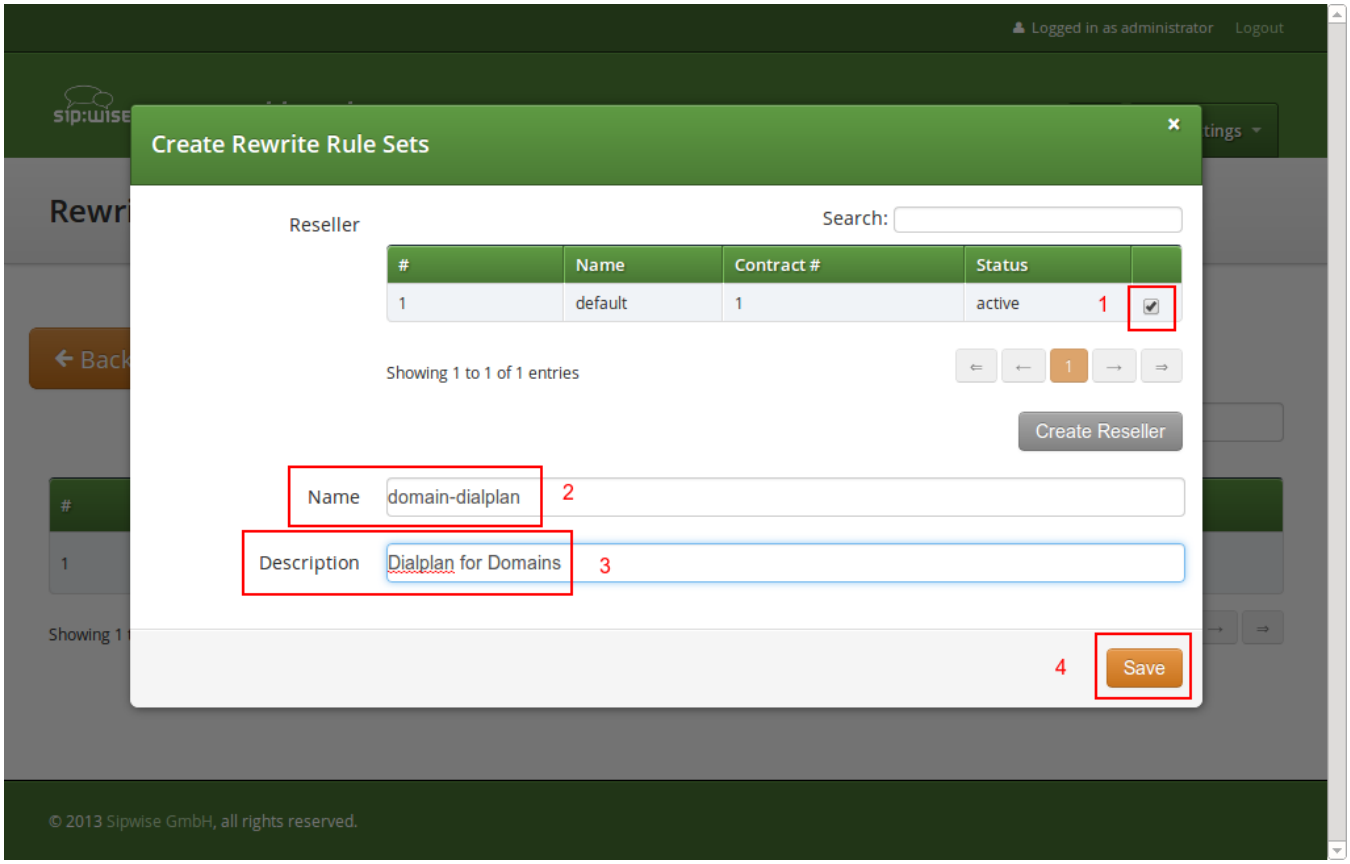
- `${caller_cc}` : This is the value taken from the subscriber's preference CC value under Number Manipulation
- `${caller_ac}` : This is the value taken from the subscriber's preference AC value under Number Manipulation
- `${caller_emergency_cli}` : This is the value taken from the subscriber's preference emergency_cli value under Number Manipulation
- `${caller_emergency_prefix}` : This is the value taken from the subscriber's preference emergency_prefix value under Number Manipulation
- `${caller_emergency_suffix}` : This is the value taken from the subscriber's preference emergency_suffix value under Number Manipulation
- `${caller_cloud_pbx_base_cli}` : This is the value taken from the *Primary Number* field from section *Details Master Data* of the *Pilot Subscriber* for a particular PBX customer.

To create a new Rewrite Rule Set, go to *SettingsRewrite Rule Sets*. There you can create a Set identified by a name. This name is later shown in your peer-, domain- and user-preferences where you can select the rule set you want to use.

Copyright © Sipwise GmbH - All rights reserved



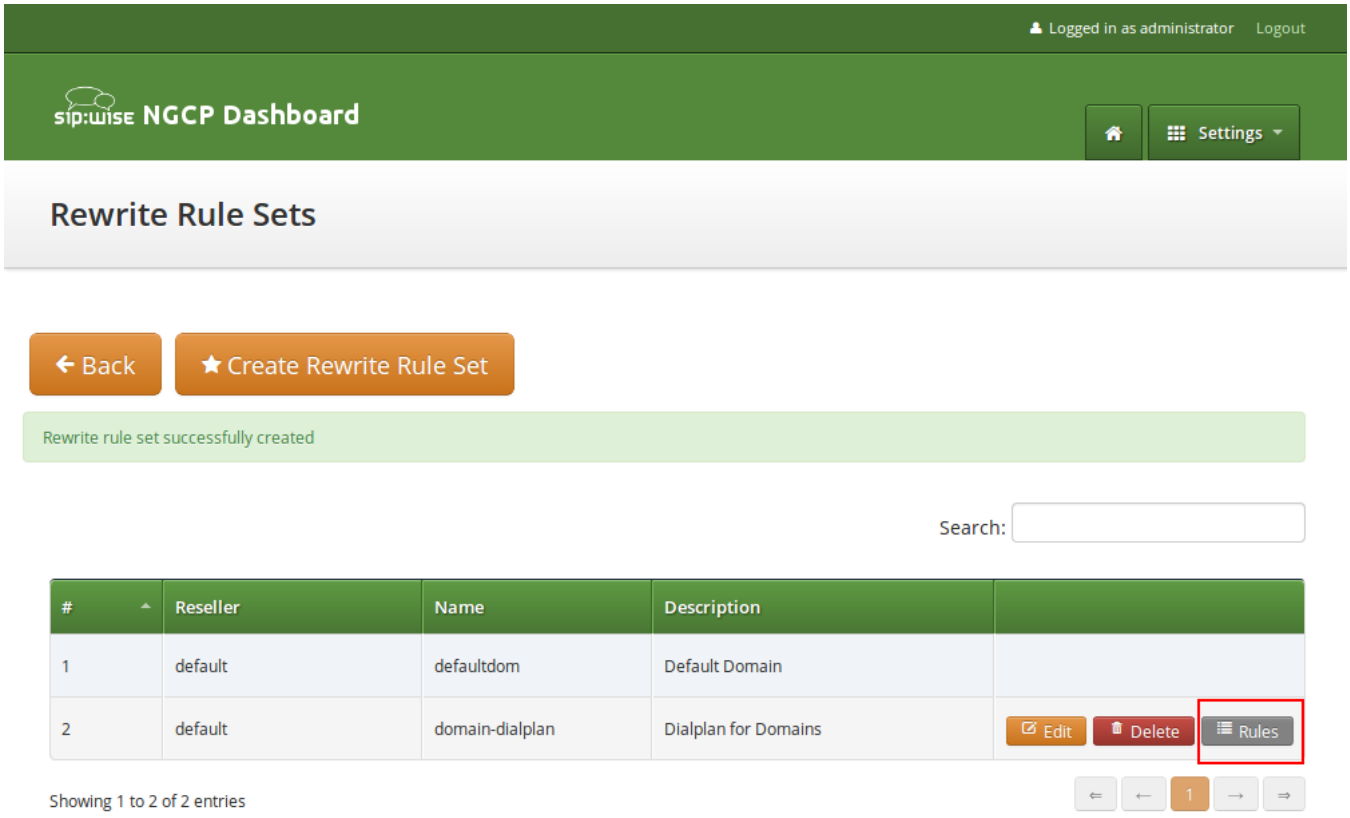
Click *Create Rewrite Rule Set* and fill in the form accordingly.



Press the *Save* button to create the set.

Copyright © Sipwise GmbH - All rights reserved

To view the *Rewrite Rules* within a set, hover over the row and click the *Rules* button.



The rules are ordered by *Caller* and *Callee* as well as direction *Inbound* and *Outbound*.

TIP In Europe, the following formats are widely accepted: *+<cc><ac><sn>*, *00<cc><ac><sn>* and *0<ac><sn>*. Also, some countries allow the areacode-internal calls where only subscriber number is dialed to reach another number in the same area. Within this section, we will use these formats to show how to use rewrite rules to normalize and denormalize number formats.

5.7.1. Inbound Rewrite Rules for Caller

These rules are used to normalize user-provided numbers (e.g. passed in *From Display Name* or *P-Preferred-Identity* headers) into E.164 format. In our example, we'll normalize the three different formats mentioned above into E.164 format.

To create the following rules, click on the *Create Rewrite Rule* for each of them and fill them with the values provided below.

Strip leading `00` or `+`

- Match Pattern: `^(00|\+)([1-9][0-9]+)$`
- Replacement Pattern: `\2`
- Description: `International to E.164`
- Direction: `Inbound`

Copyright © Sipwise GmbH - All rights reserved

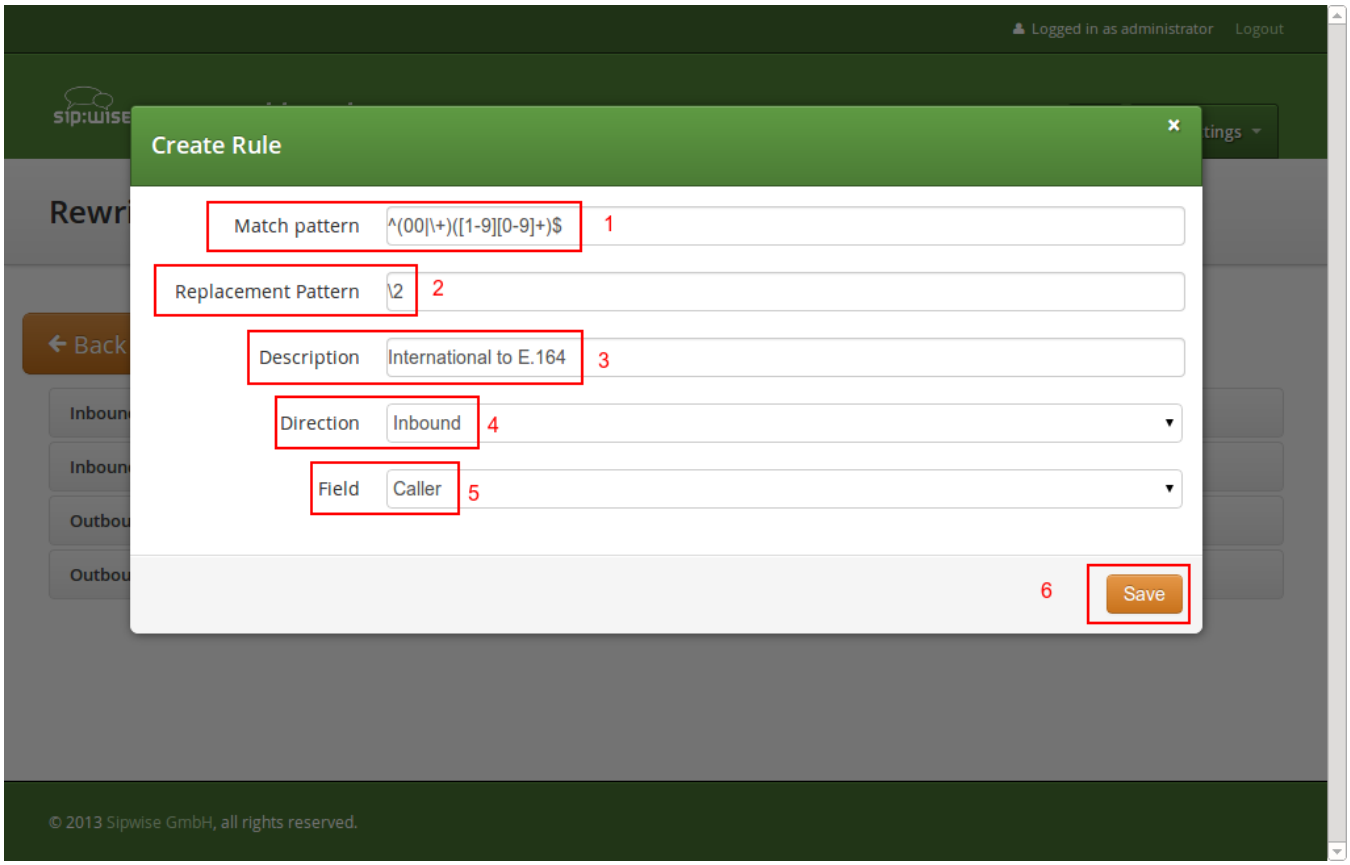
- Field: `Caller`

Replace `0` by caller's country code:

- Match Pattern: `^0([1-9][0-9]+)$`
- Replacement Pattern: `${caller_cc}\1`
- Description: `National to E.164`
- Direction: `Inbound`
- Field: `Caller`

Normalize local calls:

- Match Pattern: `^([1-9][0-9]+)$`
- Replacement Pattern: `${caller_cc}${caller_ac}\1`
- Description: `Local to E.164`
- Direction: `Inbound`
- Field: `Caller`

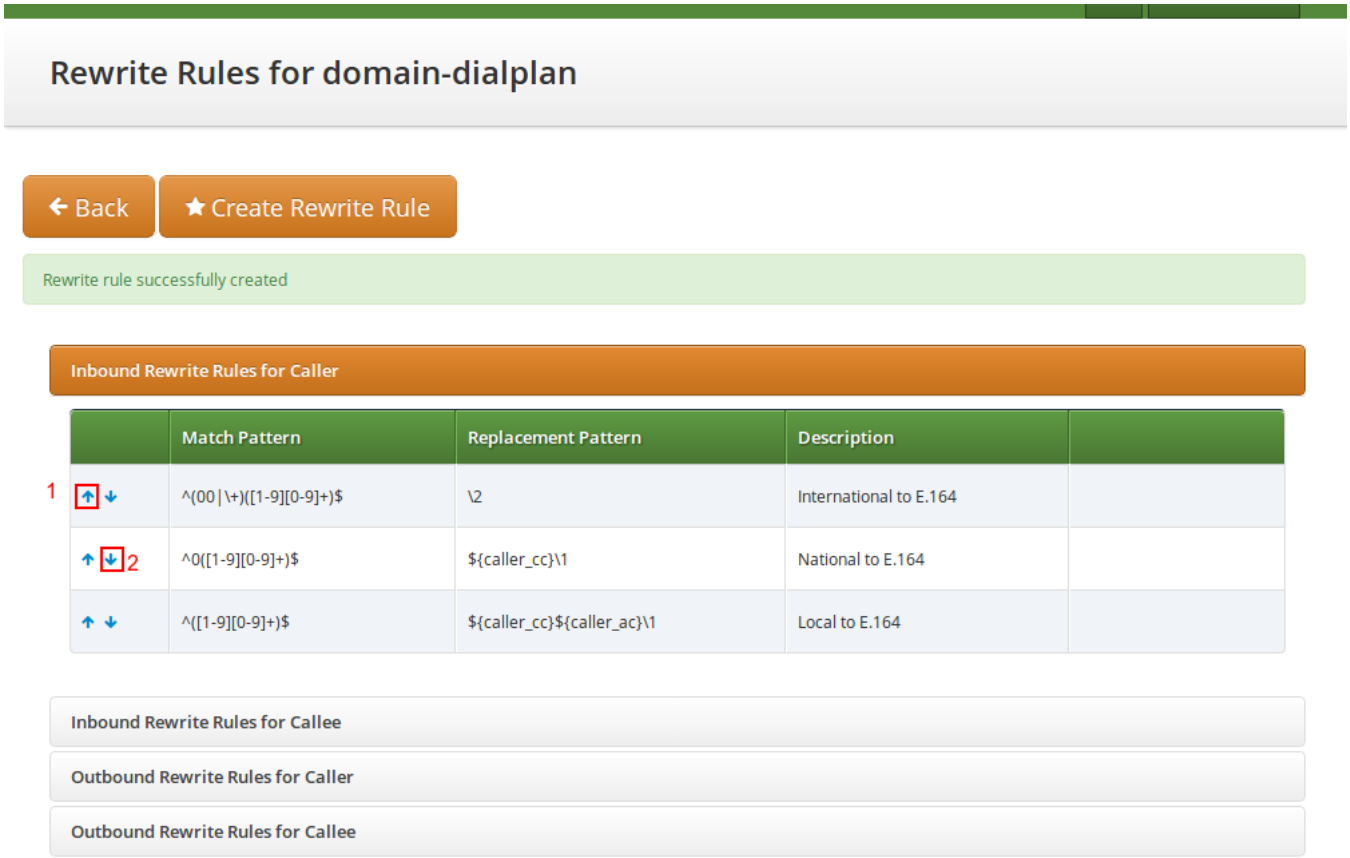


Normalization for national and local calls is possible with special variables `${caller_cc}` and `${caller_ac}` that can be used in Replacement Pattern and are substituted by the country and area code accordingly during the call routing.

Copyright © Sipwise GmbH - All rights reserved

IMPORTANT These variables are only being filled in when a call originates from a subscriber (because only then the cc/ac information is known by the system), so you can not use them when a calls comes from a SIP peer (the variables will be empty in this case).

TIP When routing a call, the rewrite processing is stopped after the first match of a rule, starting from top to bottom. If you have two rules (e.g. a generic one and a more specific one), where both of them would match some numbers, reorder them with the up/down arrows into the appropriate position.



5.7.2. Inbound Rewrite Rules for Callee

These rules are used to rewrite the number the end user dials to place a call to a standard format for routing lookup. In our example, we again allow the three different formats mentioned above and again normalize them to E.164, so we put in the same rules as for the caller.

Strip leading `00` or `+`

- Match Pattern: `^(00|\+)([1-9][0-9]+)$`
- Replacement Pattern: `\2`
- Description: `International to E.164`
- Direction: `Inbound`
- Field: `Callee`

Copyright © Sipwise GmbH - All rights reserved

Replace `0` by caller's country code:

- Match Pattern: `^0([1-9][0-9]+)$`
- Replacement Pattern: `${caller_cc}\1`
- Description: `National to E.164`
- Direction: `Inbound`
- Field: `Callee`

Normalize areacode-internal calls:

- Match Pattern: `^([1-9][0-9]+)$`
- Replacement Pattern: `${caller_cc}${caller_ac}\1`
- Description: `Local to E.164`
- Direction: `Inbound`
- Field: `Callee`

TIP Our provided rules will only match if the caller dials a numeric number. If he dials an alphanumeric SIP URI, none of our rules will match and no rewriting will be done. You can however define rules for that as well. For example, you could allow your end users to dial support and rewrite that to your support hotline using the match pattern `^support$` and the replace pattern `43800999000` or whatever your support hotline number is.

5.7.3. Outbound Rewrite Rules for Caller

These rules are used to rewrite the calling party number for a call to an end user. For example, if you want the device of your end user to show *0<ac><sn>* if a national number calls this user, and *00<cc><ac><sn>* if an international number calls, put the following rules there.

Replace Austrian country code `43` by `0`

- Match Pattern: `^43([1-9][0-9]+)$`
- Replacement Pattern: `0\1`
- Description: `E.164 to Austria National`
- Direction: `Outbound`
- Field: `Caller`

Prefix `00` for international caller

- Match Pattern: `^([1-9][0-9]+)$`
- Replacement Pattern: `00\1`
- Description: `E.164 to International`
- Direction: `Outbound`
- Field: `Caller`

 Copyright © Sipwise GmbH - All rights reserved

TIP Note that both of the rules would match a number starting with `43`, so reorder the national rule to be above the international one (if it's not already the case).

5.7.4. Outbound Rewrite Rules for Callee

These rules are used to rewrite the called party number immediately before sending out the call on the network. This gives you an extra flexibility by controlling the way request appears on a wire, when your SBC or other device expects the called party number to have a particular tech-prefix. It can be used on calls to end users too if you want to do some processing in intermediate SIP device, e.g. apply legal intercept selectively to some subscribers.

Prefix `sipsp#` for all calls

- Match Pattern: `^([0-9]+)$`
- Replacement Pattern: `sipsp#\1`
- Description: `Intercept this call`
- Direction: `Outbound`
- Field: `Callee`

5.7.5. Emergency Number Handling

There are 2 ways to handle calls from local subscribers to emergency numbers in NGCP:

- *Simple* emergency number handling: inbound rewrite rules append an emergency tag to the called number, this will be recognised by NGCP's call routing logic and the call is routed directly to a peer. Please read the next section for details of simple emergency number handling.
- An emergency *number mapping* is applied: a dedicated emergency number mapping database is consulted in order to obtain the most appropriate routing number of emergency services. This logic ensures that the caller will contact the geographically closest emergency service. Please visit the Emergency Mapping section of the handbook for more details.

NOTE If Sipwise C5 detects that the call is of an emergency type, then by default it will not apply concurrent calls limitation defined by your license. In order to change this behavior globally, you can set the config.yml option 'b2b.sbc.skip_cpslimit_license_check_emergency' to 'no' (which is by default set to 'yes'). Remember, by applying the changes the SEMS-B2B component will have to restart.

Simple Emergency Number Handling Overview

The overview of emergency call processing is as follows:

Copyright © Sipwise GmbH - All rights reserved

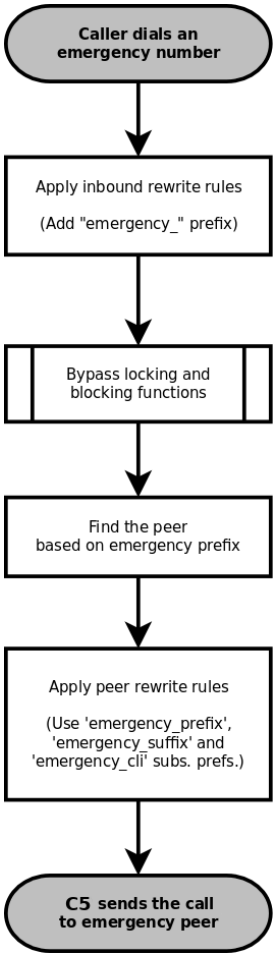


Figure 19. Simple Emergency Call Handling

Configuring Emergency Numbers is also done via Rewrite Rules.

Tagging Inbound Emergency Calls

For Emergency Calls from a subscriber to the platform, you need to define an *Inbound Rewrite Rule For Callee*, which adds a prefix `emergency_` to the number (and can rewrite the number completely as well at the same time). If the proxy detects a call to a SIP URI starting with `emergency_`, it will enter a special routing logic bypassing various checks which might make a normal call fail (e.g. due to locked or blocked numbers, insufficient credits or exceeding the max. amount of parallel calls).

Tag an Emergency Call

- Match Pattern: `^(911|112)$`
- Replacement Pattern: `emergency_\1`
- Description: `Tag Emergency Numbers`
- Direction: `Inbound`
- Field: `Callee`

To route an Emergency Call to a Peer, you can select a specific peering group by adding a peering rule with a *callee prefix* set to `emergency_` to a peering group.

 Copyright © Sipwise GmbH - All rights reserved

Normalize Emergency Calls for Peers

In order to normalize the emergency number to a valid format accepted by the peer, you need to assign an *Outbound Rewrite Rule For Callee*, which strips off the `emergency_` prefix. You can also use the variables `${caller_emergency_cli}`, `${caller_emergency_prefix}` and `${caller_emergency_suffix}` as well as `${caller_ac}` and `${caller_cc}`, which are all configurable per subscriber to rewrite the number into a valid format.

Normalize Emergency Call for Peer

- Match Pattern: `^emergency_(.+)$`
- Replacement Pattern: `${caller_emergency_prefix}${caller_ac}\1`
- Description: `Normalize Emergency Numbers`
- Direction: `Outbound`
- Field: `Callee`

5.7.6. Emergency Geo-location Formats

A tagged Emergency Call from a subscriber will have Geo-location information attached to the SDP when `emergency_location_object` preference is properly set depending on the format defined at subscriber's `emergency_location_format` preference.

These are the `emergency_location_format` formats that Sipwise C5 currently supports:

- `PIDF-LO` (TR Notruf v2)
- `cirpack` (TR Notruf v1)

PIDF-LO format

`emergency_provider_info` preference must be defined at domain level with `application/xml` as content-type and the whole XML with the provider info described. For instance:

```
<?xml version="1.0" encoding="UTF-8"?>
<emergencyCall.ProviderInfo
xmlns="urn:ietf:params:xml:ns:emergencyCall.ProviderInfo">
 <DataProviderString>Telekom</DataProviderString>
 <ProviderID>D150</ProviderID>
 <contactURI>sip:+492911234567@telekom.de;user=phone</contactURI>
 <ProviderIDSeries>BNetzA</ProviderIDSeries>
</emergencyCall.ProviderInfo>
```

`emergency_location_object` preference can be defined at subscriber level with `application/pidf+xml` as content-type and the whole XML containing the Geo-location as PIDF-LO.

It can use different location encodings but there are two mandatory elements that need a special value in order to be replaced by Sipwise C5 at the moment of initiating the emergency call:

- `timestamp`

Copyright © Sipwise GmbH - All rights reserved

- retention-expiry

```
<retention-expiry>$$expiry$$</retention-expiry>
<timestamp>$$ts$$</timestamp>
```

An example of XML document with the mandatory elements:

```
<?xml version="1.0" encoding="UTF-8"?>
<presence
 xmlns="urn:ietf:params:xml:ns:pidf"
 xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
 xmlns:gml="http://www.opengis.net/gml"
 xmlns:gs="http://www.opengis.net/pidflo/1.0"
 xmlns:cl="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"
entity="pres:123@t-mobile.de">
 <tuple id="arcband">
    <status>
      <gp:geopriv>
        <gp:location-info>
          <gml:location>
            <gs:ArcBand srsName="urn:ogc:def:crs:EPSG:: 4258"
              xmlns:gs="http://www.opengis.net/pidflo/1.0"
              xmlns:gml="http://www.opengis.net/gml">
              <gml:pos>49.8967 8.6228</gml:pos>
              <gs:innerRadius
uom="urn:ogc:def:uom:EPSG::9001">0</gs:innerRadius>
              <gs:outerRadius
uom="urn:ogc:def:uom:EPSG::9001">2005</gs:outerRadius>
              <gs:startAngle
uom="urn:ogc:def:uom:EPSG::9102">328</gs:startAngle>
              <gs:openingAngle
uom="urn:ogc:def:uom:EPSG::9102">64</gs:openingAngle>
            </gs:ArcBand>
          </gml:location>
          <con:confidence>100</con:confidence>
          <cl:civicAddress xml:lang="de">
            <cl:LOC>Mobilfunkzelle</cl:LOC>
            <cl:ADDCODE>26201F1080939A</cl:ADDCODE>
          </cl:civicAddress>
        </gp:location-info>
        <gp:usage-rules>
          <gbp:retransmission-allowed>yes</gbp:retransmission-allowed>
          <gbp:retention-expiry>$$expiry$$</gbp:retention-expiry>
        </gp:usage-rules>
      </gp:geopriv>
    </status>
    <timestamp>$$ts$$</timestamp>
  </tuple>
</presence>
```

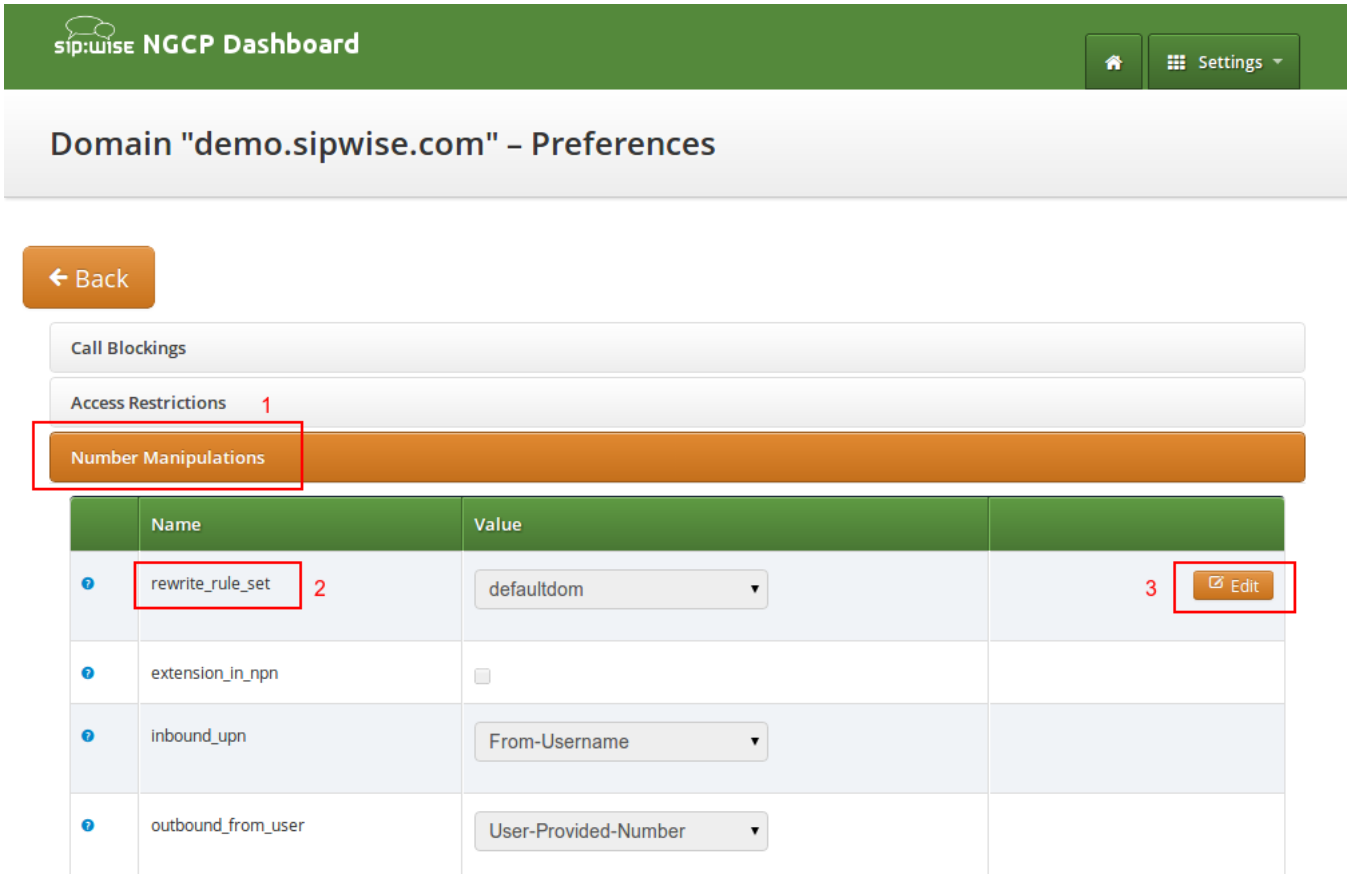

Copyright © Sipwise GmbH - All rights reserved

cirpack format

`emergency_location_object` preference can be defined at subscriber level with `application/vnd.cirpack.isdn-ext` as content-type and the hex string. For instance: `7e 0d 04 55 75 69 20 4d 61 6b 65 43 61 6c 6c`

5.7.7. Assigning Rewrite Rule Sets to Domains and Subscribers

Once you have finished to define your Rewrite Rule Sets, you need to assign them. For sets to be used for subscribers, you can assign them to their corresponding domain, which then acts as default set for all subscribers. To do so, go to *SettingsDomains* and click *Preferences* on the domain you want the set to assign to. Click on *Edit* and select the Rewrite Rule Set created before.



You can do the same in the *Preferences* of your subscribers to override the rule on a subscriber basis. That way, you can finely control down to an individual user the dial-plan to be used. Go to *Settings Subscribers*, click the *Details* button on the subscriber you want to edit, the click the *Preferences* button.

5.7.8. Creating Dialplans for Peering Servers

For each peering server, you can use one of the Rewrite Rule Sets that was created previously as explained in Configuring Rewrite Rule Sets (keep in mind that special variables `${caller_ac}` and `${caller_cc}` can not be used when the call comes from a peer). To do so, click on the name of the peering server, look for the preference called *Rewrite Rule Sets*.

If your peering servers don't send numbers in E.164 format <*cc*><*ac*><*sn*>, you need to create *Inbound Rewrite Rules* for each peering server to normalize the numbers for caller and callee to this format, e.g. by stripping leading `+` or put them from national into E.164 format.

Copyright © Sipwise GmbH - All rights reserved

Likewise, if your peering servers don't accept this format, you need to create *Outbound Rewrite Rules* for each of them, for example to append a '+' to the numbers.

5.7.9. Call Routing Verification

The Sipwise C5 provides a utility that helps with the verification of call routing among local subscribers and peers. It is called *Call Routing Verification* and employs rewrite rules and peer selection rules, in order to process calling and called numbers or SIP users and find the appropriate peer for the destination.

The *Call Routing Verification* utility performs only basic number processing and does not invoke the full number manipulation logic applied on real calls. The goal is to enable testing of rewrite rules, rather than validate the complete number processing.

- What is considered during the test:

  subscriber preferences: cli and allowed_clis

  domain / subscriber / peer rewrite rules

- What is not taken into account during the test:

  other subscriber or peer preferences

  LNP (Local Number Portability) lookup on called numbers; LNP rewrite rules

You can access the utility following the path on Admin web interface: *Tools Call Routing Verification*.

Expected input data

- Caller number/uri: 2 formats are accepted in this field:

  A simple **phone number** in international (`00431..`, `+431..`) or E.164 (`431..`) format.

  A SIP **URI** in `username@domain` format (without adding "sip:" at the beginning).

- Callee number/uri: The same applies as for Caller number/uri.
- Caller Type: Select `Subscriber` or `Peer`, depending on the source of the call.
- Caller Subscriber or Caller Peer: Optionally, you can select the subscriber or peer explicitly. Without the explicit selection, however, the *Call Routing Verification* tool is able to find the caller in the database, based on the provided number / URI.
- Caller RWR Override, Callee RWR Override, Callee Peer Override: The caller / callee rewrite rules and peer selection rules defined in domain, subscriber and peer preferences are used for call processing by default. But you can also override them by explicitly selecting another rewrite or peer selection rule.

Examples

1. Using only phone numbers and explicit subscriber selection

   Input Data:

 Copyright © Sipwise GmbH - All rights reserved

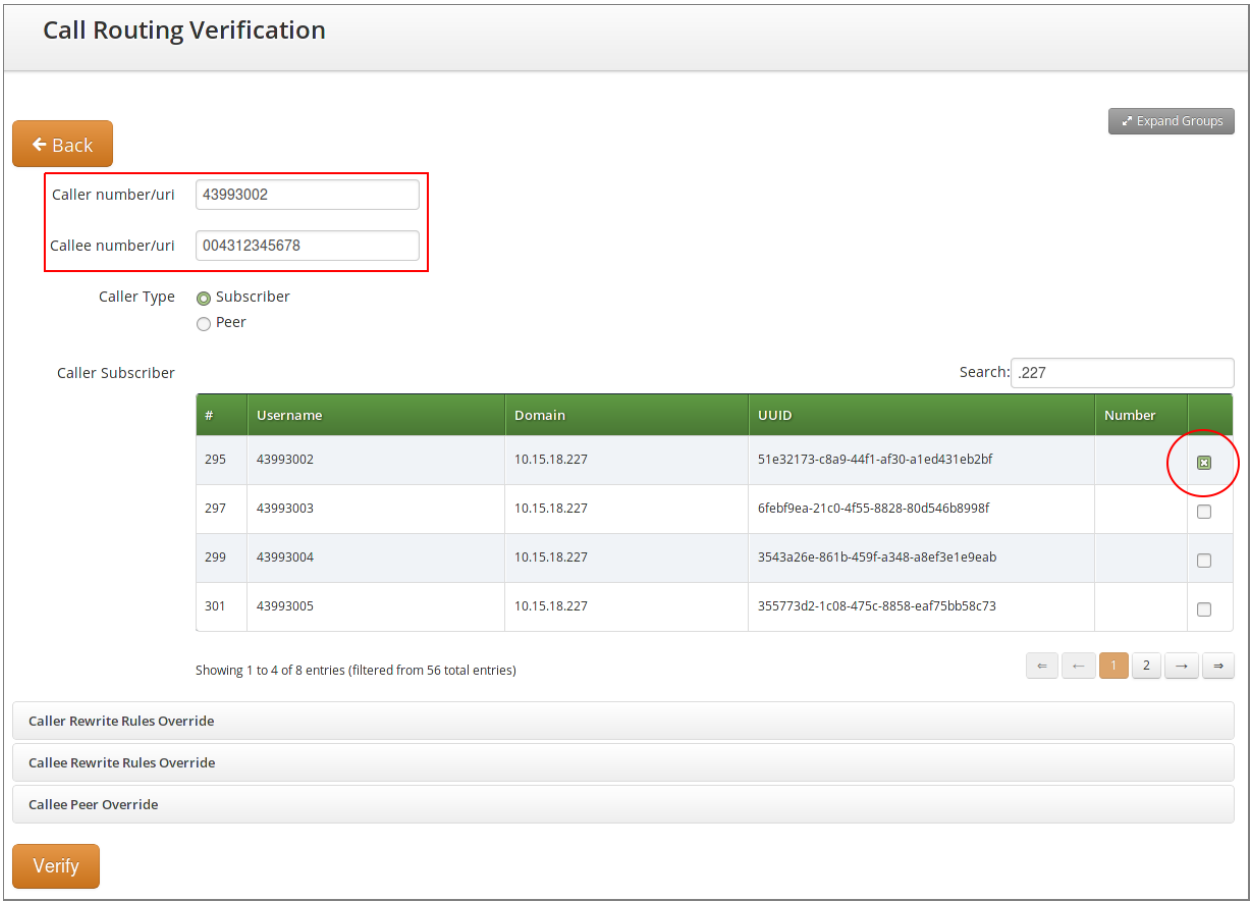


Figure 20. Call Routing Verif. - Only Numbers - Input

Result:

Copyright © Sipwise GmbH - All rights reserved

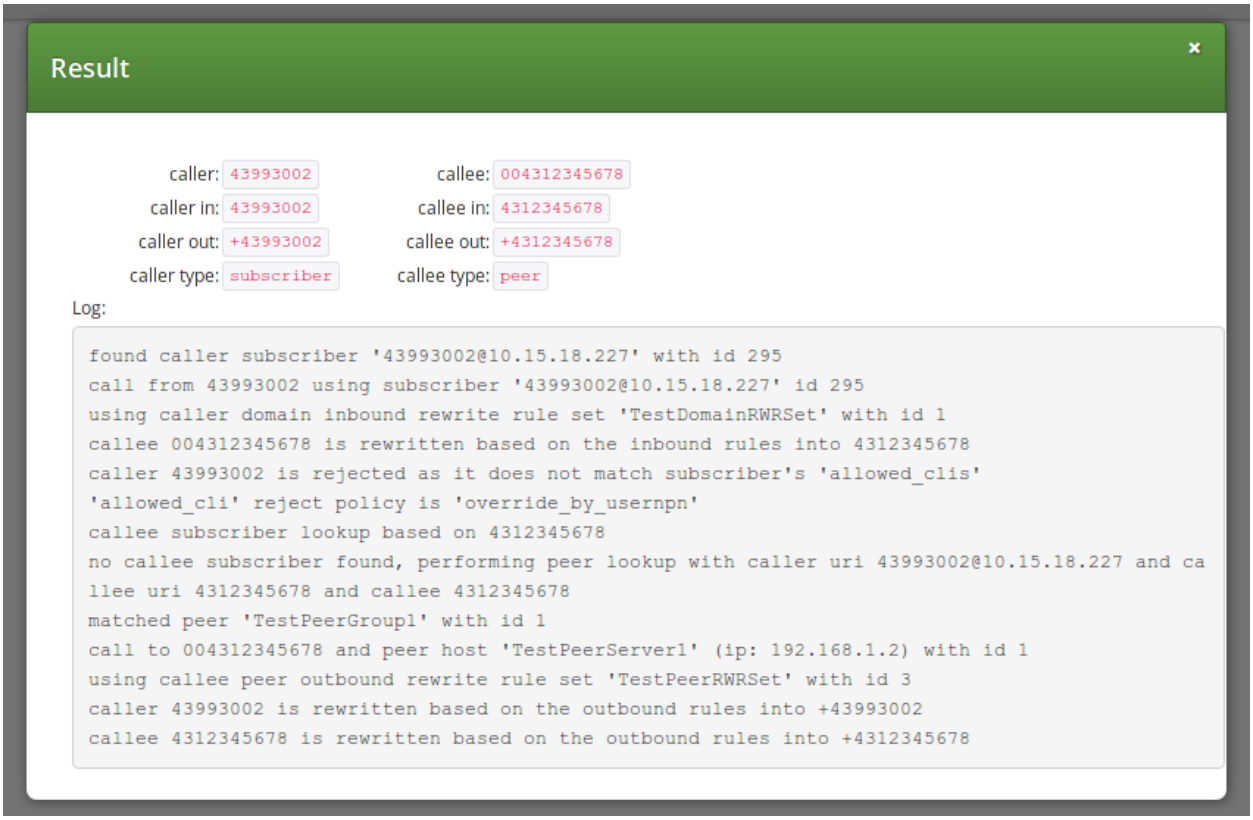


Figure 21. Call Routing Verif. - Only Numbers - Result

2. Using phone number and URI, without explicit subscriber selection

   Input Data:

 Copyright © Sipwise GmbH - All rights reserved

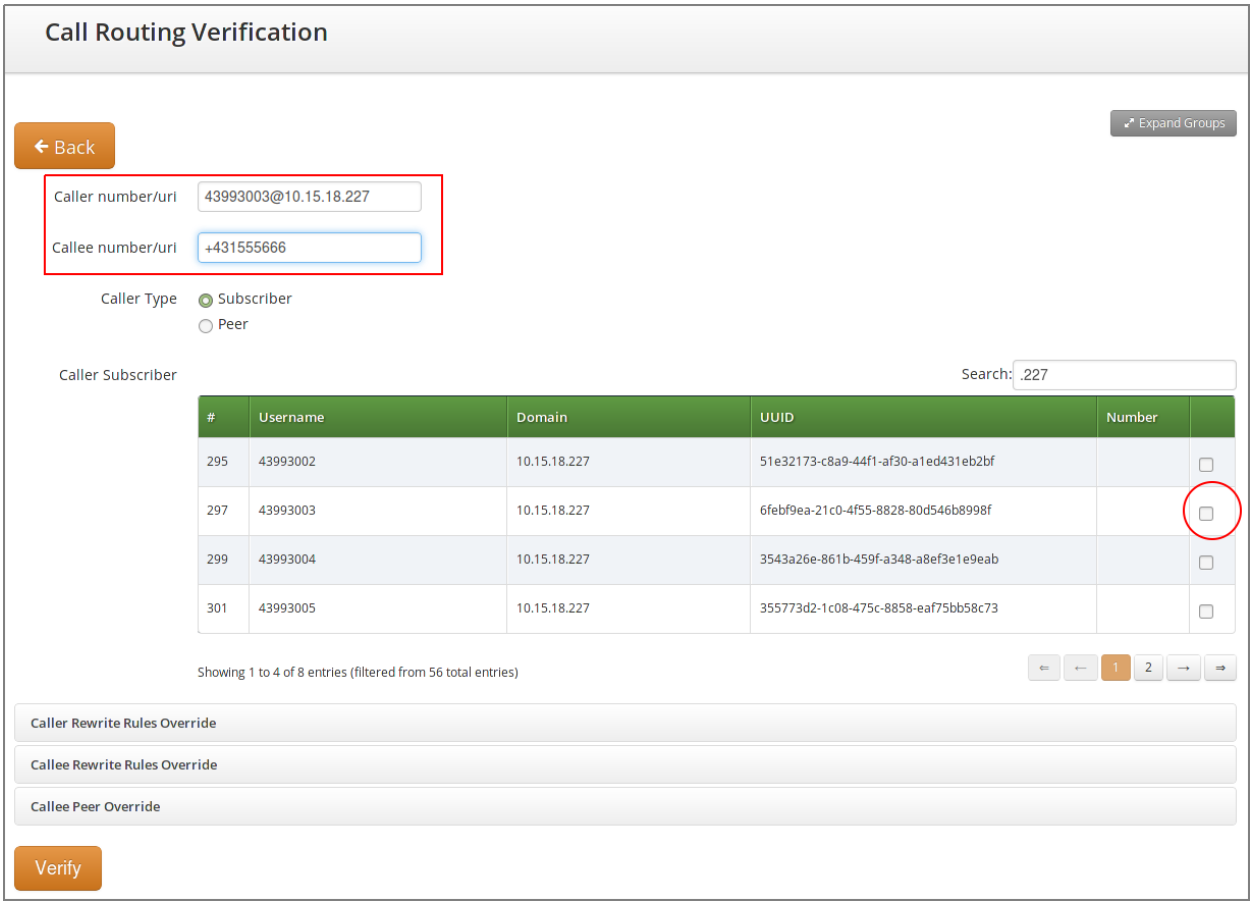


Figure 22. Call Routing Verif. - Number and URI - Input

Result:

Copyright © Sipwise GmbH - All rights reserved

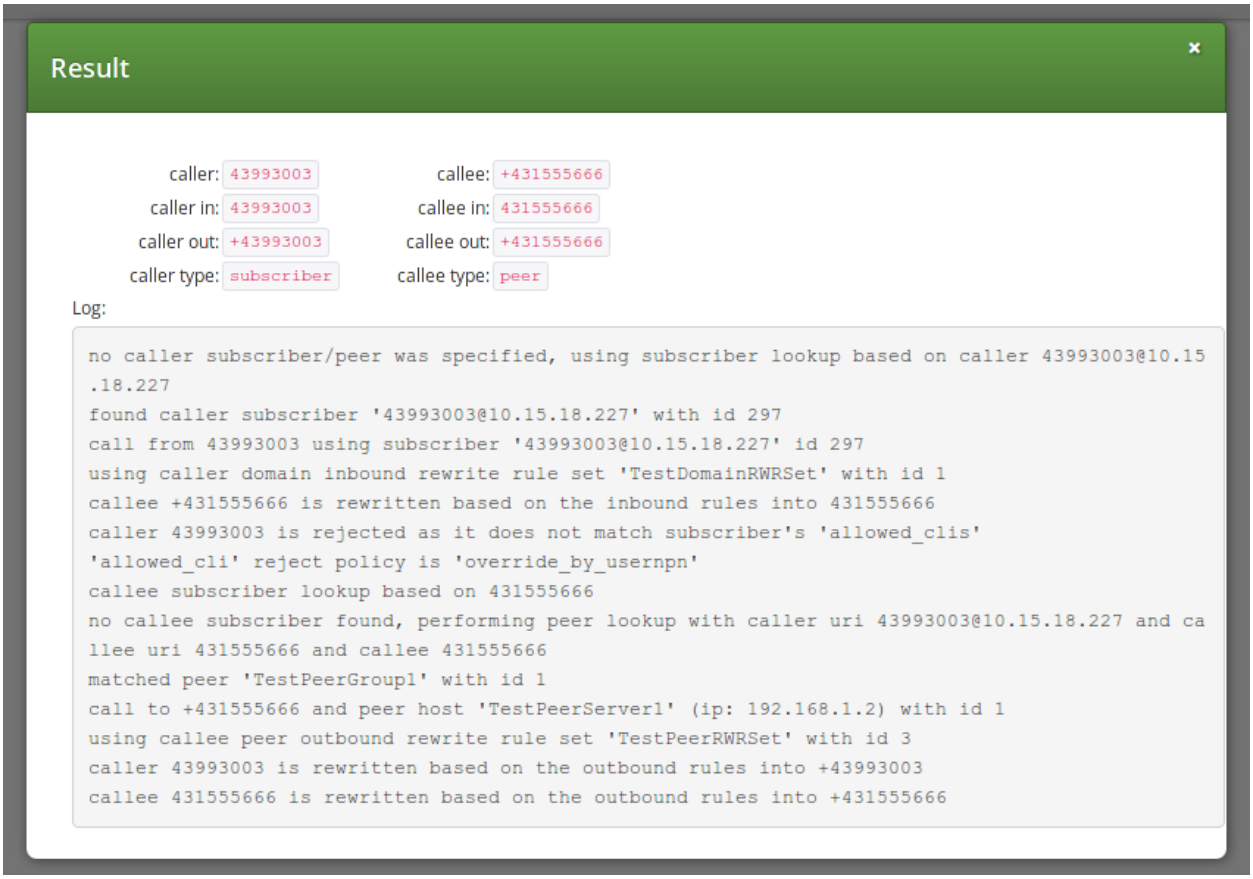


Figure 23. Call Routing Verif. - Number and URI - Result

Copyright © Sipwise GmbH - All rights reserved

Chapter 6. Billing

This chapter describes the steps necessary to rate calls and export rated CDRs (call detail records) to external systems.

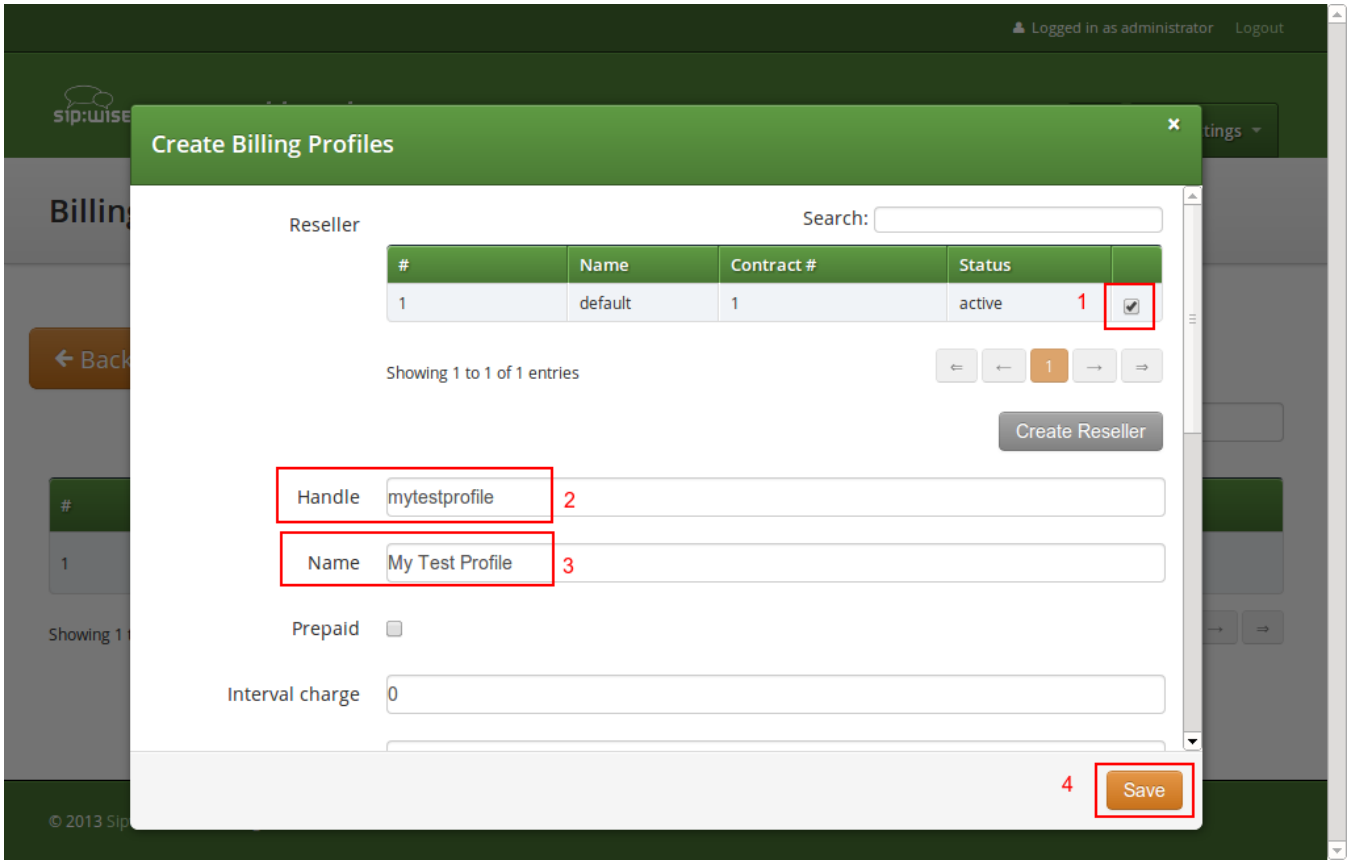
6.1. Billing Profiles

Service billing on Sipwise C5 is based on billing profiles, which may be assigned to customers and SIP peerings. The design focuses on a simple, yet flexible approach, to support arbitrary dial-plans without introducing administrative overhead for the system administrators. The billing profiles may define a base fee and free time or free money per billing interval. Unused free time or money automatically expires at the end of the billing interval.

Each profile may have call destinations (usually based on E.164 number prefix matching) with configurable fees attached. Call destination fees each support individual intervals and rates, with a different duration and/or rate for the first interval. (e.g.: charge the first minute when the call is opened, then every 30 seconds, or make it independent of the duration at all) It is also possible to specify different durations and/or rates for peak and off-peak hours. Peak time may be specified based on weekdays, with additional support for manually managed dates based on calendar days. The call destinations can finally be grouped for an overview on user's invoices by specifying a zone in two detail levels. (E.g.: national landline, national mobile, foreign 1, foreign 2, etc.)

6.1.1. Creating Billing Profiles

The first step when setting up billing data is to create a billing profile, which will be the container for all other billing related data. Go to *SettingsBilling* and click on *Create Billing Profile*.



Copyright © Sipwise GmbH - All rights reserved

The fields *Reseller*, *Handle* and *Name* are mandatory.

- **Reseller**: The reseller this billing profile belongs to.
- **Handle**: A unique, permanently fixed string which is used to attach the billing profile to a customer or SIP peering contract.
- **Name**: A free form string used to identify the billing profile in the *Admin Panel*. This may be changed at any time.
- **Ignore domain**: If enabled, the rating engine will skip matching billing fees as `destination_user_in@destination_domain` and use `destination_user_in` instead.
- **Prepaid**: Enables prepaid accounting for this profile as opposed to normal post-paid mode.
- **Prepaid library**: one of available prepaid libraries to use for the prepaid accounting
- **Advice of charge**: Enables Advice of Charge support to send call costs in the SIP INFO messages back to the caller. The *Billing Fees* are used in the cost and interval calculations.
- **Interval charge**: A base fee for the billing interval, specifying a monetary amount (represented as a floating point number) in whatever currency you want to use.
- **Interval free time**: If you want to include free calling time in your billing profile, you may specify the number of seconds that are available every billing interval. See *Creating Billing Fees* below on how to select destinations which may be called using the free time.
- **Interval free cash**: Same as for *interval free time* above, but specifies a monetary amount which may be spent on outgoing calls. This may be used for example to implement a minimum turnover for a contract, by setting the *interval charge* and *interval free cash* to the same values.
- **Fraud monthly limit**: The monthly fraud detection limit (in Cent) for accounts with this billing profile. If the call fees of an account reach this limit within a billing interval, an action can be triggered.
- **Fraud monthly lock**: a choice of *none, foreign, outgoing, incoming, global*. Specifies a lock level which will be used to lock the account and his subscribers when *fraud monthly limit* is exceeded.
- **Fraud monthly notify**: An email address or comma-separated list of email addresses that will receive notifications when *fraud monthly limit* is exceeded.
- **Fraud daily limit**: The fraud detection limit (in Cent) for accounts with this billing profile. If the call fees of an account reach this limit within a calendar day, an action can be triggered.
- **Fraud daily lock**: a choice of *none, foreign, outgoing, incoming, global*. Specifies a lock level which will be used to lock the account and his subscribers when *fraud daily limit* is exceeded.
- **Fraud daily notify**: An email address or comma-separated list of email addresses that will receive notifications when *fraud daily limit* is exceeded.
- **Currency**: The currency symbol for your currency. Any UTF-8 character may be used and will be printed in web interfaces.
- **VAT rate**: The percentage of value added tax for all fees in the billing profile. Currently for informational purpose only and not used further.
- **VAT included**: Whether VAT is included in the fees entered in web forms or uploaded to the platform. Currently for informational purpose only and not used further.

6.1.2. Creating Billing Fees

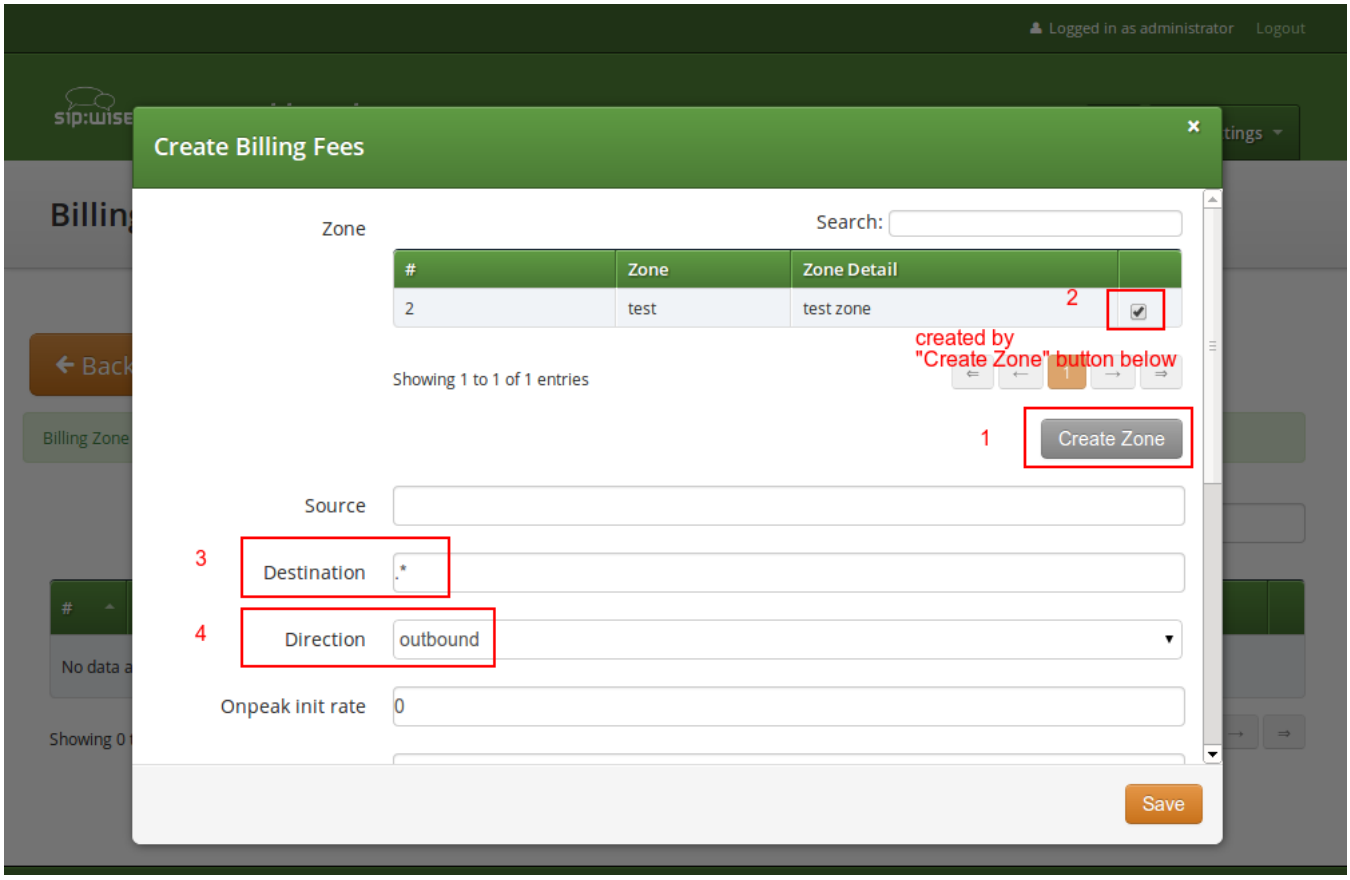
Each *Billing Profile* holds multiple *Billing Fees*.

Copyright © Sipwise GmbH - All rights reserved

To set up billing fees, click on the *Fees* button of the billing profile you want to configure. Billing fees may be uploaded using a configurable CSV file format, or entered directly via the web interface by clicking *Create Fee Entry*. To configure the CSV field order for the file upload, rearrange the entries in the *www_adminfees_csvelement_order* array in */etc/ngcp-config/config.yml* and execute the command ngcpcfg apply 'changed fees element order'. The following is an example of working CSV file to upload (pay attention to double quotes):

```
".","^1",out,"EU","ZONE
EU",5.37,60,5.37,60,5.37,60,5.37,60,0,0,regex_longest_pattern
"^01.+$","^02145.+$",out,"AT","ZONE
Test",0.06250,1,0.06250,1,0.01755,1,0.01733,1,0,regex_longest_pattern,30
,0.01,30,0.01
```

For input via the web interface, fill in the text fields accordingly.



A billing fee record essentially defines the rate per interval to charge the customer when calling a particular destination number. The properties below outline supported options in detail:

- **Zone**: A zone for a group of fees. May be used to group fees for simplified display, e.g. on invoices. (e.g. foreign zone 1)
- **Match Mode**: The mode for matching a fee's source and destination patterns against a CDR's source fields (the caller given by `<source_cli>@<source_domain>` or `<source_cli>` only) and destination fields (the callee given by `<destination_user_in>@<destination_domain>` or `<destination_user_in>` only). Each of the currently supported modes below provide different flexibility and speed:

Copyright © Sipwise GmbH - All rights reserved

1. Exact string (destination): The destination string has to match the destination from the CDR exactly. Fastest, O(log(#fees)). In csv files, this match mode is specified by `exact_destination`.
2. Prefix string: The fee's source/destination represent strings which both the source/destination from the CDR have to start with. The fee with the longest destination prefix is picked. If there are multiple, the one with the longest source prefix is picked. In contrast to regular-expression based match modes, this algorithm uses database index lookups instead of SQL `REGEXP` table scans. The performance boundary is O(length(cdr src) * length(cdr dest) * log(#fees)), hence this will be the preferred mode for tens of thousands of fees in place or high throughput (LCR, rating peer-to-peer calls). In csv files, this match mode is specified by `prefix`.
3. Regular expression - longest match: The fee's source/destination patterns represent PCREs which both have to match the source/destination from the CDR. The fee with the longest match within the destination string is picked. If there are multiple, the one with the longest match within the source string is picked. In csv files, this match mode is specified by `regex_longest_match`.
4. Regular expression - longest pattern: The fee's source/destination represent PCREs which both have to match the source/destination from the CDR. The fee with the longest (most distinctive) destination pattern is picked. If there are multiple, the one with the longest (most distinctive) source pattern is picked. In csv files, this match mode is specified by `regex_longest_pattern`.

If fees with different match mode are in place and matching, the precedence is given by above order. When omitted in file uploads, the legacy default `regex_longest_pattern` is used.

- **Source**: The source pattern (prefix ie. `123` or regular expression `^123someone@sip\.sipwise\.com$`). The legacy default `"."` regular expression (matching everything) will be set implicitly.
- **Destination**: The destination pattern (string ie. `456somebody@sip.sipwise.com`, prefix ie. `456` or regular expression `^456somebody@sip\.sipwise\.com$`). This field must be set.

  To specify a special fixed rate for any ported number in the local LNP tables belonging to an LNP provider, a fee with `exact_destination` match mode and destination `lnp:<lnp provider ID>` can be set up.

  To specify an FCI (Furnished Charging Info) destination for cases when the FCI data is retrieved from the LNP lookup, use a format fci=10050 where "10050" is the FCI data.
- **Direction**: Outbound for standard origination fees (applies to callers placing a call and getting billed for that) or Inbound for termination fees (applies to callees if you want to charge them for receiving various calls, e.g. for 800-numbers). *If in doubt, use Outbound*. If you upload fees via CSV files, use out or in, respectively.

IMPORTANT

The {match mode, source, destination, direction} combination needs to be unique for a billing profile. The system will return an error if such a set is specified twice via web interface/ or /api, or skipped when processing the file upload. When uploading fees, the *Purge Existing* checkbox allows to drop all existing fees before creating the records.

Copyright © Sipwise GmbH - All rights reserved

IMPORTANT

There are several internal services (vsc, conference, voicebox, fax2mail) which will need a specific destination entry with a domain-based destination. If you don't want to charge the same (or nothing) for those services, add a fee for destination `\.local$` there. If you want to charge different amounts for those services, break it down into separate fee entries for `@fax2mail\.local$`, `@vsc\.local$`, `@conference\.local$` and `@voicebox\.local$` with the according fees. **NOT CREATING EITHER THE CATCH-ALL FEE OR THE SEPARATE FEES FOR THE `.local` DOMAIN WILL BREAK YOUR RATING PROCESS!**

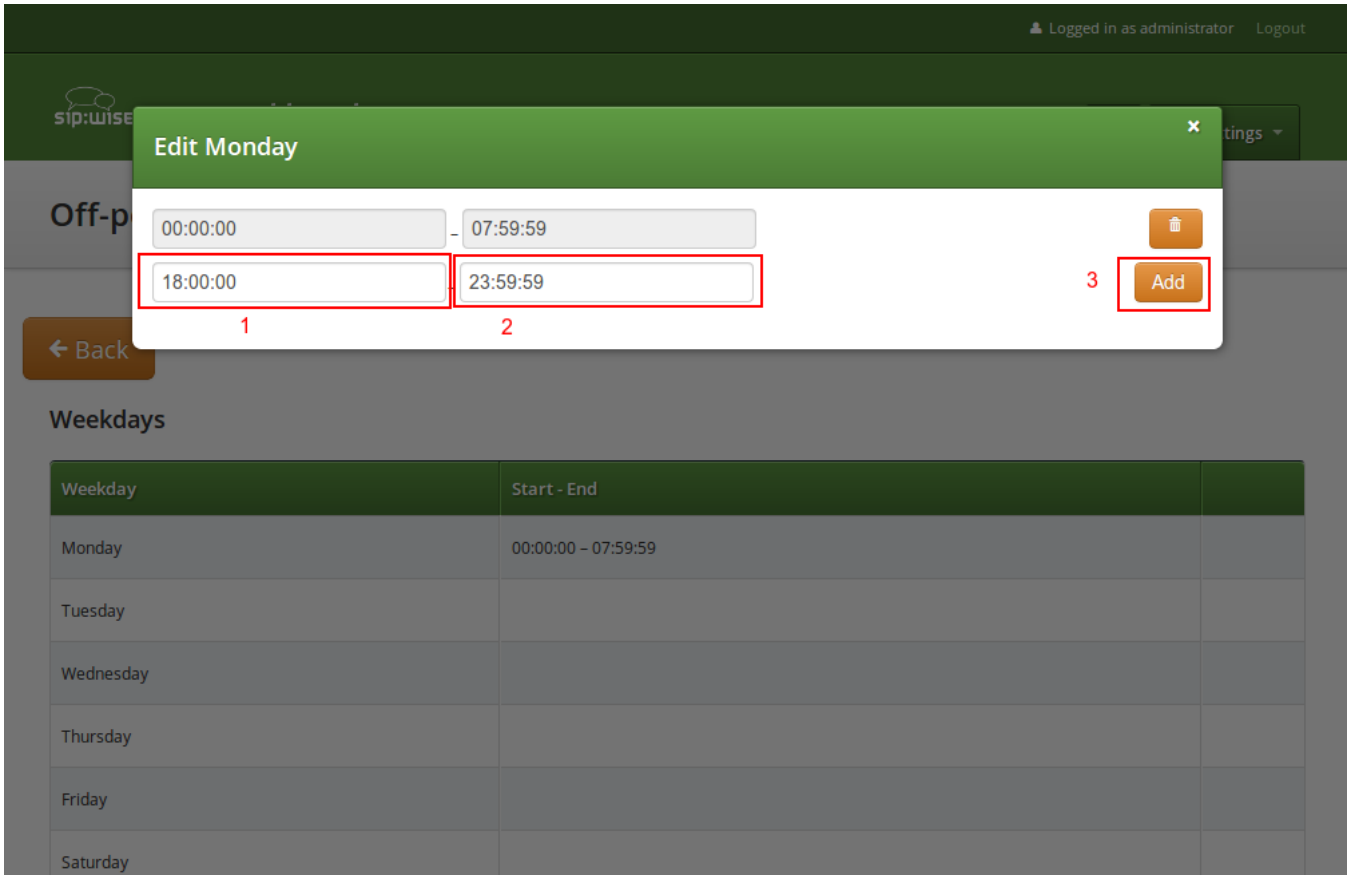
- **Onpeak init rate**: The rate for the first rating interval in cent (of whatever currency, represented as a floating point number) per second. Applicable to calls during onpeak hours.
- **Onpeak init interval**: The duration of the first billing interval, in seconds. Applicable to calls during onpeak hours.
- **Onpeak follow rate**: The rate for subsequent rating intervals in cent (of whatever currency, represented as a floating point number) per second. Applicable to calls during onpeak hours. Defaults to *onpeak init rate*.
- **Onpeak follow interval**: The duration of subsequent billing intervals, in seconds. Applicable to calls during onpeak hours. Defaults to *onpeak init interval*.
- **Offpeak init rate**: The rate for the first rating interval in cent (of whatever currency, represented as a floating point number) per second. Applicable to calls during off-peak hours. Defaults to *onpeak init rate*.
- **Offpeak init interval**: The duration of the first billing interval, in seconds. Applicable to calls during off-peak hours. Defaults to *onpeak init interval*.
- **Offpeak follow rate**: The rate for subsequent rating intervals in cent (of whatever currency, represented as a floating point number) per second. Applicable to calls during off-peak hours. Defaults to *offpeak init rate* if that one is specified, or to *onpeak follow rate* otherwise.
- **Offpeak follow interval**: The duration of subsequent billing intervals, in seconds. Applicable to calls during off-peak hours. Defaults to *offpeak init interval* if that one is specified, or to *onpeak follow interval* otherwise.
- **Onpeak use free time**: Specifies whether free time minutes may be used when calling this destination during onpeak hours. Specified in the file upload as 0, n[o], f[alse] and 1, y[es], t[rue] respectively.
- **Offpeak use free time**: Specifies whether free time minutes may be used when calling this destination during off-peak. Specified in the file upload as 0, n[o], f[alse] and 1, y[es], t[rue] respectively.
- **Onpeak extra second**: If defined, an extra rate will be charged at the given second of call time for post-paid calls. Applicable to calls started during onpeak hours.
- **Onpeak extra rate**: The rate to charge if the call time exceeds *extra second* in cent (of whatever currency, represented as a floating point number). Applicable to calls started during onpeak hours.
- **Offpeak extra second**: See onpeak extra second. Applicable to calls started during offpeak hours.
- **Offpeak extra rate**: See onpeak extra rate. Applicable to calls started during offpeak hours.

Copyright © Sipwise GmbH - All rights reserved

6.1.3. Creating Off-Peak Times

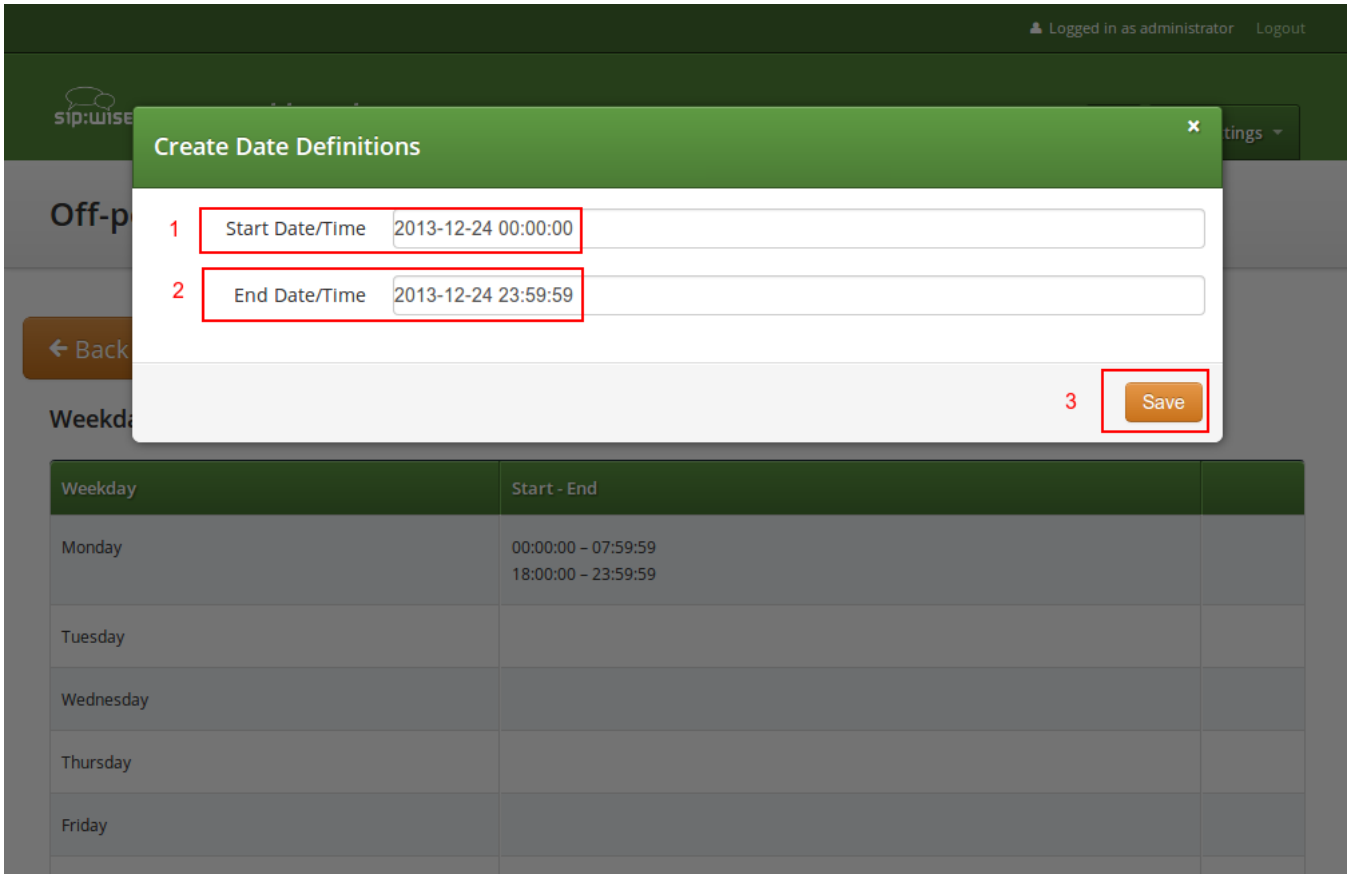
To be able to differentiate between on-peak and off-peak calls, the platform stores off-peak times for every billing profile based on weekdays and/or calendar days. To edit the settings for a billing profile, go to *SettingsBilling* and press the *Off-Peaktimes* button on the billing profile you want to configure.

To set off-peak times for a weekday, click on *Edit* next to the according weekday. You will be presented with two input fields which both receive a timestamp in the form of *hh:mm:ss* specifying a time of day for the start and end of the off-peak period. If any of the fields is left empty, the system will automatically insert 00:00:00 (*start* field) or *23:59:59* (*end* field). Click on *Add* to store the setting in the database. You may create more than one off-peak period per weekday. To delete a range, click *Delete* next to the entry. Click the *close* icon when done.



To specify off-peak ranges based on calendar dates, click on *Create Special Off-Peak Date*. Enter a date in the form of *YYYY-MM-DD hh:mm:ss* into the *Start Date/Time* input field and *End Date/Time* input field to define a range for the off-peak period.

Copyright © Sipwise GmbH - All rights reserved



6.2. Peak Time Call Rating Modes

6.2.1. Introduction to Call Rating Modes

The call rating engine component (*ngcp-rate-o-mat*) supports two different modes to consider configured off-peak/on-peak periods when calculating call costs:

- **Split-Peak-Parts mode**: CDRs reflecting calls which cross an off-peak/on-peak period transition will be split into two CDR fragments. This way it is possible for each fragment to exactly mark it as either on-peak or off-peak, and the CDR's *frag_carrier_onpeak*, *frag_reseller_onpeak* and *frag_customer_onpeak* fields can be populated accordingly.

  CDRs that are entirely within either on-peak or off-peak periods are not split and show a value of 0 for their *is_fragmented* field. CDR fragments are marked by the *is_fragmented* field showing a value of 1. If the call is crossing *n* transitions, *(n+1)* fragments are created.

  Apart from *is_fragmented*, **_onpeak* and **_cost* fields, each fragment is a copy of the original CDR, except for start_time and durations fields. The sum of durations of fragments is equal to the duration of the original CDR. Fragments are adjacent, so the *start_time* of a fragment is equal to the end time (*start_time* + *duration*) of the previous fragment.

- **Regular mode**: In regular mode, the costs are calculated by summing up init/follow interval ticks, and selecting on-peak or off-peak rates of the billing fee per tick. Resulting call costs will be identical to the sum of the costs of fragmented CDRs in Split-Peak-Parts mode, but now comprised of both on-peak an off-peak rates in a single value. Hence *frag_carrier_onpeak*, *frag_reseller_onpeak* and *frag_customer_onpeak* CDR fields cannot be provided.

Copyright © Sipwise GmbH - All rights reserved

6.2.2. Typical Use Cases for Call Rating Modes

The CDR fragmentation produced by **Split-Peak-Parts mode** can be useful when implementing:

- End-customer invoicing to **separate** call listings or costs by **off-peak and on-peak**
- Reports to compare sums of carrier and customer costs when fees with **different metering** (given by the fees' init and follow interval) are in effect

The process of the **regular mode** does not create additional CDRs, which has advantages in other situations:

- It is possible to **re-rate** CDRs, as there is no need to revert fragmentation.
- The concept of **one-CDR-per-call-leg** is kept, which simplifies external rating, reporting, call-flow visualisation etc.

6.2.3. Configuration of Call Rating Modes

The regular mode is enabled by default. To enable Split-Peak-Parts mode, set rateomat.splitpeakparts to 1 in /etc/ngcp-config/config.yml file.

6.3. Prepaid Accounting

In a normal post-paid accounting scenario, each customer accumulates debt in their billing account, which at the end of the billing interval is then billed to the customer. A *prepaid* billing profile reverses this sequence: the customer first has to provide credit to their account balance, and the costs for all calls are then deducted from that account balance. Once the balance reaches zero, no further calls from this customer are accepted, with the exception of free calls. Additionally, if the balance drops to zero while any calls are currently active, Sipwise C5 will disconnect those calls as soon as that happens.

With prepaid billing enabled, all details of the billing profile and all details of the billing fees behave as they normally do, including interval free time. If any interval free time is given, the free time will be used before the account's credit is.

IMPORTANT	For technical reasons, the system can make the distinction between on-peak and off-peak times only at call establishment time. In other words, if the currently active call fee at the moment when the call is established is an off-peak fee, then the same off-peak fee will remain active for the whole length of this call, even if the call actually transitions into an on-peak fee (and vice versa).
IMPORTANT	For technical reasons, prepaid billing can't charge local endpoint calls to Voicebox, VSC calls or calls to a Conference Room.

The Sipwise C5 platform offers advanced billing features which are especially designed for pre-paid billing scenarios. For details please visit Billing Customizations section of the handbook.

6.4. Fraud Detection and Locking

The Sipwise C5 supports a fraud detection feature, which is designed to detect accounts causing unusually high customer costs, and then to perform one of several actions upon those accounts. This feature can be enabled and configured through two sets of billing profile options described in Creating

Copyright © Sipwise GmbH - All rights reserved

Billing Profiles, namely the monthly (*fraud monthly limit*, *fraud monthly lock* and *fraud monthly notify*) and daily limits (*fraud daily limit*, *fraud daily lock* and *fraud daily notify*). Either monthly/daily limits or both of them can be active at the same time.

As soon as call costs of a CDR are determined by rate-o-mat, database tables for bookkeeping daily and monthly sums are updated. Fraud lock levels will be applied instantly in case of a *fraud event* - that is when either the *fraud monthly limit* or *fraud daily limit* got exceeded. If **fraud lock** is set to anything other than *none*, it will lock the account's subscribers accordingly (e.g. if **fraud lock** is set to *outgoing*, the account will be locked for all outgoing calls).

A background script (managed by cron daemon, running every 10 minutes by default) automatically checks recent fraud events. An email will be sent to the address given by **fraud notify**, if set. The email will contain information about which account is affected, which subscribers within that account are affected, the current account balance and the configured fraud limit, and also whether or not the account was locked in accordance with the **fraud lock** setting. It should be noted that this email is meant for the administrators or accountants etc., and not necessarily for the customer.

6.4.1. Fraud Lock Levels

Fraud lock levels are various protection (and notification) settings that are applied to subscribers of a *Customer*, if fraud detection is enabled in the currently active billing profile and the *Customer's* daily or monthly fraud limit has been exceeded.

The following lock levels are available:

- none: no account locking will happen
- foreign calls: only calls within the subscriber's own domain, and emergency calls, are allowed
- all outgoing calls: subscriber cannot place any calls, except calls to emergency destinations
- incoming and outgoing: subscriber cannot place and receive any calls, except calls to emergency destinations
- global: same restrictions as at incoming and outgoing level, additionally subscribers are not allowed to access the Customer Self Care (CSC) interface
- ported: only automatic call forwarding, due to number porting, is allowed

IMPORTANT You can override fraud detection and locking settings of a billing profile on a per-account basis via REST API or the Admin interface.

CAUTION Accounts that were automatically locked by the fraud detection feature will **not** be automatically unlocked (e.g. if either a limit is lowered or the next day/month starts). This has to be done manually through the administration panel or through the provisioning interface.

NOTE It is possible to fetch the list of fraud events and thus get fraud status of *Customers* by using the REST API and referring to the resource: /api/customerfraudevents.

NOTE Apart from the daily fraud detection check service, Sipwise C5 also provides instant, "hard" locking for prepaid use cases, by means of billing profile packages. See Billing Profile Packages for reference.

Copyright © Sipwise GmbH - All rights reserved

6.5. Billing Customizations

The standard way of doing the billing—i.e. having fixed billing intervals of a calendar month, starting on the 1st day of month—may not fit all billing profiles and intervals that Sipwise C5 platform operators would like to use.

The Sipwise C5 supports alternate ways of defining billing profiles and intervals which are especially worthy for pre-paid scenarios. New functionality is covered by the following titles:

1. Billing Networks
2. Profile Mappings Schedule
3. Profile Packages
4. Vouchers
5. Top-up
6. Balance Overviews
7. Usage Examples

Subsequent sections will provide an introduction and configuration instructions to these advanced features of Sipwise C5.

6.5.1. Billing Networks

The idea is to dynamically select billing profiles (including fees) depending on the IP network the caller's SIP client is using to connect. The caller's IP is populated in a call's CDR, and effectively processed by:

- the rating engine component (*ngcp-rate-o-mat*) and the
- prepaid interception module (libswrate).

The billing profile for rating a call is identified by matching the source IP against network ranges linked to the customer contract's billing mappings records. This feature is sometimes also referred to as *roaming*.

A *Billing Network* is defined as a series of *network blocks* where each network block consists of *a single IP address* or *an IP subnet*. Blocks of a particular billing network can be defined by either IPv4, or IPv6 addresses but not mixed.

Copyright © Sipwise GmbH - All rights reserved

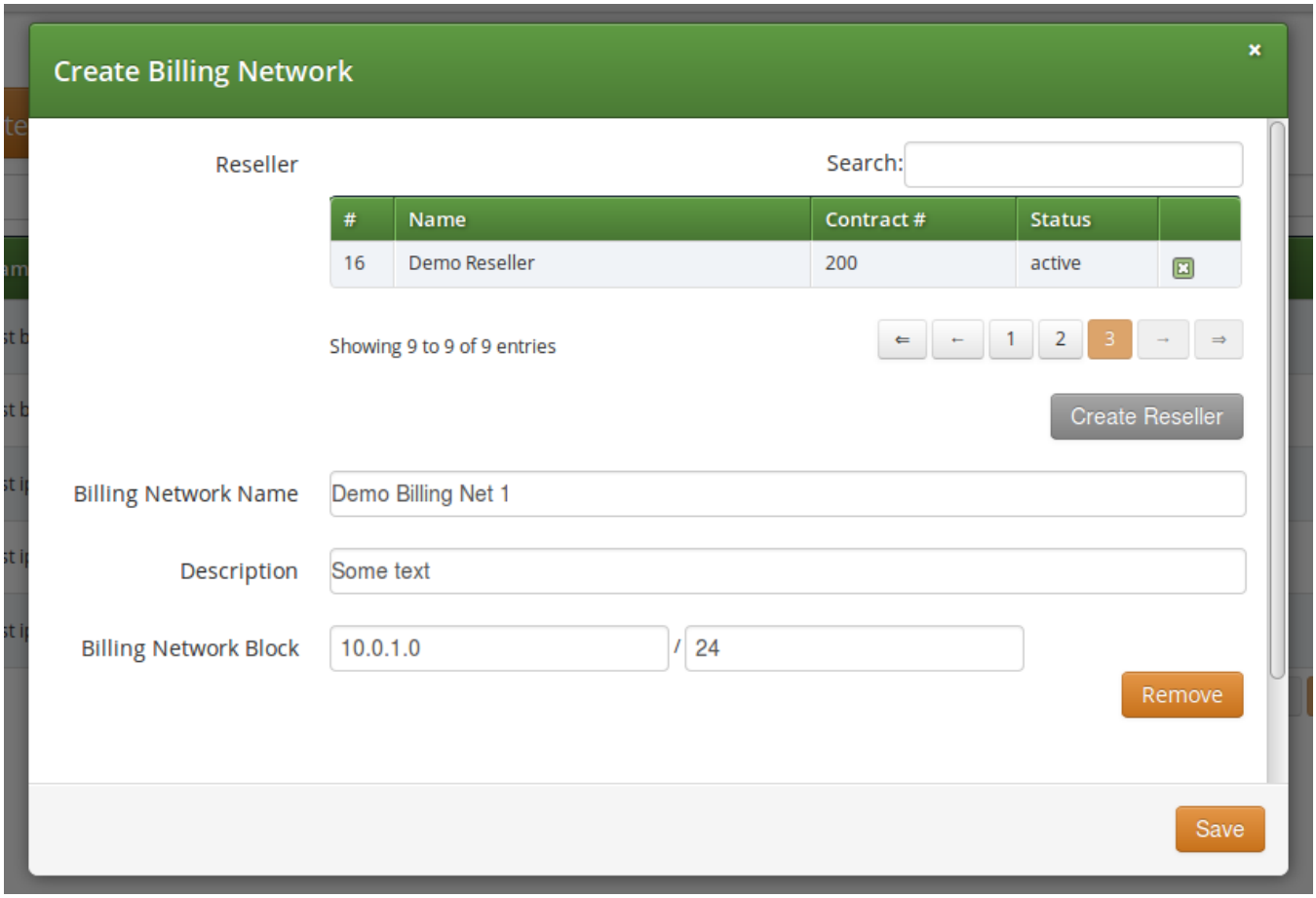


Figure 24. Creation of Billing Network

The new /api/billingnetworks/ **REST API** resource makes it possible to manage billing networks. The example billing network that is shown in the figure above may be defined through the API with this JSON structure:

```
{ "blocks" : [ { "ip" : "10.0.1.0", // subnet: 10.0.1.0 .. 10.0.1.255
                 "mask" : 24
               },
               { "ip" : "10.0.2.2" // single ip
               }
             ],
  "description" : "Some text",
  "name" : "Demo Billing Net 1", //unique per reseller
  "reseller_id" : 1
}
```

Input validation of the network blocks is automatically performed by Sipwise C5 during their definition in a way that it prevents specifying overlapping blocks by means of Interval Trees; billing networks themselves may overlap though.

Copyright © Sipwise GmbH - All rights reserved

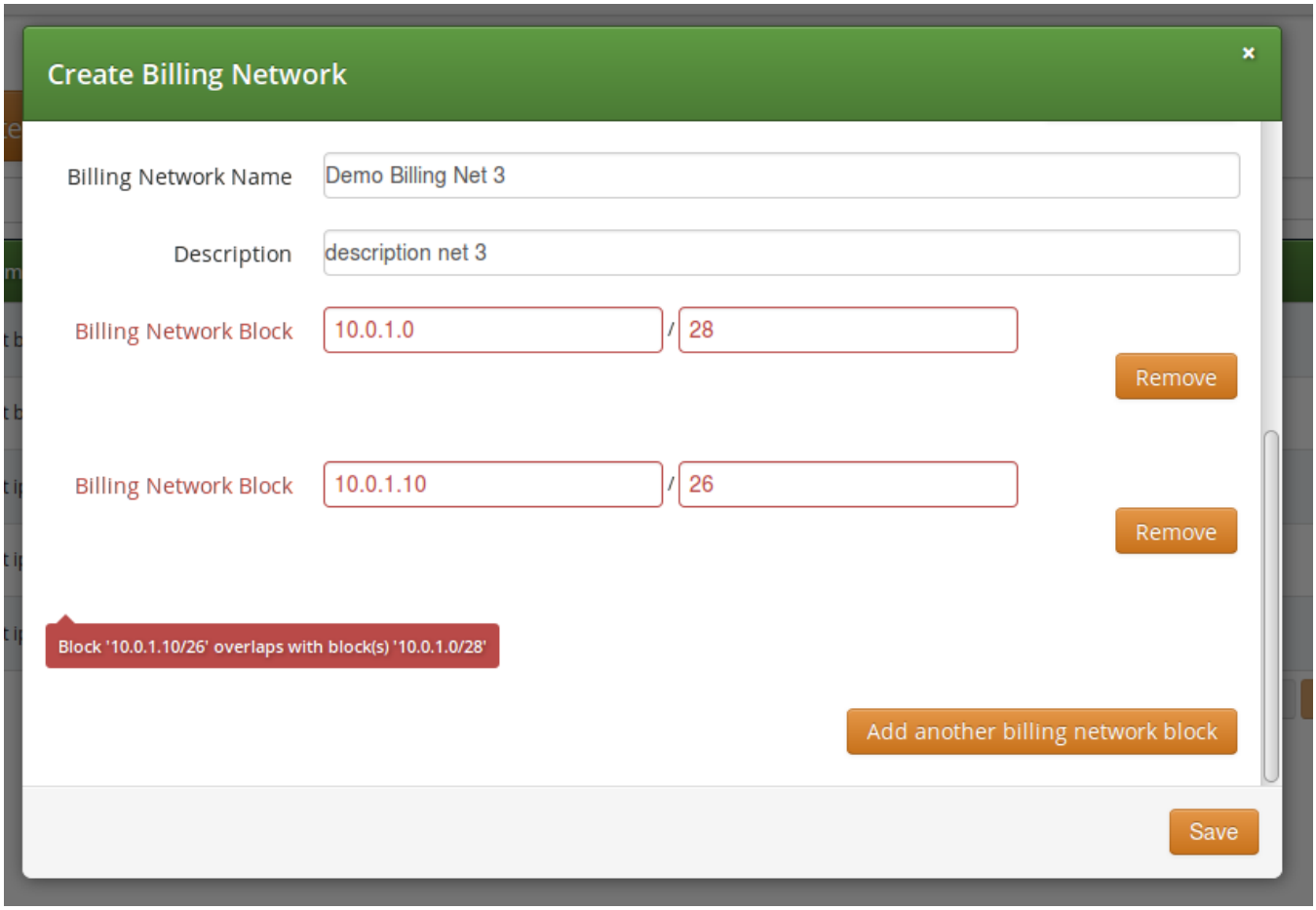


Figure 25. Overlapping Block Prevention

6.5.2. Profile Mapping Schedule

Using the default settings related to billing when creating a new *Reseller* or *Customer* on the administrative web panel results in applying the standard billing profile mapping schedule: the same billing profile is always used.

Definition of Profile Mapping Schedules

The idea of *billing profile mapping schedule* is to extend the billing mappings logic to utilize it as a schedule for billing profiles (and associated fees) for the *Customer* or *Reseller* contract. So far, billing mapping records provided only a history showing which profile was in effect at a given time in the past, which is for example required for delayed rating of calls.

Now it is also possible to define in advance, when specific billing profiles should become active in the future, e.g. to plan campaigns or special offers.

Billing profile mappings represent a schedule of overlapping time intervals with *Billing Profiles* and *Billing Networks*, which are assigned to (customer) contracts when creating or editing them.

Mapping intervals can be of type:

- open: no start time + no end time
- half-open:

 Copyright © Sipwise GmbH - All rights reserved

left-open: no start time + definite end time

right-open: definite start time + no end time

- closed: definite start time + definite end time

Schedule Example

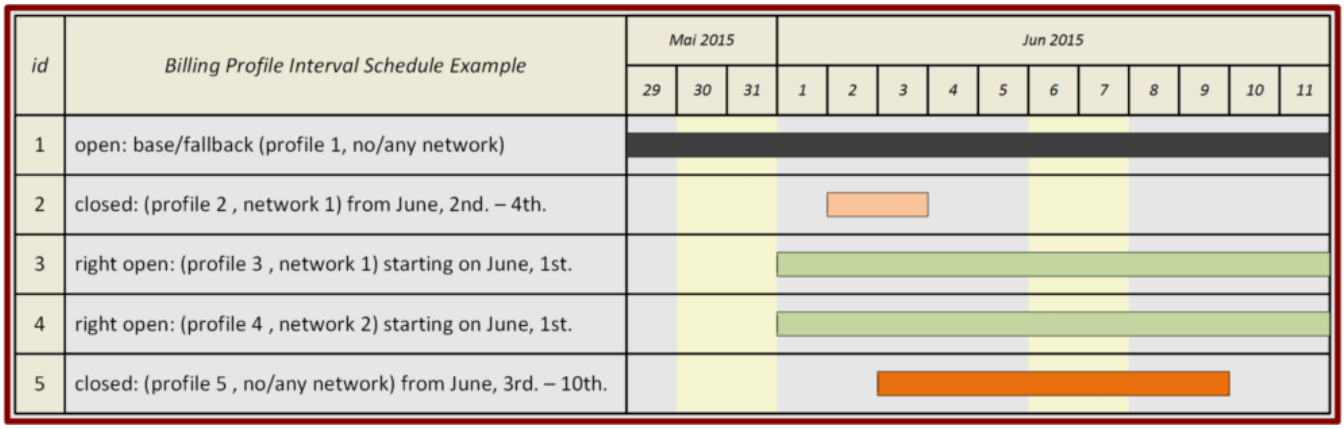


Figure 26. Profile Mapping Schedule Example

Applying the profile mapping schedule shown in the above figure will result in billing profiles being active as provided in the table below.

Table 3. Active Billing Profiles

Time	**Web Panel shows**	**Rating**		
		Caller IP in Network 1	Caller IP in Network 2	Caller IP in other network
May 30	*Profile 1*	*Profile 1*	*Profile 1*	*Profile 1*
June 1	*Profile 4*	*Profile 3*	*Profile 4*	*Profile 1*
June 2	*Profile 2*	*Profile 2*	*Profile 4*	*Profile 1*
June 5	*Profile 5*	*Profile 3*	*Profile 4*	*Profile 5*

Configuration of Schedules

A *Customer's* default billing profile mapping can be changed to scheduled mappings when editing its properties, at the parameter "Set billing profiles", selecting: schedule (billing mapping intervals)

Copyright © Sipwise GmbH - All rights reserved

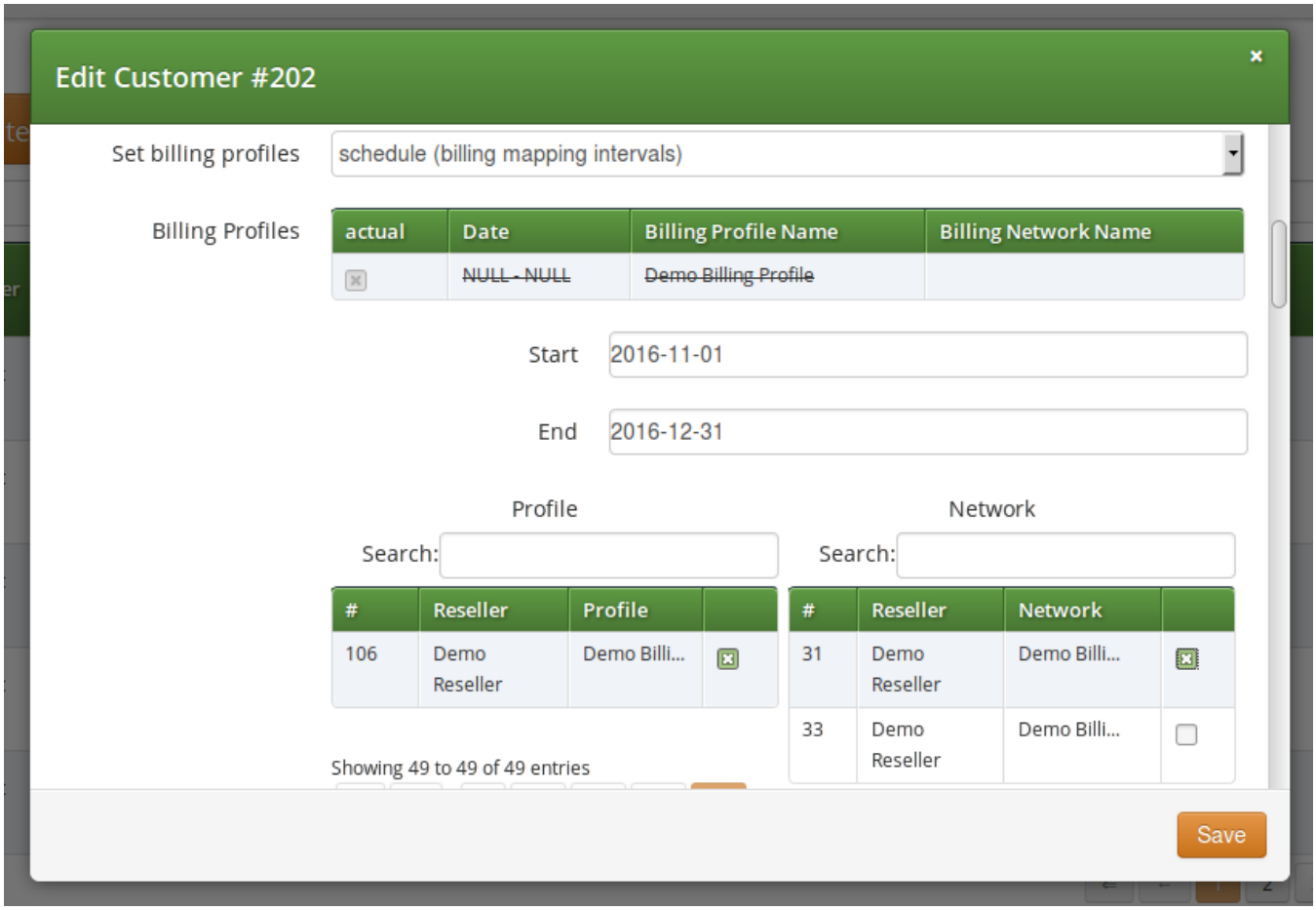


Figure 27. Profile Mapping Schedule Creation

TIP Assigning a *Billing Network* to a billing profile mapping is optional. Without selecting the network, the *Billing Profile* will be applied to all calls.

The profile mapping schedule assigned to a *Customer* is also listed among *Customer's* properties. See *Settings Customers Details Billing Profile Schedule*.

Copyright © Sipwise GmbH - All rights reserved

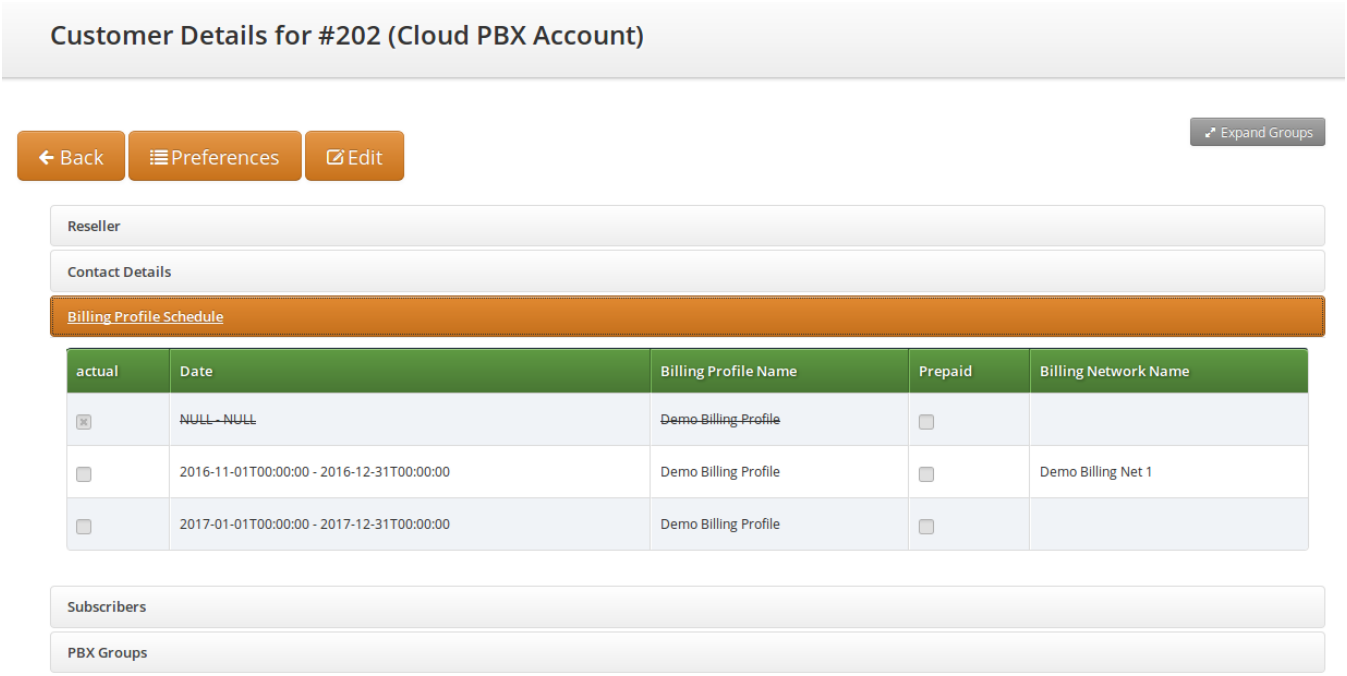


Figure 28. Profile Mapping Schedule List

NOTE Profile mappings that started in the past, like the default one, are displayed with a strike-through font in order to indicate that those can not be modified.

The currently active mapping is depicted by a checked box.

REST API for Profile Mapping Schedules

The /api/customers/ API resource was extended to provide three different modes of defining profile mappings:

1. billing_profiles field: explicitly declare profile mappings in form of (billing profile, billing network, start time, stop time) tuples
2. billing_profile_id field (legacy API spec): a single profile mapping interval is appended (billing profile, no network / any caller IP respectively, starting now)
3. profile_package_id field: profile mappings starting now are appended by using lists of (billing profile, billing network) tuples from the given profile package

With regards to *Resellers*, the /api/contracts/ API resource was enhanced as well, but supports method 1. and 2. only, and without billing networks.

Mapping Intervals

Intervals can be of open, half-open (left-open, right-open) or closed type. When specifying profile mappings discretely, allowed interval types are restricted, depending on create/update situation:

Table 4. Allowed Mapping Intervals

Copyright © Sipwise GmbH - All rights reserved

Interval Type	Start	Stop	POST (create)	PUT / PATCH (update)
open	undefined	undefined	1..*	0
left-open	undefined	defined	0	0
right-open	> now()	undefined	*	*
closed	> now()	> start	*	*

Example Profile Mapping

An example JSON structure for definition of profile mapping schedules shown in Billing Profile Schedule List :

```
{ ...,
  "billing_profile_definition" : "profiles", // i.e. use
'billing_profiles' field
  "billing_profiles" : [ { "network_id" : "236",
                           "profile_id" : "236",
                           "start" : "2016-11-01 00:00:00",
                           "stop" : "2016-12-31 00:00:00"
                         }, // closed future interval, with network
                         { "network_id" : null,
                           "profile_id" : "237",
                           "start" : "2017-01-01 00:00:00",
                           "stop" : "2017-12-31 00:00:00"
                         } ], // closed future interval, without network
  "contact_id" : 141,
  ...
}
```

6.5.3. Profile Packages

By introducing billing profile packages, general billing parameters can be defined for a customer contract:

- Balance interval duration (regular/constant or aligned to top-up events)
- The first interval's start date
- The cash-balance carry-over/discard behaviour upon interval transitions
- Subscriber lock levels and profile sets to get applied upon:
  - top-up
  - balance threshold underrun
- Initial balance and billing profiles

Profile Packages are fundamental for pre-paid billing scenarios, since in such a billing scheme the traditional, fixed monthly periods prove to be insufficient to cover the business needs of Sipwise C5 platform operator. As an example: pre-paid subscribers typically have their "billing periods" between account balance top-ups.

Copyright © Sipwise GmbH - All rights reserved

Elements of Profile Packages

A *Profile Package* consists of various elements that will be discussed in subsequent sections of Sipwise C5 handbook. In order to set the parameters of a profile package one must navigate to: *Settings Profile Packages Create Profile Package*, or alternatively, in order to update an existing profile package: select the package and press *Edit* button.

Basic Balance Intervals Setup

- Interval duration (n hours, days, weeks, months)
- Interval start mode:

  1st of month (1st): billing interval is 1 calendar month; this is the default for each *Customer* created on Sipwise C5 platform

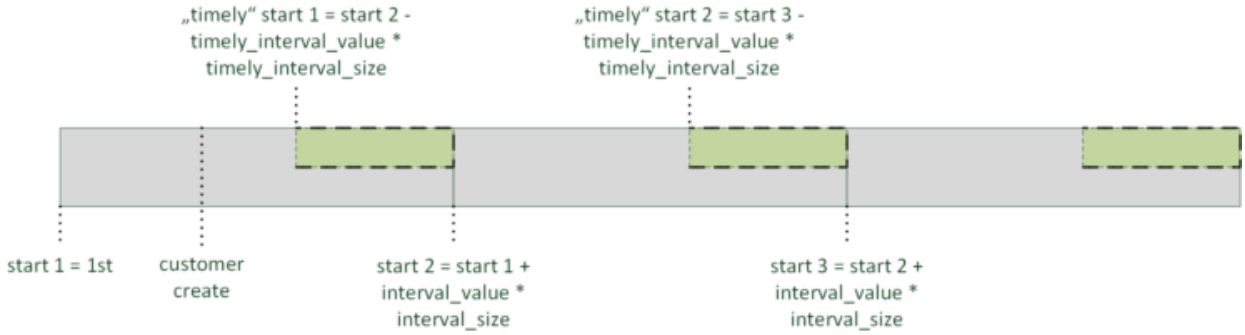


Figure 29. Interval Start Mode: 1st

  upon customer creation (create): (the initial) billing interval starts when the *Customer* is created

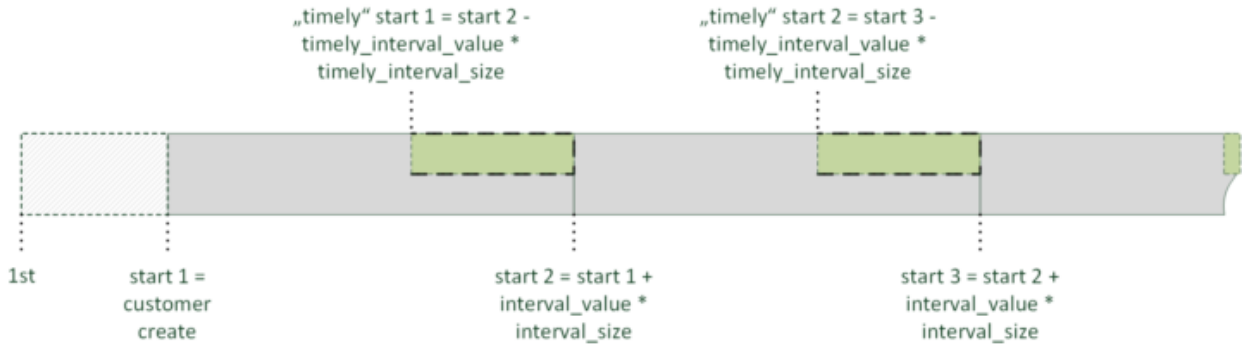


Figure 30. Interval Start Mode: create

  upon topup (topup_interval): interval starts at *first topup* event and its length is defined by interval duration parameter of the profile package

Copyright © Sipwise GmbH - All rights reserved

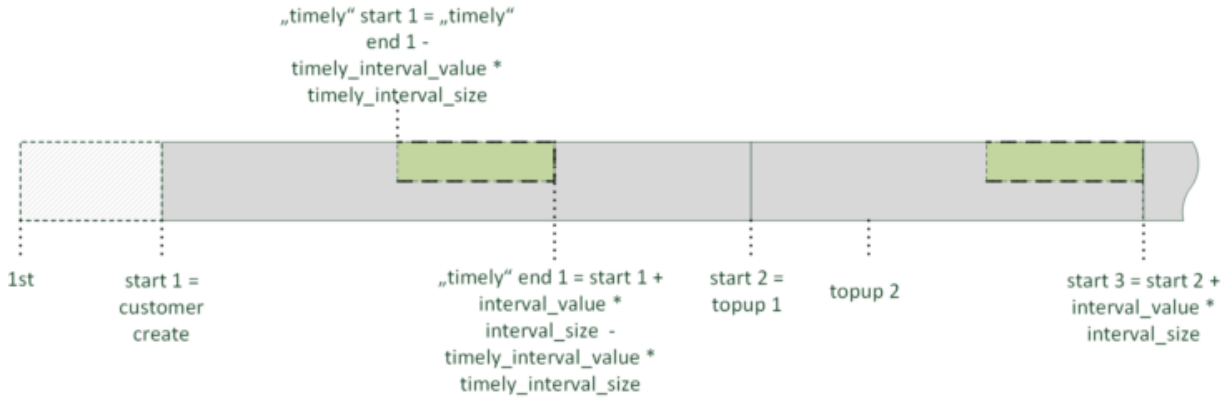


Figure 31. Interval Start Mode: topup_interval

intervals from topup to topup (topup): interval starts at *any topup* event and its length is defined by interval duration parameter of the profile package; intervals can overlap in this case

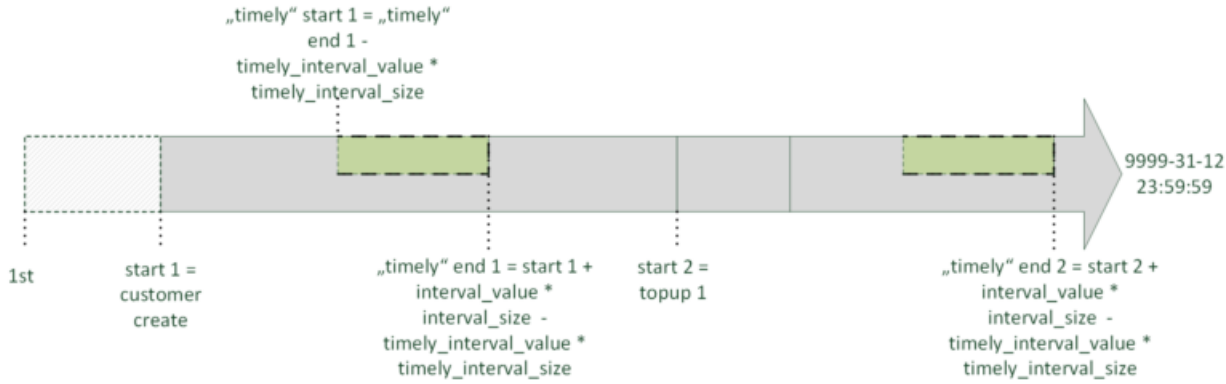


Figure 32. Interval Start Mode: topup

- Initial balance: the initial value of account balance (e.g. every new customer gets 5 Euros as a starting bonus)

Balance Carry Over

- Carry Over: balance carry over behaviour upon interval transitions:

  carry-over: always keep balance

  carry-over only if topped-up timely: keep balance in case of a *timely* top-up only; where **timely** means the topup happens within a pre-defined time span before the end of the balance interval

  discard: discard balance at the end of each interval

- Timely Duration: duration of the *timely* period
- Discard balance after intervals: for how many balance intervals the remaining account balance is kept before its disposal

Underrun Settings

- Underrun lock threshold: when account balance reaches this amount the subscriber will be locked to a restricted set of services
- Underrun lock level: this level of services will apply when an account balance underruns

Copyright © Sipwise GmbH - All rights reserved

don't change: no change in the available set of services

no lock: all services are available

foreign: only calls within subscriber's own domain are allowed

outgoing: all outgoing calls are prohibited

all calls: all calls (incoming + outgoing) are prohibited

global: all calls + access to Customer Self Care web interface are prohibited

ported: only automatic call forwarding, due to number porting, is allowed

- Underrun profile threshold: when account balance reaches this amount the *Underrun Billing Profile* will be applied

Basic Top-up Settings

- Top-up lock level: subscriber lock (unlock) levels to apply upon top-up event
- Service charge: (always) subtract this value from the voucher amount, if topup happens via the usage of a voucher

Profile mappings

A lists of (billing profile, billing network) tuples for appending profile mappings:

- Initial Billing Profile: when creating or manually changing the customers package (initial_profiles)
- Underrun Billing Profile: when the balance underruns a cash threshold (underrun_profiles)
- Top-up Billing Profile: when the customer tops-up using a voucher associated with the package (topup_profiles)

Examples

Profile Package Configuration

1. Definition of basic profile package parameters

Copyright © Sipwise GmbH - All rights reserved

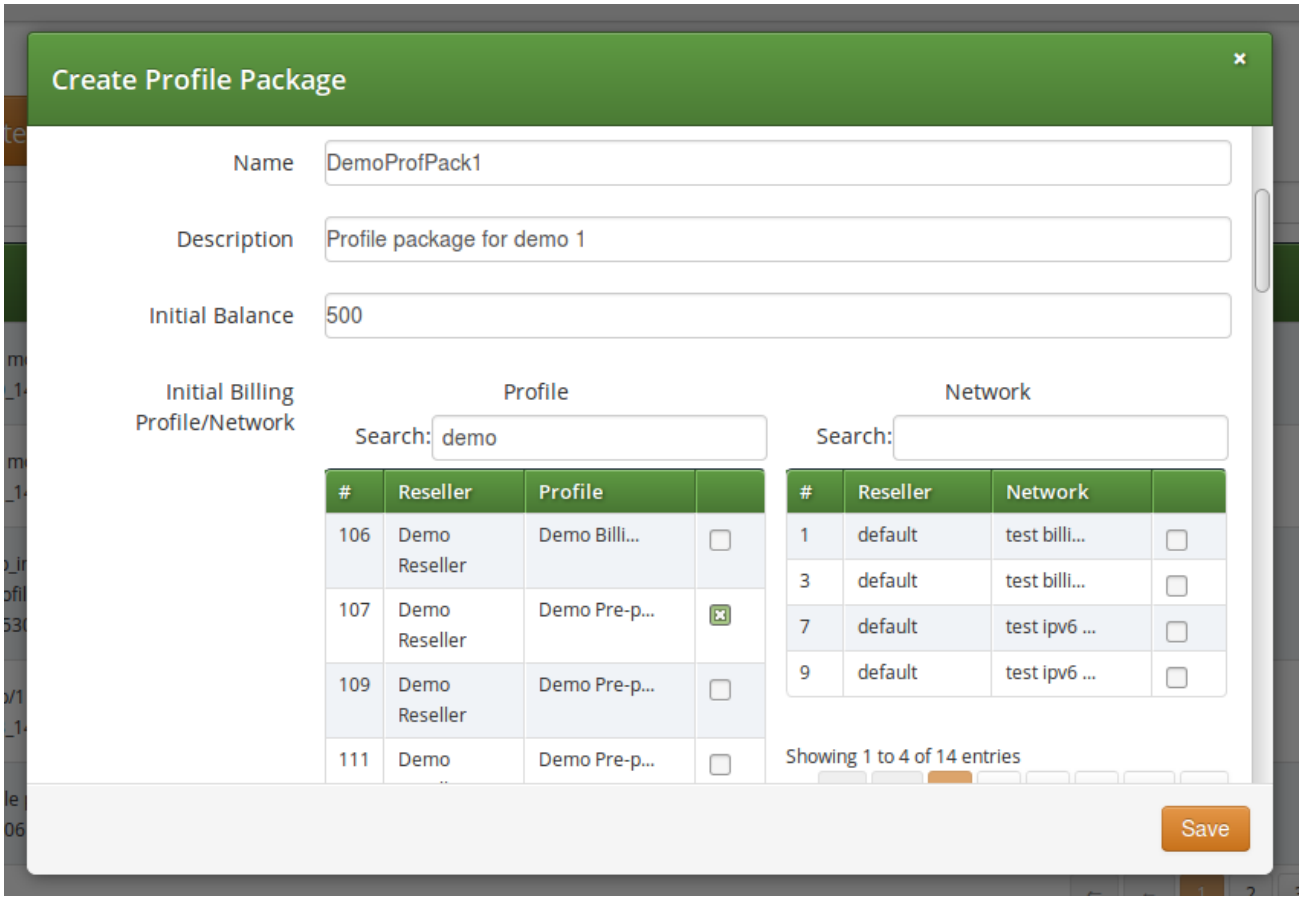


Figure 33. Basic Profile Package Parameters

2. Definition of balance interval and carry-over behaviour

Copyright © Sipwise GmbH - All rights reserved

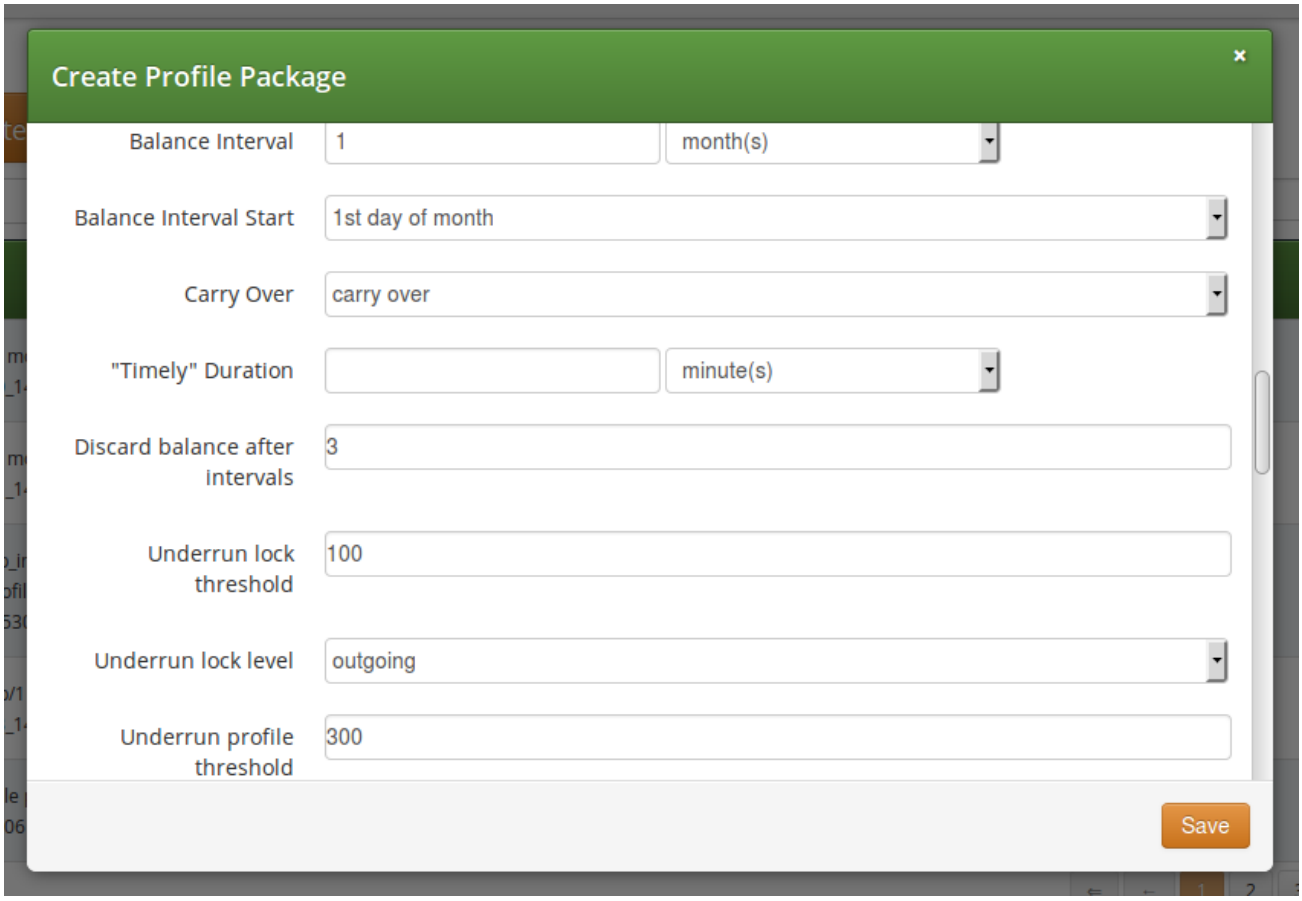


Figure 34. Balance Interval and Carry-over

3. Definition of balance underrun parameters

Copyright © Sipwise GmbH - All rights reserved

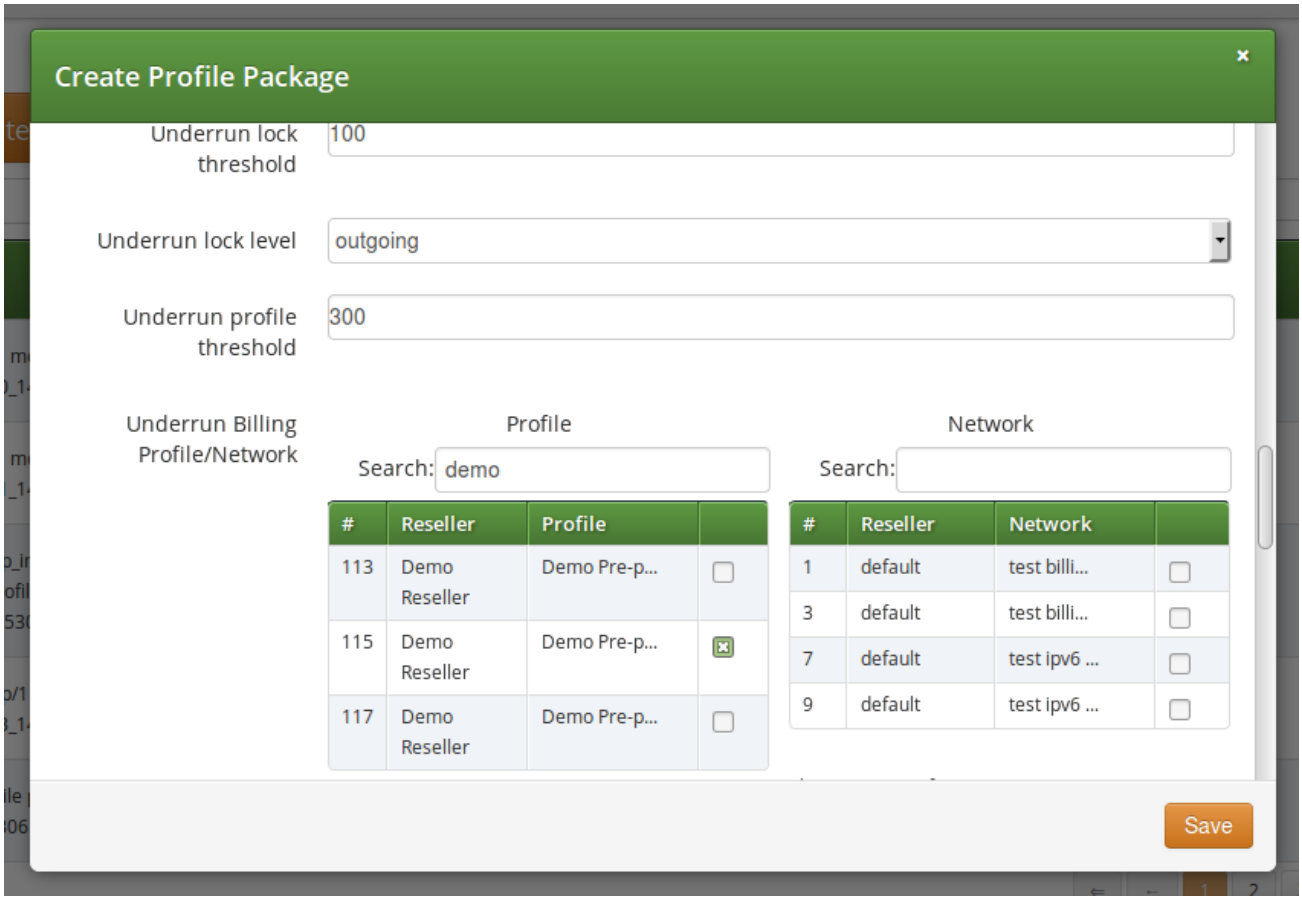


Figure 35. Balance Underrun Parameters

4. Definition of top-up settings

Copyright © Sipwise GmbH - All rights reserved

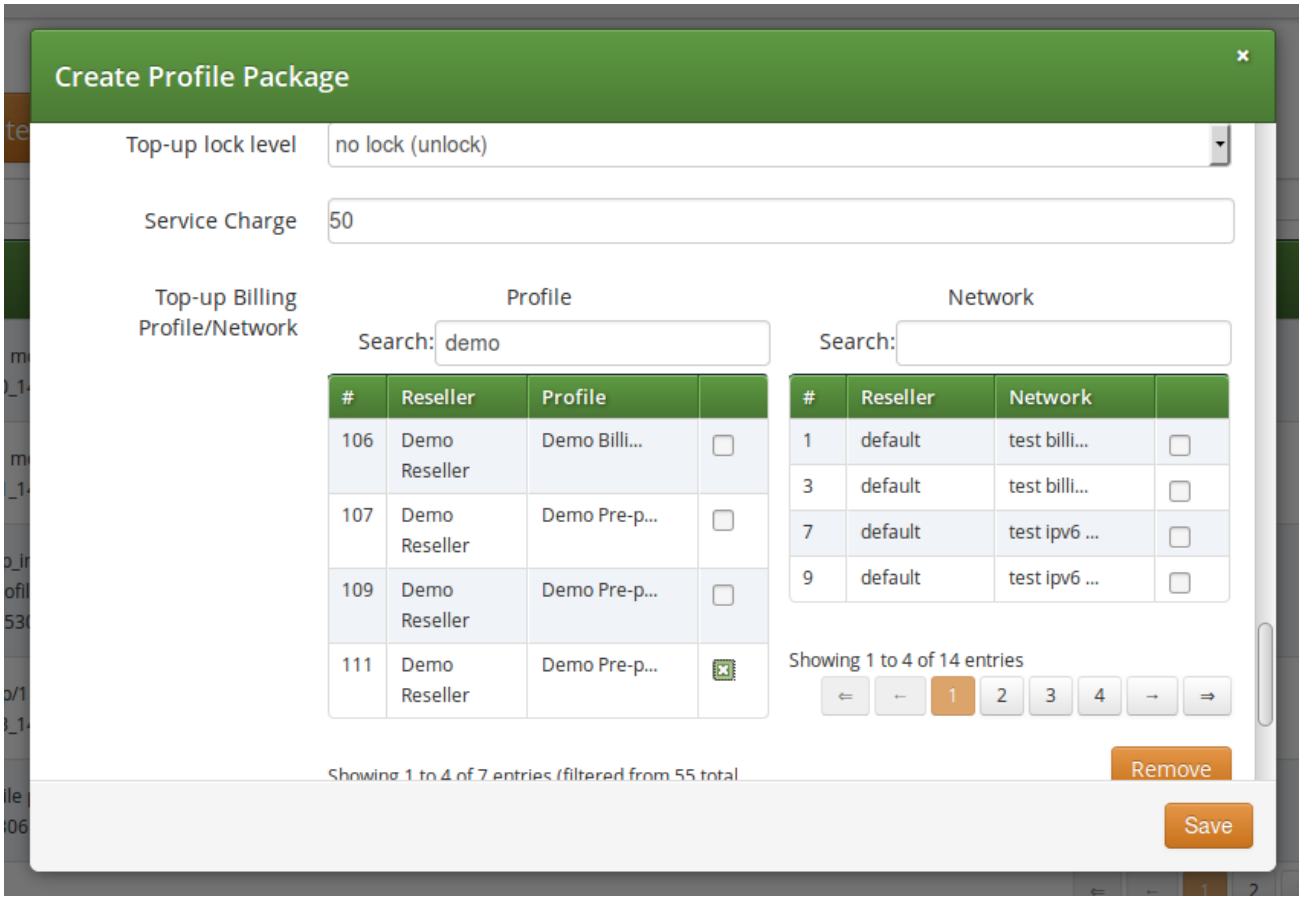


Figure 36. Balance Top-up Settings

5. Assigning a profile package to a customer

Copyright © Sipwise GmbH - All rights reserved

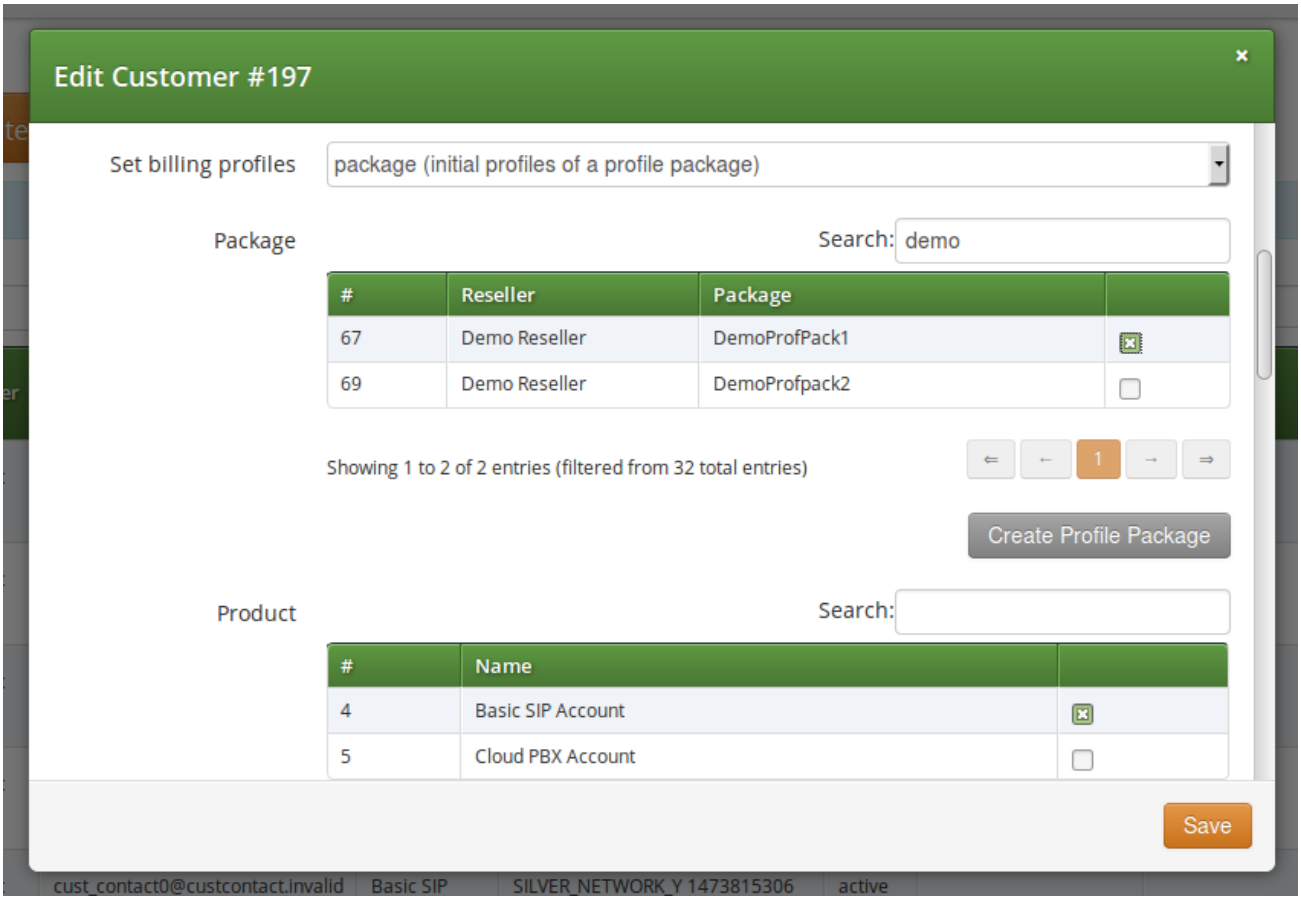


Figure 37. Assigning Profile Package to Customer

Interval start mode: top-up interval; carry-over: timely

Profile package setup:

- initial_balance: 1.0 euro
- balance_interval: 30 "day(s)"
- interval_start_mode: "topup_interval"
- carry_over_mode: "timely"
- timely_duration: 12 "day(s)"
- underrun_lock_threshold: 0.7 euro
- underrun_profile_threshold: 5.0 euro
- underrun_lock_level:...

Copyright © Sipwise GmbH - All rights reserved

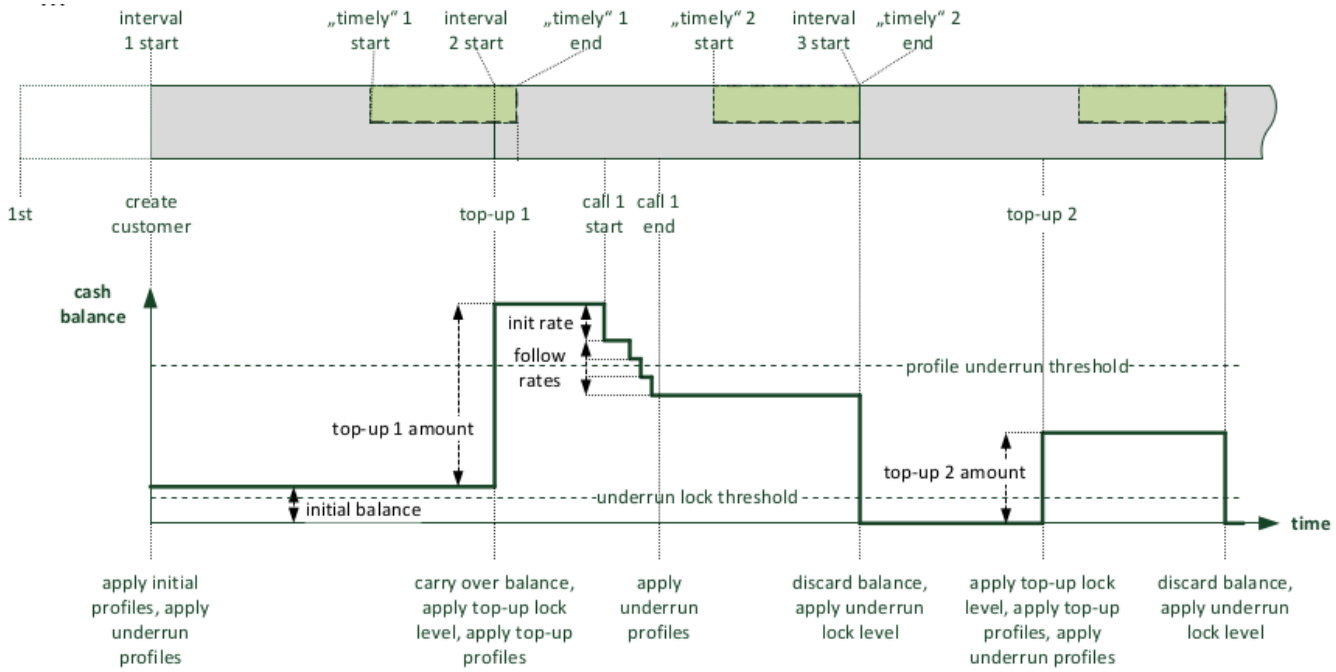


Figure 38. Example: Top-up Interval and Timely Carry-over

Interval start mode: top-up to top-up; carry-over: always

- initial_balance: 1.0 euro
- balance_interval: 30 "day(s)"
- interval_start_mode: "topup"
- carry_over_mode: "carry-over"
- notopup_discard_intervals: 1
- underrun_lock_threshold: 0.7 euro
- underrun_profile_threshold: 5.0 euro
- underrun_lock_level:...

Copyright © Sipwise GmbH - All rights reserved

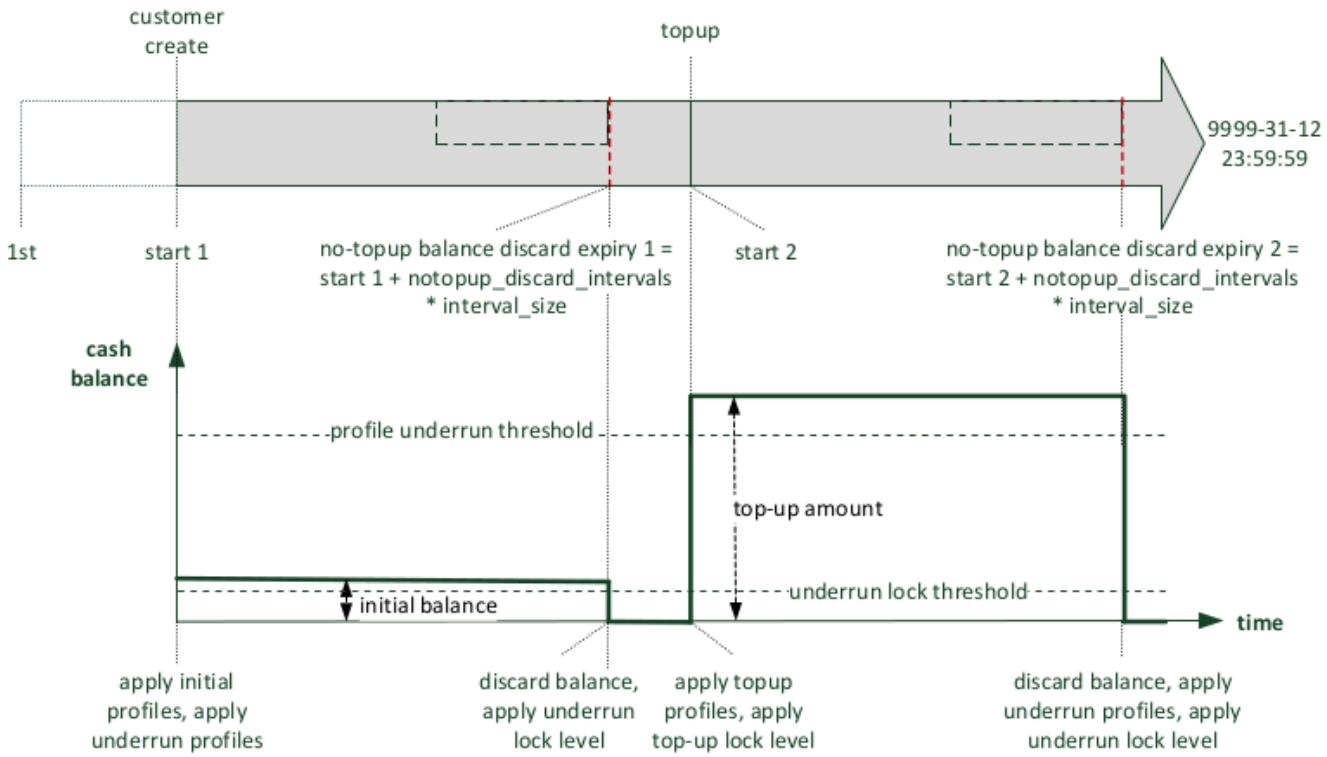


Figure 39. Example: Top-up and Always Carry-over

REST API

The new /api/profilepackages/ REST API resource makes it possible to manage billing profile package container entities, that aggregate settings of profile packages.

A sample JSON structure follows:

Copyright © Sipwise GmbH - All rights reserved

```
{
  "reseller_id" : 1,
  "status" : "active",
  "name" : "demo profile package",
  "description" : "package for 10€ ...",
  "balance_interval_start_mode" : "1st",
  "balance_interval_value" : 1,
  "balance_interval_unit" : "month",
  "carry_over_mode" : "carry_over",
  "timely_duration_unit" : null,
  "timely_duration_value" : null,
  "initial_balance" : 0,
  "initial_profiles" : [...], // required default, e.g. same as
„topup_profiles“
  "notopup_discard_intervals" : null,
  "underrun_lock_threshold" : 0,
  "underrun_lock_level" : 4,
  "underrun_profile_threshold" : 5,
  "underrun_profiles" : [...],
  "service_charge" : 10,
  "topup_lock_level" : null,
  "topup_profiles" : [ {
                         "network_id" : null, // any network
                         "profile_id" : 29
                       },
                       {
                         "network_id" : 2, // a specific billing network
                         "profile_id" : 30
                       },
                     ],
  ...
}
```

6.5.4. Vouchers

Vouchers are a typical mean of topping-up an account balance in pre-paid billing scenarios.

The definition of a voucher in the database may succeed via:

- manual entry of voucher data on the administrative web panel or through the REST API
- bulk-uploading of vouchers using a CSV (comma separated value) formatted file

In order to manage vouchers the administrator has to navigate to: *Settings Vouchers Create Billing Voucher* or select an existing one and press *Edit* button.

Copyright © Sipwise GmbH - All rights reserved

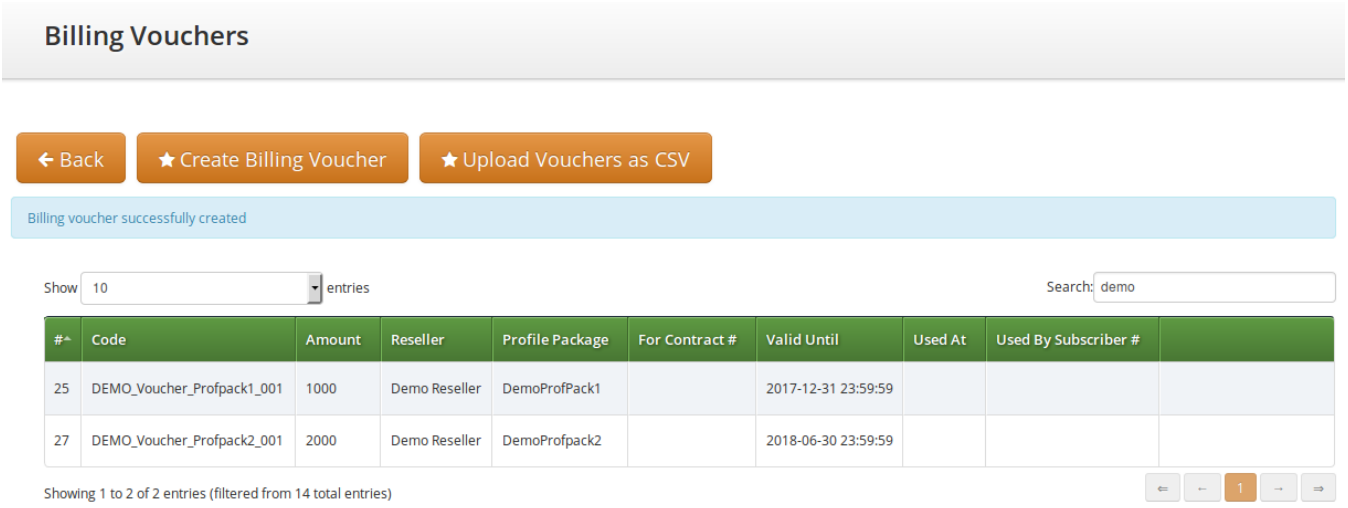


Figure 40. List of Vouchers

Properties of Vouchers

- Code: the unique code of the voucher which assures that a voucher can be used only once; this property is encrypted and displayed on the web panel to authorized users only
- Amount: the amount of money the voucher represents
- Valid until: end of validity period

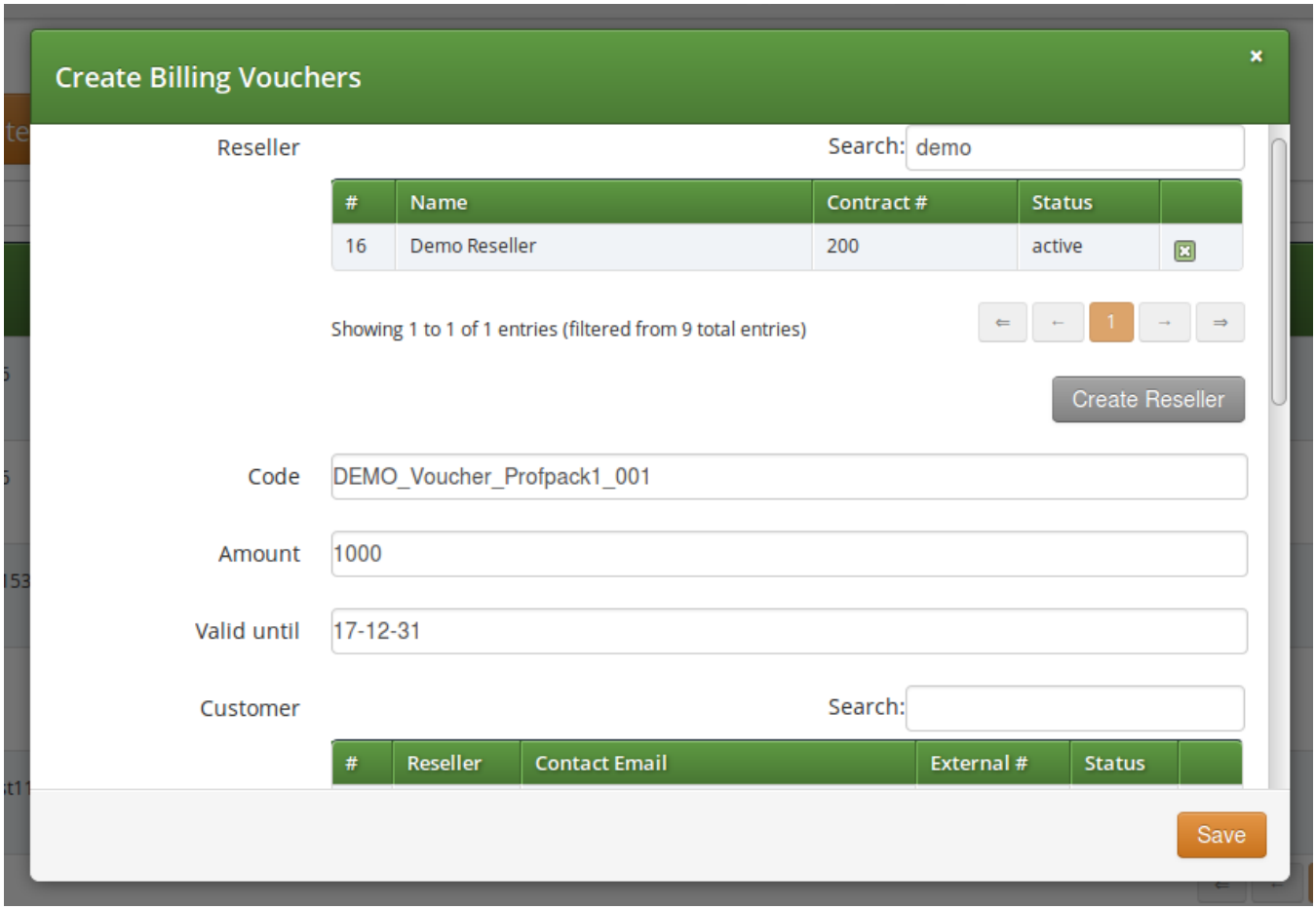


Figure 41. Voucher's Main Properties

Copyright © Sipwise GmbH - All rights reserved

Setting following properties of a voucher is optional:

- Customer: the *Customer* whom the voucher will be assigned to; subscribers of other customers can not redeem the voucher
- Package: vouchers may be associated with profile packages; if done so, some changes will be applied to the *Customer* for whom the voucher is redeemed with the top-up event:

  applying top-up profile mappings starting with the time of the top-up

  subtracting the new package's service charge from the voucher amount

  resizing the current balance interval for a gapless transition, if the new package has a different interval start mode (e.g. from "create" to "1st")

  if a new balance interval starts with the top-up, the carry-over mode of the customer's previous package applies

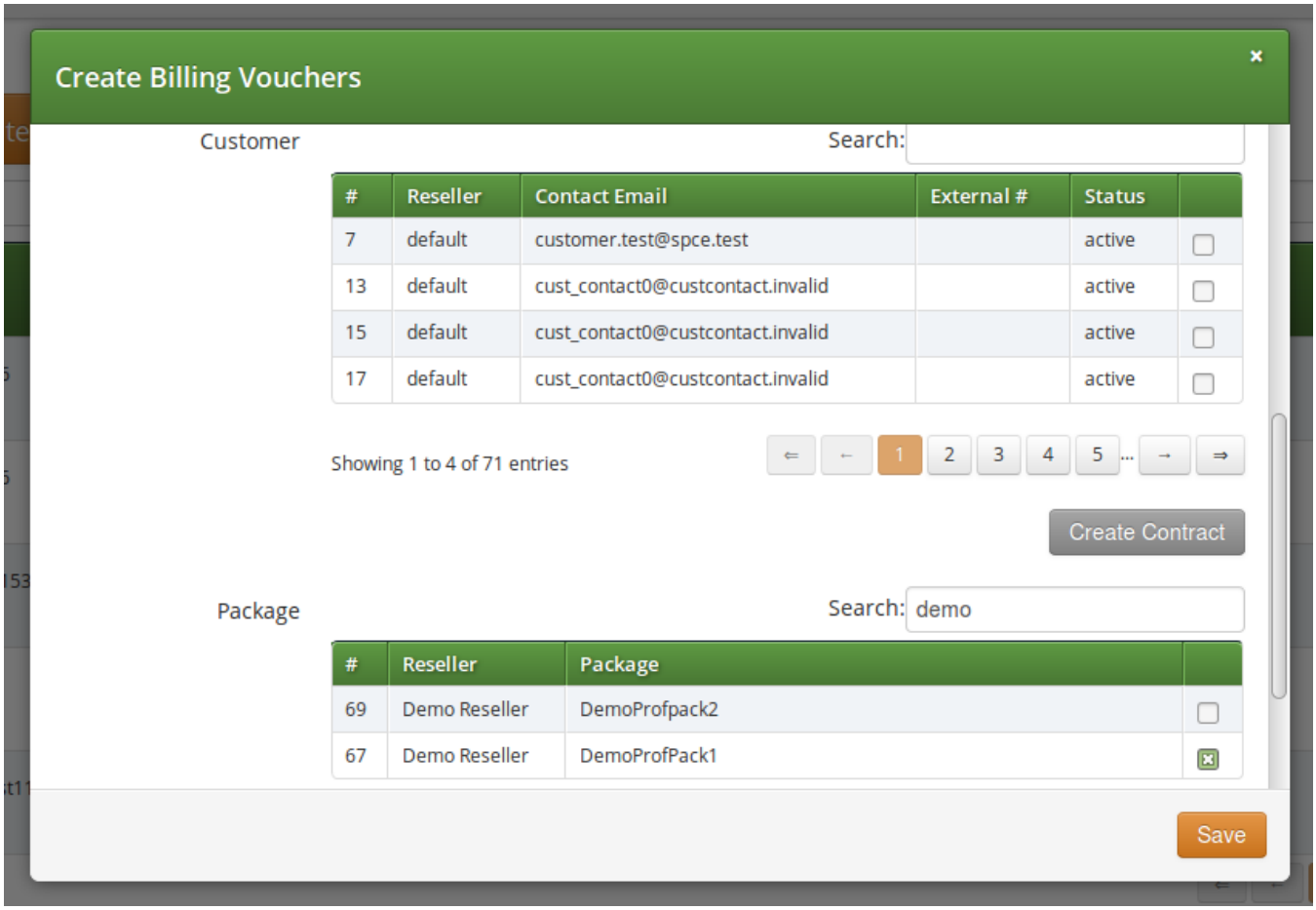


Figure 42. Voucher: Customer and Profile Package

REST API

Vouchers can be created and managed using the /api/vouchers/ REST API resource. This resource restricts invasive operations (POST, PUT, PATCH, DELETE) to authorized users.

Copyright © Sipwise GmbH - All rights reserved

```
{
  "amount" : 1000,
  "customer_id" : null, //do not restrict to a specific customer
  "valid_until" : "2017-06-05 23:59:59",
  "package_id" : "571", //switch to profile package
  "reseller_id" : 1,
  "code" : "SILVER_1_1437974823"
}
```

6.5.5. Top-up

A customer's administrator or subscriber can perform a top-up to increase the contract's cash balance. The Sipwise C5 platform supports two means of topping-up the balance:

1. Top-up Cash: Directly specify the cash amount to add
2. Top-up Voucher: Specify the code of a voucher, which was set up in advance

The Sipwise C5 platform provides 2 interfaces to perform top-ups:

1. through the REST API: use a CRM or third-party REST-API Broker (which i.e. coordinates with an App-Store purchase process) to finally instruct Sipwise C5 to perform a top-up. This is the **recommended** method.
2. through the administrative web interface:

   One has to select the *Customer*, then *Details Contract Balance* and finally press *Top-up Cash* or *Top-up Voucher*.

Top-up Cash

When doing top-up with cash one needs to supply the amount of top-up in the currency of the customer contract. Optionally one can assign a *Profile Package* to the top-up event which will activate that profile package for the customer.

Copyright © Sipwise GmbH - All rights reserved

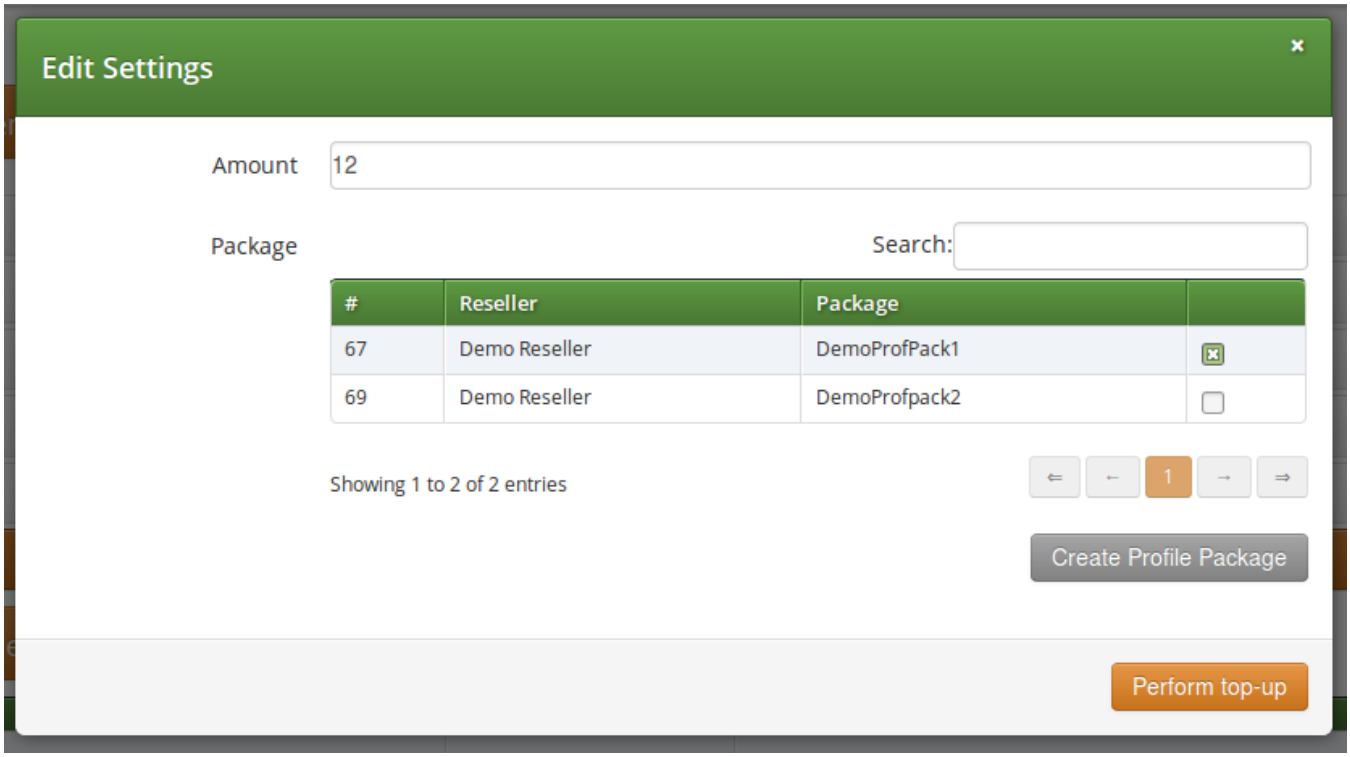


Figure 43. Balance Top-up with Cash

It is also possible to perform top-up through the **REST API**: POST /api/topupcash

```
{
  "subscriber_id" : "73",
  "amount" : 100,
  "package_id" : null,
}
```

Top-up Voucher

Selecting *Top-up Voucher* option will provide a simple list of available vouchers from which the administrator can choose the voucher. If a *Profile Package* is assigned to the voucher, that package will be activated for the customer on the top-up event.

Copyright © Sipwise GmbH - All rights reserved

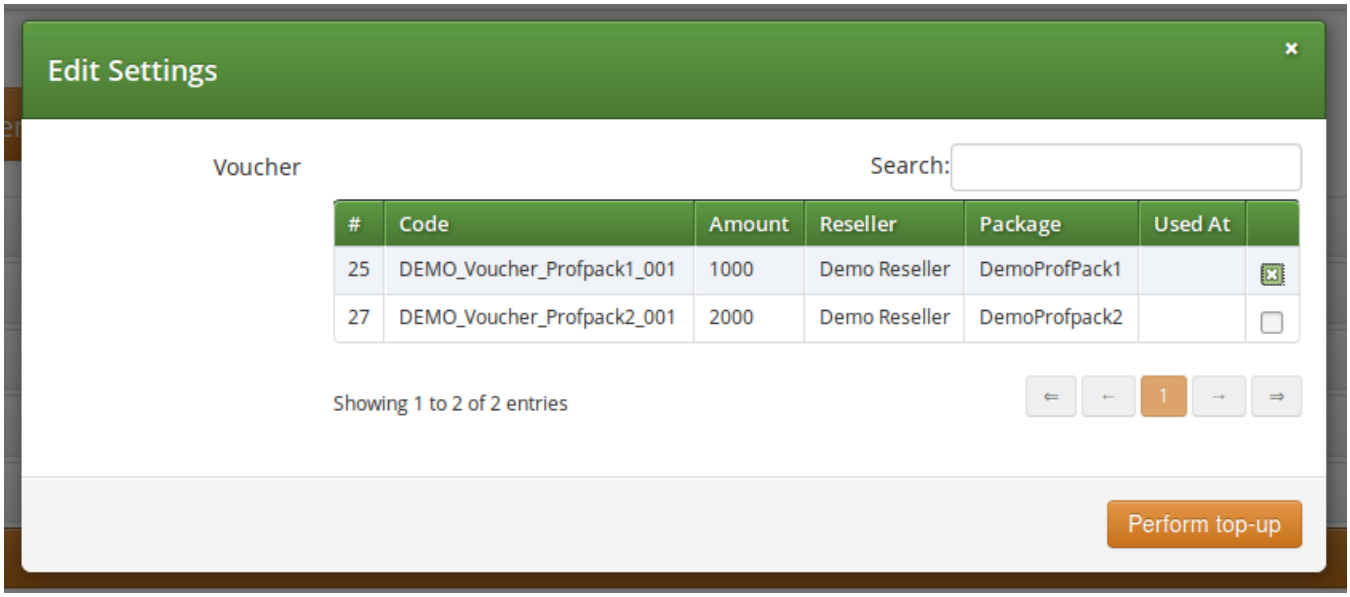


Figure 44. Balance Top-up with Voucher

It is also possible to perform top-up through the **REST API**: POST /api/topupvouchers

```
{
  "subscriber_id" : "73",
  "code" : "SILVER_1_1437974390“
  "request_token" : "uuid_from_3rdparty_relay" // optional request
identifier
                                                // for lookups in the
top-up log
}
```

6.5.6. Balance Overviews

The actual contract balance and logs of top-up or balance interval change events are a kind of financially important information and that's why those are provided on the administrative web interface for each customer. One should navigate to: *Settings Customers select the customer Details*.

The various information details available on the web interface are discussed in subsequent sections of the handbook.

Contract Balance

This part of the overviews shows the actual financial state of the customer's balance and the current profile package and balance interval.

 Copyright © Sipwise GmbH - All rights reserved

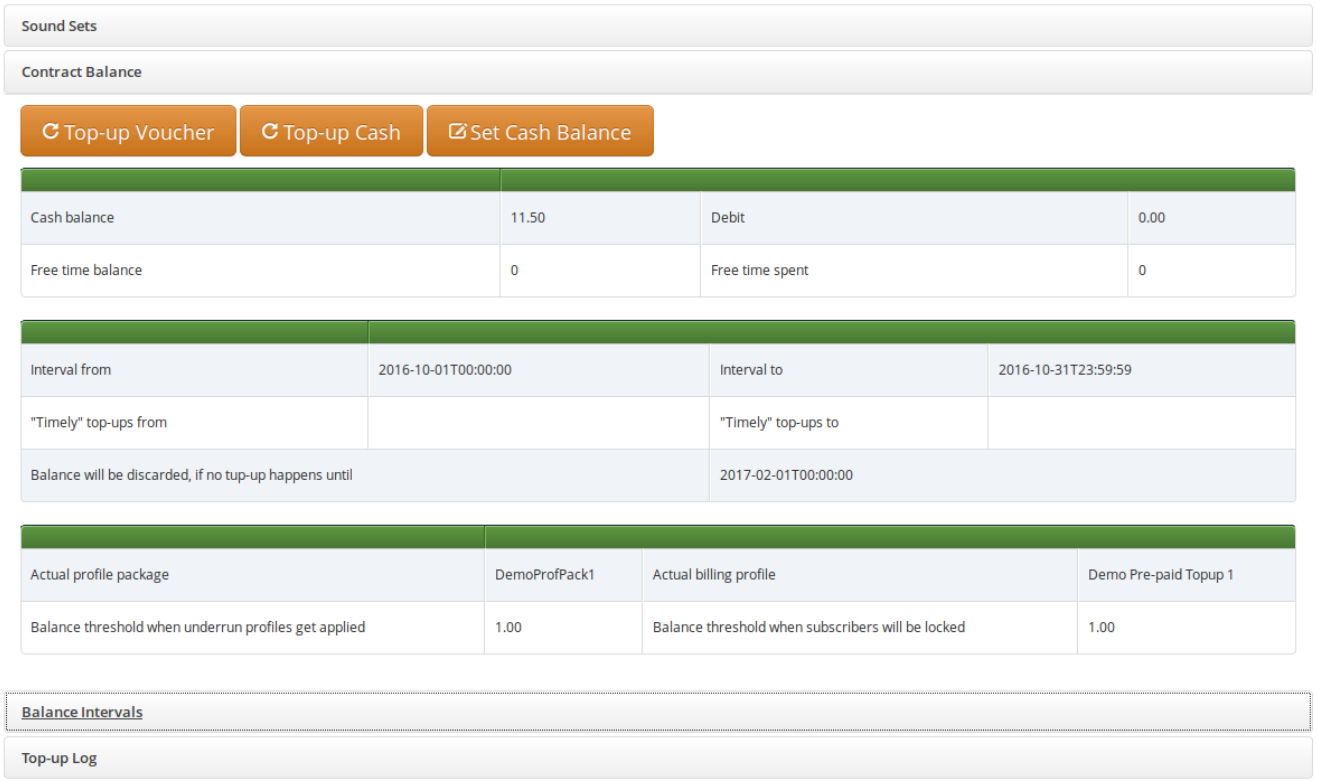


Figure 45. Contract Balance Status

Another functionality assigned to *Contract Balance* section is the manual top-up. Both top-up with cash and top-up with voucher can be performed from here.

Balance Intervals

This table shows the balance intervals that have been in use, including the current interval.

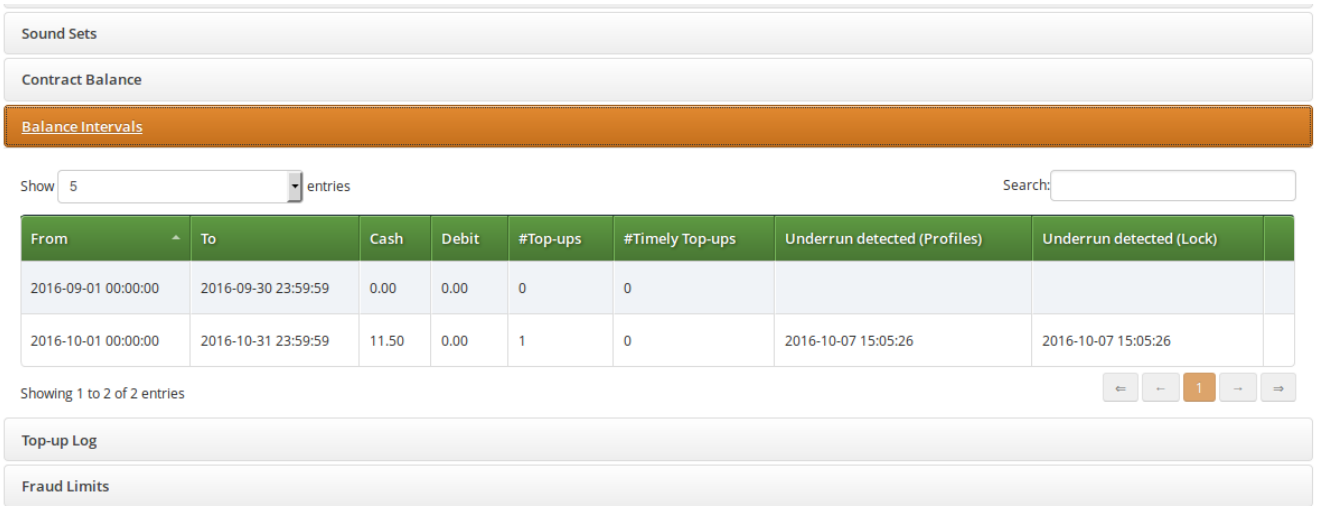


Figure 46. List of Balance Intervals

Content of the balance intervals table is:

- From, To: starting and end points of the time interval
- Cash: the contract's cash balance value at the end of the interval (former int.), or currently (actual

Copyright © Sipwise GmbH - All rights reserved

int.)

- Debit: the total spent amount of money in the actual interval

> **NOTE** While "Cash" shows the remaining amount, "Debit" shows the spent amount. With a post-paid billing scenario only "Debit" field would be populated, with pre-paid both fields will display an amount.

- No. of Top-ups: how many top-up events happened within the interval
- No. of Timely Top-ups: how many timely top-up events happened within the interval
- Underrun detected (Profiles or Lock): the time of last underrun event when either an underrun billing profile, or a subscriber lock was activated

Top-up Log

Each successful or failing top-up request has to be logged. The log records represent an audit trail and reflect any data changes in the course of the top-up request.

In case of an error during the top-up operation the error message and any parseable fields of failed top-up attempts is recorded.

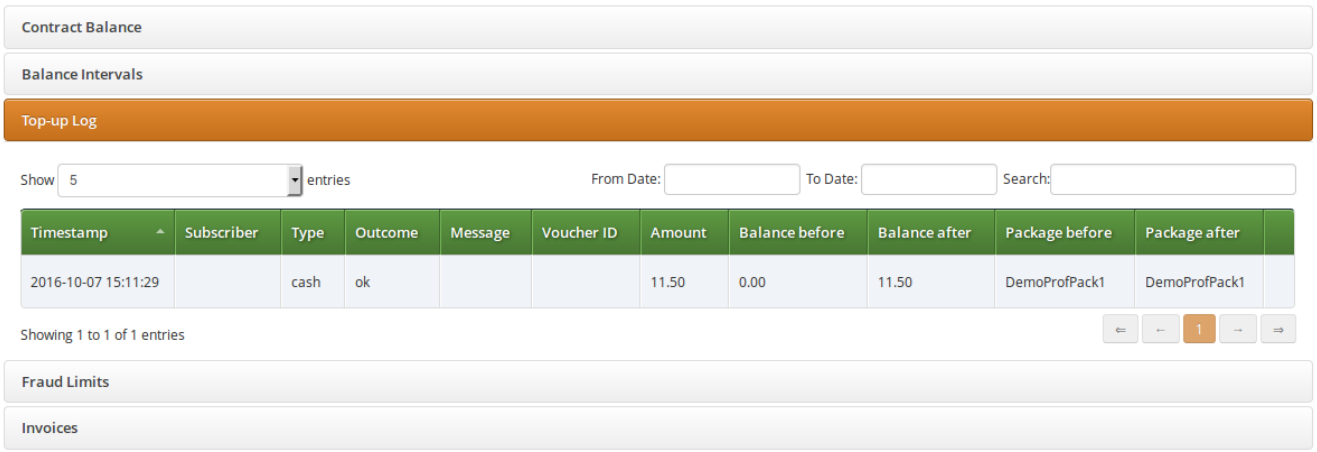


Figure 47. Balance Top-up Log

Content of the top-up log table is:

- Timestamp: when the top-up happened
- Subscriber: the ID of the subscriber who performed the top-up
- Type: cash or voucher
- Outcome: ok or failed
- Message: error message, if Outcome="failed"
- Voucher ID: ID of voucher, if Type="voucher"
- Amount: the amount by which the balance was modified (after the *Service Charge* was subtracted from the voucher's value)
- Balance before: balance's value before top-up
- Balance after: balance's value after top-up

 Copyright © Sipwise GmbH - All rights reserved

- Package before: the name of the *Profile Package* that was active before top-up
- Package after: the name of the *Profile Package* that became active after top-up

The top-up log table can also be queried using the readonly /api/topuplogs **REST API** resource.

An example of the response:

```
{
  "_embedded" : {
    "ngcp:topuplogs" : [{
      "_links" : {...},
      "amount" : null,
      "cash_balance_after" : null,
      "cash_balance_before" : null,
      "contract_balance_after_id" : null,
      "contract_balance_before_id" : null,
      "contract_id" : 2565,
      "id" : 373,
      "lock_level_after" : null,
      "lock_level_before" : null,
      "message" : ..., //error reason
      "outcome" : "failed",
      "package_after_id" : null,
      "package_before_id" : null,
      "profile_after_id" : null,
      "profile_before_id" : null,
      "request_token" : "1444956281_6", // = “panel“ for panel UI
requests
      "subscriber_id" : 1804,
      "timestamp" : "2015-10-16 02:45:19",
      "type" : "voucher", // "cash" or "voucher"
      "username" : "administrator",
      "voucher_id" : null }]
  },
  "_links" : { ... },
  "total_count" : 1
}
```

6.5.7. Usage Examples

After getting to know the concepts of customized billing solution on Sipwise C5 platform, it's worth seeing some practical examples for the usage of those advanced features.

The starting point is the setup of *Profile Packages* for our fictive customers: A, B and C. There are 4 different packages defined, with corresponding vouchers:

- **Initial:**

  Balance interval: 1 month

  Timely duration: 1 month

Copyright © Sipwise GmbH - All rights reserved

Interval start mode: topup_interval

Carry-over mode: carry_over_timely

- **Silver:**

  Balance interval: 1 month

  Timely duration: 1 month

  Interval start mode: "topup_interval"

  Carry-over mode: "carry_over_timely"

  Service charge: 2 EUR

  Underrun lock level: "no lock"

  Voucher value: 10 EUR

- **Gold:**

  Balance interval: 1 month

  Interval start mode: "topup_interval"

  Carry-over mode: "carry_over"

  Service charge: 5 EUR

  Underrun lock level: "no lock"

  Voucher value: 20 EUR

- **Extension:**

  Balance interval: 1 month

  Timely duration: 1 month

  Interval start mode: "topup_interval"

  Carry-over mode: "carry_over_timely"

  Service charge: 2 EUR

  Underrun lock level: "no lock"

  Voucher value: 2 EUR

Customer A—Silver Package

1. Customer A tops up 10 EUR with a “silver” voucher. 2 EUR are deducted as service charge. Remaining balance is 8 EUR starting on the date of the top- up.
2. Customer A doesn’t top-up balance within the next month, so remaining balance is set to 0 after one month, and billing profiles and lock levels are set to the balance-underrun definition of the “silver” package.

 Copyright © Sipwise GmbH - All rights reserved

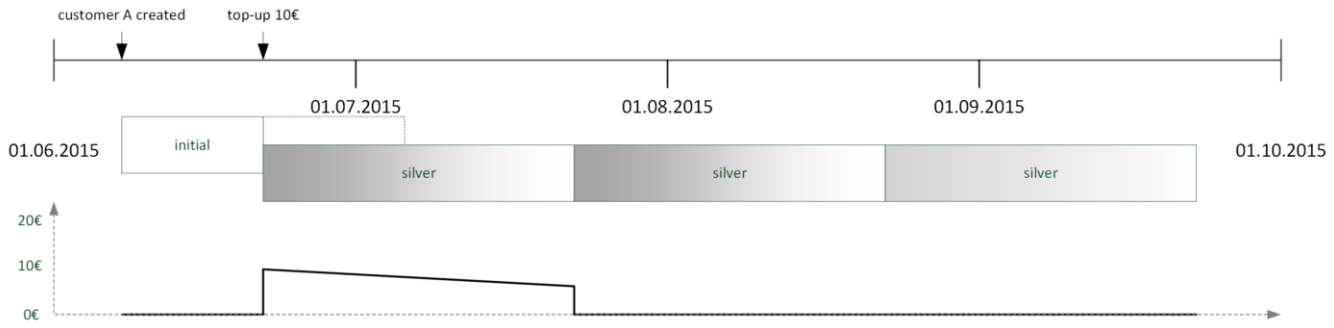


Figure 48. Usage Example: Silver Package

Customer B—Silver and Extension Package

1. Customer B tops up 10 EUR with the “silver” voucher. 2 EUR are deducted as service charge. Remaining balance is 8 EUR starting on the date of the top-up.
2. Customer B tops up 2 EUR using an “extension” voucher on the last day. 2 EUR are deducted as service charge and the interval is extended for one month, carrying over his old balance.
3. Customer B doesn't top-up balance within the next month, so remaining balance is set to 0 after the month, and billing profiles and lock levels are set to the balance-underrun definition of the “extension” package.

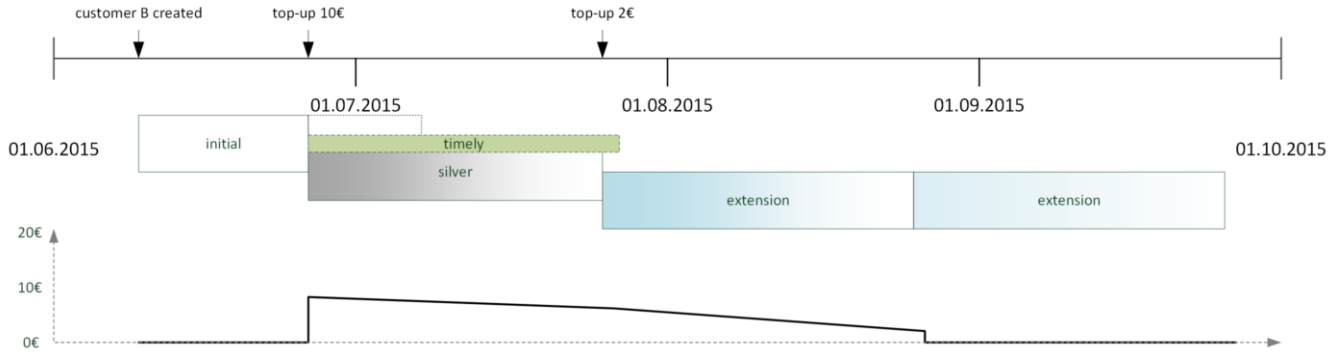


Figure 49. Usage Example: Silver + Extension Package

Customer C—Gold Package

Customer C tops up 20 EUR with the “gold” voucher. 5 EUR are deducted as service charge. Remaining balance is 15 EUR starting on the date of the top-up. Balance is carried over after each month until used up.

Copyright © Sipwise GmbH - All rights reserved

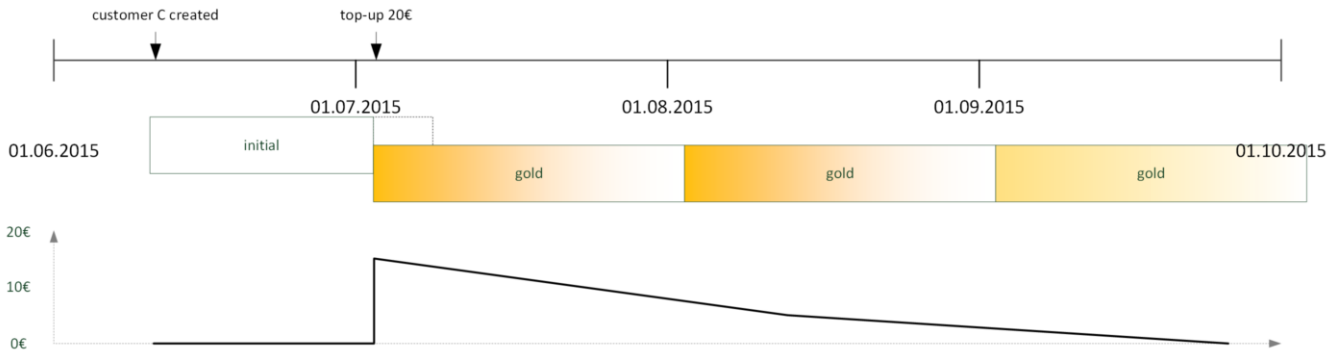


Figure 50. Usage Example: Gold Package

6.6. Notes on Billing and Call Rating

Cash balance with post-paid billing profile

Customers with a post-paid billing profile may have a positive account cash balance value. This is the regular case when using a post-paid billing profile showing a *free cash* greater than '0'.

TIP You can set the free cash (and the free time) in the billing profile. The account balance will be set and managed (i.e. refilled or carried over) automatically for subsequent balance intervals.

In case the account has a positive cash balance, the cost of the call will be deducted from that balance and not considered as additional cost of that particular call for the customer.

IMPORTANT The rating engine (*ngcp-rate-o-mat*) in Sipwise C5 will write '0' instead of the real cost of a call in the CDR, if the source customer's (who initiated the call) account has a positive cash balance! The purpose of this is to reflect the usage of free cash in the CDR for the particular call.

NOTE It might happen, for instance, that a customer's billing profile is changed from pre-paid to post-paid, and the customer already had a positive cash balance on his account. In that case the same call rating mechanism is involved as for the free cash.

6.7. Billing Data Export

Regular billing data export is done using CSV (*comma separated values*) files which may be downloaded from the platform using the *cdrexport* user which has been created during the installation.

There are two types of exports. One is *CDR* (Call Detail Records) used to charge for calls made by subscribers, and the other is *EDR* (Event Detail Records) used to charge for provisioning events like enabling certain features.

6.7.1. Glossary of Terms

Billing records contain fields that hold data of various entities that play a role in the phone service offered by Sipwise C5. For a better understanding of billing data please refer to the glossary provided here:

 Copyright © Sipwise GmbH - All rights reserved

- **Account**: the customer's account that is charged for calls of its subscriber(s)
- **Carrier**: a SIP peer that sends incoming calls to, or receives outgoing calls from NGCP. A carrier may charge fees for the outgoing calls from Sipwise C5 (outbound billing fee), or for the incoming calls to Sipwise C5 (inbound billing fee).
- **Contract**: the service contract that represents a customer, a reseller or a SIP peer; a contract on Sipwise C5 contains the billing profile (billing fees) too
- **Customer**: the legal entity that represents any number of subscribers; this entity receives the bills for calls of its subscriber(s)
- **Provider**: either the reseller that holds a subscriber who is registered on NGCP, or the SIP peer that handles calls between an external subscriber and NGCP
- **Reseller**: the entity who is the direct, administrative service provider of a group of customers and subscribers registered on NGCP; Sipwise C5 operator may also charge a reseller for the calls initiated or received by its subscribers
- **User**: the subscriber who either is registered on NGCP, or is an external call party

6.7.2. File Name Format

In order to be able to easily identify billing files, the file names are constructed by the following fixed-length fields:

```
<prefix><separator><version><separator><timestamp><separator><sequence
number><suffix>
```

The definition of the specific fields is as follows:

Table 5. CDR/EDR export file name format

File name element	Length	Description
<prefix>	7	A fixed string. Always sipwise.
<separator>	1	A fixed character. Always _.
<version>	3	The format version, a three digit number. Currently 007.
<timestamp>	14	The file creation timestamp in the format YYYYMMDDhhmmss.
<sequence number>	10	A unique 10-digit zero-padded sequence number for quick identification.
<suffix>	4	A fixed string. Always .cdr or .edr.

A valid example filename for a CDR billing file created at 2012-03-10 14:30:00 and being the 42nd file exported by the system, is:

```
sipwise_007_20130310143000_0000000042.cdr
```

Copyright © Sipwise GmbH - All rights reserved

6.7.3. File Format

Each billing file consists of three parts: one header line, zero to 5000 body lines and one trailer line.

File Header Format

The billing file header is one single line, which is constructed by the following fields:

```
<version>,<number of records>
```

The definition of the specific fields is as follows:

Table 6. CDR/EDR export file header line format

Body Element	Length	Type	Description
<version>	3	zero-padded uint	The format version. Currently 007.
<number of records>	4	zero-padded uint	The number of body lines contained in the file.

A valid example for a Header is:

```
007,0738
```

File Body Format for Call Detail Records (CDR)

The body of a CDR consists of a minimum of zero and a default maximum of 5000 lines. The platform operator can configure the maximum number of lines kept in a file by updating the cdrexport.max_rows_per_file parameter in /etc/ngcp-config/config.yml file. Each line holds one call detail record in CSV format and is constructed by a configurable set of fields, all of them enclosed in single quotes.

The following table defines the **default set of fields** that are inserted into the CDR file, for exports related to *system* scope. The list of fields is defined in /etc/ngcp-config/config.yml file, cdrexport.admin_export_fields parameter.

Table 7. Default set of system CDR fields

Body Element	Length	Type	Description
CDR_ID	1-10	uint	Internal CDR ID.
UPDATE_TIME	19	timestamp	Timestamp of last modification, including date and time (with seconds precision).

Copyright © Sipwise GmbH - All rights reserved

Body Element	Length	Type	Description
SOURCE_USER_ID	36	string	Internal UUID of calling party subscriber. Value is 0 if calling party is external.
SOURCE_PROVIDER_ID	0-255	string	Internal ID of the contract of calling party provider (i.e. reseller or peer).
SOURCE_EXTERNAL_SUBSCRIBER_ID	0-255	string	External, arbitrary ID of calling party subscriber. (A string value shown as "External ID" property of an Sipwise C5 subscriber.)
SOURCE_SUBSCRIBER_ID	1-11	uint	Internal ID of calling party subscriber. Value is 0 if calling party is external.
SOURCE_EXTERNAL_CONTRACT_ID	0-255	string	External, arbitrary ID of calling party customer. (A string value shown as "External ID" property of an Sipwise C5 customer/peer.)
SOURCE_ACCOUNT_ID	1-11	uint	Internal ID of calling party customer.
SOURCE_USER	0-255	string	SIP username of calling party.
SOURCE_DOMAIN	0-255	string	SIP domain of calling party.
SOURCE_CLI	0-64	string	CLI of calling party in E.164 format.
SOURCE_CLIR	1	uint	1 for calls with CLIR, 0 otherwise.
SOURCE_IP	0-64	string	IP Address of the calling party.
DESTINATION_USER_ID	36	string	Internal UUID of called party subscriber. Value is 0 if called party is external.
DESTINATION_PROVIDER_ID	0-255	string	Internal ID of the contract of called party provider (i.e. reseller or peer).
DESTINATION_EXTERNAL_SUBSCRIBER_ID	0-255	string	External, arbitrary ID of called party subscriber. (A string value shown as "External ID" property of an Sipwise C5 subscriber.)
DESTINATION_SUBSCRIBER_ID	1-11	uint	Internal ID of called party subscriber. Value is 0 if calling party is external.
DESTINATION_EXTERNAL_CONTRACT_ID	0-255	string	External, arbitrary ID of called party customer. (A string value shown as "External ID" property of an Sipwise C5 customer/peer.)
DESTINATION_ACCOUNT_ID	1-11	uint	Internal ID of called party customer.

Copyright © Sipwise GmbH - All rights reserved

Body Element	Length	Type	Description
DESTINATION_USER	0-255	string	Final SIP username of called party.
DESTINATION_DOMAIN	0-255	string	Final SIP domain of called party.
DESTINATION_USER_IN	0-255	string	Incoming SIP username of called party, after applying inbound rewrite rules.
DESTINATION_DOMAIN_IN	0-255	string	Incoming SIP domain of called party, after applying inbound rewrite rules.
DESTINATION_USER_DIALED	0-255	string	The user-part of the SIP Request URI as received by NGCP.
PEER_AUTH_USER	0-255	string	Username used to authenticate towards peer.
PEER_AUTH_REALM	0-255	string	Realm used to authenticate towards peer.
CALL_TYPE	3-4	string	The type of the call - one of: call: normal call cfu: call forward unconditional cfb: call forward busy cft: call forward timeout cfna: call forward not available cfs: call forward for SMS cfr: call forward on response cfo: call forward on overflow

Copyright © Sipwise GmbH - All rights reserved

Body Element	Length	Type	Description
CALL_STATUS	2-8	string	The final call status - one of: ok: successful call busy: called party busy noanswer: no answer from called party cancel: cancel from caller offline called party offline timeout: no reply from called party other: unspecified, see CALL_CODE field for details
CALL_CODE	3	string	The final SIP status code.
INIT_TIME	23	timestamp	Timestamp of call initiation (SIP 'INVITE' received from calling party). Includes date, time with milliseconds (3 decimals).
START_TIME	23	timestamp	Timestamp of call establishment (final SIP response received from called party). Includes date, time with milliseconds (3 decimals).
DURATION	4-13	fixed precision (3 decimals)	Length of call (calculated from START_TIME) including milliseconds (3 decimals).
END_TIME	23	timestamp	START_TIME plus DURATION.
INIT_TIME_TRUNCATED	19	timestamp	INIT_TIME without milliseconds.
START_TIME_TRUNCATED	19	timestamp	START_TIME without milliseconds.
END_TIME_TRUNCATED	19	timestamp	END_TIME without milliseconds.
TIMEZONE	100	string	The name of the local subscriber's active timezone, defined by the contact of the subscriber/contract/reseller. The caller's timezone is used for outgoing calls. For calls from external subscribers, the callee's timezone is used. System timezone (defined in config.yml) is used for transit calls.

Copyright © Sipwise GmbH - All rights reserved

Body Element	Length	Type	Description
INIT_TIME_LOCALIZED	23	timestamp	INIT_TIME, converted to TIMEZONE.
START_TIME_LOCALIZED	23	timestamp	START_TIME, converted to TIMEZONE.
END_TIME_LOCALIZED	23	timestamp	END_TIME, converted to TIMEZONE.
INIT_TIME_LOCALIZED_TRUNCATED	19	timestamp	INIT_TIME_LOCALIZED without milliseconds.
START_TIME_LOCALIZED_TRUNCATED	19	timestamp	START_TIME_LOCALIZED without milliseconds.
END_TIME_LOCALIZED_TRUNCATED	19	timestamp	END_TIME_LOCALIZED without milliseconds.
CALL_ID	0-255	string	The SIP Call-ID.
RATING_STATUS	2-7	string	The internal rating status of the CDR - one of: unrated: not rated ok: successfully rated failed: error while rating Currently always ok or unrated, depending on whether rating is enabled or not.
RATED_AT	0-19	datetime	Time of rating, including date and time (with seconds precision). Empty if CDR is not rated.
SOURCE_CARRIER_COST	7-14	fixed precision (6 decimals)	The originating carrier cost that the carrier (i.e. SIP peer) charges for the calls routed to his network, or empty if CDR is not rated. *PLEASE NOTE: Only available in system exports, not for resellers.*
SOURCE_CUSTOMER_COST	7-14	fixed precision (6 decimals)	The originating customer cost, or empty if CDR is not rated.

Copyright © Sipwise GmbH - All rights reserved

Body Element	Length	Type	Description
SOURCE_CARRIER_ZONE	0-127	string	Name of the originating carrier billing zone, or onnet if data is not available. *PLEASE NOTE: Only available in system exports, not for resellers.*
SOURCE_CUSTOMER_ZONE	0-127	string	Name of the originating customer billing zone, or empty if CDR is not rated.
SOURCE_CARRIER_DETAIL	0-127	string	Description of the originating carrier billing zone, or platform internal if data is not available. *PLEASE NOTE: Only available in system exports, not for resellers.*
SOURCE_CUSTOMER_DETAIL	0-127	string	Description of the originating customer billing zone, or empty if CDR is not rated.
SOURCE_CARRIER_FREE_TIME	1-10	uint	The number of free time seconds used on originating carrier side, or empty if CDR is not rated. *PLEASE NOTE: Only available in system exports, not for resellers.*
SOURCE_CUSTOMER_FREE_TIME	1-10	uint	The number of free time seconds used from the originating customer's account balance, or empty if CDR is not rated.
DESTINATION_CARRIER_COST	7-14	fixed precision (6 decimals)	The terminating carrier cost, or empty if CDR is not rated. *PLEASE NOTE: Only available in system exports, not for resellers.*
DESTINATION_CUSTOMER_COST	7-14	fixed precision (6 decimals)	The terminating customer cost, or empty if CDR is not rated.
DESTINATION_CARRIER_ZONE	0-127	string	Name of the terminating carrier billing zone, or onnet if data is not available. *PLEASE NOTE: Only available in system exports, not for resellers.*

Copyright © Sipwise GmbH - All rights reserved

Body Element	Length	Type	Description
DESTINATION_CUSTOMER_ZONE	0-127	string	Name of the terminating customer billing zone, or empty if CDR is not rated.
DESTINATION_CARRIER_DETAIL	0-127	string	Description of the terminating carrier billing zone, or empty if CDR is not rated. *PLEASE NOTE: Only available in system exports, not for resellers.*
DESTINATION_CUSTOMER_DETAIL	0-127	string	Description of the terminating customer billing zone, or empty if CDR is not rated.
DESTINATION_CARRIER_FREE_TIME	1-10	uint	The number of free time seconds used on terminating carrier side, or empty if CDR is not rated. *PLEASE NOTE: Only available in system exports, not for resellers.*
DESTINATION_CUSTOMER_FREE_TIME	1-10	uint	The number of free time seconds used from the terminating customer's account balance, or empty if CDR is not rated.
SOURCE_RESELLER_COST	7-14	fixed precision (6 decimals)	The originating reseller cost, or empty if CDR is not rated. *PLEASE NOTE: Only available in system exports, not for resellers.*
SOURCE_RESELLER_ZONE	0-127	string	Name of the originating reseller billing zone, or empty if CDR is not rated. *PLEASE NOTE: Only available in system exports, not for resellers.*
SOURCE_RESELLER_DETAIL	0-127	string	Description of the originating reseller billing zone, or empty if CDR is not rated. *PLEASE NOTE: Only available in system exports, not for resellers.*

Copyright © Sipwise GmbH - All rights reserved

Body Element	Length	Type	Description
SOURCE_RESELLER_FREE_TIME	1-10	uint	The number of free time seconds used from the originating reseller's account balance, or empty if CDR is not rated. *PLEASE NOTE: Only available in system exports, not for resellers.*
DESTINATION_RESELLER_COST	7-14	fixed precision (6 decimals)	The terminating reseller cost, or empty if CDR is not rated. *PLEASE NOTE: Only available in system exports, not for resellers.*
DESTINATION_RESELLER_ZONE	0-127	string	Name of the terminating reseller billing zone, or empty if CDR is not rated. *PLEASE NOTE: Only available in system exports, not for resellers.*
DESTINATION_RESELLER_DETAIL	0-127	string	Description of the terminating reseller billing zone, or empty if CDR is not rated. *PLEASE NOTE: Only available in system exports, not for resellers.*
DESTINATION_RESELLER_FREE_TIME	1-10	uint	The number of free time seconds used from the terminating reseller's account balance, or empty if CDR is not rated. *PLEASE NOTE: Only available in system exports, not for resellers.*
<line_terminator>	1	string	Always \n (special char LF - ASCII 0x0A).

A valid example of one body line of a rated CDR is (line breaks added for clarity):

Copyright © Sipwise GmbH - All rights reserved

```
'15','2013-03-26 22:09:11','a84508a8-d256-4c80-a84e-
820099a827b0','1','','1','',
'2','testuser1','192.168.51.133','4311001','0','192.168.51.1',
'94d85b63-8f4b-43f0-b3b0-221c9e3373f2','1','','3','','4','testuser3',
'192.168.51.133','testuser3','192.168.51.133','testuser3','','','call','
ok','200',
'2013-03-25 20:24:50.890','2013-03-25 20:24:51.460','10.880','44449842',
'ok','2013-03-25 20:25:27','0.00','24.00','onnet','testzone','platform
internal',
'testzone','0','0','0.00','200.00','','foo','','foo','0','0',
'0.00','','','0','0.00','','','0'
```

The format of the **CDR export files generated for *resellers*** (as opposed to the complete system-wide export) is identical except for a few missing fields.

NOTE Please check the description of fields in the table above, in order to see which fields are omitted for *reseller* related CDR exports.

The list of fields for *reseller* CDR export is defined in /etc/ngcp-config/config.yml file, cdrexport.reseller_export_fields parameter.

Extra fields that can be exported to CDRs

Supplementary Data

There are fields in CDR database that contain **supplementary data** related to subscribers. This data is not used by Sipwise C5 for CDR processing but rather provides the system administrator with a possibility to include supplementary information in CDRs.

NOTE *This informational section is meant for problem solving / debugging purpose:* The supplementary data listed in following table is stored in `provisioning.voip_preferences` database table.

Table 8. Supplementary data in CDR fields

Body Element	Length	Type	Description
SOURCE_GPP0	0-255	string	Supplementary data field 0 of calling party.
SOURCE_GPP1	0-255	string	Supplementary data field 1 of calling party.
SOURCE_GPP2	0-255	string	Supplementary data field 2 of calling party.
SOURCE_GPP3	0-255	string	Supplementary data field 3 of calling party.
SOURCE_GPP4	0-255	string	Supplementary data field 4 of calling party.
SOURCE_GPP5	0-255	string	Supplementary data field 5 of calling party.
SOURCE_GPP6	0-255	string	Supplementary data field 6 of calling party.
SOURCE_GPP7	0-255	string	Supplementary data field 7 of calling party.

Copyright © Sipwise GmbH - All rights reserved

Body Element	Length	Type	Description
SOURCE_GPP8	0-255	string	Supplementary data field 8 of calling party.
SOURCE_GPP9	0-255	string	Supplementary data field 9 of calling party.
DESTINATION_GPP0	0-255	string	Supplementary data field 0 of called party.
DESTINATION_GPP1	0-255	string	Supplementary data field 1 of called party.
DESTINATION_GPP2	0-255	string	Supplementary data field 2 of called party.
DESTINATION_GPP3	0-255	string	Supplementary data field 3 of called party.
DESTINATION_GPP4	0-255	string	Supplementary data field 4 of called party.
DESTINATION_GPP5	0-255	string	Supplementary data field 5 of called party.
DESTINATION_GPP6	0-255	string	Supplementary data field 6 of called party.
DESTINATION_GPP7	0-255	string	Supplementary data field 7 of called party.
DESTINATION_GPP8	0-255	string	Supplementary data field 8 of called party.
DESTINATION_GPP9	0-255	string	Supplementary data field 9 of called party.

Account balance details (prepaid calls)

There are fields in CDR database that show **changes in cash or free time balance**. In addition to that, a history of billing packages / profiles may also be present, since Sipwise C5 vouchers, that are used to top-up, may also be set up to cause a transition of profile packages. (Which in turn can result in changing the billing profile/applicable fees). Therefore the billing package and profile valid at the time of the CDR are recorded and exposed as fields for CDR export.

TIP Such fields may also be required to integrate Sipwise C5 with legacy billing systems.

NOTE Please be aware that pre-paid billing functionality is only available in *Sipwise C5 PRO* and *Sipwise C5 CARRIER* products.

The name of CDR data field consists of the elements listed below:

1. source|destination: decides if the data refers to calling (source) or called (destination) party
2. carrier|reseller|customer: the account owner, whose billing data is referred
3. data type:
    1. cash_balance|free_time_balance _ before|after: cash balance or free time balance, before or after the call
    2. profile_package_id|contract_balance_id: internal ID of the active pre-paid billing profile or the account balance

Examples:

- source_customer_cash_balance_before
- destination_customer_profile_package_id

Copyright © Sipwise GmbH - All rights reserved

IMPORTANT For calls spanning multiple balance intervals, the latter one will be selected, that is the balance interval where the call ended.

IMPORTANT There are some limitations in rating **pre-paid** calls, please visit Pre-paid Billing section for details.

Distinguish between on-net and off-net calls CDRs

On-net calls (made only between devices on your network) are sometimes treated differently from off-net calls (terminated to or received from a peer) in external billing systems.

To distinguish between on-net and off-net calls in such a billing systems, check the **source_user_id** and **destination_user_id** fields. For on-net calls, both fields will have a different from zero value (actually, a UUID).

File Body Format for Event Detail Records (EDR)

The body of an EDR consists of a minimum of zero and a maximum of 5000 lines. The platform operator can configure the maximum number of lines kept in a file by updating the eventexport.max_rows_per_file parameter in /etc/ngcp-config/config.yml file. Each line holds one call detail record in CSV format and is constructed by the fields as per the subsequent table.

The following table defines the **default set of fields** that are inserted into the EDR file, for exports related to *system* scope. The list of fields is defined in /etc/ngcp-config/config.yml file, eventexport.admin_export_fields parameter.

Table 9. Default set of system EDR fields

Body Element	Length	Type	Description
EVENT_ID	1-11	uint	Internal EDR ID.

Copyright © Sipwise GmbH - All rights reserved

Body Element	Length	Type	Description
TYPE	0-255	string	The type of the event - one of: start_profile: A subscriber profile has been newly assigned to a subscriber. end_profile: A subscriber profile has been removed from a subscriber. update_profile: A subscriber profile has been changed for a subscriber. start_huntgroup: A subscriber has been provisioned as PBX / hunting group. end_huntgroup: A subscriber has been deprovisioned as PBX / hunting group. start_ivr: A subscriber has a new call-forward to Auto-Attendant. end_ivr: A subscriber has removed a call-forward to Auto-Attendant.
CONTRACT_EXTERNAL_ID	0-255	string	The external ID of the customer. (A string value shown as "External ID" property of an Sipwise C5 customer.)
COMPANY	0-127	string	The company name of the customer's contact.
SUBSCRIBER_EXTERNAL_ID	0-255	string	The external ID of the subscriber. (A string value shown as "External ID" property of an Sipwise C5 subscriber.) *PLEASE NOTE: This field is empty in case of* `start_huntgroup` *and* `end_huntgroup` *events.*
PRIMARY_NUMBER	0-64	string	The VoIP number of the subscriber with the highest ID (DID or primary number).

Copyright © Sipwise GmbH - All rights reserved

Body Element	Length	Type	Description
OLD_PROFILE_NAME	0-255	string	The old status of the event. Depending on the event_type: start_profile: Empty. end_profile: The name of the subscriber profile which got removed from the subscriber. update_profile: The name of the former subscriber profile which got updated. start_huntgroup: Empty. end_huntgroup: Empty. start_ivr: Empty. end_ivr: Empty.
NEW_PROFILE_NAME	0-255	string	The new status of the event. Depending on the event_type: start_profile: The name of the subscriber profile which got assigned to the subscriber. end_profile: Empty. update_profile: The name of the new subscriber profile which got applied. start_huntgroup: Empty. end_huntgroup: Empty. start_ivr: Empty. end_ivr: Empty.
TIMESTAMP	23	timestamp	Timestamp of event. Includes date, time with milliseconds (3 decimals).
RESELLER_ID	1-11	uint	Internal ID of the reseller which the event belongs to. *PLEASE NOTE: Only available in system exports, not for resellers.*
<line_terminator>	1	string	A fixed character. Always \n (special char LF - ASCII 0x0A).

A valid example of one body line of an EDR is (line breaks added for clarity):

Copyright © Sipwise GmbH - All rights reserved

```
"1","start_profile","sipwise_ext_customer_id_4","Sipwise GmbH",
"sipwise_ext_subscriber_id_44","436667778","","1","2014-06-19
11:34:31","1"
```

The format of the **EDR export files generated for *resellers*** (as opposed to the complete system-wide export) is identical except for a few missing fields.

NOTE Please check the description of fields in the table above, in order to see which fields are omitted for *reseller* related EDR exports.

The list of fields for *reseller* EDR export is defined in /etc/ngcp-config/config.yml file, eventexport.reseller_export_fields parameter.

Extra fields that can be exported to EDRs

There are fields in EDR database that contain **supplementary data** related to subscribers, for example subscriber phone numbers are such data.

Table 10. Supplementary data in EDR fields

Body Element	Length	Type	Description
SUBSCRIBER_PROFILE_SET_NAME	0-255	string	The subscriber's profile set name.
PILOT_SUBSCRIBER_PROFILE_SET_NAME	0-255	string	The profile set name of the subscriber's pilot subscriber.
PILOT_SUBSCRIBER_PROFILE_NAME	0-255	string	The profile name of the subscriber's pilot subscriber.
FIRST_NON_PRIMARY_ALIAS_USERNAME_BEFORE	0-255	string	The subscriber's non-primary alias with lowest ID, before number updates during the operation.
FIRST_NON_PRIMARY_ALIAS_USERNAME_AFTER	0-255	string	The subscriber's non-primary alias with lowest ID, after number updates during the operation.
PILOT_FIRST_NON_PRIMARY_ALIAS_USERNAME_BEFORE	0-255	string	The non-primary alias with lowest ID of the subscriber's pilot subscriber, before number updates during the operation.
PILOT_FIRST_NON_PRIMARY_ALIAS_USERNAME_AFTER	0-255	string	The non-primary alias with lowest ID of the subscriber's pilot subscriber, after number updates during the operation.
NON_PRIMARY_ALIAS_USERNAME	0-255	string	The non-primary alias of a subscriber affected by an `update_profile`, `start_profile` or `end_profile` event to track number changes.
PRIMARY_ALIAS_USERNAME_BEFORE	0-255	string	The subscriber's primary alias, before number updates during the operation.
PRIMARY_ALIAS_USERNAME_AFTER	0-255	string	The subscriber's primary alias, after number updates during the operation.

Copyright © Sipwise GmbH - All rights reserved

Body Element	Length	Type	Description
PILOT_PRIMARY_ALIAS_USERNAME_BEFORE	0-255	string	The primary alias of the subscriber's pilot subscriber, before number updates during the operation.
PILOT_PRIMARY_ALIAS_USERNAME_AFTER	0-255	string	The primary alias of the subscriber's pilot subscriber, after number updates during the operation.
FIRST_NON_PRIMARY_ALIAS_USERNAME_BEFORE_AFTER	0-255	string	Equals FIRST_NON_PRIMARY_ALIAS_USERNAME_BEFORE, if the value is not NULL, otherwise it's the same as FIRST_NON_PRIMARY_ALIAS_USERNAME_AFTER.
PILOT_FIRST_NON_PRIMARY_ALIAS_USERNAME_BEFORE_AFTER	0-255	string	Equals PILOT_FIRST_NON_PRIMARY_ALIAS_USERNAME_BEFORE, if the value is not NULL, otherwise it's the same as PILOT_FIRST_NON_PRIMARY_ALIAS_USERNAME_AFTER.

File Trailer Format

The billing file trailer is one single line, which is constructed by the following fields:

```
<md5 sum>
```

The `<md5 sum>` is a 32 character hexadecimal MD5 hash of the *Header* and *Body*.

To validate the billing file, one must remove the Trailer before computing the MD5 sum of the file. The `ngcp-cdr-md5` program included in the `ngcp-cdr-exporter` package can be used to validate the integrity of the file.

Given a CDR-file named as `sipwise_001_20071110123000_0000000004.cdr`, the output of the integrity check for an intact CDR file would be:

```
$ ngcp-cdr-md5 sipwise_001_20071110123000_0000000004.cdr
/tmp/ngcp-cdr-md5.sipwise_001_20071110123000_0000000004.cdr.oqkd4P2zXI:
OK
```

If the file has been altered during transmission, the output of the integrity check would be:

```
$ ngcp-cdr-md5 sipwise_001_20071110123000_0000000004.cdr
/tmp/ngcp-cdr-md5.sipwise_001_20071110123000_0000000004.cdr.hUtuhtKEn1:
FAILED
md5sum: WARNING: 1 of 1 computed checksum did NOT match
```


Copyright © Sipwise GmbH - All rights reserved

6.7.4. File Transfer

Billing files are created twice per hour at minutes 25 and 55 and are stored in the home directory of the cdrexport user. If the amount of records within the transmission interval exceeds the threshold of 5000 records per file, multiple billing files are created. If no billing records are found for an interval, a billing file without body data is constructed for detection of lost billing files on the 3rd party side.

CDR and EDR files are fetched by a 3rd party billing system using SFTP or SCP with either public key or password authentication using the username cdrexport.

If public key authentication is chosen, the public key file has to be stored in the file `.ssh/authorized_keys` below the home directory of the cdrexport user (i.e. `/home/jail/home/cdrexport/.ssh/authorized_keys`). Otherwise, a password has to be set for the user.

The 3rd party billing system is responsible for deleting CDR files after fetching them.

NOTE The cdrexport user is kept in a jailed environment on the system, so it has only access to a very limited set of command-line utilities.

Copyright © Sipwise GmbH - All rights reserved

Chapter 7. Features

The Sipwise C5 provides plenty of subscriber features to offer compelling VoIP services to end customers, and also to cover as many deployment scenarios as possible. In this chapter, we provide the features overview and describe their function and use cases.

7.1. About the Admin Web Interface

This section is going to give some hints to the reader about the Admin web interface of Sipwise C5. The notes here are generic and apply to most of the features that we discuss in the handbook in subsequent chapters.

7.1.1. Filtering the Lists / Datatables

When you look at or want to change various settings on Admin web interface you will see datatables or lists of particular items, e.g. Subscribers, Peering Groups, etc. Sometimes this kind of list can be really long and then it's difficult to find the desired item there. To help the system administrator, the Sipwise C5 offers search filters for each of the lists / datatables. You have to type a search string (arbitrary text) in the *Search* textbox and the system will automatically filter the complete datatable for records that match the search string.

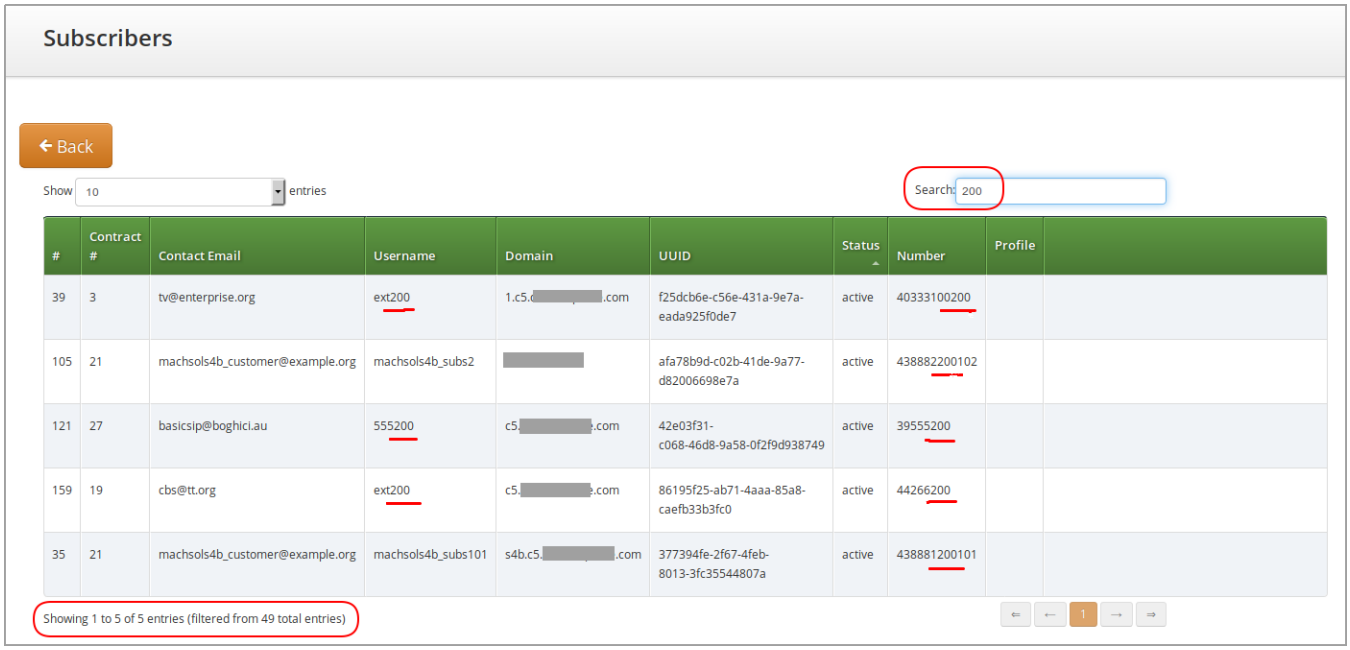


Figure 51. Filtered List of Subscribers

The Search String

The previous example shows what happens if you type a search string in the *Search* textbox. The search string will be applied to all visible columns of the datatable as a filter and all matching records are kept displayed.

The `symbol can be used as *wildcard` for zero-or-more characters.

 Copyright © Sipwise GmbH - All rights reserved

NOTE The * is prepended and appended implicitly to the string entered in *Search* textbox to make filtering easier, for almost all datatables / lists.

While the search pattern is typically matched to values of all columns visible in the datatable, in some cases (i.e. unindexed columns) may be excluded from searching.

7.1.2. Call History

Each call appears in the subscriber's *Call History*, except globally suppressed ones (if suppressing is configured), and you can apply search filters to the table as in case of other datatables.

The *Call History* datatable behaves slightly differently when it comes to wildcard usage. The * wildcard needs to be entered explicitly by the user if needed.

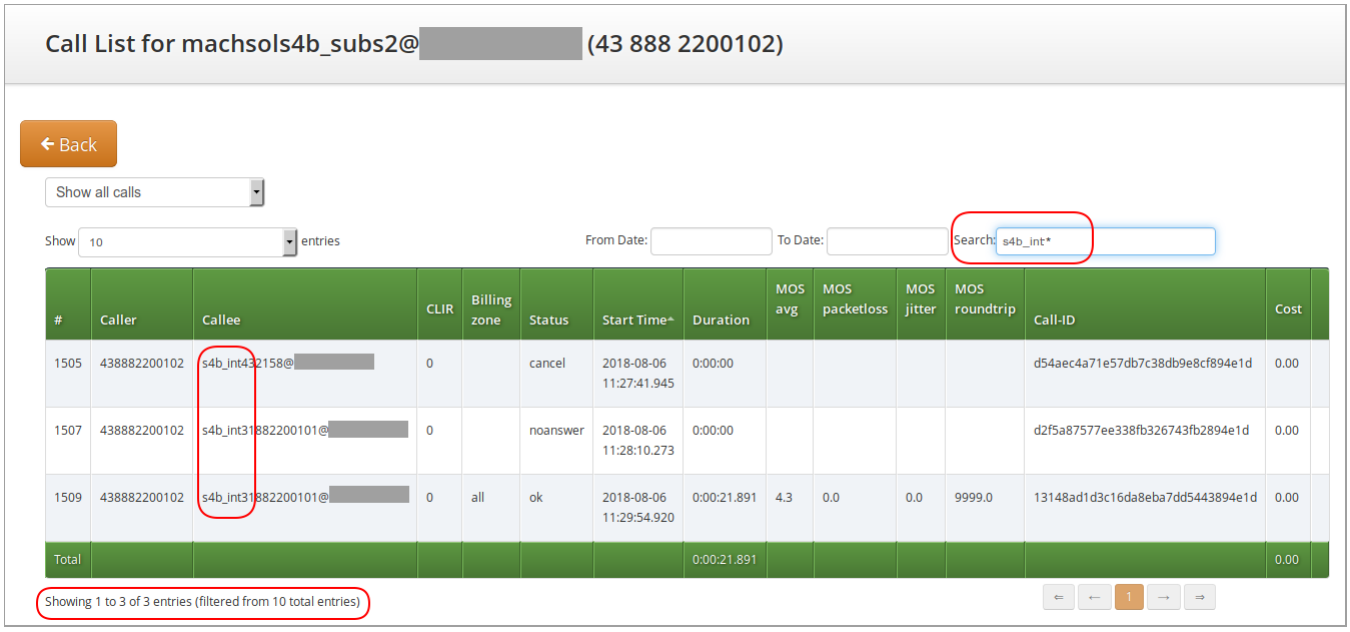


Figure 52. Filtered Call History

CAUTION Be aware that acceptable response times of the administrative web interface rely on utilizing available database indexes, which is impossible with a leading wildcard in the search string. Wildcards at the end of the search pattern do not impact performance.

7.2. Managing System Administrators

The Sipwise C5 offers the platform operator with an easy to use interface to manage users with administrative privileges. Such users are representatives of resellers, and are entitled to manage configuration of services for *Customers*, *Subscribers*, *Domains*, *Billing Profiles* and other entities on Sipwise C5.

Administrators, as user accounts, are also used for client authentication on the REST API of NGCP.

There are two administrators, whose account is enabled by default. Both of them belong to the *default reseller*. These users are the *superusers* of Sipwise C5 administrative web interface (the so-called "admin panel"), and they have the right to modify administrators of other *Resellers* as well. These users are:

Copyright © Sipwise GmbH - All rights reserved

- "administrator" is a default administrative account. It is fully manageable by the system owner.
- "sipwise" is solely for the Sipwise support access. This user can be only enabled or disabled but nor modified neither removed.

7.2.1. Configuring Administrators

Configuration of access rights of system administrators is possible through the admin panel of NGCP. In order to do that, please navigate to *Settings Administrators*.

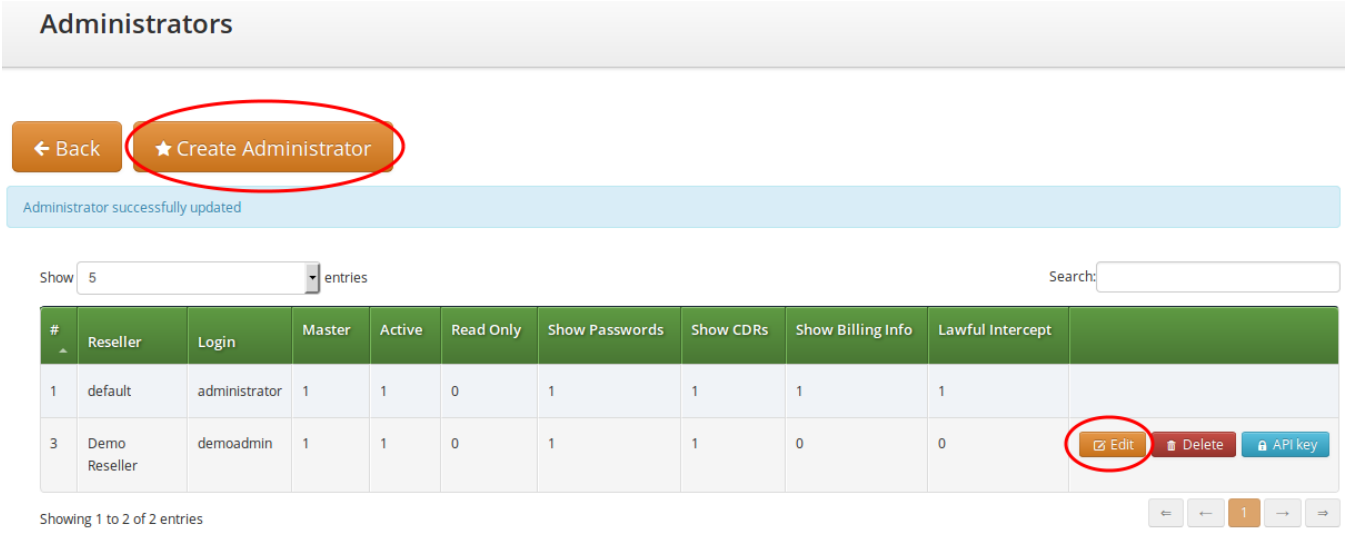


Figure 53. List of System Administrators

You have 2 options:

- If you'd like to **create** a new administrator user press *Create Administrator* button.
- If you'd like to **update** an existing administrator user press *Edit* button in its row.

There are some generic attributes that have to be set for each administrator:

 Copyright © Sipwise GmbH - All rights reserved

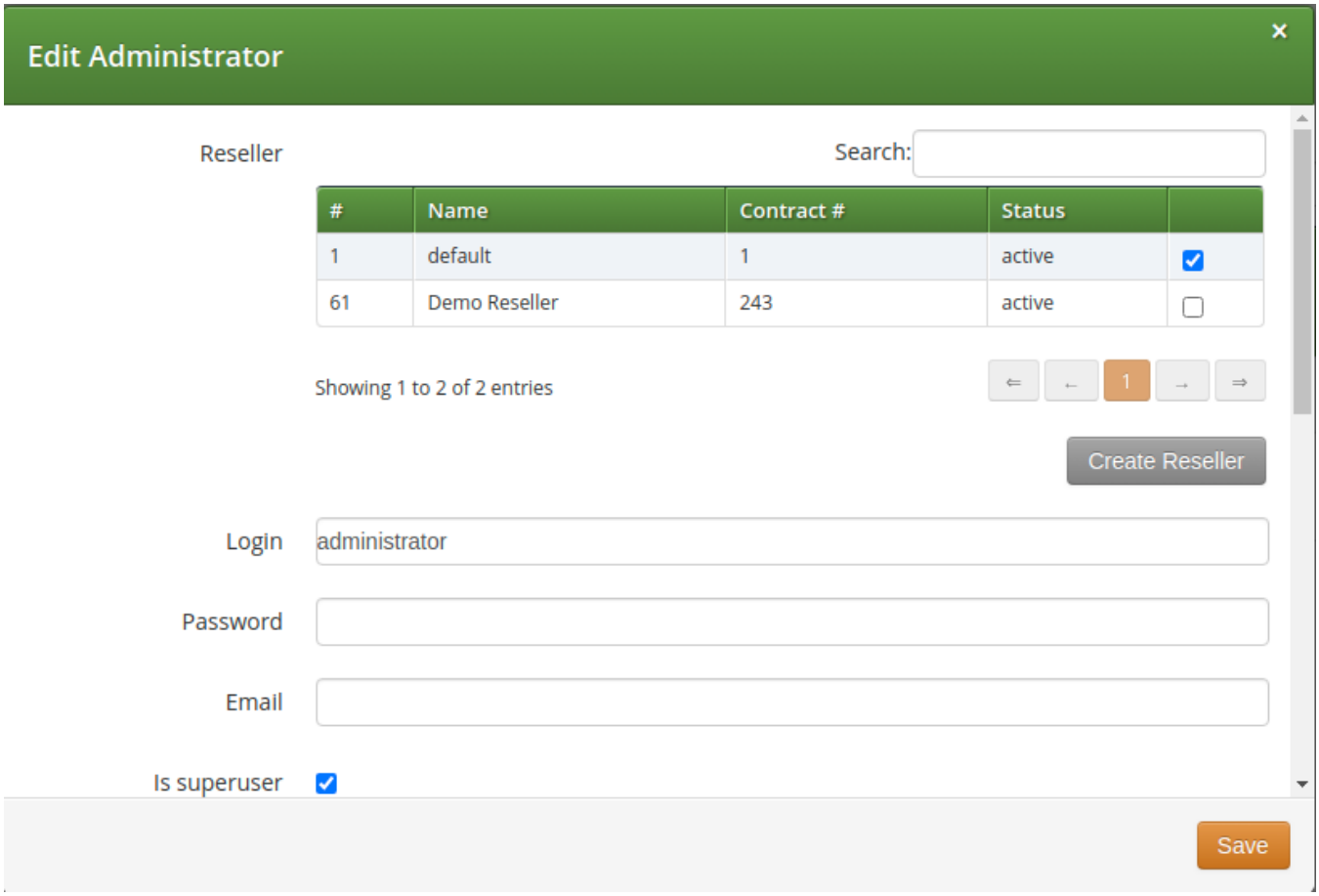


Figure 54. Generic System Administrator Attributes

- *Reseller*: each administrator user must belong to a *Reseller*. There is always a default reseller (ID: `1`, Name: `default`), but the administrator has to be assigned to his real reseller, if such an entity (other than `default`) exists.
- *Login*: the login name of the administrator user
- *Password*: the password of the administrator user for logging in the admin panel, or for authentication on REST API
- *Email*: the email of the administrator user, used for resetting password.

NOTE Due to the fact that administrators can request a password reset via email, no administrator is able to change a password or API key other than to himself. Administrators with is_system, is_superuser and is_master flags are the only ones that can change the username and email of other administrators, while the ones without those flags can only change their own email and password.

The second set of attributes is a list of access rights that are discussed in subsequent section of the handbook.

7.2.2. Access Rights of Administrators

The various access rights of administrators are shown in the figure and summarized in the table below.

Copyright © Sipwise GmbH - All rights reserved

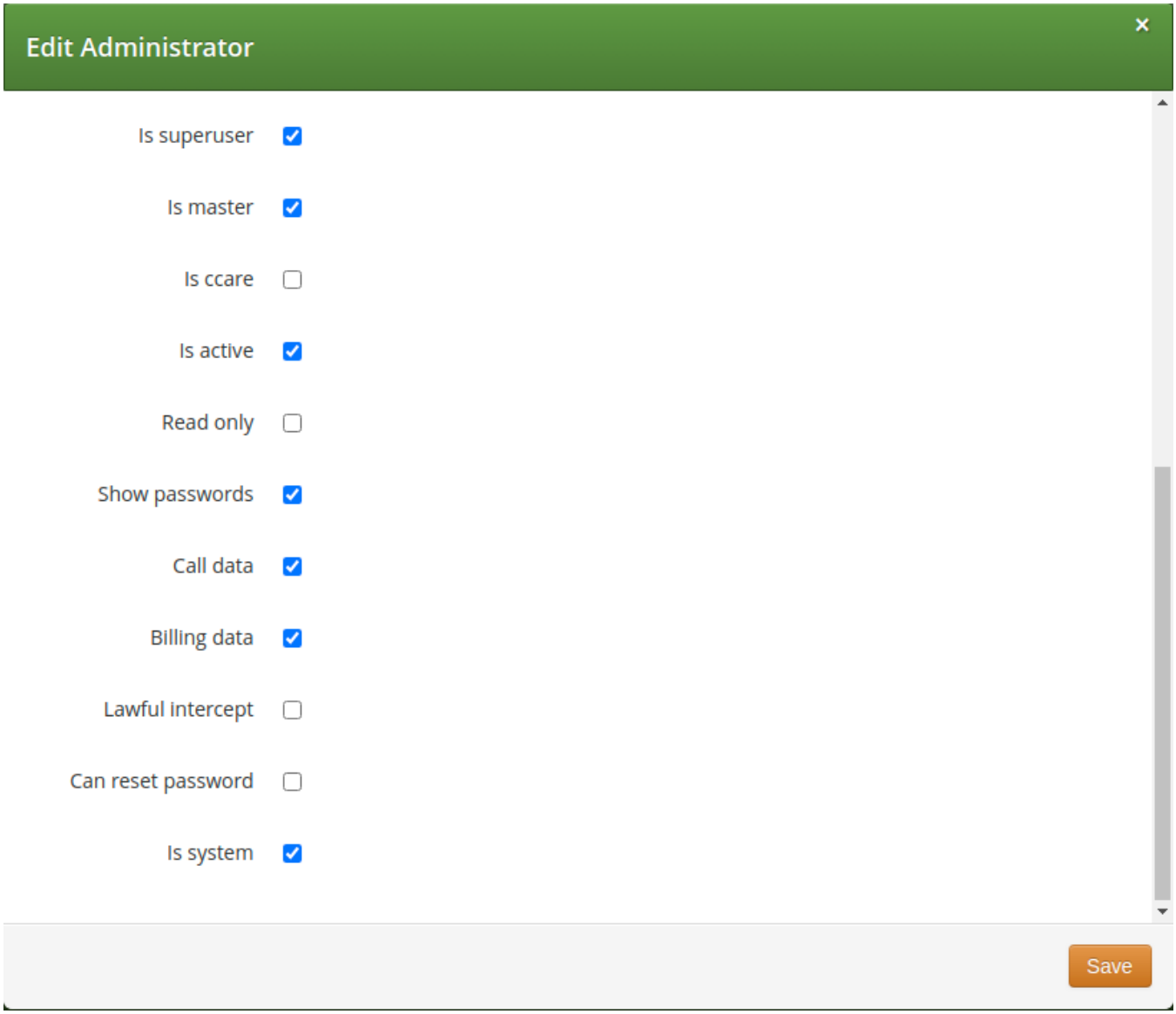


Figure 55. Access Rights of System Administrators

Table 11. Access Rights of System Administrators

Label in admin list	Access Right	Description
not shown	`Is superuser`	The user is allowed to modify data on Reseller level and—among others—is able to modify administrators of other resellers. There should be only 1 user on Sipwise C5 with this privilege.
Master	`Is master`	The user is allowed to create, delete or modify other Admins who belong to the same Reseller.

Copyright © Sipwise GmbH - All rights reserved

Label in admin list	Access Right	Description
Customer Care	`Is ccare`	The user is allowed to create, delete or modify Customers and Subscribers. If mixed with **Is superuser** it defines whether the user can modify the data only within a Reseller the user belongs to, or across all Resellers. The user can access to relevant information required to create or modify Customers or Subscribers, such as Domains, Billing Profiles, Email Templates, but the user cannot access the entries (e.g: see the detailed info about a Billing Profile), nor modify them.
Active	`Is active`	The user account is active, i.e. the admin user can login on the web panel or authenticate himself on REST API; otherwise user authentication will fail.
Read Only	`Read only`	The user will only be able to list various data but is not allowed to modify anything. * For the **web interface** this means that *Create…* and *Edit* buttons will be hidden or disabled. * For the **REST API** this means that only GET, HEAD, OPTIONS HTTP request methods are accepted, and Sipwise C5 will reject those targeting data modification: PUT, PATCH, POST, DELETE.
Show Passwords	`Show passwords`	The user sees subscriber passwords (in plain text) on the web interface. NOTE: Admin panel user passwords and subscriber web passwords are stored in an unreadable way (cryptographic hash digest) in the database, while subscriber SIP passwords are stored in plain text. The latter happens on purpose, e.g. to make subscriber data migration possible.
Show CDRs	`Call data`	This privilege has effect on 2 items that will be displayed on admin panel of NGCP, when *Subscriber Details* is selected: 1. PBX *Groups* list 2. *Captured Dialogs* list
Show Billing Info	`Billing data`	Some REST API resources that are related to billing are disabled: HTTP requests on /api/vouchers, /api/topupcash and /api/topupvoucher resources are rejected.
Lawful Intercept	`Lawful intercept`	Visible only for System administrators. If the privilege is selected then the REST API for interceptions (that is: /api/interceptions) is enabled; if the privilege is not selected then the interceptions API is disabled. Administrators with this flag do not have access to anything but the administrators in UI and API and /api/interceptions in the API NOTE: This means that besides enabling LI in config.yml configuration file one also needs to enable the API via the LI privilege of an administrator user, so that Sipwise C5 can really provide LI service.
Can Reset Password	`Can Reset Password`	The user is allowed to request a password reset.

Copyright © Sipwise GmbH - All rights reserved

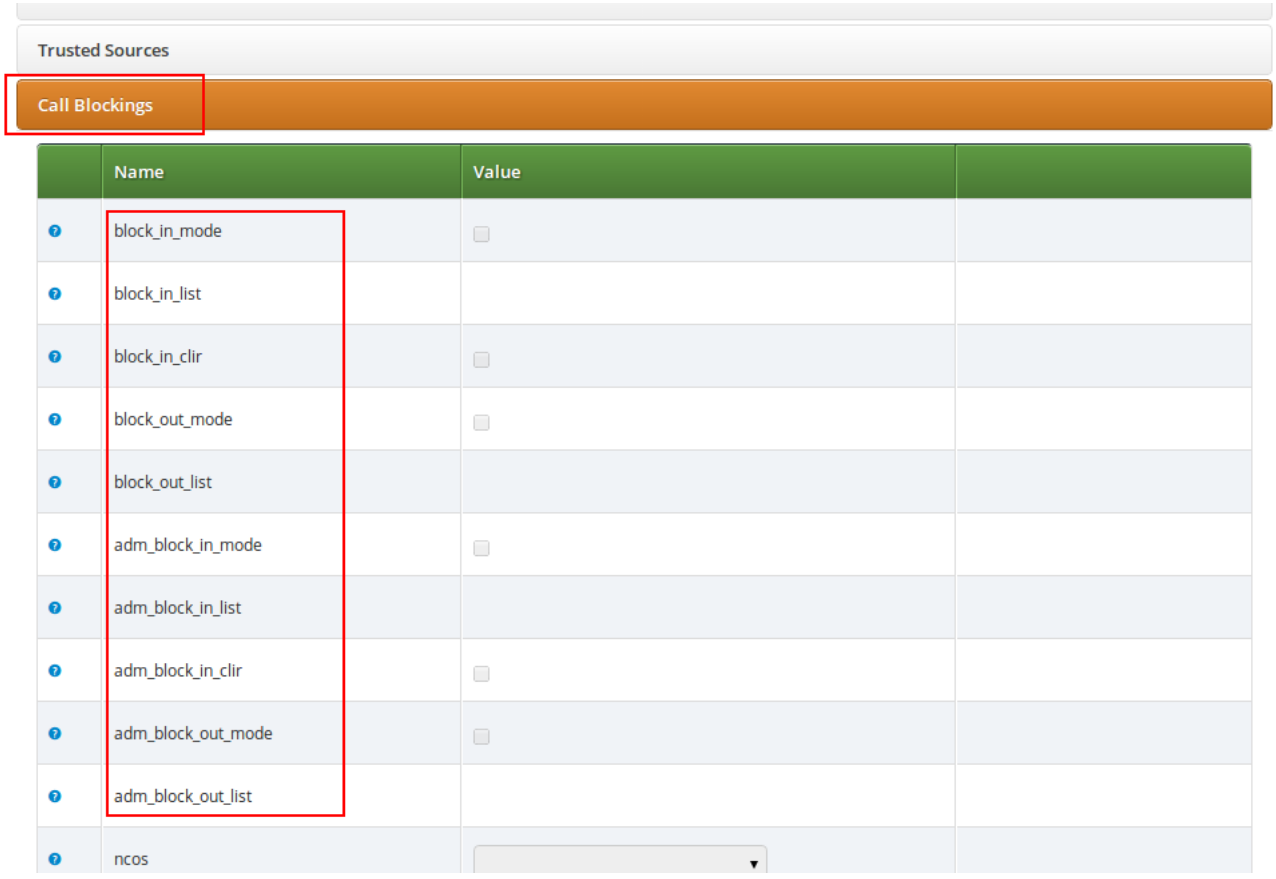
Label in admin list	Access Right	Description
System	`Is system`	The user is allowed to create, delete or modify Lawful Intercept Admins which are otherwise invisible to other types of administrators.

7.3. Access Control for SIP Calls

There are two different methods to provide fine-grained call admission control to both subscribers and admins. One is *Block Lists*, where you can define which numbers or patterns can be called from a subscriber to the outbound direction and which numbers or patterns are allowed to call a subscriber in the inbound direction. The other is *NCOS (Network Class of Service) Levels*, where the admin predefines rules for outbound calls, which are grouped in certain levels. The subscriber can then choose the level, or the admin can restrict a subscriber to a certain level. Also Sipwise C5 offers some options to restrict the IP addresses that subscriber is allowed to use the service from. The following sections describe these features in detail.

7.3.1. Block Lists

Block Lists provide a way to control which users/numbers can call or be called, based on a subscriber level, and can be found in the *Call Blockings* section of the subscriber preferences.



Block Lists are separated into *Administrative Block Lists* (*adm_block_**) and *Subscriber Block Lists* (*block_**). They both have the same behaviour, but Administrative Block Lists take higher precedence. Administrative Block Lists are only accessible by the system administrator and can thus be used to override any Subscriber Block Lists, e.g. to block certain destinations. The following break-down of the various block features apply to both types of lists.

 Copyright © Sipwise GmbH - All rights reserved

Block Modes

Block lists can either be *whitelists* or *blacklists* and are controlled by the User Preferences *block_in_mode*, *block_out_mode* and their administrative counterparts.

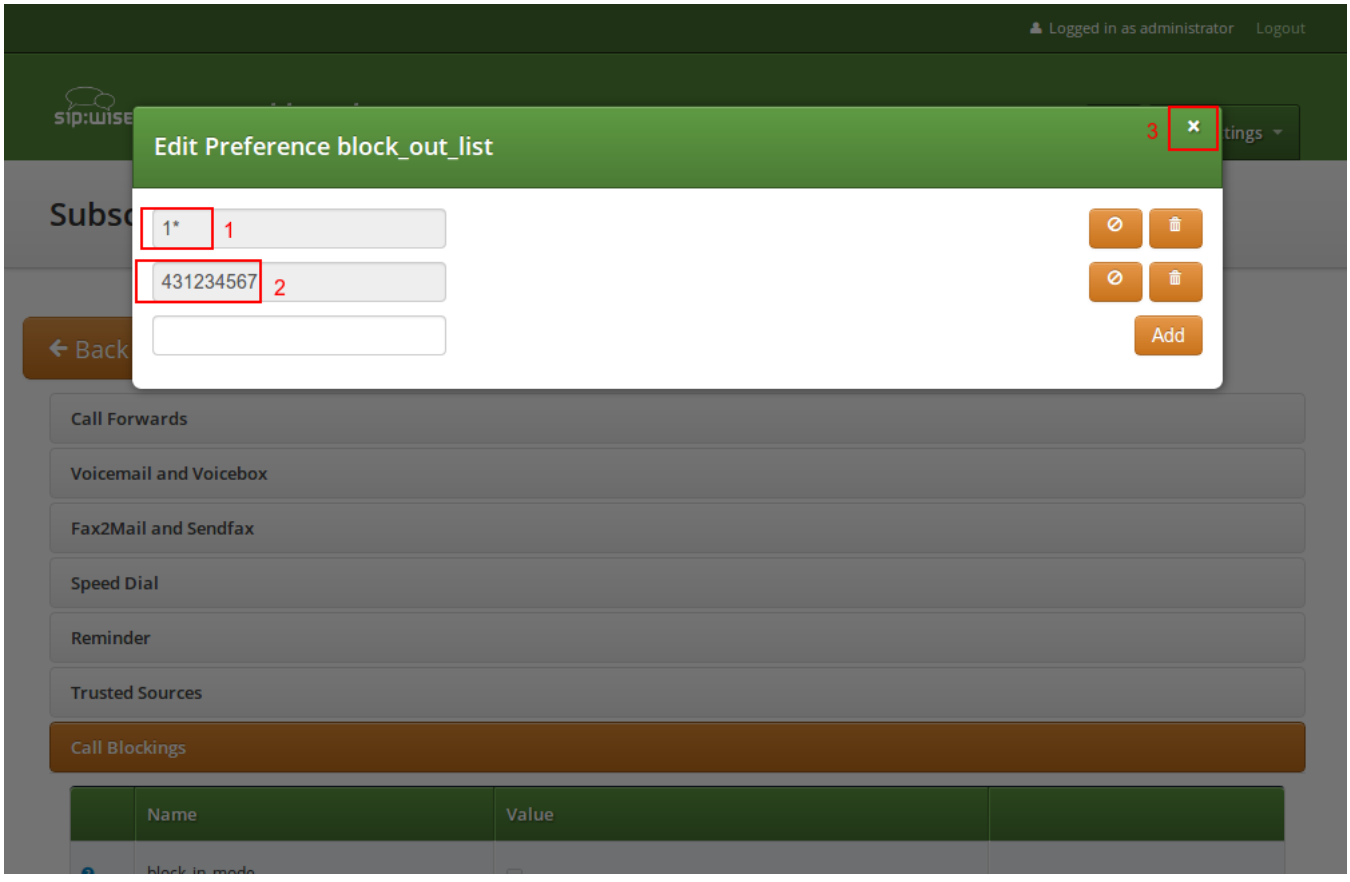
- The *blacklist* mode (option is not checked tells the system to **allow anything except the entries in the list**. Use this mode if you want to block certain numbers and allow all the rest.
- The *whitelist* mode indicates to **reject anything except the entries in the list**. Use this mode if you want to enforce a strict policy and allow only selected destinations or sources.

You can change a list mode from one to the other at any time.

Block Lists

The list contents are controlled by the User Preferences *block_in_list*, *block_out_list* and their administrative counterparts. Click on the *Edit* button in the *Preferences* view to define the list entries.

In block list entries, you can provide shell patterns like * and []. The behavior of the list is controlled by the *block_xxx_mode* feature (so they are either allowed or rejected). In our example above we have *block_out_mode* set to *blacklist*, so all calls to US numbers and to the Austrian number +431234567 are going to be rejected.



Click the *Close* icon once you're done editing your list.

Block Anonymous Numbers

For incoming call, the User Preference *block_in_clir* and *adm_block_in_clir* controls whether or not to reject incoming calls with number suppression (either "[Aa]nonymous" in the display- or user-part of the

Copyright © Sipwise GmbH - All rights reserved

From-URI or a header *Privacy: id* is set). This flag is independent from the Block Mode.

7.3.2. NCOS (Network Class of Service) Levels

NCOS Levels provide predefined lists of allowed or denied destinations for outbound calls of local subscribers. Compared to *Block Lists*, they are much easier to manage, because they are defined on a global scope, and the individual levels can then be assigned to each subscriber. Again there is the distinction for the user- and administrative- levels.

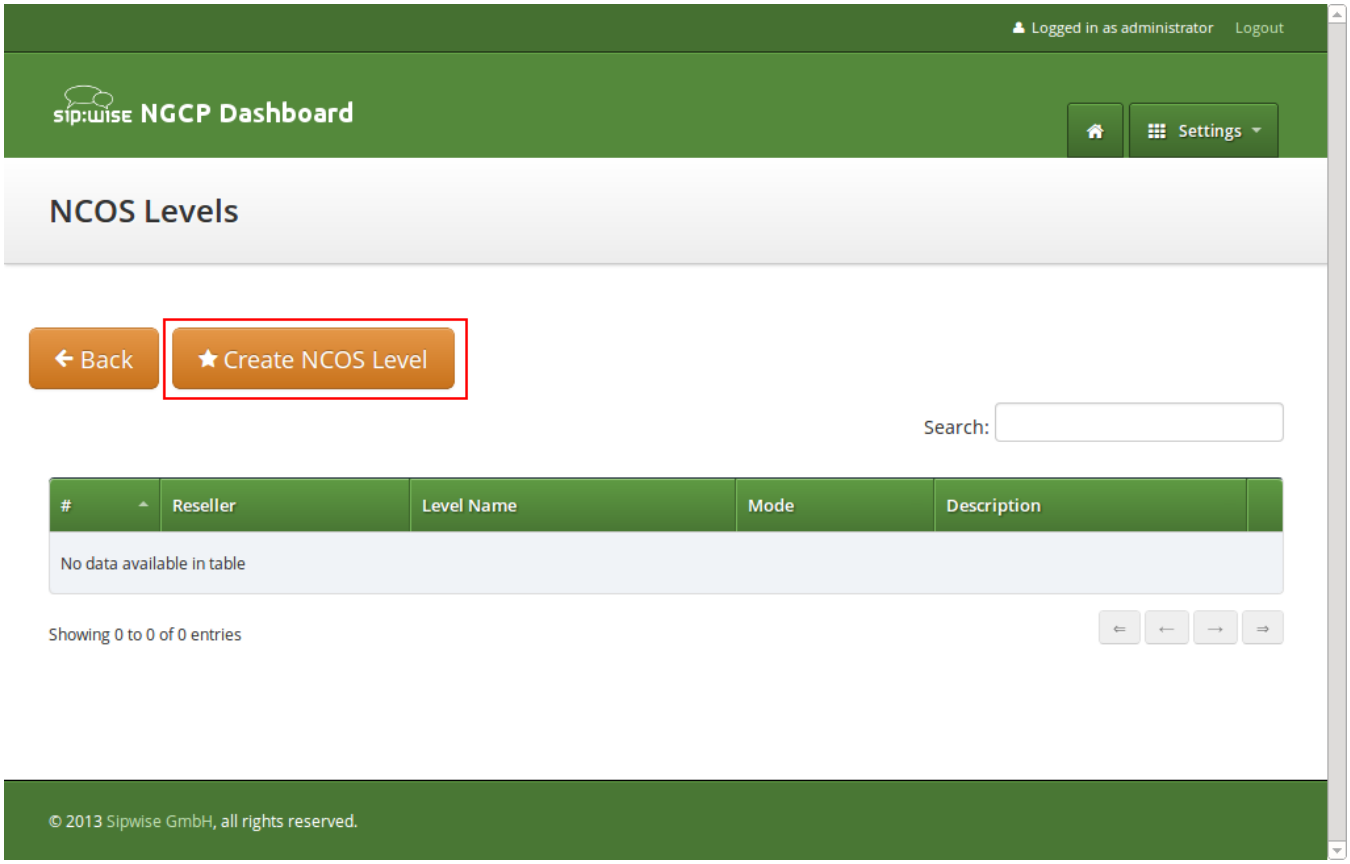
In a case of a conflict, when the Block Lists feature allows a number and NCOS Levels rejects the same number or vice versa, the call will be rejected.

NCOS levels can either be *whitelists* or *blacklists*.

- The *blacklist* mode indicates to **allow everything except the entries in this level**. Use this mode if you want to block specific destinations and allow all the rest.
- The *whitelist* mode indicates to **reject anything except the entries in this level**. Use this mode if you want to enforce a strict policy and allow only selected destinations.

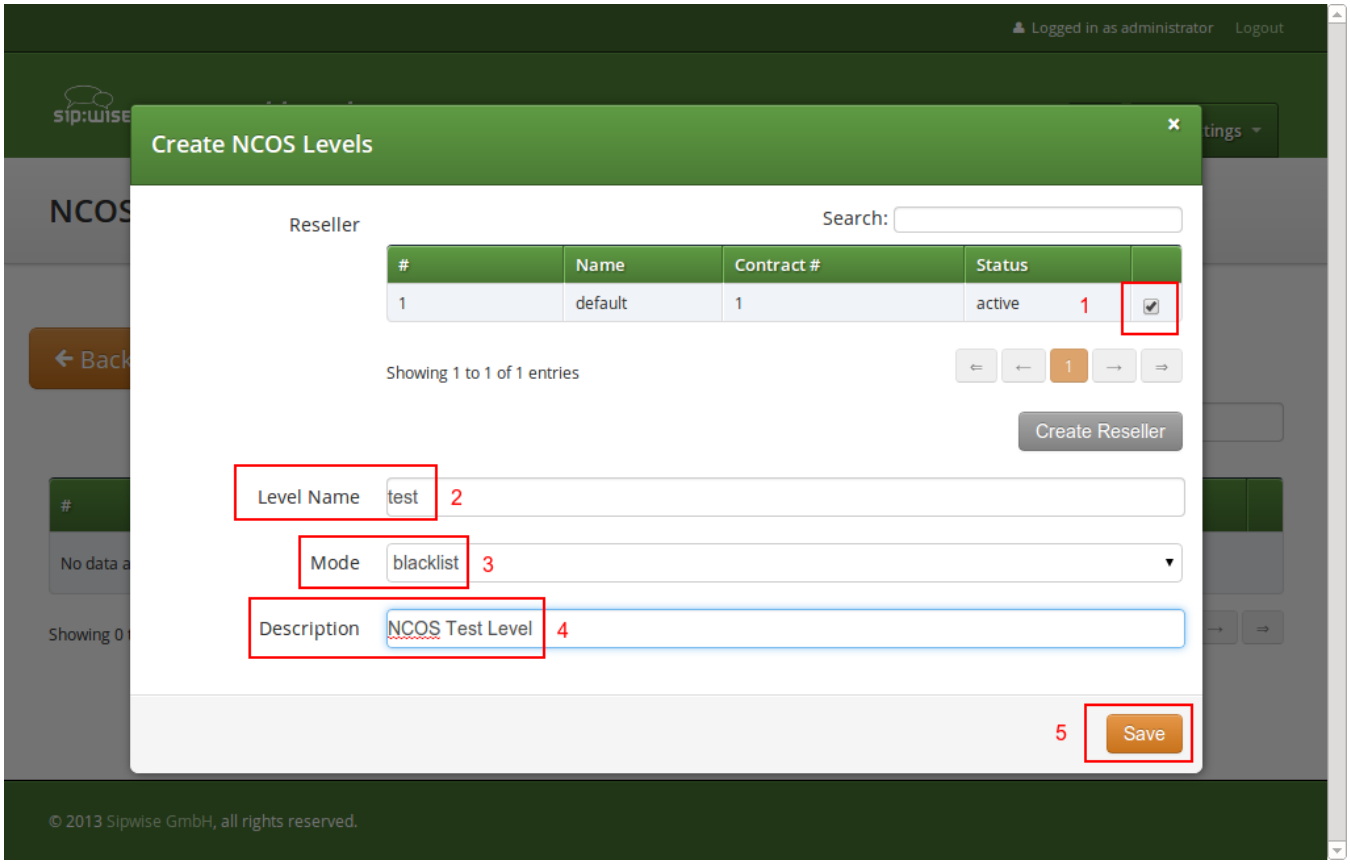
Creating NCOS Levels

To create an NCOS Level, go to *SettingsNCOS Levels* and press the *Create NCOS Level* button.



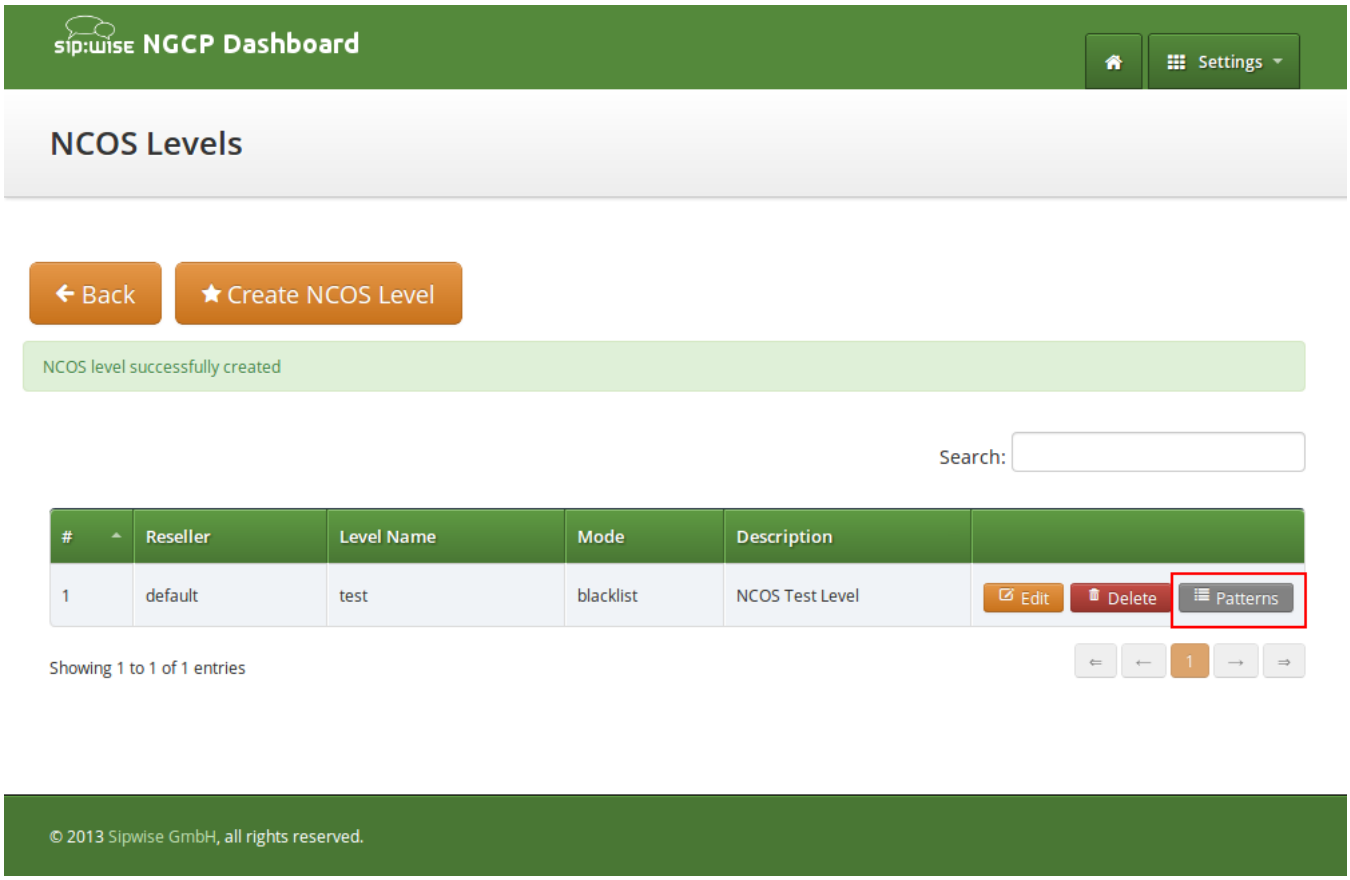
Select a reseller, enter a name, select the mode and add a description, then click the *Save* button.

 Copyright © Sipwise GmbH - All rights reserved



Creating Rules per NCOS Level

To define the rules within the newly created NCOS Level, click on the *Patterns* button of the level.



There are 2 groups of patterns where you can define matching rules for the selected NCOS Level:

- NCOS Number Patterns: here you can define number patterns that will be matched against the called number and allowed or blocked, depending on whitelist / blacklist mode. The patterns are regular expressions.
- NCOS LNP Carriers: here you can select predefined *LNP Carriers* that will be allowed (whitelist mode) or prohibited (blacklist mode) to route calls to them. For each of them you can restrict the matching to a predefined number pattern. (See Local LNP Database in the handbook for the description of LNP functionality)

NOTE Sipwise C5 performs number matching always with the dialed number and not with the number generated after LNP lookup that is: either the original dialed number prefixed with an LNP carrier code, or the routing number.

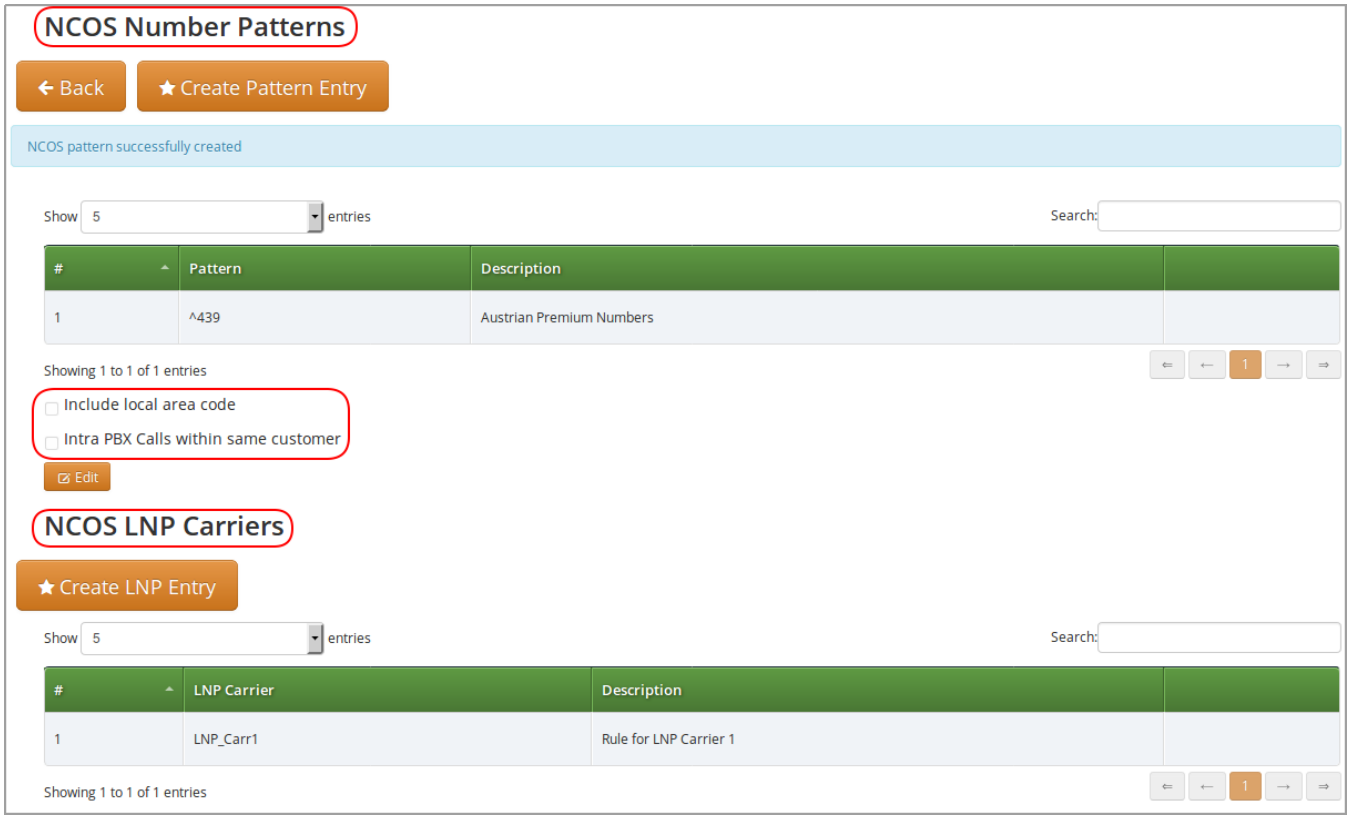


Figure 56. NCOS Patterns List

In the ***NCOS Number Patterns*** view you can create multiple patterns to define your level, one after the other. Click on the *Create Pattern Entry* Button on top and fill out the form.

 Copyright © Sipwise GmbH - All rights reserved

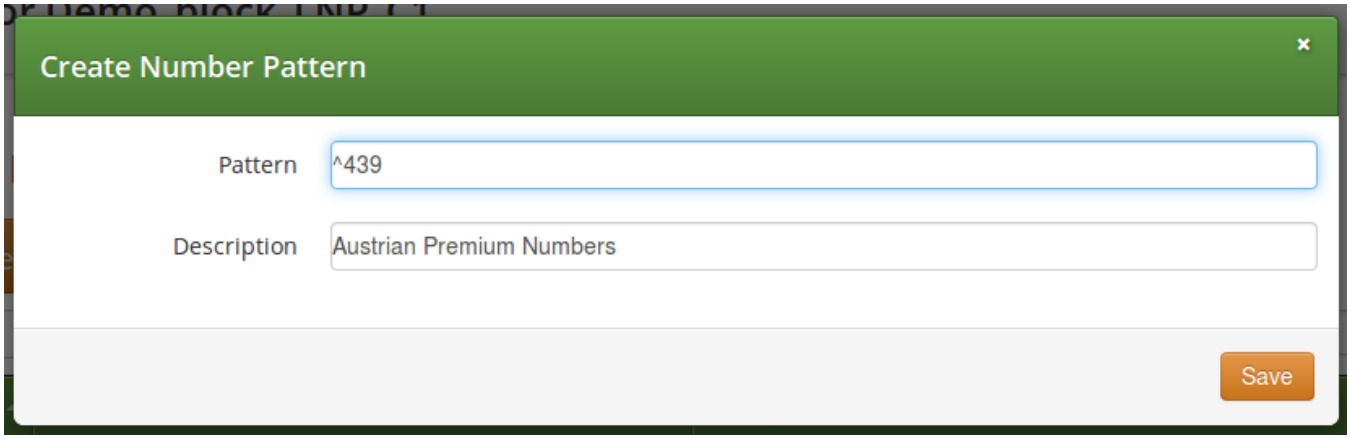


Figure 57. Create NCOS Number Pattern

In this example, we block (since the mode of the level is *blacklist*) all numbers starting with 439. Click the *Save* button to save the entry in the level.

There are ***2 options*** that help you to easily define specific number ranges that will be allowed or blocked, depending on whitelist / blacklist mode:

- *Include local area code*: all subscribers within the caller's local area, e.g. if a subscriber has country-code 43 and area-code 1, then selecting this checkbox would result in the implicit number pattern: ^431.
- *Intra PBX calls within same customer*: all subscribers that belong to the same PBX customer as the caller himself.

In the ***NCOS LNP Carriers*** view you can select specific LNP Carriers—i.e. carriers that host the called ported numbers—that will be allowed or blocked for routing calls to them (whitelist / blacklist mode, respectively).

An example of *NCOS LNP Carrier* definition:

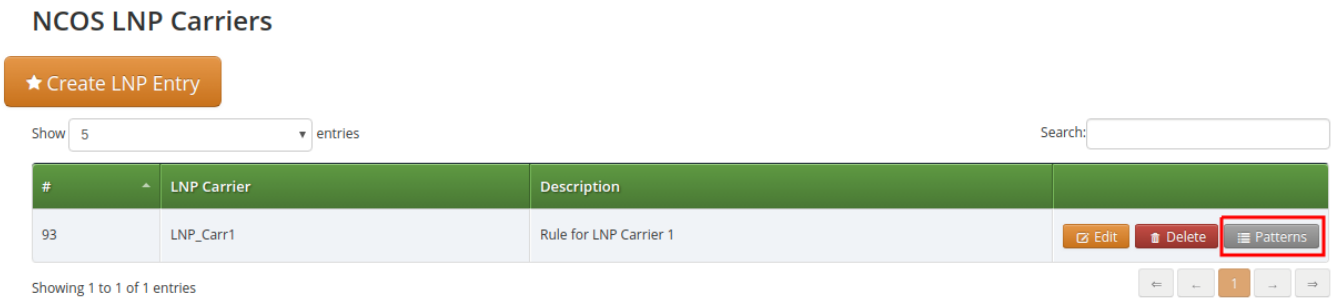


Figure 58. Create NCOS LNP Carrier

In the above example we created a rule that blocks calls to "LNP_Carr1" carrier, supposing we use blacklist mode of the NCOS Level.

In the ***LNP NCOS Number Patterns*** view you can create multiple patterns to restrict *NCOS LNP Carrier* matching, one after the other. Click on the *Create LNP Pattern Entry* Button on top and fill out the form.

Copyright © Sipwise GmbH - All rights reserved

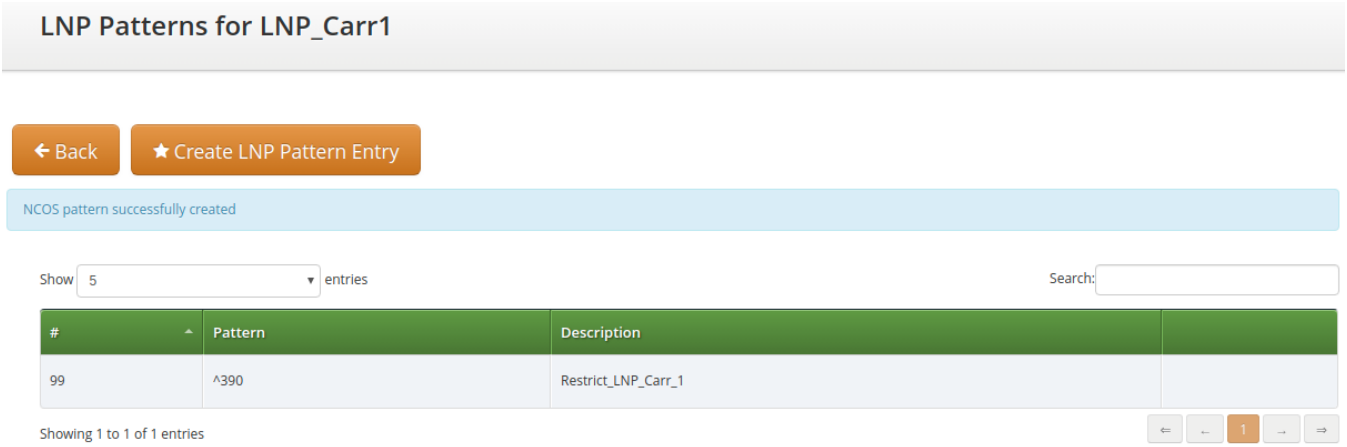


Figure 59. Create NCOS LNP Carrier Pattern

Considering the example before and adding the pattern shown in the picture, the rule now blocks only calls to "LNP_Carr1" carrier that starts with 390.

TIP There might be situations when phone number patterns may not be strictly aligned with telephony providers, for instance in case of full number portability in a country. In such cases using *NCOS LNP Carriers* patterns still allows for defining NCOS levels that allow / block calls to mobile numbers, for example. In order to achieve this goal you have to list all LNP carriers in the NCOS patterns that are known to host mobile numbers.

The below table gives an overview of all the possible combinations of *NCOS* and *NCOS LNP Carrier*:

Table 12. NCOS combinations

TYPE	NCOS	NCOS_LNP	RESULT
Whitelist	empty table	empty table	Blocked
Whitelist	empty table	no match	Blocked
Whitelist	empty table	match	Allowed
Whitelist	no match	empty table	Blocked
Whitelist	no match	no match	Blocked
Whitelist	no match	match	Blocked*
Whitelist	match	empty table	Allowed
Whitelist	match	no match	Blocked*
Whitelist	match	match	Allowed
Blacklist	empty table	empty table	Allowed
Blacklist	empty table	no match	Allowed
Blacklist	empty table	match	Blocked
Blacklist	no match	empty table	Allowed
Blacklist	no match	no match	Allowed
Blacklist	no match	match	Blocked

 Copyright © Sipwise GmbH - All rights reserved

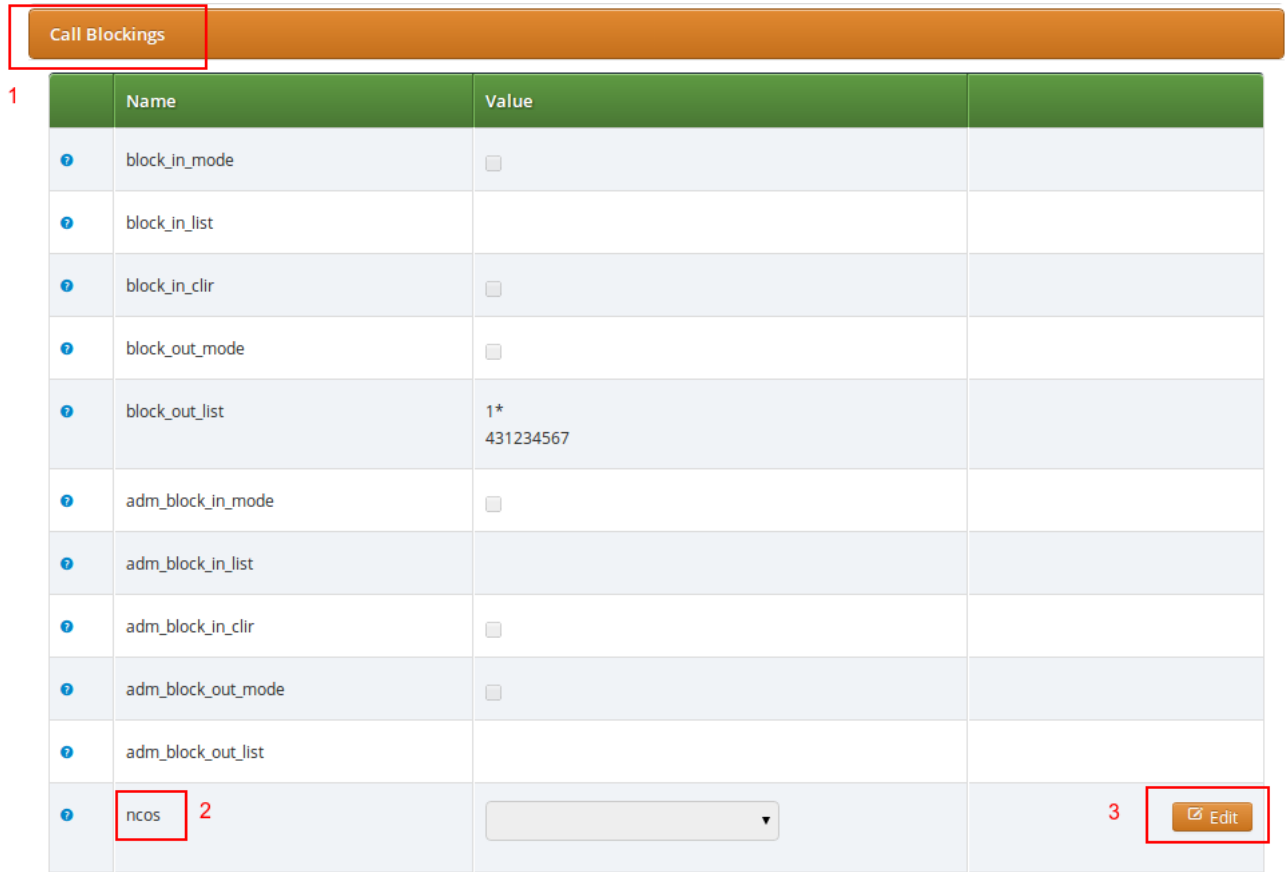
TYPE	NCOS	NCOS_LNP	RESULT
Blacklist	match	empty table	Blocked
Blacklist	match	no match	Blocked
Blacklist	match	match	Blocked

- = different behaviour compared with the previous versions (< mr7.5)

The parameter kamailio.proxy.lnp.strictly_check_ncos contained in /etc/ngcp-config/config.yml specify whether the NCOS LNP should be evaluated even if the LNP lookup was not previously executed (because not required by the inbound/outbound call) or if it didn't return any occurrence. If set to *yes*, a whitelist NCOS will fail if the LNP lookup doesn't return any match. The parameter has no impact on blacklist NCOS.

Assigning NCOS Levels to Subscribers/Domains

Once you've defined your NCOS Levels, you can assign them to local subscribers. To do so, navigate to *SettingsSubscribers*, search for the subscriber you want to edit, press the *Details* button and go to the *Preferences* View. There, press the *Edit* button on either the *ncos* or *adm_ncos* setting in the *Call Blockings* section.



You can assign the NCOS level to all subscribers within a particular domain. To do so, navigate to *SettingsDomains*, select the domain you want to edit and click *Preferences*. There, press the *Edit* button on either *ncos* or *admin_ncos* in the *Call Blockings* section.

Note: if both domain and subscriber have same NCOS preference set (either *ncos* or *adm_ncos*, or both) the subscriber's preference is used. This is done so that you can override the domain-global setting on

Copyright © Sipwise GmbH - All rights reserved

the subscriber level.

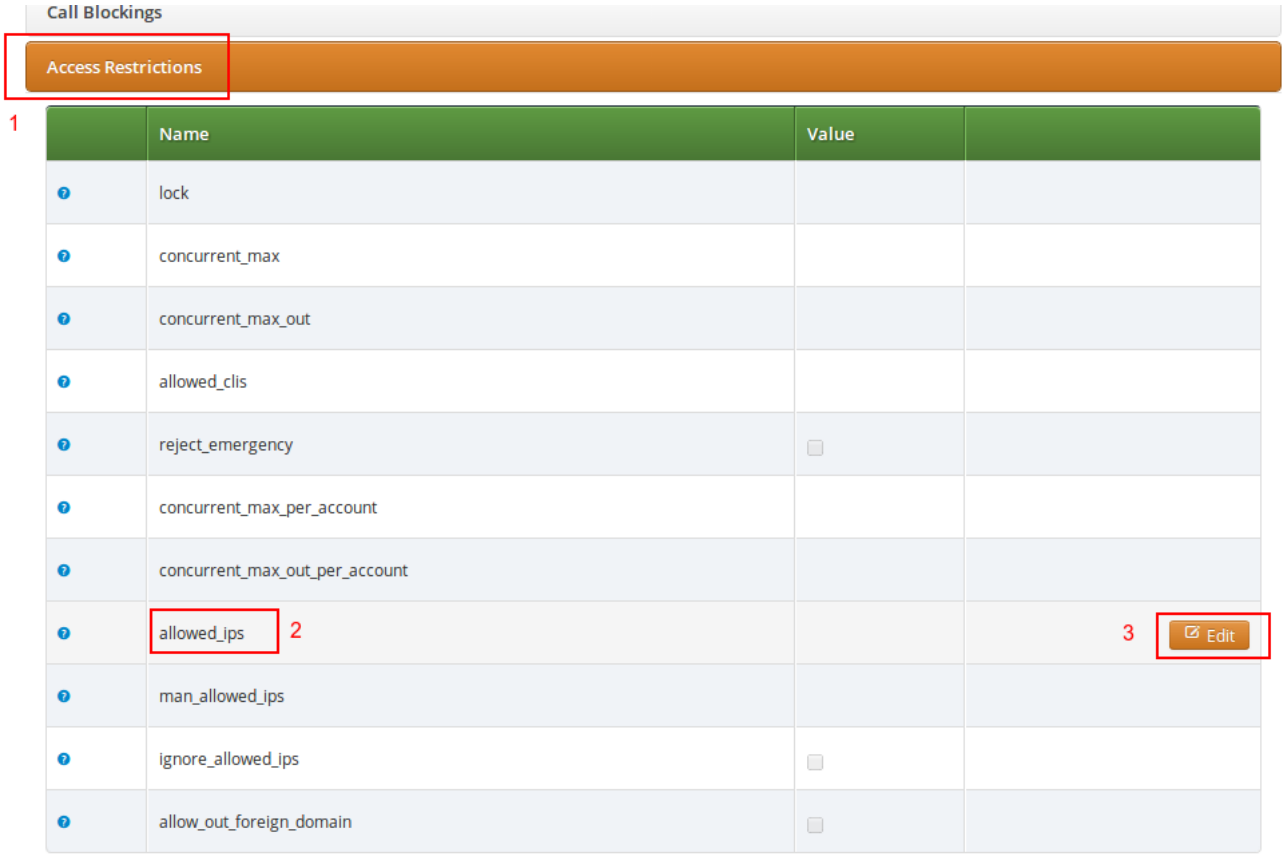
Assigning NCOS Level for Forwarded Calls to Subscribers/Domains

In some countries there are regulatory requirements that prohibit subscribers from forwarding their numbers to special numbers like emergency, police etc. While Sipwise C5 does not deny provisioning Call Forward to these numbers, the administrator can prevent the incoming calls from being actually forwarded to numbers defined in the NCOS list: select the appropriate NCOS level in the domain's or subscriber's preference *adm_cf_ncos*. This NCOS will apply only to the Call Forward from the subscribers and not to the normal outgoing calls from them.

7.3.3. IP Address Restriction

The Sipwise C5 provides subscriber and domain preference *allowed_ips* to restrict the IP addresses that a particular subscriber or any subscribers within the respective domain is allowed to use the service from. If the REGISTER or INVITE request comes from an IP address that is not in the allowed list, Sipwise C5 will reject it with a 403 message. Also a voice message can be played when the call attempt is rejected (if configured).

By default, *allowed_ips* is an empty list which means that subscriber is not restricted. If you want to configure a restriction, navigate to *SettingsSubscribersPreferences* or *SettingsDomainsPreferences*, and search for the *allowed_ips* preference in the *Access Restrictions* section.



Press the Edit button to the right of empty drop-down list.

You can enter multiple allowed IP addresses or IP address ranges one after another. Click the *Add* button to save each entry in the list. Click the *Delete* button if you want to remove some entry.

 Copyright © Sipwise GmbH - All rights reserved

7.3.4. CLI-based Access Control

The Sipwise C5 provides subscriber preference *upn_block_list* to restrict the CLI that subscriber is allowed to use the service from. If the INVITE request comes with a CLI that is not in the allowed list, Sipwise C5 will reject it with a 403 message. Also a voice message can be played when the call attempt is rejected (if configured).

The restriction is applied to User-Provided Number (UPN) which is obtained from the configurable source based on the setting of *inbound_upn* preference in the *Number Manipulation* section in the Domain and/or User preferences, after it has been rewritten with Inbound Rewrite Rules for Caller.

In case the *inbound_upn* preference is set to the "From Display-Name" the UPN value can be alphanumeric so the access control supports the alphanumeric (caller name) matching as well. If the incoming message does not have the Display-Name, though, the UPN value will be taken from the From-Username.

The *inbound_upn* preference has a slightly different meaning if *kamailio.proxy.multiple_headers_for_valid_upn* is set to yes inside config.yml.

In this particular case the *inbound_upn* preference only defines which Sip Header must be checked first and, in case of failure, the check is repeated evaluating the username part of the following headers:

- P-Preferred-Identity
- P-Asserted-Identity
- From

In case of positive result, that value will be used as valid UPN and the subsequent headers will be not evaluated, otherwise if all additional checks fail, restrictions are applied as described above.

By default, *upn_block_list* is an empty list which means that subscriber is not restricted. If you want to configure a restriction, navigate to *SettingsSubscribers*, search for the subscriber you want to edit, press *Details* and then *Preferences* and press *Edit* for the *upn_block_list* preference in the *Call Blockings* section to define the list entries.

In block list entries, you can provide shell patterns like * and []. The CLI-based block list can either be *whitelist* or *blacklist*.

- The *blacklist* mode indicates to **allow everything except the entries in this list**. This is the default mode of operation and is effective when the preference *upn_block_mode* is unset.
- The *whitelist* mode indicates to **reject anything except the entries in this list**. In order to switch to this mode, set the preference *upn_block_mode* (it is a toggle between whitelist/blacklist).

If separate preference *upn_block_clir* is enabled, outgoing anonymous calls from this user will be dropped.

If the caller's UPN is allowed it is also checked according to *allowed_clis* preference as usual and can be rewritten according to *allowed_clis_reject_policy* for correct calling number presentation on outgoing calls. This step happens after Access Control.

7.3.5. Call Limit Control

There's a set of preferences that limits calls to and from subscribers. The option *concurrent_max_total*

Copyright © Sipwise GmbH - All rights reserved

defines the maximum number of concurrent calls (incoming and outgoing) for a subscriber, while the option *concurrent_max_out_total* limits only subscriber's outbound concurrent calls and the option *concurrent_max_in_total* only subscriber's inbound concurrent calls.

Preferences *concurrent_max*, *concurrent_max_out*, and *concurrent_max_in* have the same effect, excluding calls to voicemail, application server and intra-PBX calls.

It's also possible to limit the number of concurrent calls of a subscriber compared to the number of calls made or received by all subscribers within the same customer (account). The options *concurrent_max_per_account*, *concurrent_max_out_per_account*, *concurrent_max_in_per_account* permit to apply this limit. To better understand how they work, suppose we have two subscribers A and B, owned by the same customer. If we set *concurrent_max_per_account*=2 on B preferences and A is placing two calls, then B can not receive or place new calls at the same time. For instance, an administrator may define this restriction to some non-manager subscribers, in which *concurrent_max_per_account*=2. Hence, they will be able to make a maximum of two calls if there are no other calls in place within the customer. On manager subscribers, the limit can be defined differently, or even not set at all. In the last case, calls will be always allowed.

When *concurrent_max_total* limit is reached, announcement set on *max_calls_in* is played to those who try to call that subscriber. The same announcement is played for *concurrent_max*, *concurrent_max_per_account*, *concurrent_max_in_total*, *concurrent_max_in*, *concurrent_max_in_per_account*. When *concurrent_max_out_total* limit is reached, announcement set on *max_calls_out* is played. The same announcement is played for *concurrent_max_out* or *concurrent_max_out_per_account*.

Options *concurrent_max*, *concurrent_max_out* and *concurrent_max_in* are configurable on peers as well.

Furthermore, options *concurrent_max*, *concurrent_max_out*, *concurrent_max_in* and their *_total* version (*concurrent_max_total* and so on), are configurable on customer as well. In this case the limits are applied considering the number of calls of all the subscribers belonging to that account.

7.4. Call Forwarding and Call Hunting

The Sipwise C5 provides the capabilities for normal *call forwarding* (deflecting a call for a local subscriber to another party immediately or based on events like the called party being busy or doesn't answer the phone for a certain number of seconds) and *serial call hunting* (sequentially executing a group of deflection targets until one of them succeeds). Targets can be stacked, which means if a target is also a local subscriber, it can have another call forward or hunt group which is executed accordingly.

7.4.1. Call Forward Types

Currently 7 different types of Call Forward are available in Sipwise C5:

- **Call Forward Unconditional (CFU)**: The call forward is always executed, completely disregarding the subscriber state.
- **Call Forward Busy (CFB)**: The call forward is executed when the subscriber returns a busy state.
- **Call Forward Timeout (CFT)**: The call forward is executed when no answer is received from the subscriber before the timeout expiration. Timeout is configurable in *ringtimeout* subscriber preference.
- **Call Forward Unavailable (CFNA)**: The call forward is executed when the subscriber has no

 Copyright © Sipwise GmbH - All rights reserved

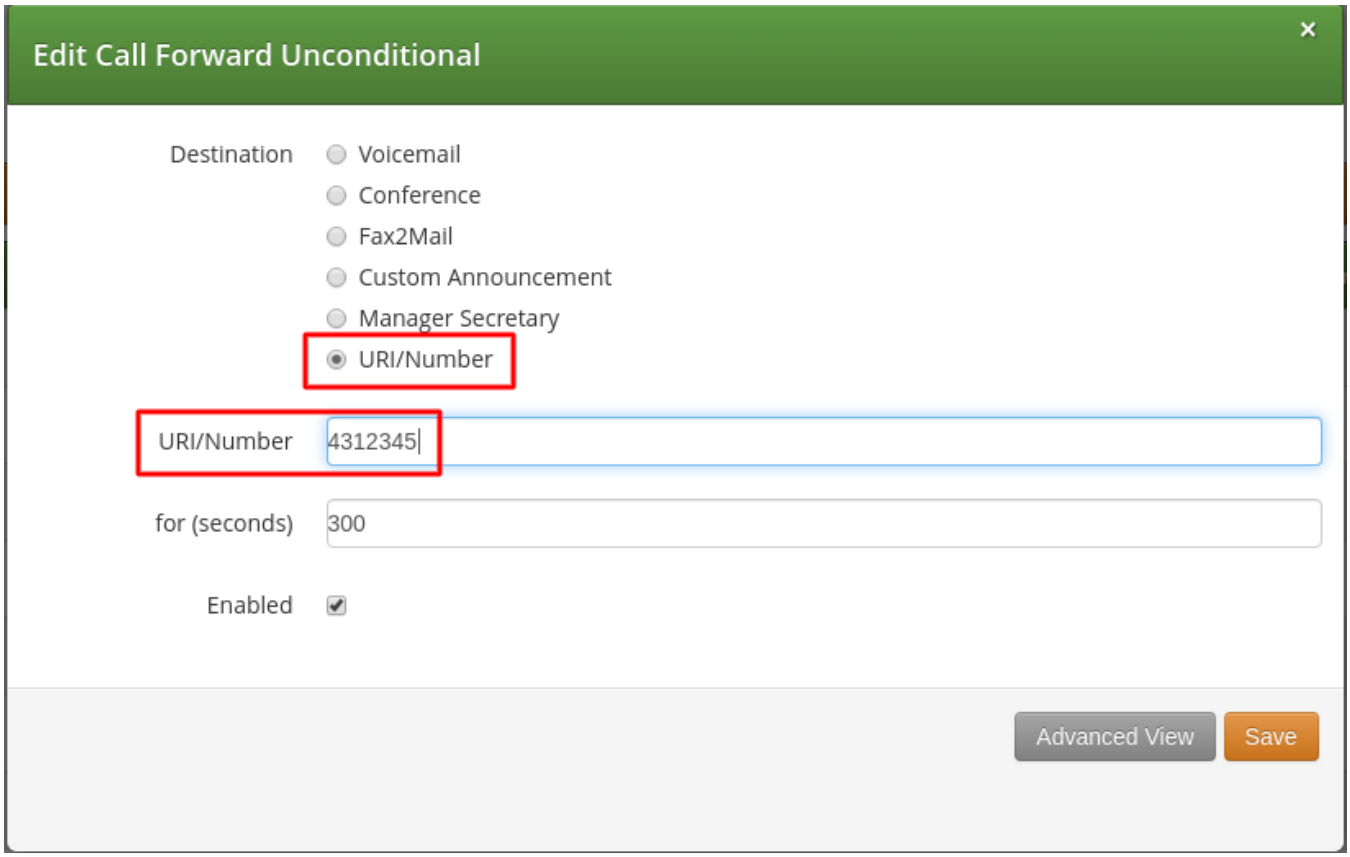
endpoint registered.

CAUTION The Call Forward Unavailable is also executed if the callee responds with *480 Temporarily Unavailable*, which may be the case when a subscriber's endpoint (e.g. an IP-PBX) is registered but the callee user is not available.

- **Call Forward SMS (CFS)**: The SMS forward is always executed, completely disregarding the subscriber state. SMS service has to be enabled, see the SMS (Short Message Service) subchapter for a detailed description on how to activate it.
- **Call Forward on Response (CFR)**: The call forward is executed only for particular reply codes received back from the destination endpoint. The list of the reply codes and the activation mode can be configured in *rerouting_codes* and *rerouting_mode* subscriber's preferences. Example: suppose that *rerouting_codes* is set to 503, *rerouting_mode* to whitelist and the CFR is configured. If that subscriber receives a call and it replies back a with code 503, then the call will be re-routed to the destination configured in the CFR. For all the other reply codes the CFR will be NOT executed.
- **Call Forward on Overflow (CFO)**: The call forward is executed when the new incoming call for the subscriber exceeds the limit configured in *concurrent_max_in_total*, *concurrent_max_in* or *concurrent_max_in_per_account* subscriber's preferences. If none of the preferences is set then the CFO will be NOT executed.

7.4.2. Setting a simple Call Forward

Go to your *Subscriber Preferences* and click *Edit* on the Call Forward Type you want to set (e.g. *Call Forward Unconditional*).



If you select *URI/Number* in the *Destination* field, you also have to set a *URI/Number*. The timeout defines for how long this destination should be tried to ring.

Copyright © Sipwise GmbH - All rights reserved

7.4.3. Call Forward Destinations

- **Voicemail**: Calls are forwarded to the Voicemail Application Server where the caller can leave a message.
- **Conference**: Calls are forwarded to the conference room. The subscriber is the host of the conference.
- **Fax2Mail**: Calls are forwarded to the Fax Server and the caller is supposed to leave a fax message. Note: The Fax2Mail feature must be enabled in the subscriber's preferences.
- **Custom Announcement**: A custom announcement is played back to the caller. Select an announcement from the *Custom announcement* list.
- **Manager Secretary**: Calls are forwarded to numbers defined in the "manager_secretary_numbers" subscriber preference. The "manager_secretary" feature must be enabled.
- **URI/Number**: The call is forwarded to the provided SIP-URI string or a number (See the *Call Forward Destination Extra Parameters* section below).

Call Forward Destination Options

- **URI/Number**: A destination to forward calls to. This option is only valid for the *URI/Number* destination type. Specify a valid SIP-URI string or a plain number.
- **for (seconds)**: Sets the ringing time, after which the call is forwarded to the next number on the list (if configured).
- **Custom Announcement**: Custom Announcements are created in Sound Sets and must have the name like 'custom_announcement_0', where the trailing symbol is a digit from 0 to 9.
- **Enabled**: Defines whether the Call Forward rule is being used or not.

7.4.4. Advanced Call Hunting

Beside call forwarding to a single destination, Sipwise C5 offers the possibility to activate call forwarding in a more sophisticated way:

- to multiple destinations (*Destination Set*)
- only during a pre-defined time set (*Time Set*)
- only for specific callers (*Source Set*)
- only for specific callee (*B-Number Set*)

If you want to define such more detailed call forwarding rules, you need to change into the *Advanced View* when editing your call forward. There, you can select multiple *Destination Set - Time Set - Source Set - B-Number Set* groups that determine all conditions under which the call will be forwarded.

Explanation of call forward parameters

- A ***Destination Set*** is a list of destinations where the call will be routed to, one after another, according to the order of their assigned priorities. See the Destination Sets subchapter for a detailed description.
- A ***Time Set*** is a time period definition, i.e. when the call forwarding has to be active. See the Time Sets subchapter for a detailed description.
- A ***Source Set*** is a list of number patterns that will be matched against the calling party number; if the

 Copyright © Sipwise GmbH - All rights reserved

calling number matches the call forwarding will be executed. See the Source Sets subchapter for a detailed description.

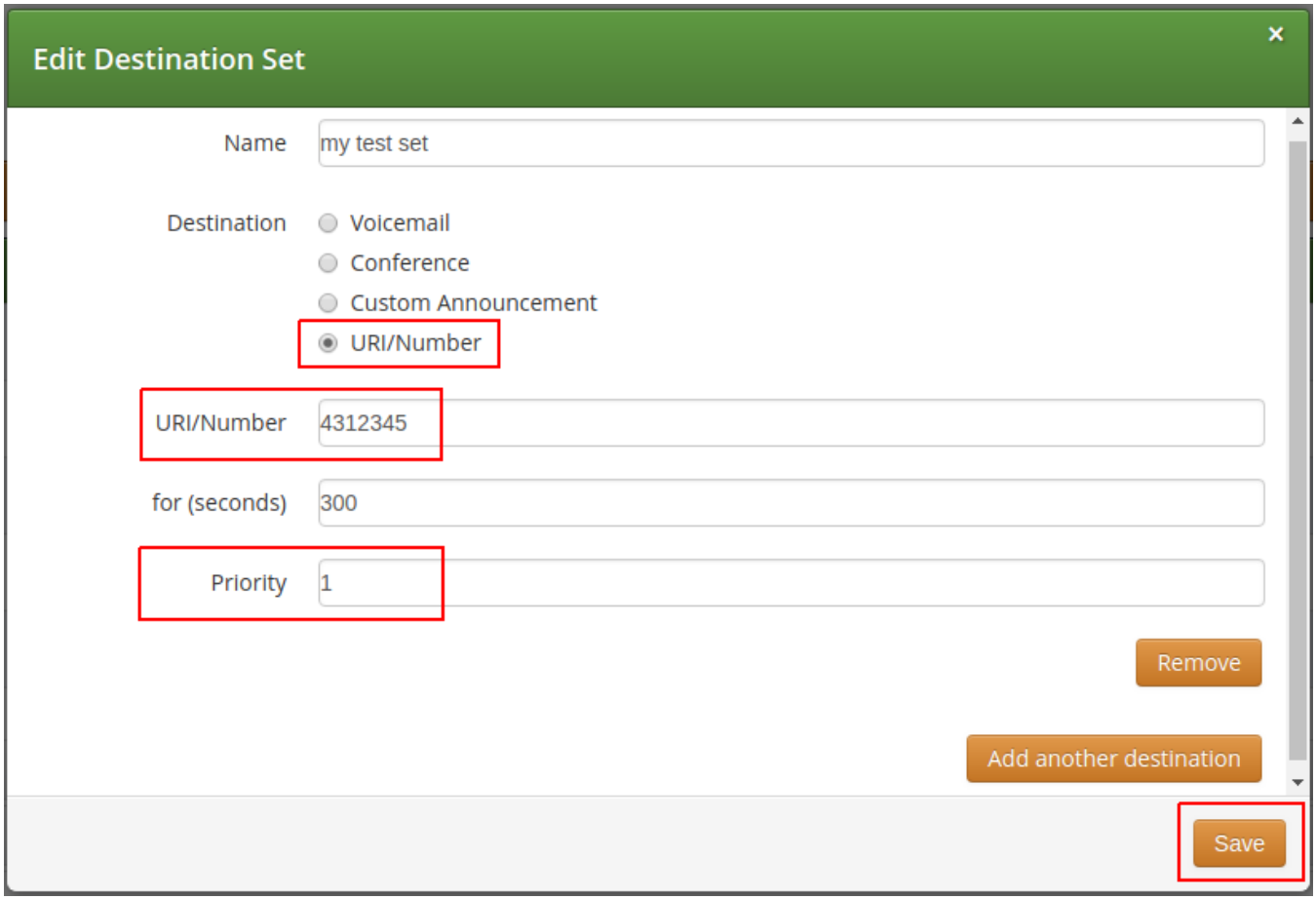
- A ***B-Number Set*** is a list of number patterns that will be matched against the called party number; if the callee number matches the call forwarding will be executed. See the B-Number Sets subchapter for a detailed description.

7.4.5. Show Call Forward Destination to Caller (colp_cf)

When using extension numbers belonging to the same customer pbx, there is the possibility to take advantage of the "colp_cf" subscriber preference. If "colp_cf" subscriber's preference is enabled, during a call forward, Sipwise C5 will try to send to the original caller some information about the final destination using the P-Asserted-Identity header. This extra header can be read from the calling user agent while ringing or during the conversation, and display the correct forward destination in place of the original dialed destination.

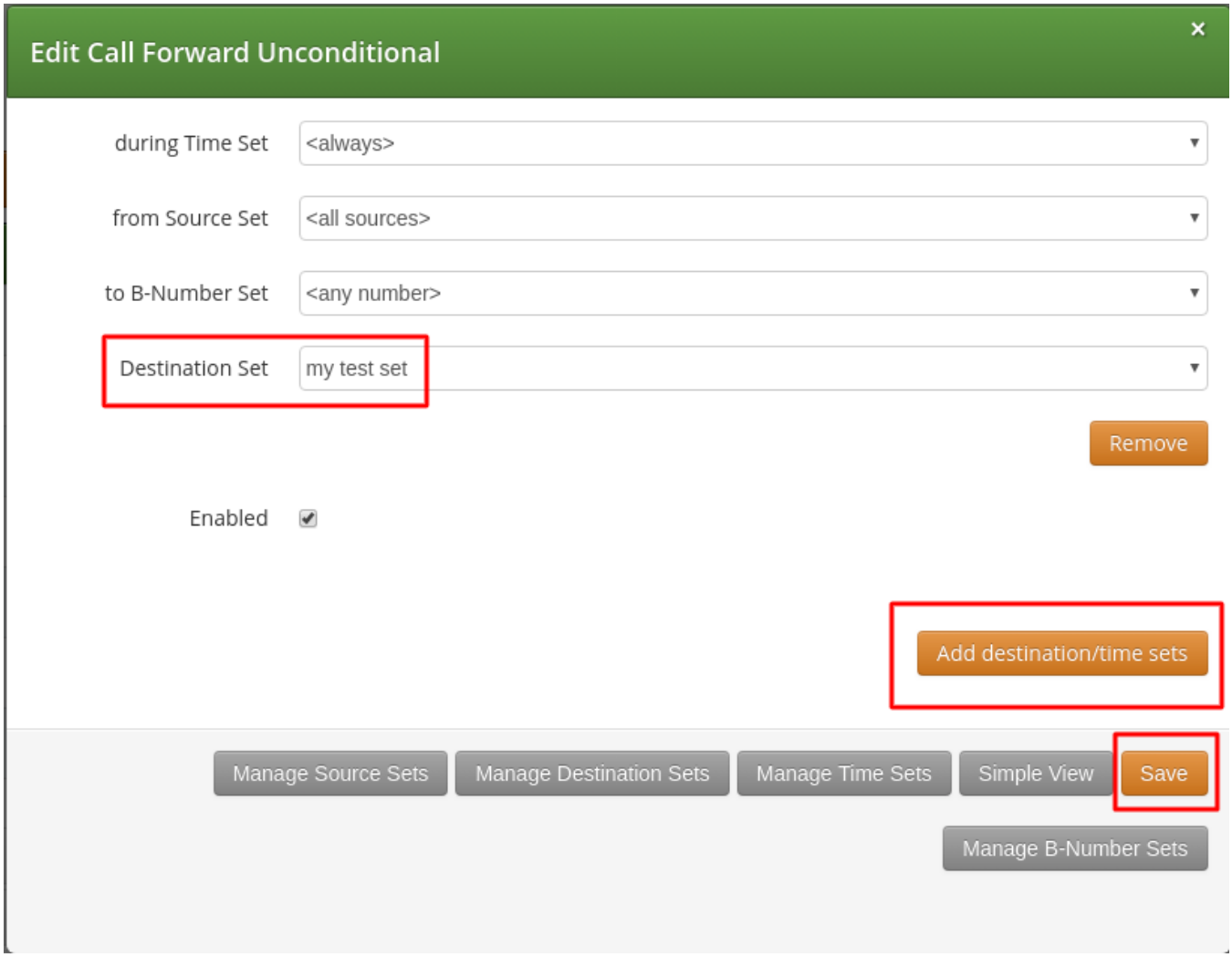
Configuring Destination Sets

Click on *Manage Destination Sets* to see a list of available sets. The *quickset_cfu* has been implicitly created during our creation of a simple call forward. You can edit it to add more destinations, or you can create a new destination set.



When you close the *Destination Set* Overview, you can now assign your new set in addition or instead of the *quickset_cfu* set.

Copyright © Sipwise GmbH - All rights reserved

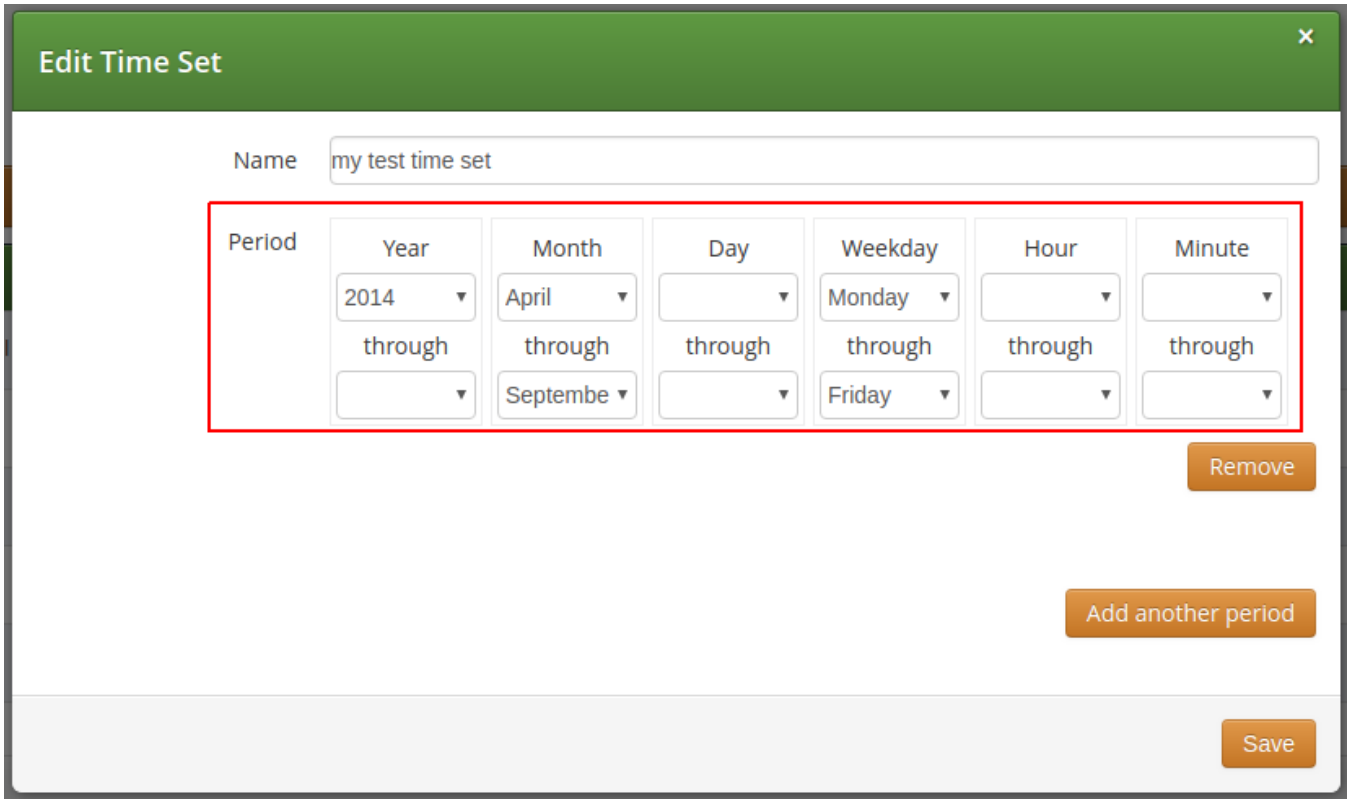


Press *Save* to store your settings.

Configuring Time Sets

Click on *Manage Time Sets* in the advanced call-forward menu to see a list of available time sets. By default there are none, so you have to create one.

Copyright © Sipwise GmbH - All rights reserved



You need to provide a *Name*, and a list of *Periods* where this set is active. If you only set the top setting of a date field (like the *Year* setting in our example above), then it's valid for only this setting (like the full year of *2013* in our case). If you provide the bottom setting as well, it defines a period (like our *Month* setting, which means from beginning of April to end of September). For example, if a CF is set with the following timeset: "hour { 10-12 } minute { 20-30 }", the CF will be matched within the following time ranges:

- from 10.20am to 10:30am
- from 11.20am to 11:30am
- from 12.20am to 12:30am

IMPORTANT the period is a *through* definition, so it covers the full range. If you define an *Hour* definition *8-16*, then this means from *08:00* to *16:59:59* (unless you filter the *Minutes* down to something else).

If you close the *Time Sets* management, you can assign your new time set to the call forwards you're configuring.

Configuring Source Sets

Once the *Advanced View* of the call forward definition has been opened, you will need to press the *Manage Source Sets* button to start defining new Source Sets or managing an existing one. The following image shows the Source Set definition dialog:

Copyright © Sipwise GmbH - All rights reserved

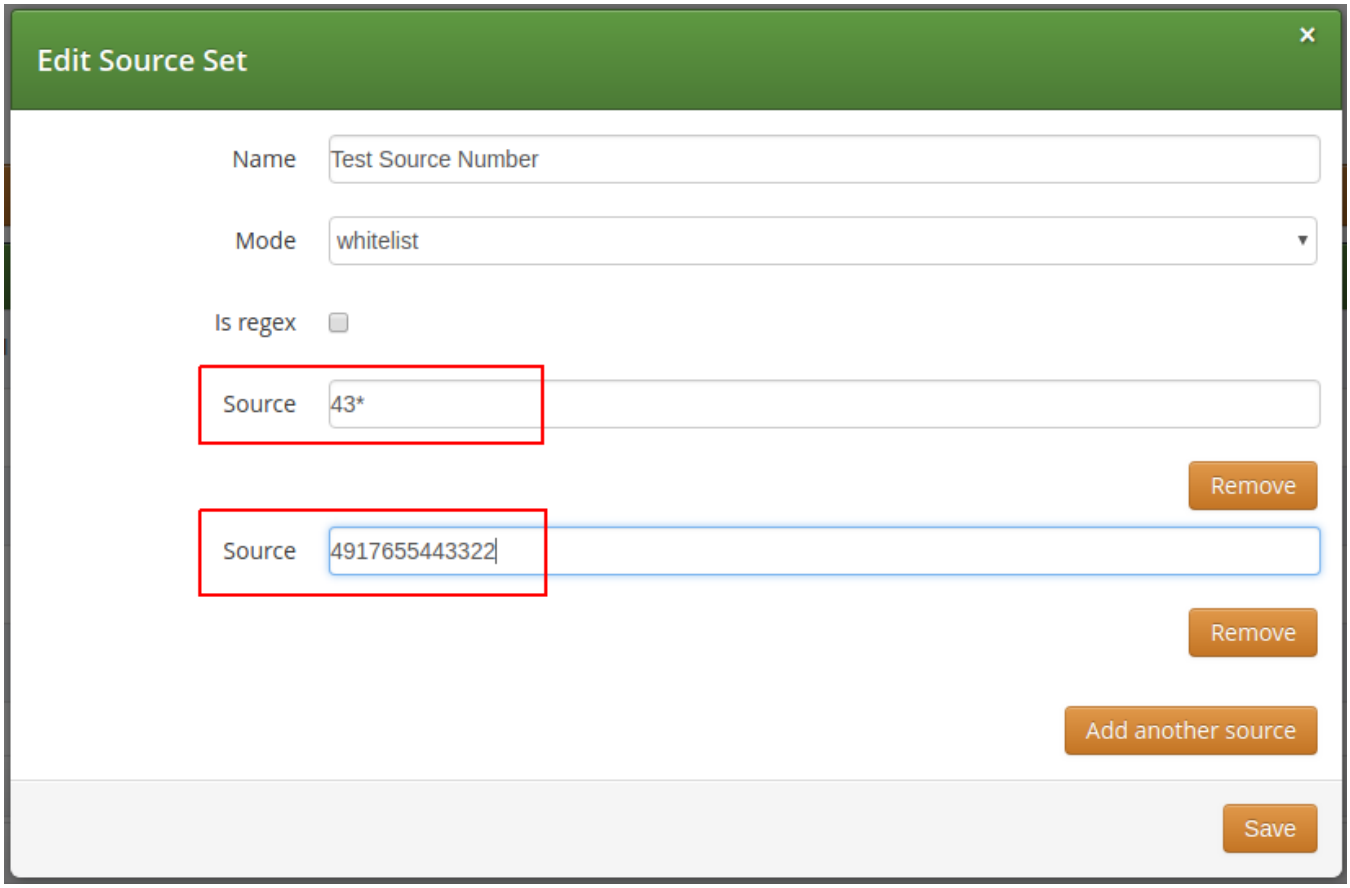


Figure 60. Creating a Call Forward Source Set

You will need to fill in the Name field first, the Mode: whitelist or blacklist, the is_regex flag and finally in the Source field you can enter:

- A simple phone number in E.164 format
- A pattern, in order to define a range of numbers. You can use "*" (matches a string of 0 to any number of characters), "?" (matches any single character), "[abc]" (matches a single character that is part of the explicitly listed set: a, b or c) and "[0-9]" (matches a single character that falls in the range 0 to 9) as wildcards, as usual in shell patterns. Examples:

  "431*" (all numbers from Vienna / Austria)

  "49176[0-5]77*" (German numbers containing fixed digits and a variable digit in 0-5 range in position 6)

  "43130120??" (numbers from Vienna with fixed prefix and 2 digits variable at the end)
- A perl compatible regular expressions (only if is_regex if set). Capturing groups can be formed using parentheses and referenced in the *Destination Set* via \\1, \\2,...
- The constant string "anonymous" that indicates a suppressed calling number (CLIR)

You can add more patterns to the Source Set by pressing the *Add another source* button. When you finished adding all patterns, press the *Save* button. You will then see the below depicted list of Source Sets:

 Copyright © Sipwise GmbH - All rights reserved

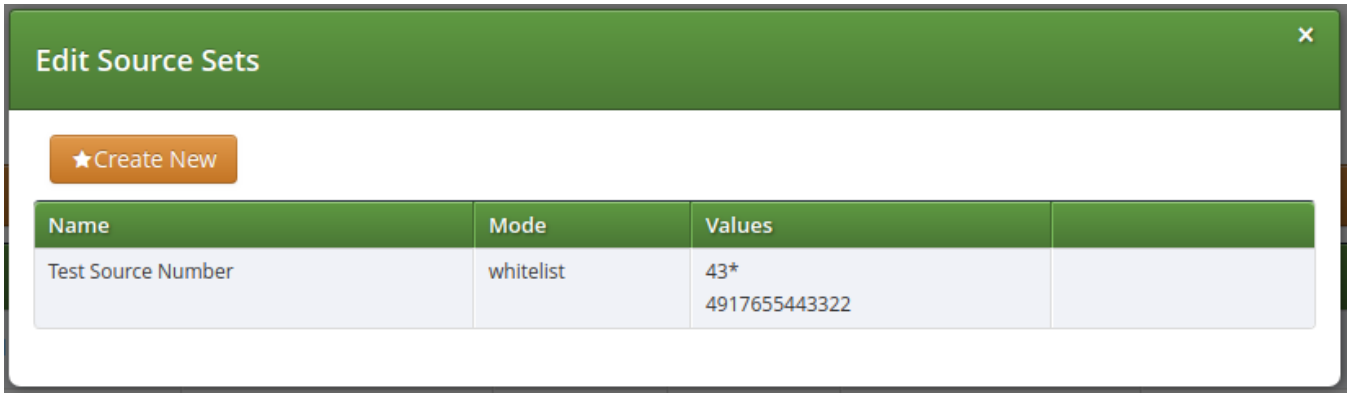


Figure 61. List of Call Forward Source Sets

Configuring B-Number Sets

Once the *Advanced View* of the call forward definition has been opened, you will need to press the *Manage B-Number Sets* button to start defining new B-Number Sets or managing an existing one. The following image shows the B-Number Set definition dialog:

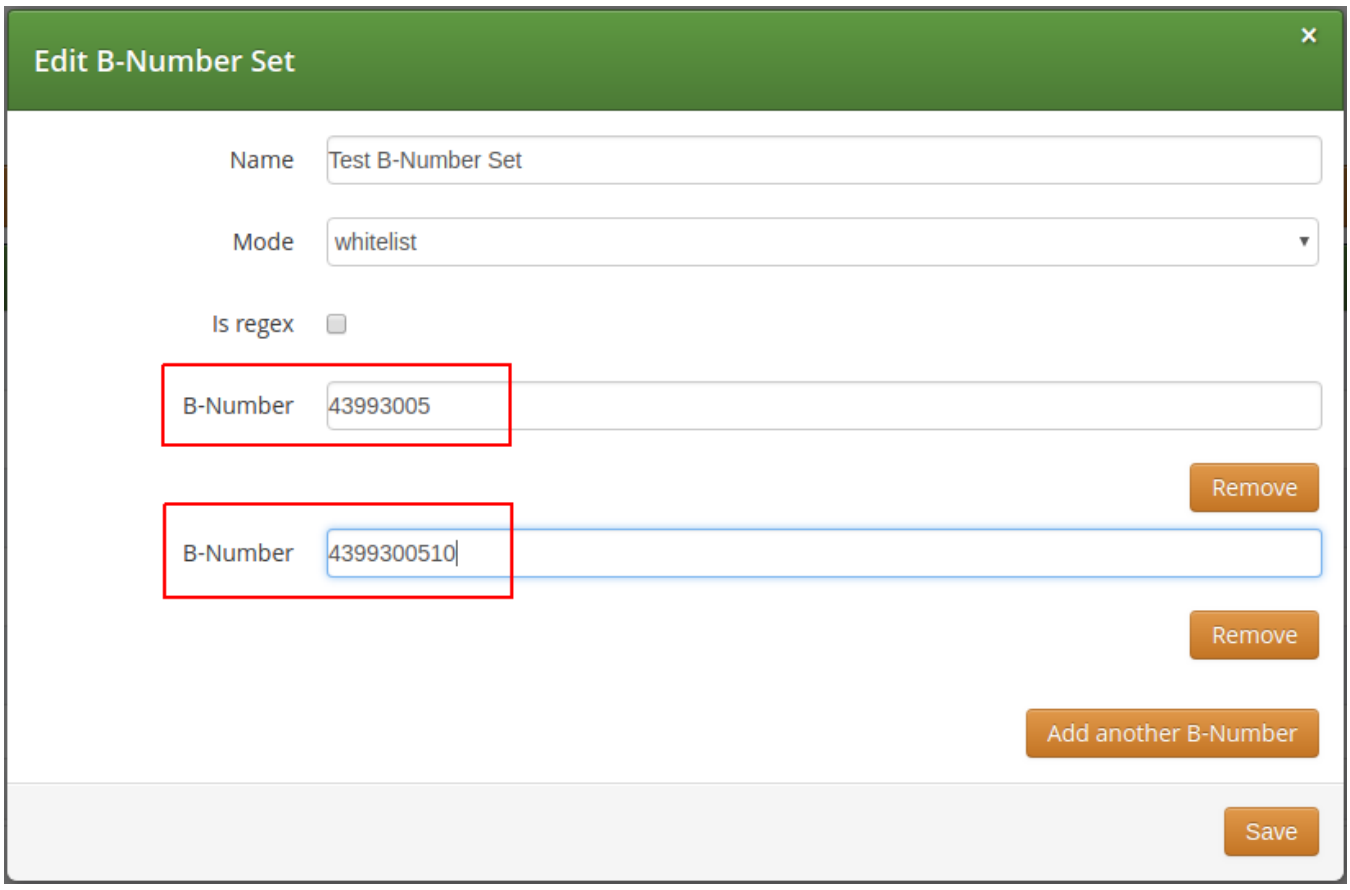


Figure 62. Creating a Call Forward B-Number Set

You will need to fill in the Name field first, the Mode: whitelist or blacklist, the is_regex flag and finally in the B-Number field you can enter:

- A simple phone number in E.164 format
- A pattern, in order to define a range of numbers. You can use "*" (matches a string of 0 to any

Copyright © Sipwise GmbH - All rights reserved

number of characters), "?" (matches any single character), "[abc]" (matches a single character that is part of the explicitly listed set: a, b or c) and "[0-9]" (matches a single character that falls in the range 0 to 9) as wildcards, as usual in shell patterns. Examples:

"431*" (all numbers from Vienna / Austria)

"49176[0-5]77*" (German numbers containing fixed digits and a variable digit in 0-5 range in position 6)

"43130120??" (numbers from Vienna with fixed prefix and 2 digits variable at the end)

- A perl compatible regular expressions (only if is_regex if set). Capturing groups can be formed using parentheses and referenced in the *Destination Set* via \\1, \\2,...

You can add more patterns to the B-Number Set by pressing the *Add another B-Number* button. When you finished adding all patterns, press the *Save* button. You will then see the below depicted list of B-Number Sets:

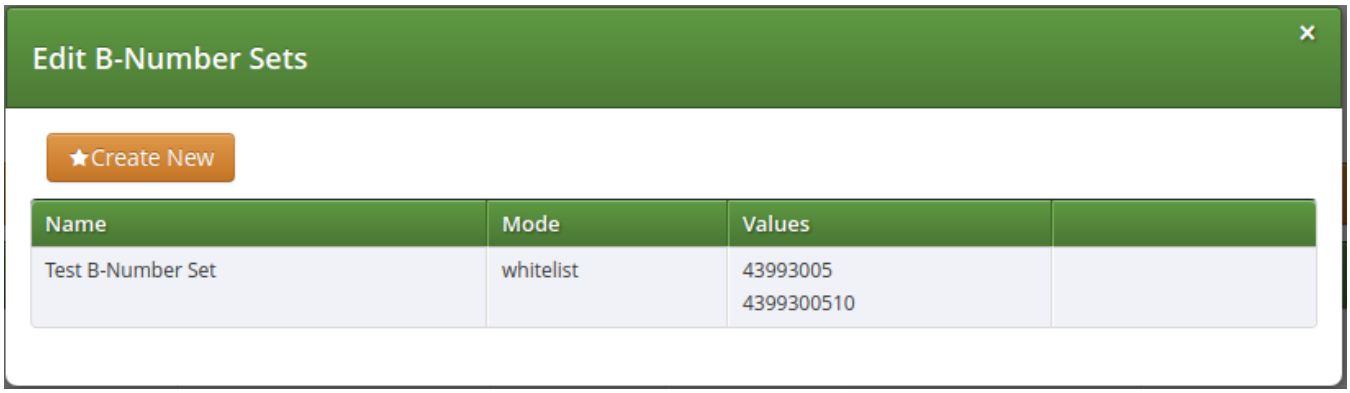


Figure 63. List of Call Forward B-Number Sets

Finalizing the call forward definition

As additional step you can define a Destination Set as described in Destination Sets subchapter. For our example, we have defined the following Destination Set:

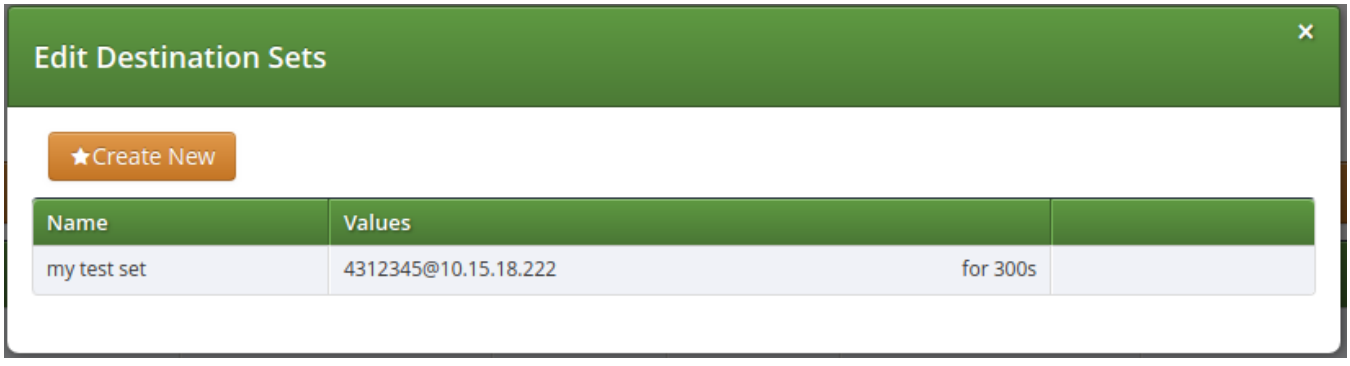


Figure 64. List of Call Forward Destination Sets

A final step of defining the call forward settings is selecting a Destination, a Time Set, a Source Set and a B-Number Set, as shown in the image below. *Please note* that there is no specific Time Set selected in our example, that means the call forward rule is valid (as shown) <always>.

 Copyright © Sipwise GmbH - All rights reserved

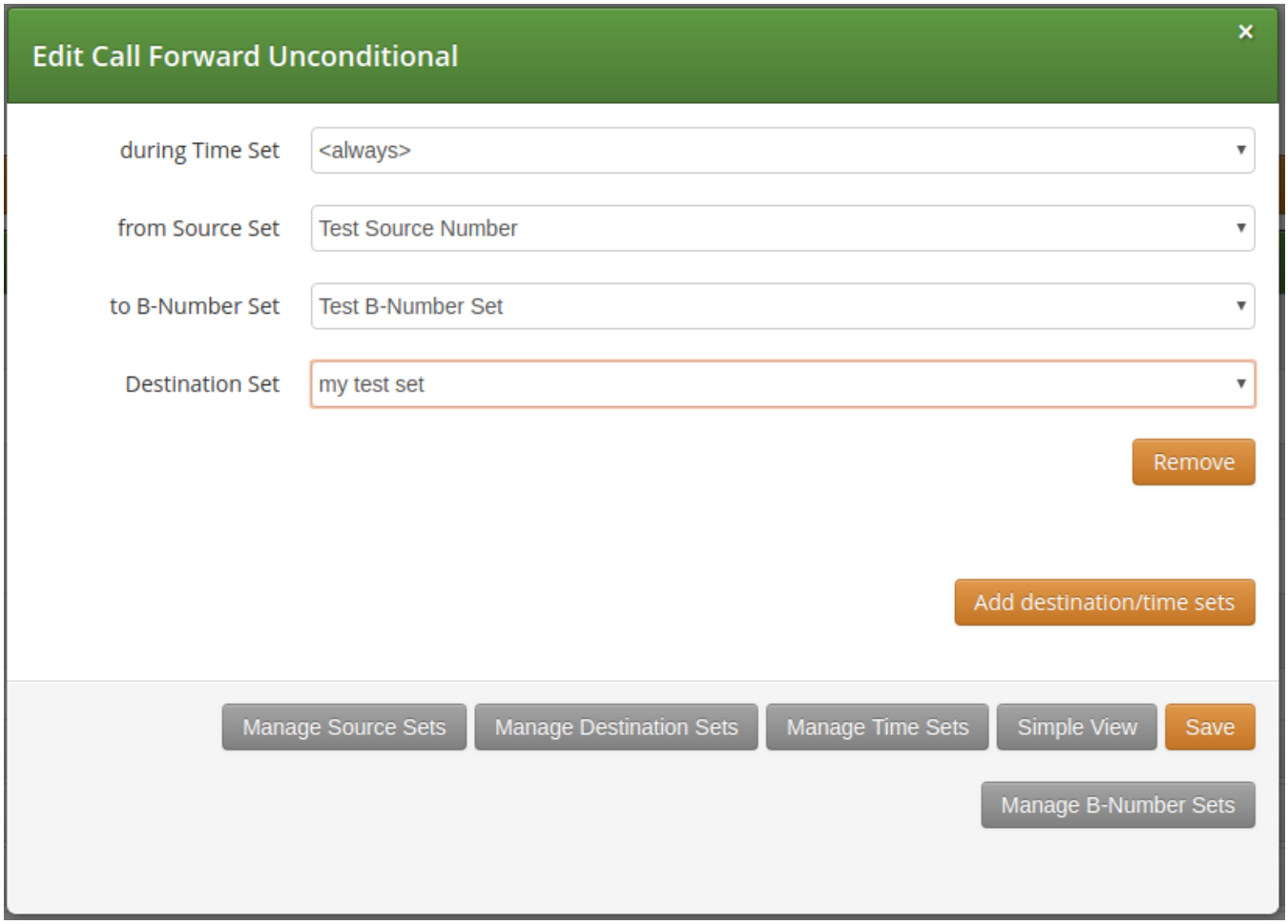


Figure 65. Definition of a Call Forward with Source and Destination Sets

Once all the settings have been defined and the changes are saved, you will see the call forward entry (in our example: *Call Forward Unconditional*), with the names of the selected Destination, Time Set, Source Sets and B-Number Set provided, at *SubscriberPreferences Call Forwards* location on the web interface:

Copyright © Sipwise GmbH - All rights reserved

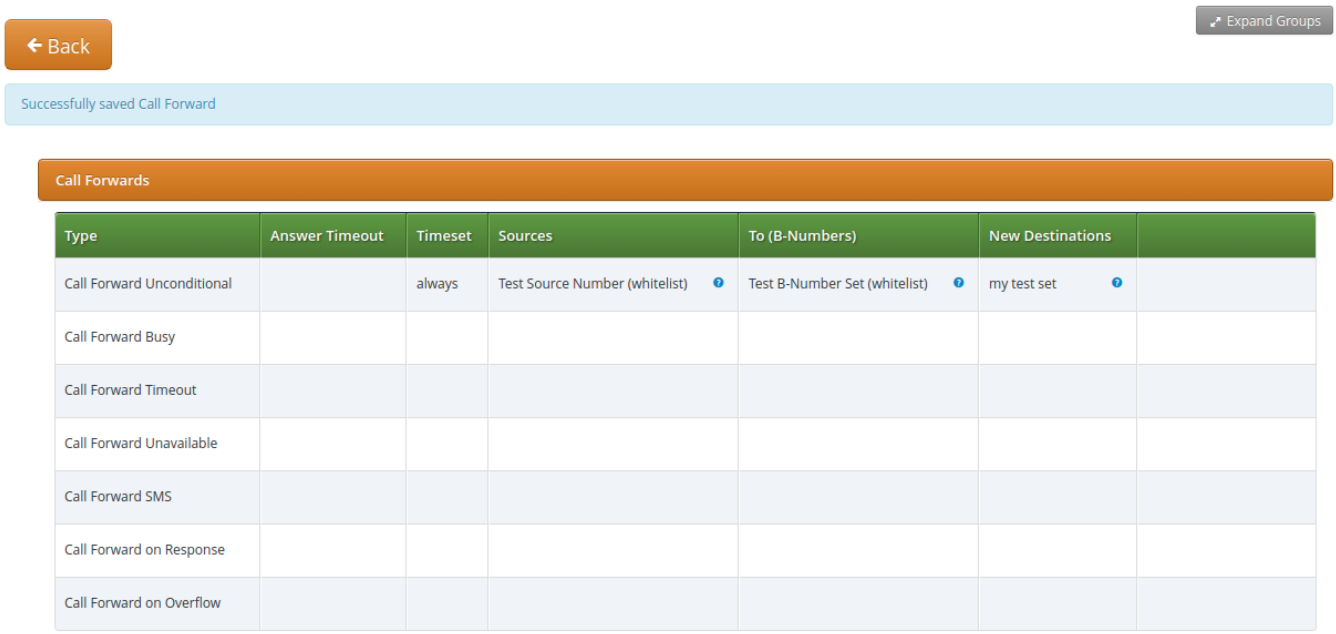


Figure 66. List of Call Forward with Source and Destination Sets

7.5. Call Forking by Q value

The Sipwise C5 platform allows you to register multiple devices under the same subscriber. By the default, the maximum number of the device you can register is 5. This value is configurable via *kamailio proxymax_registrations_per_subscriber* preference in *config.yml*.

If a customer registers multiple devices, Sipwise C5 – once receives a call for that user – send the call to all the registered devices, in parallel. All the devices will ring at the same time. This is called Parallel Forking, and this is the default behavior. The Sipwise C5 can also do the so-called Serial Forking, which means let ring one device first, then after a timeout let ring the next device, and so on and so forth. The Serial Forking feature can be activated setting subscriber/domain preference *serial_forking_by_q_value*.

7.5.1. The Q value

Serial Forking is based on SIP Contact's parameter called 'Q value', which is a priority number, set by the clients during their Registration. The q value is a floating point number in a range 0 to 1.0 specified as a parameter in the Contact header field.

In case the client doesn't set the q value, Sipwise C5 set a default value of 'q=-1' in the database.

Q value can be also specified during the creation of a subscriber's permanent registration (*Details Registered DevicesCreate Permanent Registration*).

The Sipwise C5 can apply two different type of algorithm to the q values in order to achieve two different types of serial forking called *standard* and *probability*.

7.5.2. The Standard Method

This method uses the q values as a pure priority index. The higher the q value number, the more priority that device has. Contacts with q value 1.0 have maximum priority, so such contacts will be always tried first in serial forking. Contacts with q value 0 have the lowest priority and they will be tried after all other

 Copyright © Sipwise GmbH - All rights reserved

contacts with higher priority.

NOTE In case two or more contacts have the same q value, then they are tried in parallel. This allow to create a very flexible mix of Serial and Parallel forking.

This method can be activated setting *Standard* in subscriber/domain preference *serial_forking_by_q_value*.

7.5.3. The Probability Method

This method uses the q values as the weight of the contact. The higher the q value number, the more probability that the device has to ring first. Equals q values means equals probability to be tried. Contacts with q values equals to 0 or lower are not considered by the ordering algorithm, but added at the end of the list as backup option if all other contacts fail.

NOTE Differently from the *standard* method there is no possibility to have parallel forking. This algorithm can be useful to load-balance the calls in case of endpoints in ACTIVE-ACTIVE configuration.

This method can be activated setting *Probability* in subscriber/domain preference *serial_forking_by_q_value*.

7.5.4. Advanced Configurations

If a subscriber with Serial Forking enabled receives a call, Sipwise C5 calls the registered devices one after the another. The forking is stopped only in the following cases:

- there are no more devices to try to contact
- one of the ringing devices answers the call
- one of the ringing devices replies with the SIP code 600, 603, 604, 606 or one of the response codes defined in *stop_forking_code_lists* subscriber/domain preference.
- a Call Forward on Timeout is set and the ringtimeout is reached.

Sipwise C5 allows you also to define how long each single device has to ring during a Serial Forking call. To do that, set the subscriber/domain preference *contact_ringtimeout* to the desired value.

NOTE In case both *contact_ringtimeout* and Call Forward on Timeout are configured, CFT timeout has higher priority. To clarify this concept, please take a look at the following examples: Case 1: CFT timeout lower than the total ringtimeout of the contacts. For example: CFT timeout = 100, contact_ringtimeout = 40 and 3 devices registered: 1st device will ring for 40 seconds, 2nd device will ring for other 40 seconds, 3rd device will ring only for 20 seconds because of the CFT. Case 2: CFT timeout higher than the total ringtimeout of the contacts. For example: CFT timeout = 100, contact_ringtimeout = 40 and 2 devices registered: 1st device will ring for 40 seconds, 2nd device will ring till reaching the CFT timeout (60 seconds in this case).

7.6. Local Number Porting

The Sipwise C5 platform comes with two ways of accomplishing local number porting (LNP):

Copyright © Sipwise GmbH - All rights reserved

- one is populating the integrated LNP database with porting data,
- the other is accessing external LNP databases via the Sipwise LNP daemon using the LNP API.

NOTE Accessing external LNP databases is available for PRO and CARRIER products only.

7.6.1. Local LNP Database

The local LNP database provides the possibility to define LNP Carriers (the owners of certain ported numbers or number blocks) and their corresponding LNP Numbers belonging to those carriers. It can be configured on the admin panel in *SettingsNumber Porting* or via the API. The LNP configuration can be populated individually or via CSV import/export both on the panel and the API.

LNP Carriers

LNP Carriers are defined by an arbitrary *Name* for proper identification (e.g. *British Telecom*) and contain a *Prefix* which can be used as routing prefix in LNP Rewrite Rules and subsequently in Peering Rules to route calls to the proper carriers. The LNP prefix is written to CDRs to identify the selected carrier for post processing and analytics purposes of CDRs. LNP Carrier entries also have an *Authoritative* flag indicating that the numbers in this block belong to the carrier operating Sipwise C5 . This is useful to define your own number blocks, and in case of calls to those numbers reject the calls if the numbers are not assigned to local subscribers (otherwise they would be routed to a peer, which might cause call loops). Finally the *Skip Rewrite* flag skips executing of LNP Rewrite Rules if no number manipulation is desired for an LNP carrier.

LNP Numbers

LNP Carriers contain one or more LNP Numbers. Those LNP Numbers are defined by a *Number* entry in E164 format (<*cc*><*ac*><*sn*>) used to match a number against the LNP database. Number matching is performed on a longest match, so you can define number blocks without specifying the full subscriber number (e.g. a called party number *431999123* is going to match an entry *431999* in the LNP Numbers).

For an LNP Numbers entry, an optional *Routing Number* can be defined. This is useful to translate e.g. premium 900 or toll-free 800 numbers to actual routing numbers. If a Routing Number is defined, the called party number is implicitly replaced by the Routing Number and the call processing is continued with the latter. For external billing purposes, the optional *Type* tag of a matched LNP number is recorded in CDRs.

An optional *Start Date* and *End Date* makes it possible to schedule porting work-flows up-front by populating the LNP database with certain dates, and the entries are only going to become active with those dates. Empty values for start indicate a start date in the past, while empty values for end indicate an end time in the future during processing of a call, allowing to define infinite date ranges. As intervals can overlap, the LNP number record with a start time closest to the current time is selected.

Enabling local LNP support

In order to activate Local LNP during routing, the feature must be activated in *config.yml*. Set *kamailio proxylnpenable* to *yes* and *kamailioproxylnptype* to *local*.

LNP Routing Procedure

When a call arrives at the system, the calling and called party numbers are first normalized using the

 Copyright © Sipwise GmbH - All rights reserved

Inbound Rewrite Rules for Caller and *Inbound Rewrite Rules for Callee* within the rewrite rule set assigned to the calling party (a local subscriber or a peer).

If the called party number is not assigned to a local subscriber, or if the called party is a local subscriber and has the subscriber/domain preference *lnp_for_local_sub* set, the LNP lookup logic is engaged, otherwise the call proceeds without LNP lookup. The further steps assume that LNP is engaged.

If the call originated from a peer, and the peer preference *caller_lnp_lookup* is set for this peer, then an LNP lookup is performed using the normalized calling party number. The purpose for that is to find the LNP prefix of the calling peer, which is then stored as *source_lnp_prefix* in the CDR, together with the selected LNP number's *type* tag (*source_lnp_type*). If the LNP lookup does not return a result (e.g. the calling party number is not populated in the local LNP database), but the peer preference *default_lnp_prefix* is set for the originating peer, then the value of this preference is stored in *source_lnp_prefix* of the CDR.

Next, an LNP lookup is performed using the normalized called party number. If no number is found (using a longest match), no further manipulation is performed.

If an LNP number entry is found, and the *Routing Number* is set, the called party number is replaced by the routing number. Also, if the *Authoritative* flag is set in the corresponding LNP Carrier, and the called party number is not assigned to a local subscriber, the call is rejected. This ensures that numbers allocated to the system but not assigned to subscribers are dropped instead of routed to a peer.

IMPORTANT

If the system is serving a local subscriber with only the routing number assigned (but not e.g. the premium number mapping to this routing number), the subscriber will not be found and the call will either be rejected if the called party premium number is within an authoritative carrier, or the call will be routed to a peer. This is due to the fact that the subscriber lookup is performed with the dialled number, but not the routing number fetched during LNP. So make sure to assign e.g. the premium number to the local subscriber (optionally in addition to the routing number if necessary using alias numbers) and do not use the LNP routing number mechanism for number mapping to local subscribers.

Next, if the LNP carrier does not have the *Skip Rewriting* option set, the *LNP Rewrite Rules for Callee* are engaged. The rewrite rule set used is the one assigned to the originating peer or subscriber/domain via the *rewrite_rule_set* preference. The variables available in the match and replace part are, beside the standard variables for rewrite rules:

- `${callee_lnp_prefix}`: The prefix stored in the LNP Carrier
- `${callee_lnp_basenumber}`: The actual number entry causing the match (may be shorter than the called party number due to longest match)

Typically, you would create a rewrite rule to prefix the called party number with the *callee_lnp_prefix* by matching `^([0-9]+)$` and replacing it by `${callee_lnp_prefix}\1`.

Once the LNP processing is completed, the system checks for further preferences to finalize the number manipulation. If the terminating local subscriber or peer has the preference *lnp_add_npdi* set, the Request URI user-part is suffixed with `;npdi`. Next, if the preference *lnp_to_rn* is set, the Request URI user-part is suffixed with `;rn=LNP_ROUTING_NUMBER`, where *LNP_ROUTING_NUMBER* is the *Routing Number* stored for the number entry in the LNP database, and the originally called number is kept in place. For example, if *lnp_to_rn* is set and the number *1800123* is called, and this number has a routing number *1555123* in the LNP database, the resulting Request-URI is

Copyright © Sipwise GmbH - All rights reserved

`sip:1800123;rn=1555123@example.org`.

Finally, the *destination_lnp_prefix* in the CDR table is populated either by the prefix defined in the Carrier of the LNP database if a match was found, or by the *default_lnp_prefix* preference of the destination peer or subscriber/domain.

Blocking Calls Using LNP Data

The Sipwise C5 provides means to allow or block calls towards ported numbers that are hosted by particular LNP carriers. Please visit Creating Rules per NCOS Level in the handbook to learn how this can be achieved.

Transit Calls using LNP

If a call originated from a peer and the peer preference *force_outbound_calls_to_peer* is set to *force_nonlocal_lnp* (the *if callee is not local and is ported* selection in the panel), the call is routed back to a peer selected via the peering rules.

This ensures that if a number once belonged to your system and is ported out, but other carriers are still sending calls to you (e.g. selecting you as an anchor network), the affected calls can be routed to the carrier the number got ported to.

CSV Format

The LNP database can be exported to CSV, and in the same format imported back to the system. On import, you can decide whether to drop existing data prior to applying the data from the CSV.

The CSV file format contains the fields in the following order:

Table 13. LNP CSV Format

Name	Description
Carrier Name	The *Name* in the LNP Carriers table (string, e.g. *My Carrier*)
Carrier Prefix	The *Prefix* in the LNP Carriers table (string, e.g. *DD55*)
Number	The *Number* in the LNP Numbers table (E164 number, e.g. *1800666*)
Routing Number	The *Routing Number* in the LNP Numbers table (E164 number or empty, e.g. *1555666*)
Start	The *Start* in the LNP Numbers table (YYYY-MM-DD or empty, e.g. *2016-01-01*)
End	The *End* in the LNP Numbers table (YYYY-MM-DD or empty, e.g. *2016-12-30*)
Authoritative	The *Authoritative* flag in the LNP Carriers table (0 or 1)
Skip Rewrite	The *Skip Rewrite* flag in the LNP Carriers table (0 or 1)

 Copyright © Sipwise GmbH - All rights reserved

Name	Description
Type	The *Type* tag in the LNP Numbers table (alphanumeric string, e.g. *mobile*)

Local LNP returned values

If a match in the local LNP table is found corresponding LNP Carrier code will be stored in CDR data.

Additionally two dedicated headers can be added to the outgoing SIP message:

- P-NGCP-LNP-Number: The returned LNP number, if any
- P-NGCP-LNP-Status: The LNP query return code (200 if successful, 404 if no entry found)

This feature is not enabled by default, but can be activated with the following parameters:

- kamailio->proxy->lnp->add_reply_headers->enable : *no*
- kamailio->proxy->lnp->add_reply_headers->number : *P-NGCP-LNP-Number*
- kamailio->proxy->lnp->add_reply_headers->status : *P-NGCP-LNP-Status*

7.6.2. External LNP via LNP API

External LNP relies on the *NGCP LNP Daemon (ngcp-lnpd)* which kamailio-proxy is talking to via a defined JSONRPC protocol. The proxy sends the A and B number to *ngcp-lnpd*, which translates it to a SIP INVITE message sent to an external server. *ngcp-lnpd* supports at the moment 2 different types of redirection servers:

- Squire SIP-to-INAP gateway: it performs an SS7 INAP request to fetch the LNP result, which is passed back as a binary blob in a 3xx response to *ngcp-lnpd*. *ngcp-lnpd* extracts the TCAP body of the response and returns the information back to the proxy.
- Plain 302 Redirecting: it performs an internal lookup to fetch the LNP result, which is passed back as SIP headers in a 3xx response to *ngcp-lnpd*. *ngcp-lnpd* extracts the redirecting number from the username of the Contact header and the LNP code from the X-LNP header (the name is configurable), and then returns this information back to the proxy.

Enabling LNP lookup via API

In order to activate LNP lookup via API during call routing, the feature must be activated in /etc/ngcp-config/config.yml. Set these parameters:

- kamailio->proxy->lnp->enable : *yes*
- kamailio->proxy->lnp->type : *api*
- lnpd->enable : *yes*

There is a possibility to explicitly allow (whitelist) or deny (blacklist) certain number ranges for which an LNP lookup may be done. The relevant configuration parameters are at kamailio->proxy->lnp->lnp_request_whitelist and kamailio->proxy->lnp->lnp_request_blacklist. For each entry in the list a POSIX regex expression may be used, see the following example:

Copyright © Sipwise GmbH - All rights reserved

```
    lnp:
      lnp_request_whitelist:
        - '^9'
        - '^800'
      lnp_request_blacklist:
        - '^1'
        - '^900'
        - '^110'
        - '^112'
```

Interpretation of the above lists (that are based on numbers represented in national format):

- whitelist: *do* LNP lookup for any called number that starts with '9' or '800'
- blacklist: *do not* perform LNP lookup for any called number that starts with '1', '900', '110' or '112'

IMPORTANT If both whitelist and blacklist are defined, the LNP lookup is only performed when the called number matches any of the whitelist patterns and does not match any of the blacklist patterns.

Squire SIP-to-INAP gateway: Refine LNP and FCI decoding

Preconfigured parameters for Squire SIP-to-INAP gateway method should already make it possible to correctly decode the LNP number and FCI code contained in the received TCAP body. If the external server replies with a non-standard TCAP body, it is possible to fine tune the information extraction. Edit the following parameters in order to point to the correct fields:

- kamailio->proxy->lnp->api->tcap_field_lnp : *ConnectArg.destinationRoutingAddress.0*
- kamailio->proxy->lnp->api->tcap_field_opcode : *end.components.0.invoke.opCode*
- kamailio->proxy->lnp->api->tcap_field_fci : *end.components.0.invoke.parameter*

Plain 302 Redirecting: Refine LNP and FCI decoding

A preconfigured setup to connect to a plain 302 redirecting server is already deployed on the system and only has to be activated. In case you need to change the name of the header that will return the LNP code, please check Configuration of LNP daemon.

The Redundancy Feature

It is possible to set up *LNP daemon* to provide a kind of redundant service to the Proxy. This means the *LNP daemon* will send its LNP query to more LNP serving nodes that are predefined in a list. (See Configuration of LNP daemon chapter for details.) The LNP query may happen in 2 ways:

- **round-robin**: *LNP daemon* sends the query to one of the serving nodes then waits for the response for a configurable timeout. If it does not get the response in time, it sends the LNP query to the next serving node.
- **parallel**: *LNP daemon* sends the query to all of the serving nodes then waits for the response, and will accept the first response that it receives.

Copyright © Sipwise GmbH - All rights reserved

Configuration of Sipwise LNP Daemon

LNP daemon takes its active configuration from /etc/ngcp-lnpd/config.yml file. The file is generated automatically—when a new Sipwise C5 configuration is applied (ngcpcfg apply...)—from the main Sipwise C5 configuration file: /etc/ngcp-config/config.yml and a template: /etc/ngcp-config/template/etc/ngcp-lnpd/config.yml.tt2. System administrators are only expected to modify the lnpd.config section of main configuration file /etc/ngcp-config/config.yml.

A sample *LNP daemon* configuration file (/etc/ngcp-lnpd/config.yml) looks like:

```
daemon:
        json-rpc:
                ports:
                        - 54321
                        - 12345
                interfaces:
                        - 127.0.0.1
                        - 192.168.1.90
                        - ::1

        sip:
                port: 5095
                address: 0.0.0.0

        threads: 4
        foreground: false
        pidfile: /run/ngcp-lnpd.pid
        loglevel: 6

instances:
        default_sigtran:
                module: sigtran
                destination: 192.168.1.99
                from-domain: test.example.com
                headers:
                        - header: INAP-Service-Key
                          value: 2
                reply:
                        tcap: raw-tcap
        redundant:
                module: sigtran
                destinations:
                        - 192.168.1.99
                        - 192.168.1.95
                        - 192.168.1.90
                mechanism: round-robin
                retry-time: 30
                timeout: 5
                from-domain: test.example.com
                headers:
                        - header: INAP-Service-Key
                          value: 2
```

Copyright © Sipwise GmbH - All rights reserved

```
                reply:
                        tcap: raw-tcap
        parallel:
                module: sigtran
                destinations:
                        - 192.168.1.99
                        - 192.168.1.95
                        - 192.168.1.90
                mechanism: parallel
                retry-time: 30
                timeout: 10
                from-domain: test.example.com
                headers:
                        - header: INAP-Service-Key
                          value: 2
                reply:
                        tcap: raw-tcap
        mock1:
                module: mock-tcap
                numbers:
                        - number: '4311003'
                          routing-number: '4318881003'
                reply:
                        tcap: raw-tcap
        default_sipredir:
                module: sipredir
                destination: 192.168.1.1:8888
                from-domain: test.example.com
                headers: []
                reply:
                        user: username
                        dom: domain
                        code: portability
                reply-headers:
                        - header: X-LNP
                          map-to: portability
```

The corresponding Sipwise C5 main configuration file contains:

Copyright © Sipwise GmbH - All rights reserved

```
    daemon:
      foreground: 'false'
      json-rpc:
        ports:
          - '54321'
          - '12345'
      loglevel: '6'
      sip:
        port: '5095'
      threads: '4'
    instances:
    << These are the same entries as in /etc/ngcp-lnpd/config.yml file
>>
```

Description of configuration parameters in /etc/ngcp-config/config.yml file

- daemon section:

  foreground: determines if the LNP daemon runs as foreground or background process

  json-rpc.ports: port numbers where LNP daemon listens for incoming JSONRPC requests from Sipwise C5 Proxy

  loglevel: how detailed information LNP daemon writes in its log file

  sip.port: listening port number used for SIP sessions with LNP serving nodes; LNP daemon will listen on first available (shared) *sip_int* IP address that is taken from /etc/ngcp-config/network.yml file

  threads: number of threads LNP daemon will use internally; this value determines how many requests the daemon can serve in parallel

- instances section: at least one `default` instance must be defined here. Others are also useful for providing redundancy, please check `redundant` and `parallel` entries above.

  module: either `sigtran` or `sipredir` depending on the type of external server

  IMPORTANT

  The module `mock-tcap` is only meant for developers. In this case the LNP daemon does not produce a SIP request that it sends to LNP serving nodes, but instead it uses the `numbers` parameter to match a called number with a routing number. The `numbers` parameter contains a list of number—routing-number pairs and is used as a database for number lookups. Finally LNP daemon returns the routing number as a response on LNP query.

  destinations: list of nodes to which LNP daemon sends the LNP query

  mechanism: either `parallel` or `round-robin`, defining the method of redundant queries

  retry-time: a period of time in seconds while LNP daemon considers an LNP serving node being unreachable after an LNP query timeout

  timeout: the period of time while LNP daemon waits for a response on an LNP query from one of the LNP serving nodes

Copyright © Sipwise GmbH - All rights reserved

NOTE `retry-time` and `timeout` are used with both the parallel and the round-robin redundancy methods

from-domain: the domain that will be used in SIP *From* header when LNP daemon sends the LNP query

headers: this is a list of `header` name—`value` pairs; these custom headers will be included in SIP request that LNP daemon sends to an LNP serving node. Necessary only when `sigtran` module is used.

reply: determines the json keys and values returned to Sipwise C5 Proxy.

tcap: currently only `raw-tcap` is supported, which means LNP daemon will not decode the TCAP response it gets from an LNP serving node but forwards the raw TCAP message body. Only for `sigtran` module.

user: username extracted from the Contact header. Only for `sipredir` module.

dom: domain extracted from the Contact header. Only for `sipredir` module.

code: LNP portability code extracted from the custom header. Only for `sipredir` module.

`reply-headers`: this is a list of `header` name—`map-to` pairs; these custom headers will be read by LNP daemon from the 302 reply message and internally translated into a variable that can be used in the `reply` parameter to return values Sipwise C5 Proxy. At least the header that provides the portability information has to be defined (X-LNP by default). Necessary only when `sipredir` module is used.

Selection of Sipwise LNP Daemon Instances

By default the instance with name `default` is used for all the lnp queries. To dynamically select which instance use, or to completely skip the lnp query for a particular call, the lnp api module is looking into the SIP message for the header with name *P-NGCP-Lnpd_Instance*:

- if present and not empty, the instance with the name equal to the header content is used
- if present but empty, the lnp api lookup is skipped
- if not present, the `default` instance is used

LNP API number manipulation

The country code assigned to the calling subscriber can be prepended to the redirecting number obtained from the LNP lookup. To enable this feature please set the following parameter:

- kamailio->proxy->lnp->api->add_caller_cc_to_lnp_dst : *no*

The obtained result can be also rewritten using dedicated LNP Rewrite Rules for Callee. The rewrite rule set used is the one assigned to the originating peer or subscriber/domain via the rewrite_rule_set preference. This variable is available in the match and replace part, beside the standard variables for rewrite rules:

- `${callee_lnp_prefix}`: lnp redirecting code

This feature is not enabled by default, but can be activated with the following parameters:

- kamailio->proxy->lnp->api->apply_callee_lnp_rewrite : *no*

 Copyright © Sipwise GmbH - All rights reserved

LNP API returned values

As for Local LNP, the LNP number and the FCI code are stored in CDR data.

Additionally two dedicated headers can be added to the outgoing SIP message:

- P-NGCP-LNP-Number: The returned LNP number, if any
- P-NGCP-LNP-Status: The LNP query return code (200 if successful, 404 if no entry found, 408 in case of connection timeout or 500 if another general error happens)

This feature is not enabled by default, but can be activated with the following parameters:

- kamailio->proxy->lnp->add_reply_headers->enable : *no*
- kamailio->proxy->lnp->add_reply_headers->number : *P-NGCP-LNP-Number*
- kamailio->proxy->lnp->add_reply_headers->status : *P-NGCP-LNP-Status*

LNP API advanced setup

Sipwise C5 *config.yml* provides some advanced configuration to control what happens to the call based on whether the LNP lookup is completed succesfully or not (timeout, malformed or missing data in the message):

- kamailio->proxy->lnp->api->check_if_user_is_local_after_succ_lookup : *no*. By default in case of successful LNP lookup, the system assumes that the callee is reachable via a peer connection. Set this option to `yes` instead to force the system to check if the redirecting number belongs to a local subscriber.
- kamailio->proxy->lnp->api->check_if_user_is_local_after_fail_lookup : *no*. By default in case of failed LNP lookup, the system tries to reach the callee using the originally dialed number via a peer connection. Set this option to `yes` instead to force the system to check if the originally number belongs to a local subscriber.
- kamailio->proxy->lnp->api->reply_error_on_lnp_failure : *no*. If set to `yes` it will return a SIP error message back to the caller in case of failed LNP lookup.

7.7. P-Early-Media

The Sipwise C5 platform supports P-Early-Media Sip Header according to RFC5009.

7.7.1. Implementation

The Sipwise C5 platform tries to find a P-Early-Media SIP header on every INVITE message coming from peers or subscribers. If the header is found, it tries to parse the content, in particular it looks for the following parameters:

- sendonly
- recvonly
- inactive
- sendrecv
- supported

Copyright © Sipwise GmbH - All rights reserved

- gated

Any other parameter found will be silently discarded.

The first four parameters are also known as *directional parameters* and they will influence how the Sipwise C5 platform will behave when a service, potentially producing early media, is involved during the call. The P-Early-media header will be also transparently sent to the b leg of the call and to the callee which must honour the contents of the header before playing early media.

A typical example is when a subscriber with an announce preference activated is called, under normal conditions the early media with the announce is played before connecting to the callee. The early media however won't be played if the caller is not willing to receive the early audio stream (*inactive* and *sendonly* directional parameters set in P-Early-Media).

There are cases when Sipwise C5 produces an early media to alert the user of an error (typical examples are when the callee is offline or pstn termination is not available). In this situations, if the caller is not willing to receive the early audio stream, Sipwise C5 will just return a SIP error without playing the early media announce.

The presence of the *supported* parameter forces the Sipwise C5 platform to send back P-Early-Media in 18X replies even if the callee user agent does not include it. This will help the caller user agent to understand that we are going to send early media and an audio channel must be allocated and opened. The Sipwise C5 platform will add the missing P-Early-Media back from the callee towards the caller, according to the following schema:

- if caller sends P-Early-Media: ***sendonly*** Sipwise C5 will reply with P-Early-Media: ***recvonly***
- if caller sends P-Early-Media: ***recvonly*** Sipwise C5 will reply with P-Early-Media: ***sendonly***
- if caller sends P-Early-Media: ***sendrecv*** Sipwise C5 will reply with P-Early-Media: ***sendrecv***
- if caller sends P-Early-Media: ***inactive*** Sipwise C5 will reply with P-Early-Media: ***inactive***

The *gated* parameter is recognized by Sipwise C5 but it doesn't affect its mode of operation. This parameter, if present, is just passed transparently to the callee.

7.7.2. Configuration

It's possible to alter how Sipwise C5 manages P-Early-Media just changing the following parameters inside the *config.yml* file:

- kamailio.proxy.p_early_media.default_auth: this parameter will instruct how Sipwise C5 behaves when a P-Early-Media without directional parameters is received. The default behaviour will permit early media streams on both directions (sendrecv).
- kamailio.proxy.p_early_media.play_on_missing: this parameter will drive the Sipwise C5 behaviour when the caller sends an INVITE message without P-Early-Media header. If set to yes (default), Sipwise C5 will be allowed to play early media for its own services. If set to no Sipwise C5 will skip early media since it's assuming the caller is not willing to receive early media.

7.8. Emergency Mapping

As opposed to the Simple Emergency Number Handling solution, Sipwise C5 supports an advanced emergency call handling method, called *emergency mapping*. The main idea is: instead of obtaining a

 Copyright © Sipwise GmbH - All rights reserved

statically assigned emergency prefix / suffix from subscriber preferences, Sipwise C5 retrieves an emergency routing prefix from a central emergency call routing table, according to the current location of the calling subscriber.

The following figure shows the overview of emergency call processing when using *emergency mapping* feature:

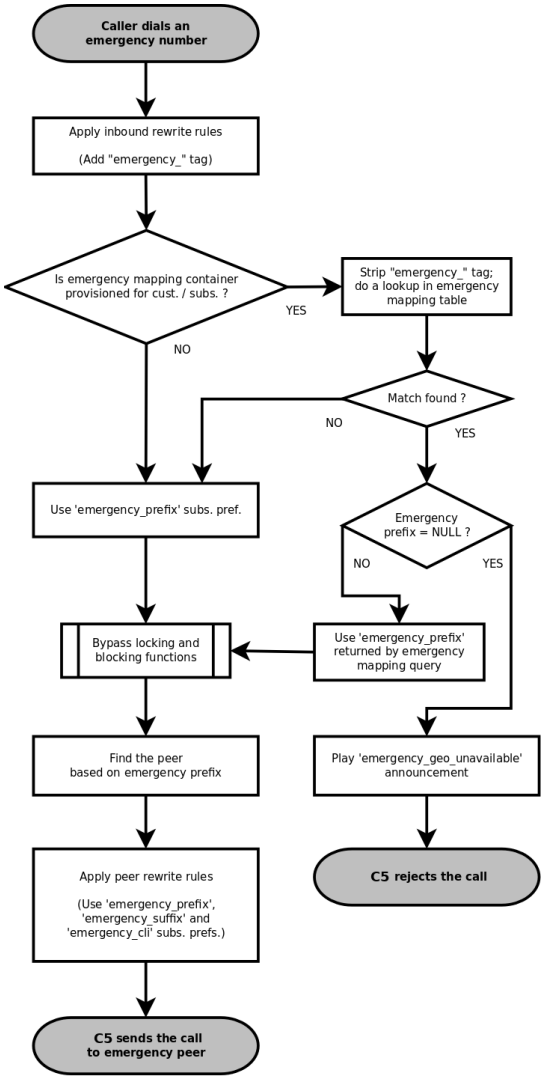


Figure 67. Emergency Call Handling with Mapping

7.8.1. Emergency Mapping Description

Emergency numbers per geographic location are mapped to different routing prefixes not deriveable from an area code or the emergency number itself. This is why a **global emergency mapping table** related to resellers is introduced, allowing to map emergency numbers to their geographically dependent routing numbers.

The geographic location is referenced by a location ID, which has to be populated by a north-bound provisioning system. No towns, areas or similar location data is stored on Sipwise C5 platform. The locations are called *Emergency Containers* on NGCP.

The actual emergency number mapping is done per location (per *Emergency Container*), using the so-called *Emergency Mapping* entries. An *Emergency Mapping* entry assigns a routing prefix, valid only in a

Copyright © Sipwise GmbH - All rights reserved

geographic area, to a generic emergency number (for example '112' in Europe, '911' in the U.S.A.) or a country specific one (for example '133').

NOTE Sipwise C5 performs an exact match on the emergency number in the emergency routing table.

Emergency Containers may be assigned to various levels of the client hierarchy within NGCP. The following list shows such levels with each level overriding the settings of the previous one:

1. Customer or Domain
2. Customer Location, which is a territory representing a subset of the customer's subscribers, defined as one or more IP subnets.
3. Subscriber

NOTE Please be aware that *Customer Location* is not necessarily identical to the "location" identified through an *Emergency Container*.

Once the emergency routing prefix has been retrieved from the emergency mapping table, call processing continues in the same way as in case of simple emergency call handling.

7.8.2. Emergency Mapping Configuration

The administrative web panel of Sipwise C5 provides the configuration interface for emergency mapping. Please navigate to *Settings* *Emergency Mapping* menu item first, in order to start configuring the mapping.

An *Emergency Container* must be created, before the mapping entries can be defined. Press *Create Emergency Container* to start this. An example of a container is shown here:

 Copyright © Sipwise GmbH - All rights reserved

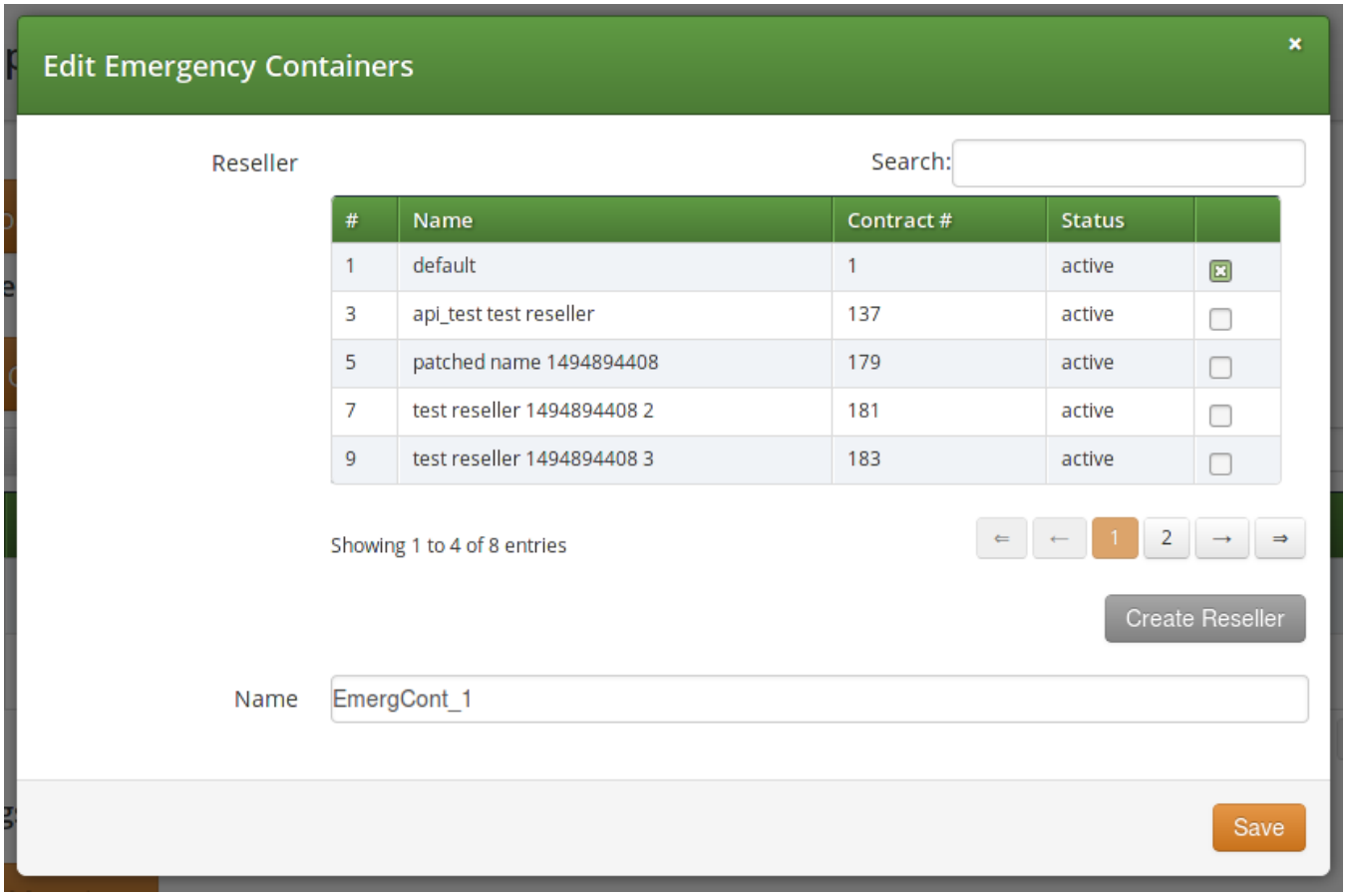


Figure 68. Creating an Emergency Container

You have to select a Reseller that this container belongs to, and enter a Name for the container, which is an arbitrary text.

TIP The platform administrator has to create as many containers as the number of different geographic areas (locations) the subscribers are expected to be in.

As the second step of emergency mapping provisioning, the *Emergency Mapping* entries must be created. Press *Create Emergency Mapping* to start this step. An example is shown here:

Copyright © Sipwise GmbH - All rights reserved

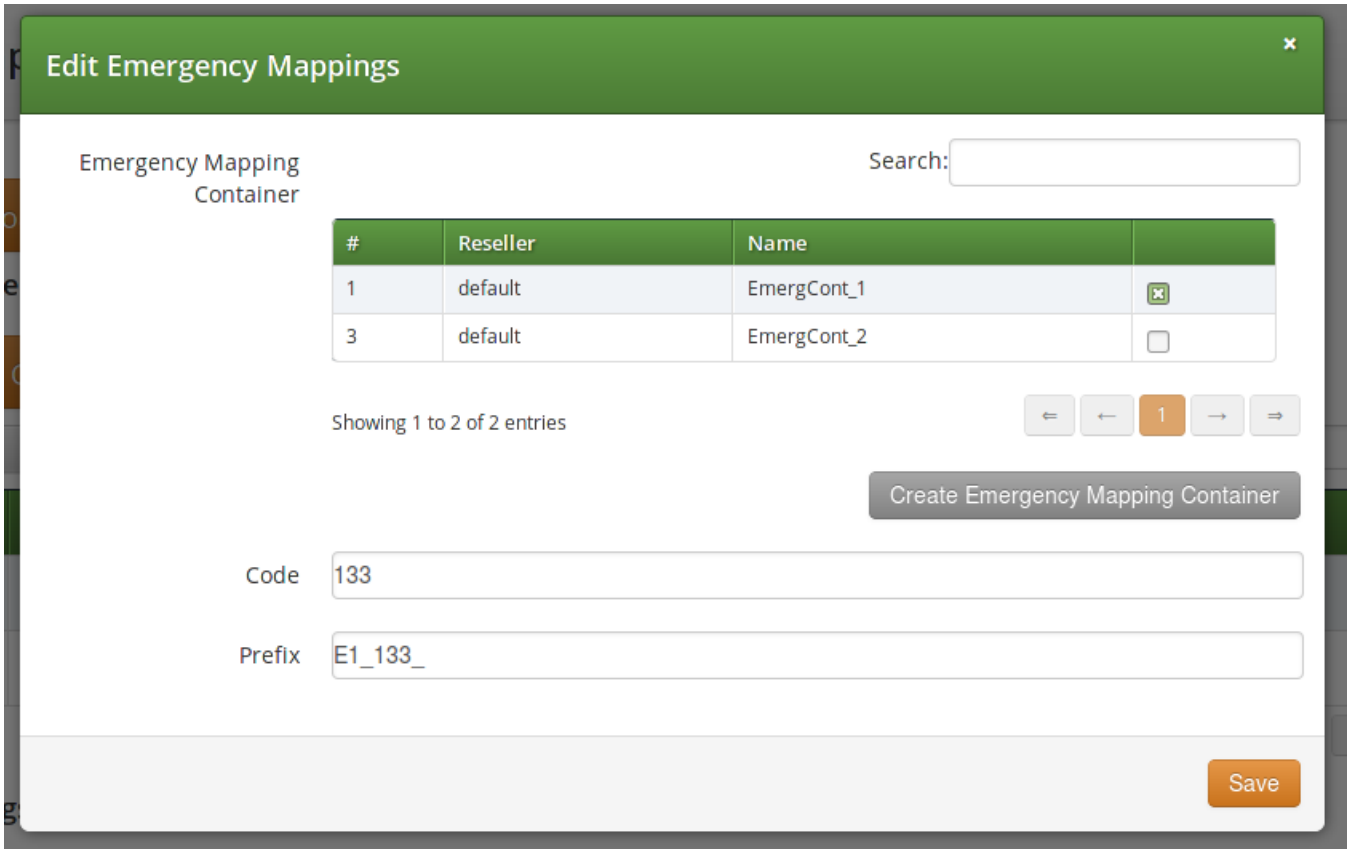


Figure 69. Creating an Emergency Mapping Entry

The following parameters must be set:

- Container: select an emergency mapping container (i.e. a location ID)
- Code: the emergency number that subscribers will dial
- Prefix: the routing prefix that belongs to the particular emergency service within the selected location

Once all the necessary emergency mappings have been defined, the platform administrator will see a list of containers and mapping entries:

 Copyright © Sipwise GmbH - All rights reserved

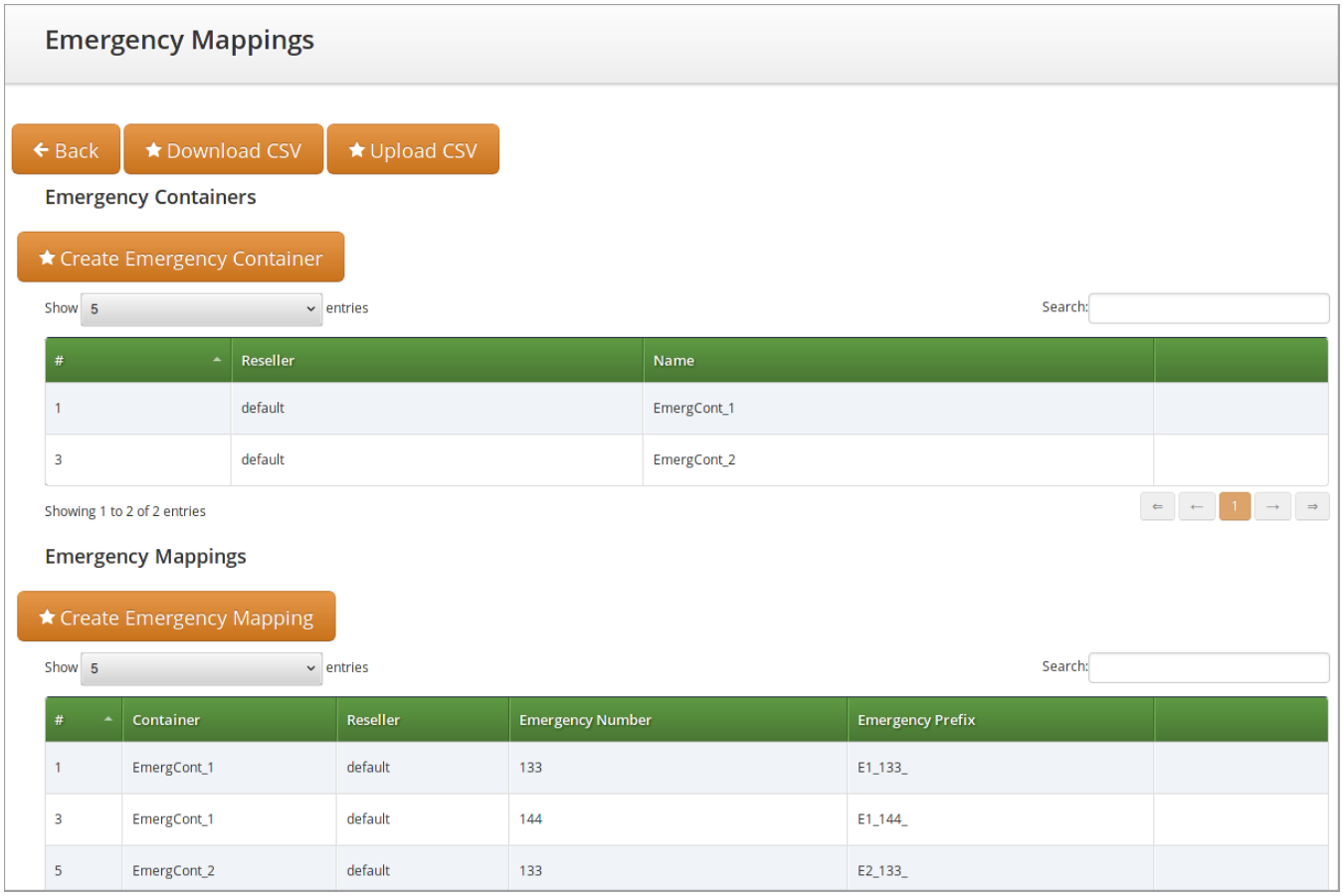


Figure 70. Emergency Mapping List

The emergency number mapping is now defined. As the next step, the platform administrator has to assign the emergency containers to *Customers* / *Domains* / *Customer Locations* or *Subscribers*. We'll take an example with a *Customer*: select the customer, then navigate to *Details* *Preferences* *Number Manipulations*. In order to assign a container, press the *Edit* button and then select one container from the drop-down list:

Copyright © Sipwise GmbH - All rights reserved

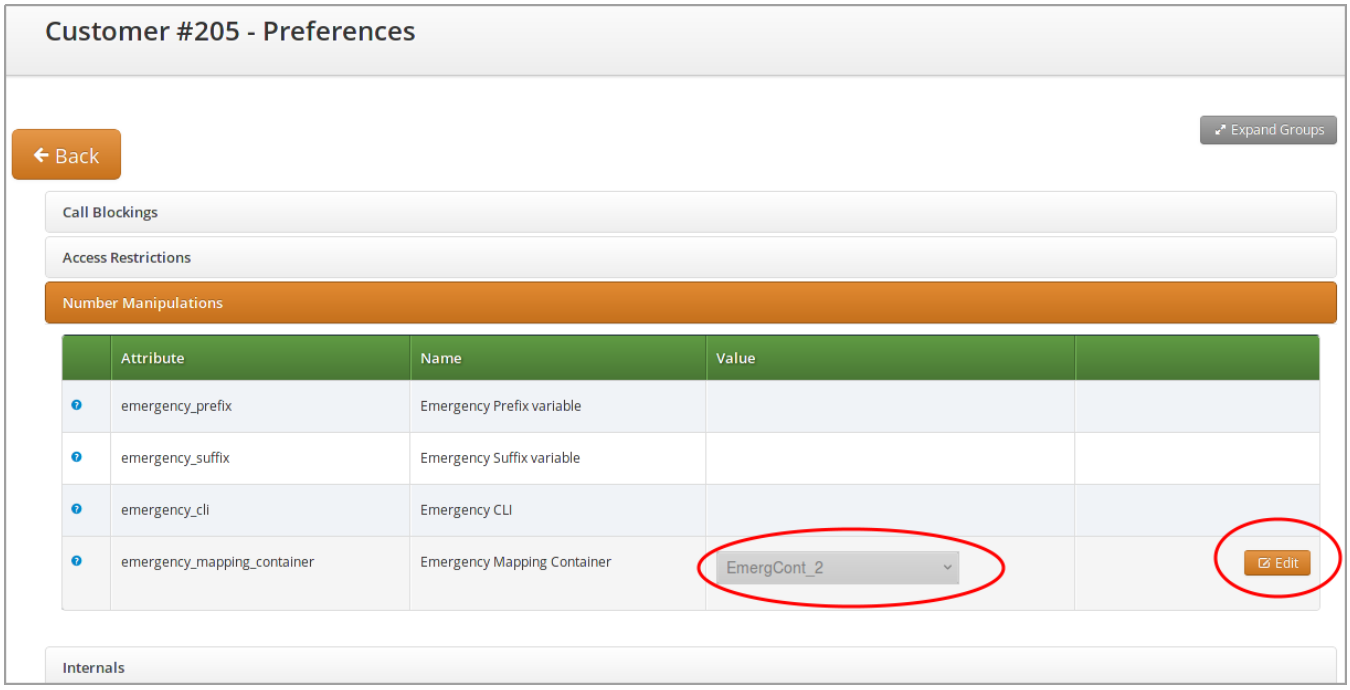


Figure 71. Assigning an Emergency Mapping Container

Rewrite Rules for Emergency Mapping

Once emergency containers and emergency mapping entries are defined, Sipwise C5 administrator has to ensure that the proper number manipulation takes place, before initiating any emergency call towards peers.

IMPORTANT Please don't forget to define the rewrite rules for peers—particularly: *Outbound Rewrite Rules for Callee*—as described in Normalize Emergency Calls for Peers section of the handbook.

Emergency Calls Not Allowed

There is a special case when the dialed number is recognized as an emergency number, but the emergency number is not available for the geographic area the calling party is located in.

In such a case the emergency mapping lookup will return an emergency prefix, but the value of this will be NULL. Therefore the call is rejected and an announcement is played. The announcement is a newly defined sound file referred as emergency_geo_unavailable.

It is possible to configure the rejection code and reason in /etc/ngcp-config/config.yml file, the parameters are: kamailio.proxy.early_rejects.emergency_invalid.announce_code and kamailio.proxy.early_rejects.emergency_invalid.announce_reason.

Bulk Upload or Download of Emergency Mapping Entries

The Sipwise C5 offers the possibility to upload / download emergency mapping entries in form of CSV files. This operation is available for each reseller, and is very useful if a reseller has many mapping entries.

Downloading Emergency Mapping List

 Copyright © Sipwise GmbH - All rights reserved

One has to navigate to *Settings Emergency Mapping* menu and then press the *Download CSV* button to get the list of mapping entries in a CSV file. First the reseller must be selected, then the *Download* button must be pressed. As an example, the entries shown in "Emergency Mapping List" picture above would be written in the file like here below:

```
EmergCont_1,133,E1_133_
EmergCont_1,144,E1_144_
EmergCont_2,133,E2_133_
```

The **CSV file** has a plain text format, each line representing a mapping entry, and contains the following **fields**:

- Container name, as defined in *Emergency Containers*
- Emergency Number
- Emergency Prefix

Uploading Emergency Mapping List

Uploading a CSV file with emergency mapping entries may be started after pressing the *Upload CSV* button. The following data must be provided:

- Reseller: selected from the list
- Upload mapping: the CSV file must be selected after pressing the *Choose File* button
- Purge existing: an option to purge existing emergency mapping entries that belong to the selected reseller, before populating the new mapping data from the file

Copyright © Sipwise GmbH - All rights reserved

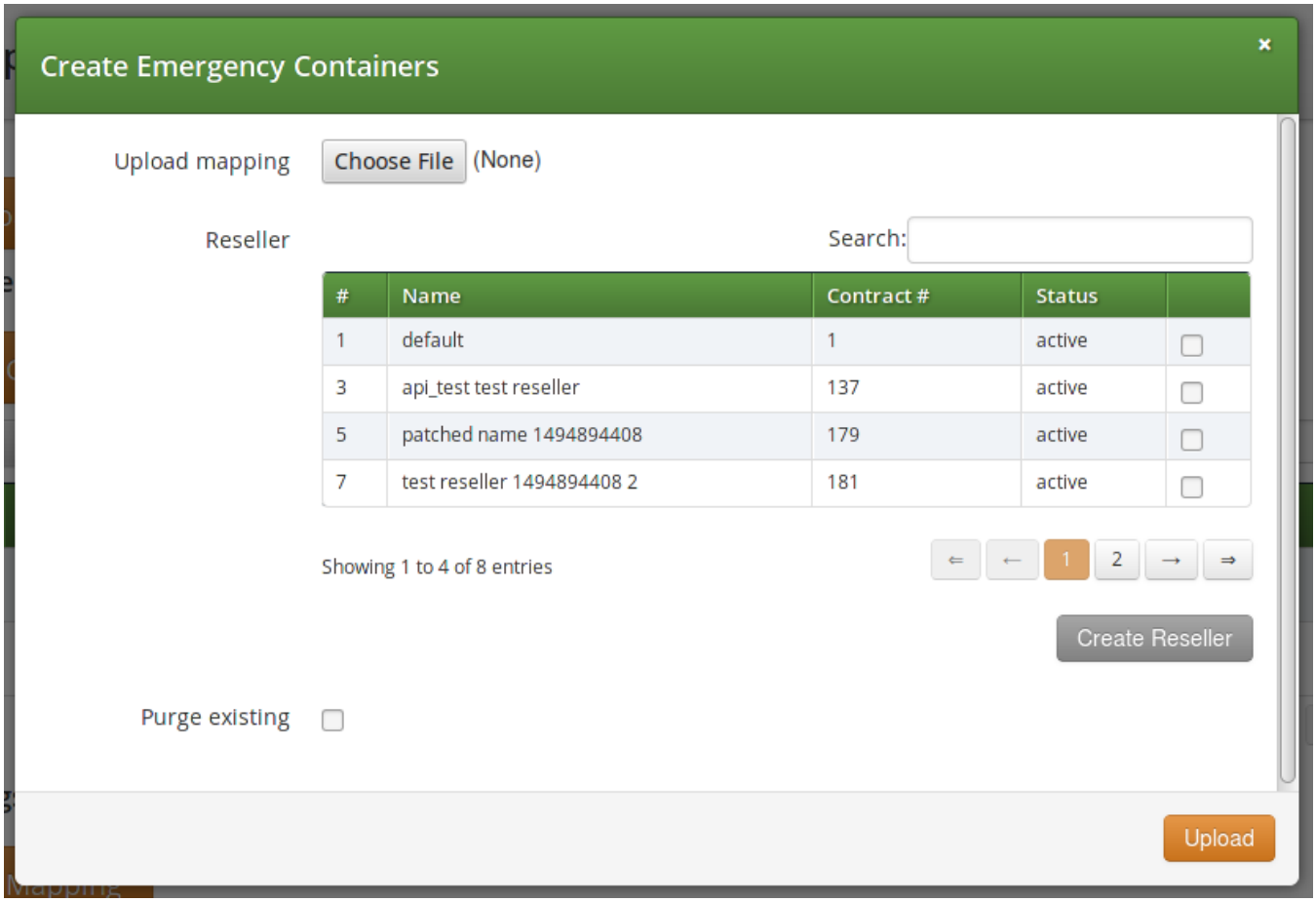


Figure 72. Uploading Emergency Mapping Data

The CSV file for the upload has the same format as the one used for download.

7.9. Emergency Priorization

The Sipwise C5 can potentially host *privileged subscribers* that offer emergency or at least prioritized services (civil defence, police etc.). In case of an emergency, the platform has to be free'd from any SIP flows (calls, registrations, presence events etc.) which do not involve those privileged subscribers.

Such an exceptional condition is called ***emergency mode*** and it can be activated for all domains on the system, or only for selected domains.

Once emergency mode is activated, Sipwise C5 will immediately apply the following restrictions on new SIP requests or existing calls:

- Any SIP requests (calls, registrations etc.) from subscribers within the affected domains, who are not marked as privileged, are rejected.
- Any calls from peers not targeting privileged subscribers are rejected.
- Any active calls which do not have a privileged subscriber involved are terminated.

Calls from non-privileged subscribers to emergency numbers are still allowed.

 Copyright © Sipwise GmbH - All rights reserved

7.9.1. Call-Flow with Emergency Mode Enabled

Typical call-flows of emergency mode will be shown in this section of the handbook. We have the following assumptions:

- Emergency priorization has been enabled on system-level
- There is a domain for which the emergency mode has been activated
- There is a privileged subscriber in that domain
- A generic peering connection has been configured for non-emergency calls
- A dedicated peering connection has been configured for emergency calls

The examples do not show details of SIP messages, but rather give a high-level overview of the call-flows.

1. A **non-privileged** subscriber makes a call **to another non-privileged subscriber**. Result: the call will be **rejected**.

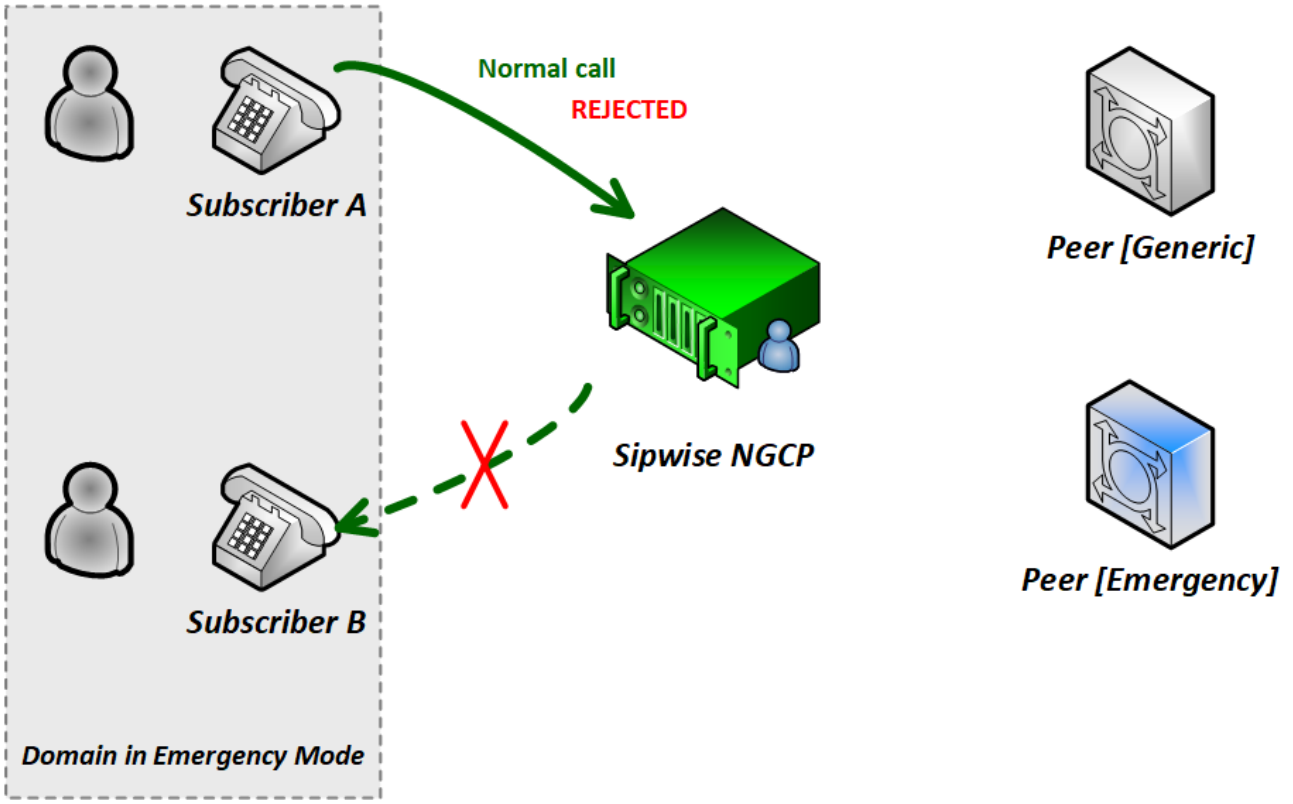


Figure 73. Call-flow in Emergency Mode 1. (Std to Std)

2. A **non-privileged** subscriber makes a call **to an external subscriber (via peer)**. Result: the call will be **rejected**.

Copyright © Sipwise GmbH - All rights reserved

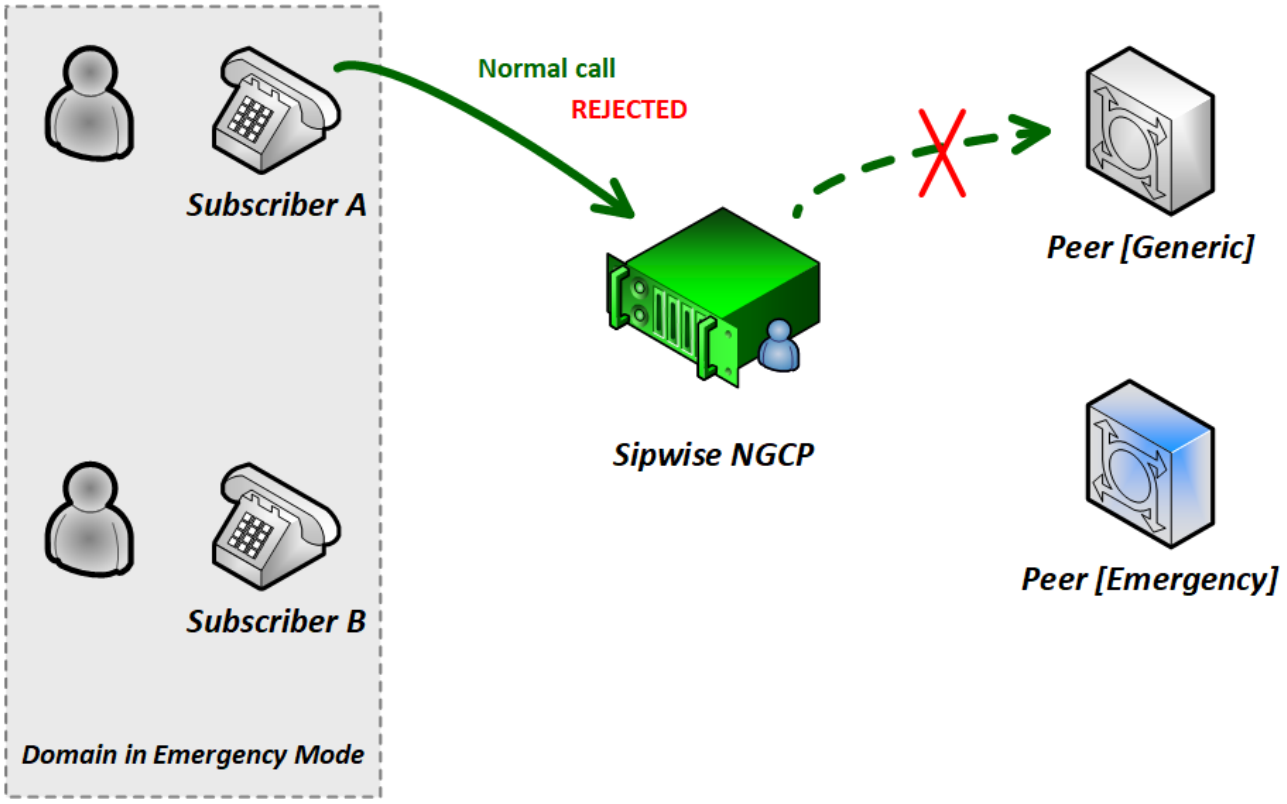


Figure 74. Call-flow in Emergency Mode 2. (Std to Peer)

3. A **non-privileged** subscriber makes a call **to a privileged subscriber**. Result: the call will be **accepted**.

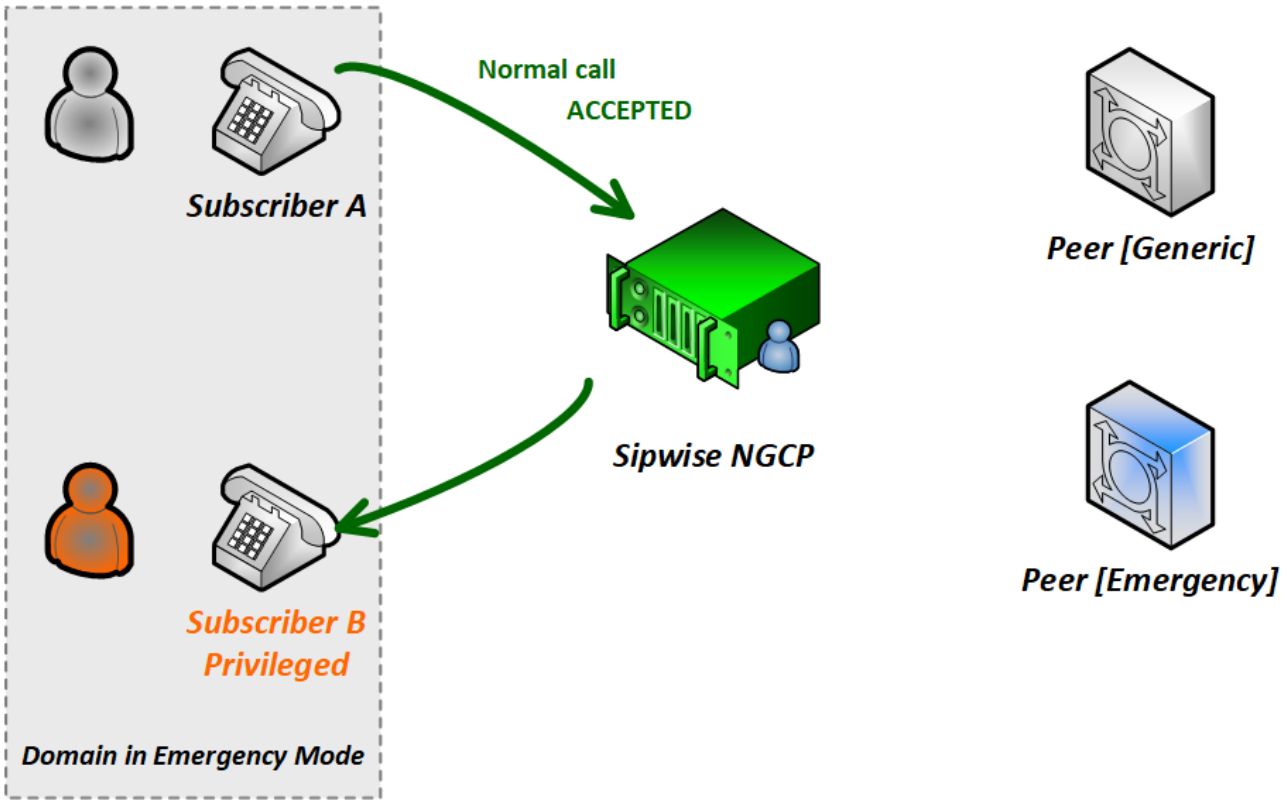


Figure 75. Call-flow in Emergency Mode 3. (Std to Priv)

 Copyright © Sipwise GmbH - All rights reserved

4. A **non-privileged** subscriber makes a call **to an emergency number**. Result: the call will be **accepted**.

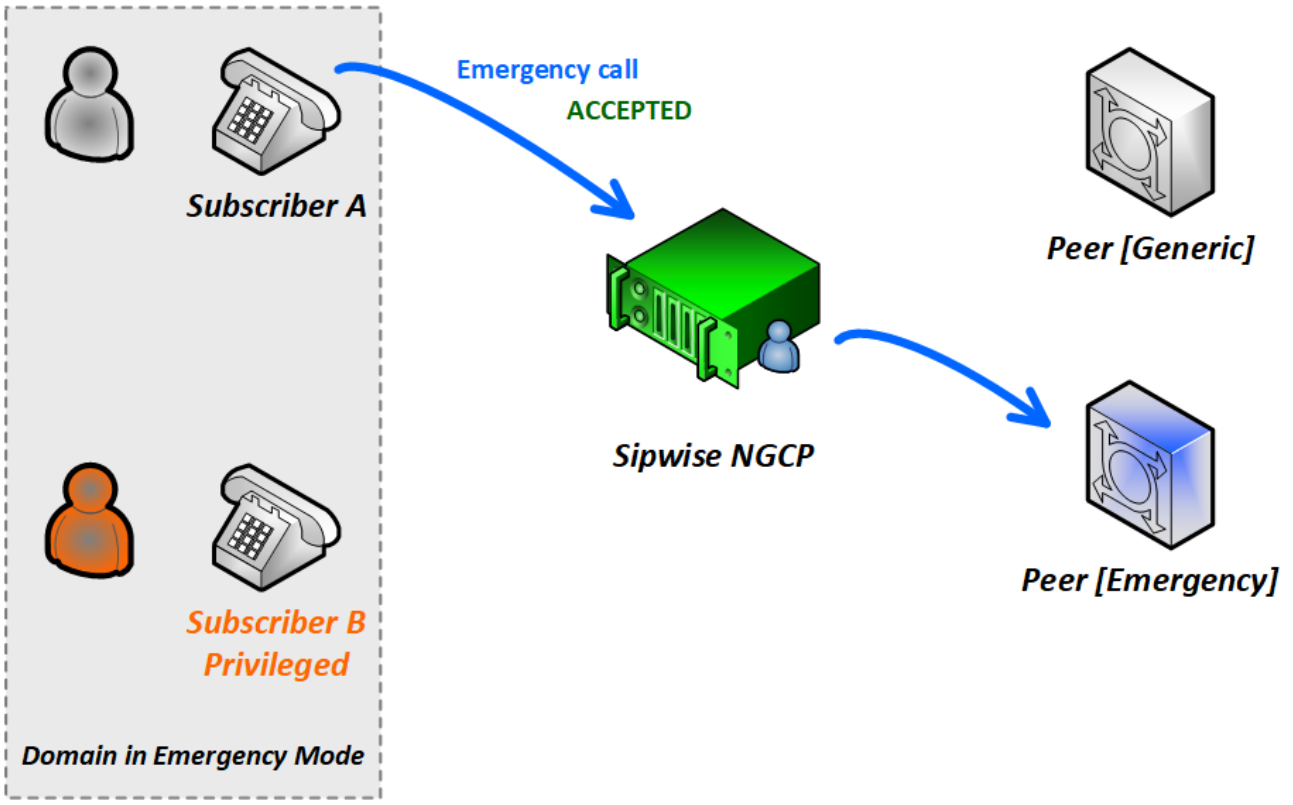


Figure 76. Call-flow in Emergency Mode 4. (Std to Emerg)

5. A **privileged** subscriber makes a call **to a non-privileged subscriber**. Result: the call will be **accepted**.

Copyright © Sipwise GmbH - All rights reserved

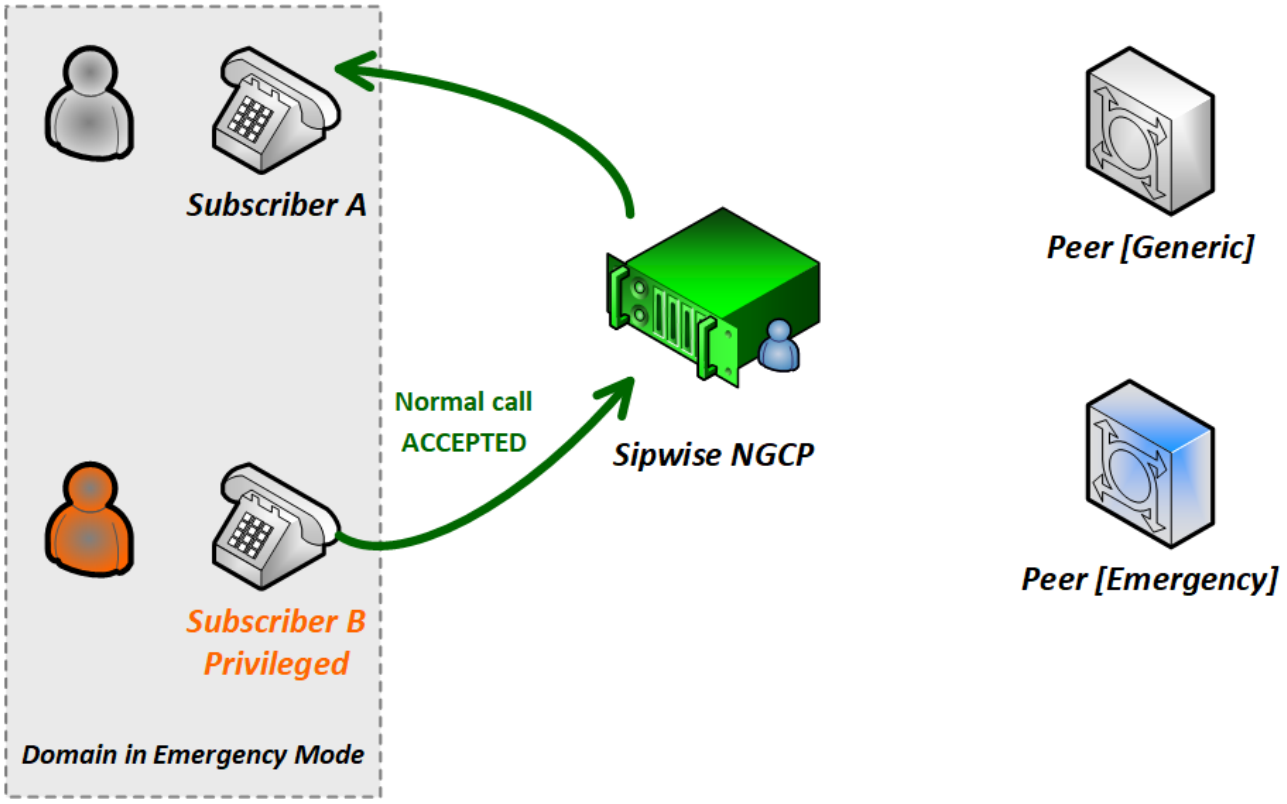


Figure 77. Call-flow in Emergency Mode 5. (Priv to Std)

6. A **privileged** subscriber makes a call **to an external subscriber (via peer)**. Result: the call will be **accepted**.

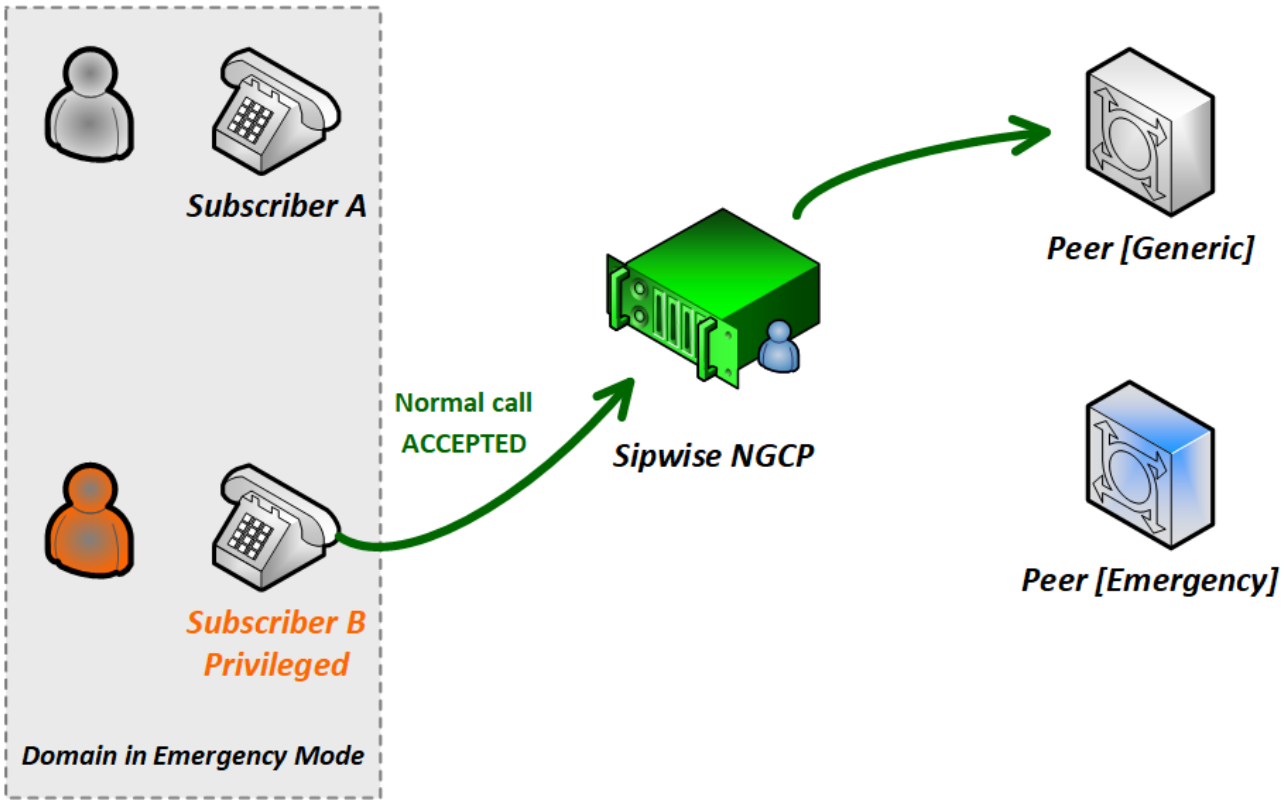


Figure 78. Call-flow in Emergency Mode 6. (Priv To Peer)

 Copyright © Sipwise GmbH - All rights reserved

7.9.2. Configuration of Emergency Mode

The platform operator has to perform 2 steps of configuration so that the emergency mode can be activated. After the configuration is completed it is necessary to explicitly activate emergency mode, which can be accomplished as described in Activating Emergency Mode later.

1. System-level Configuration

The emergency priorization function must be enabled for the whole system, otherwise emergency mode can not be activated. The platform operator has to set kamailio.proxy.emergency_priorization.enabled configuration parameter value to "yes" in the main configuration file /etc/ngcp-config/config.yml. Afterwards changes have to be applied in the usual way, with the command: `ngcpcfg apply "Enabled emergency priorization"`

In order to learn about other parameters related to emergency priorization please refer to kamailio part of the handbook.

2. Subscriber-level Configuration

The platform operator (or any administrator user) has the capability to declare a subscriber privileged, so that the subscriber can initiate and receive calls when emergency mode has been activated on the NGCP. In order to do that the administrator has to navigate to *Settings* *Subscribers* *select the subscriber* *Details* *Preferences* *Internals* *emergency_priorization* on the **administrative web interface**, and press the *Edit* button.

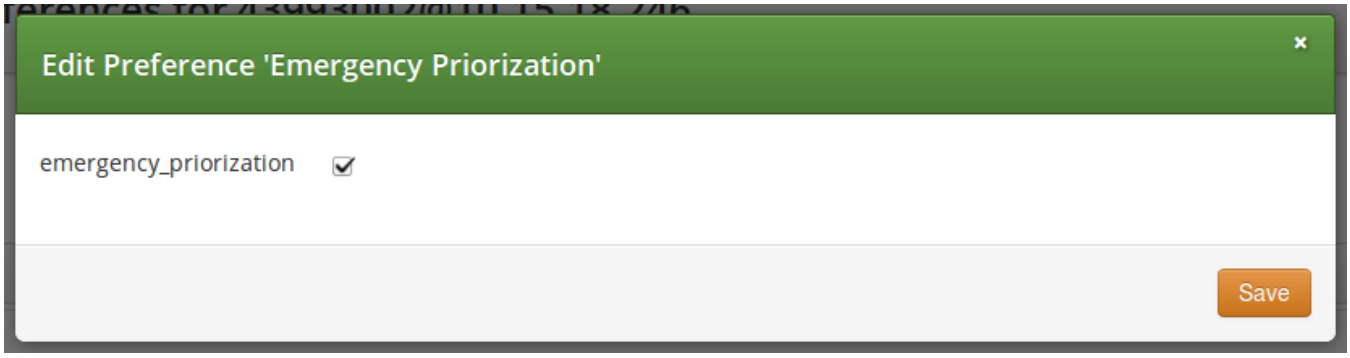


Figure 79. Emergency Priorization of Subscriber

The checkbox *emergency_priorization* has to be ticked and then press the *Save* button.

The same privilege can be added via the **REST API** for a subscriber: a HTTP PUT/PATCH request must be sent on `/api/subscriberpreferences/id` resource and the `emergency_priorization` property must be set to "true".

7.9.3. Activating Emergency Mode

The platform operator can activate emergency mode for a single or multiple domains in 3 different ways:

- via the administrative web interface
- via the REST API
- via a command-line tool

Copyright © Sipwise GmbH - All rights reserved

IMPORTANT	The interruption of ongoing calls is only possible with the command-line tool! Activating emergency mode for domains via the web interface or REST API will only affect upcoming calls.

1. Activate emergency mode via web interface: this way of activation is more appropriate if only a single (or just a few) domain is affected. Please navigate to *Settings* *Domains* *select a domain* *Preferences* *Internals* *emergency_mode_enabled* *Edit*.

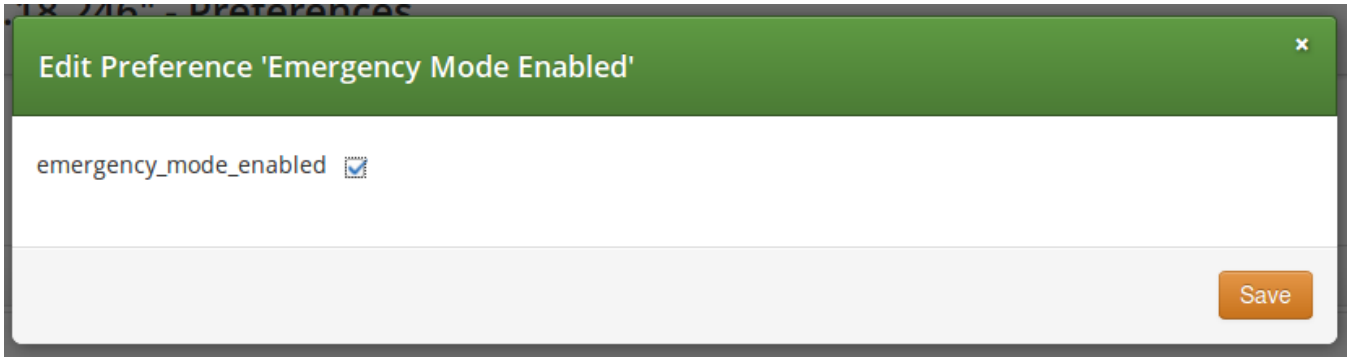


Figure 80. Activate Emergency Mode of Domain

The checkbox *emergency_mode_enabled* has to be ticked and then press the *Save* button.

2. Activate emergency mode via REST API: this way of activation is more appropriate if only a single (or just a few) domain is affected.

For that purpose a HTTP PUT/PATCH request must be sent on `/api/domainpreferences/id` resource and the `emergency_mode_enabled` property must be set to "true".

3. Activate emergency mode using a command-line tool: Sipwise C5 provides a built-in script that may be used to enable/disable emergency mode for some particular or all domains.

- Enable emergency mode:

```
> ngcp-emergency-mode enable <all|[domain1 domain2 ...]>
```

- Disable emergency mode:

```
> ngcp-emergency-mode disable <all|[domain1 domain2 ...]>
```

- Query the status of emergency mode:

```
> ngcp-emergency-mode status <all|[domain1 domain2 ...]>
```

7.10. SIP Message Filtering

Copyright © Sipwise GmbH - All rights reserved

7.10.1. Header Filtering

Adding additional SIP headers to the initial INVITEs relayed to the callee (second leg) is possible by creating a patchtt file for the following template:

/etc/ngcp-config/templates/etc/sems-b2b/etc/ngcp.sbcprofile.conf.tt2.

The following section can be changed:

```
header_filter=whitelist
header_list=[%IF kamailio.proxy.debug == "yes"%]P-NGCP-CFGTEST,[%END%]
P-R-Uri,P-D-Uri,P-Preferred-Identity,P-Asserted-
Identity,Diversion,Privacy,
Allow,Supported,Require,RAck,RSeq,Rseq,User-Agent,History-Info,Call-Info
[%IF kamailio.proxy.presence.enable == "yes"%],Event,Expires,
Subscription-State,Accept[%END%][%IF kamailio.proxy.allow_refer_method
== "yes"%],Referred-By,Refer-To,Replaces[%END%]
```

By default the system will remove from the second leg all the SIP headers which are not in the above list. If you want to keep some additional/custom SIP headers, coming from the first leg, into the second leg you need to add them at the end of the *header_list=* list. After that, as usual, you need to apply and push the changes. In this way the system will keep your headers in the INVITE sent to the destination subscriber/peer.

WARNING DO NOT TOUCH the list if you don't know what you are doing.

7.10.2. Codec Filtering

Sometimes you may need to filter some audio CODEC from the SDP payload, for example if you want to force your subscribers to do not talk a certain codecs or force them to talk a particular one. To achieve that you need to change the /etc/ngcp-config/config.yml, in the following section:

```
sdp_filter:
      codecs: PCMA,PCMU,telephone-event
      enable: yes
      mode: whitelist
```

In the example above, the system is removing all the audio CODECS from the initial INVITE except G711 alaw,ulaw and telephone-event. In this way the callee will be notified that the caller is able to talk only PCMA. Another example is the blacklist mode:

```
sdp_filter:
      codecs: G729,G722
      enable: yes
      mode: blacklist
```

In this way the G729 and G722 will be removed from the SDP payload. In order to apply the changes, run

Copyright © Sipwise GmbH - All rights reserved

```
ngcpcfg apply 'Enable CODEC filtering'
ngcpcfg push all
```

WARNING Codec filtering feature applies to SDP attributes and it does not remove a whole SDP media session in case all the related attributes match a filter rule. For example, this SDP video media session:

```
m=video 9078 RTP/AVPF 96 97
a=rtpmap:96 VP8/90000
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=42801F
```

in combination with:

```
sdp_filter:
      codecs: PCMA,PCMU,telephone-event
      enable: yes
      mode: whitelist
```

will result in no filtering at all because both codecs id 96 97 are not whitelisted and an empty media session is not allowed.

7.10.3. Codec Filtering with user preferences

Codec filtering management is also possible using the following preferences:

- codecs_list
- codecs_filter
- codecs_id_filter
- codecs_id_list

These preferences offer an alternative way to filter codecs without updating the configuration file and restarting the `kamailio-proxy` service. Since they can be applied at domain/peer/subscriber preferences level, they feature a more granular management compared when configuration is set via config.yml file, which is system wide.

The "codecs_list" preference allows you to enter a list of codecs names, valid names are G722, PCMU, PCMA, speex, GSM, G723, DVI4, L16, QCELP, CN, MPA, G728, DVI4, G729, AMR, AMR*WB, opus, telephone*event, CelB, JPEG, H261, H263, H263*1998, MPV, MP2T, nv, vp8, vp9, h264.

The "codecs_id_list" preference allows you to achieve the same result but in this case you will have to enter the corresponding CODEC-ID(s) payload type.

Once the list is filled, you have to specify if the codecs are going to be blacklisted or whitelisted by `kamailio-proxy`.This is done through the `codecs_filter` or the `codecs_id_filter` preferences.

 Copyright © Sipwise GmbH - All rights reserved

If "codecs_list" and "codecs_id_list" are set (and the corresponding codecs_list/codec_id_list), both filters will be applied sequentially: the list of codecs resulting from "codecs_list" filtering will be used as input for the second "codecs_id_list" filtering.

By default, the codecs filter preferences are cleared (blacklisted) causing all codecs listed inside the codecs lists preferences to be filtered out. Differently, if the codecs filters are set, only the ones inside the codecs list will be retained causing the others to be filtered out.

WARNING The `codecs_filter` preference is tied to `codecs_list` and only operates on codecs entered inside that list. Likewise for `codecs_id_filter` and `codecs_id_list`, respectively. Ensure not mixing them.

Examples:

- Example 1

  codecs_list: "G729"

  codecs_filter: unset (false)

  SDP IN:

  ```
  m=audio 6000 RTP/AVP 8 0 18 101
  a=rtpmap:8 PCMA/8000
  a=rtpmap:0 PCMU/8000
  a=rtpmap:18 G729/8000
  a=rtpmap:101 telephone-event/8000
  ```

  SDP OUT:

  ```
  m=audio 30000 RTP/AVP 8 0 101
  a=rtpmap:8 PCMA/8000
  a=rtpmap:0 PCMU/8000
  a=rtpmap:101 telephone-event/8000
  ```

  Codec G729 has been filtered out because blacklisted in codecs_list.

- Example 2

  codecs_id_list: "18"

  codecs_id_filter: set (true)

  SDP IN:

  ```
  m=audio 6000 RTP/AVP 8 0 18 101
  a=rtpmap:8 PCMA/8000
  a=rtpmap:0 PCMU/8000
  a=rtpmap:18 G729/8000
  a=rtpmap:101 telephone-event/8000
  ```

  SDP OUT:

Copyright © Sipwise GmbH - All rights reserved

```
m=audio 30108 RTP/AVP 18
a=rtpmap:18 G729/8000
```

Codec G729 is retained because whitelisted, PCMA PCMU have been filtered out because they are not whitelisted.

7.10.4. Enable History and Diversion Headers

It may be useful and mandatory - specially with NGN interconnection - to enable SIP History header and/or Diversion header for outbound requests to a peer or even for on-net calls. In order to do so, you should enable the following preferences in Domain's and Peer's Preferences:

- Domain's Preferences: *inbound_uprn* = **Forwarder's NPN**
- Peer's Preferences: *outbound_history_info* = **UPRN**
- Peer's Preferences: *outbound_diversion* = **UPRN**
- Domain's Preferences: *outbound_history_info* = **UPRN** (if you want to allow History Header for on-net call as well)
- Domain's Preferences: *outbound_diversion* = **UPRN** (if you want to allow Diversion Header for on-net call as well)

7.10.5. User Agent Filtering

It could be useful to filter the received REGISTER and INVITE requests based on the User-Agent header's value. For example, if you want to force your subscribers to use certain type/model of devices. To do that system wide you need to change the /etc/ngcp-config/config.yml, in the following section:

```
kamailio:
  lb:
    block_useragents:
      action: reject
      block_empty: no
      block_absent: no
      enable: yes
      mode: whitelist
      ua_patterns:
      - Yealink.*
```

In the example above, the system allows all the requests, which have the User-Agent header beginning with 'Yealink' value. All other UACs will be rejected with the *403 Forbidden* message. To silently drop the received message, it is possible to specify the *drop* action instead of the default *reject*.

Another example is the blacklist mode:

 Copyright © Sipwise GmbH - All rights reserved

```
kamailio:
  lb:
    block_useragents:
      action: drop
      block_empty: no
      block_absent: no
      enable: yes
      mode: blacklist
      ua_patterns:
      - friendly-scanner
```

In the example above Sipwise C5 drops all requests, which have the User-Agent header equal to 'friendly-scanner'. Because of the *drop* action these messages will be dropped silently, without providing a response to the sender.

Another example with *block_empty:* and *block_absent:* options set to 'yes':

```
kamailio:
  lb:
    block_useragents:
      action: reject
      block_empty: yes
      block_absent: yes
      enable: yes
      mode: whitelist
      ua_patterns: []
```

In the example above Sipwise C5 rejects all requests, which have the User-Agent header absent or the User-Agent header with no value. Such requests will be rejected with a *403 Forbidden* message. It's possible to set only one of the options (*block_empty: / block_absent:*) to 'yes'. As well as it's possible to keep both of them enabled.

As usually, in order to apply the changes, run:

```
ngcpcfg apply 'Enable User-Agent filtering'
ngcpcfg push all
```

IMPORTANT Please remember that the applying of the changes require a restart of the LB component, which is recommended to be done only during non-working hours.

Regardless of the system-wide configuration (UA filtering enabled globally or not), it is possible to define a specific User-Agent filtering for each Domain or Subscriber. In order to do so, you should configure the following fields in Domain's or Subscriber's Preferences section:

- ua_filter_list: Contains wildcard list of allowed or denied SIP User-Agents matched against the User-Agent header.
- ua_filter_mode: Specifies the operational mode of the SIP User-Agent Filter List: Blacklist or

Copyright © Sipwise GmbH - All rights reserved

Whitelist.

- ua_reject_missing: Rejects any request if no User-Agent header is given.

In case of rejection the response with code kamailio.proxy.early_rejects.block_admin.announce_code and reason kamailio.proxy.early_rejects.block_admin.announce_reason will be sent back to the subscriber.

7.11. SIP Trunking with SIPconnect

7.11.1. User provisioning

For the purpose of external SIP-PBX interconnect with Sipwise C5 the platform admin should create a subscriber with multiple aliases representing the numbers and number ranges served by the SIP-PBX.

- Subscriber username - any SIP username that forms an "email-style" SIP URI.
- Subscriber Aliases - numbers in the global E.164 format without leading plus.

To configure the Subscriber, go to *SettingsSubscribers* and click *Details* on the row of your subscriber. There, click on the *Preferences* button on top.

You should look into the *Number Manipulations* and *Access Restrictions* sections in particular, which control the calling and called number presentation.

7.11.2. Inbound calls routing

Enable preference *Number Manipulationse164_to_ruri* for routing inbound calls to SIP-PBX. This ensures that the Request-URI will comprise a SIP-URI containing the dialed alias-number as user-part, instead of the user-part of the registered AOR (which is normally a static value).

Fallback behaviours

Sipwise C5 can enhance pbx integration providing three different extended dialing modes:

- Strict number matching: dialing arbitrary extension behind subscriber number is not allowed
- Extended matching, send dialed number with extension: the system will locate the subscriber by longest matching prefix, only the whole dialled number (base number + extension) will be written inside the Request uri user part.
- Extended matching, send base matching number: the system will locate the subscriber by longest matching prefix, only the base number or subscriber username will be written inside the Request uri user part.
- Extended matching, send primary number with extension: the system will locate the subscriber by longest matching prefix, the new Request uri user part will be equal to the primary number with appended the extension (calculated using the exceeding part of the callee).

Extended dialing modes are editable under *Number Manipulationsextended_dialing_mode*.

Fallback procedure is on by default and usually applies to extended matching or e164_to_ruri modes. If the callee UAC replies with a 404 not found error, the fallback feature triggers the generation of a new INVITE. In this case the new Request URI will be set accordingly to the *extended_dialing_mode* and *e164_to_ruri* values.

 Copyright © Sipwise GmbH - All rights reserved

The whole 404 fallback procedure can however be disabled activating the *no_404_fallback* option under subscriber's or domain's preferences.

7.11.3. Number manipulations

The following sections describe the recommended configuration for correct call routing and CLI presentation according to the SIPconnect 1.1 recommendation.

Rewrite rules

The SIP PBX by default inherits the domain dialplan which usually has rewrite rules applied to normal Class 5 subscribers with inbound rewrite rules normalizing the dialed number to the E.164 standard. If most users of this domain are Class 5 subscribers the dialplan may supply calling number in national format - see Configuring Rewrite Rule Sets. While the SIP-PBX trunk configuration can be sometimes amended it is a good idea in sense of SIPconnect recommendation to send only the global E.164 numbers.

Moreover, in mixed environments with Sipwise C5 Cloud PBX sharing the same domain with SIP trunking (SIP-PBX) customers the subscribers may have different rewrite rules sets assigned to them. The difference is caused by the fact that the dialplan for Cloud PBX is fundamentally different from the dialplan for SIP trunks due to extension dialing, where the Cloud PBX subscribers use the break-out code (see Preparing PBX Rewrite Rules) to dial numbers outside of this PBX.

The SIPconnect compliant numbering plan can be accommodated by assigning Rewrite Rules Set to the SIP-PBX subscriber. Below is a sample Rewrite Rule Set for using the global E.164 numbers with plus required for the calling and called number format compliant to the recommendation.

Inbound Rewrite Rule for Caller

- Match Pattern: `^(00|\+)([1-9][0-9]+)$`
- Replacement Pattern: `\2`
- Description: `International to E.164`
- Direction: `Inbound`
- Field: `Caller`

Inbound Rewrite Rule for Callee

- Match Pattern: `^(00|\+)([1-9][0-9]+)$`
- Replacement Pattern: `\2`
- Description: `International to E.164`
- Direction: `Inbound`
- Field: `Callee`

Outbound Rewrite Rule for Caller

- Match Pattern: `^([1-9][0-9]+)$`
- Replacement Pattern: `+\1`

Copyright © Sipwise GmbH - All rights reserved

- Description: `For the calls to SIP-PBX add plus to E.164`
- Direction: `Outbound`
- Field: `Caller`

Outbound Rewrite Rule for Callee

- Match Pattern: `^([1-9][0-9]+)$`
- Replacement Pattern: `+\1`
- Description: `For the calls to SIP-PBX add plus to E.164`
- Direction: `Outbound`
- Field: `Callee`

Assign the aforementioned Rewrite Rule Set to the SIP-PBX subscribers.

WARNING Outbound Rewrite Rules for Callee shall NOT be applied to the calls to normal SIP UAs like IP phones since the number with plus does not correspond to their SIP username.

User parameter

The following configuration is needed for your platform to populate the From and To headers and Request-URI of the INVITE request with "user=phone" parameter as per RFC 3261 Section 19.1.1 (if the user part of the URI contains telephone number formatted as a telephone-subscriber).

- Domain's Preferences: *outbound_from_user_is_phone* = Y
- Domain's Preferences: *outbound_to_user_is_phone* = Y

Forwarding number

The following is our common configuration that covers the calling number presentation in a variety of use-cases, including the incoming calls, on-net calls and Call Forward by the platform:

- Domain's Preferences: *inbound_uprn* = **Forwarder's NPN**
- Domain's Preferences: *outbound_from_user* = **UPRN (if set) or User-Provided Number**
- Domain's Preferences: *outbound_pai_user* = **UPRN (if set) or Network-Provided Number**
- Domain's Preferences: *outbound_history_info* = **UPRN** (if the called user expects History-Info header)
- Domain's Preferences: *outbound_diversion* = **UPRN** (if the called user expects Diversion header)
- Domain's Preferences: *outbound_to_user* = **Original (Forwarding) called user** if the callee expects the number of the subscriber forwarding the call, otherwise leave default.

The above parameters can be tuned to operator specifics as required. You can override these settings in the Subscriber Preferences if particular subscribers need special settings.

TIP On outgoing call from SIP-PBX subscriber the Network-Provided Number (NPN) is set to the *cli* preference prefilled with main E.164 number. In order to have the full alias number as NPN on outgoing call set preference *extension_in_npn* = Y.

Copyright © Sipwise GmbH - All rights reserved

Externally forwarded call

If the call forward takes place inside the SIP-PBX it can use one of the following specification for signaling the diversion number to the platform:

- using **Diversion** method (RFC 5806): configure Subscriber's Preferences: *inbound_uprn* = **Forwarder's NPN / Received Diversion**
- using **History-Info** method (RFC 7044): Sipwise C5 platform extends the History-Info header received from the PBX by adding another level of indexing according to the specification RFC 7044.

Allowed CLIs

- For correct calling number presentation on outgoing calls, you should include the pattern matching all the alias numbers of SIP-PBX or each individual alias number under the *allowed_clis* preference.
- If the signalling calling number (usually taken from From user-part, see *inbound_upn* preferences) does not match the *allowed_clis* pattern, the *user_cli* or *cli* preference (Network-Provided Number) will be used for calling number presentation.

7.11.4. Registration

SIP-PBX can use either Static or Registration Mode. While SIPconnect 1.1 continues to require TLS support at MUST strength, one should note that using TLS for signaling does not require the use of the SIPS URI scheme. SIPS URI scheme is obsolete for this purpose.

Static Mode

While SIPconnect 1.1 allows the use of Static mode, this poses additional maintenance overhead on the operator. The administrator should create a static registration for the SIP-PBX: go to Subscribers, *Details Registered DevicesCreate Permanent Registration* and put address of the SIP-PBX in the following format: sip:username@ipaddress:5060 where username=username portion of SIP URI and ipaddress = IP address of the device.

Registration Mode

It is recommended to use the Registration mode with SIP credentials defined for the SIP-PBX subscriber.

IMPORTANT	The use of RFC 6140 style "bulk number registration" is discouraged. The SIP-PBX should register one AOR with email-style SIP URI. The Sipwise C5 will take care of routing the aliases to the AOR with *e164_to_ruri* preference.

Trusted Sources

If a SIP-PBX cannot perform the digest authentication, you can authenticate it by its source IP address in Sipwise C5. To configure the IP-based authentication, go to the subscriber's preferences (*Details PreferencesTrusted Sources*) and specify the IP address of the SIP-PBX in the *Source IP* field.

To authenticate multiple subscribers from the same IP address, use the*From* field to distinguish these subscribers.

When this feature is configured for a subscriber, Sipwise C5 authenticates all calls that arrive from the

Copyright © Sipwise GmbH - All rights reserved

specified IP address without challenging them.

IMPORTANT	If the same IP address and the FROM field are mistakenly specified as trusted for different subscribers, Sipwise C5 will not know which subscriber to charge for the call and will randomly select one.

7.12. Trusted Subscribers

In some cases, when you have a device that cannot authenticate itself against Sipwise C5, you may need to create a *Trusted Subscriber*. Trusted Subscribers use IP-based authentication and they have a Permanent SIP Registration URI in order to receive messages from Sipwise C5.

In order to make a regular subscriber trusted, perform the following extra steps:

- Create a permanent registration via (*SubscribersDetails Registered DevicesCreate Permanent Registration*)
- Add the IP address of the device as Trusted Source in your subscriber's preferences (*Details PreferencesTrusted Sources*).

This way, all SIP messages coming from the device IP will be considered trusted (and get authenticated only by the source IP). All the SIP messages forwarded to the devices will be sent to the SIP URI specified in the subscriber's permanent registration.

7.13. Peer Probing

The basic way of selecting the appropriate peering server, where an outbound call can be routed to, has already been described in Routing Order Selection of the handbook.

7.13.1. Introduction to Peer Probing Feature

The Sipwise C5 provides a web admin panel and API capabilities to configure peering servers in order to terminate calls to non-local subscribers. Those peering servers may become *temporarily unavailable* due to overloading or networking issues. The Sipwise C5 will fail over to another peering server (matching the corresponding peering rules) after a timeout configured at system level (see the `b2b.sbc.outbound_timeout` configuration parameter; 6 sec by default), if no provisional response (a response with a code in the range of 100 to 199) is received for the outbound INVITE request.

Even if this timer is set much lower, like 3 sec, the call setup time is increased significantly. This is even more true if multiple peering servers fail at the same time, which will sum up the individual timeouts, finally *causing call setup times reach the order of tens of seconds*.

To optimize the call setup time in such scenarios, a new feature is implemented to *continuously probe peering servers* via SIP messages, and mark them as unavailable on timeout or when receiving unexpected response codes. Appropriate SIP response codes from the peering servers will mark them as available again.

Peering servers *marked as unavailable* are then *skipped during call routing* in the peering selection process, which significantly shortens the call setup times if peering servers fail.

 Copyright © Sipwise GmbH - All rights reserved

7.13.2. Configuration of Peer Probing

The system administrator has to configure the peer probing feature in 2 steps:

1. System-level configuration enables the peer probing feature in general on the Sipwise C5 and determines the operational parameters, such as timeouts, the SIP method used for probing requests, etc.
2. Peering server configuration will add / remove a peering server to the list of probed endpoints.

System-level Configuration

The parameters of peer probing are found in the main system configuration file /etc/ngcp-config/config.yml. You can see the complete list of configuration parameters in kamailio of the handbook, while the most significant ones are discussed here.

Enabling peer probing system-wide happens through the `kamailio.proxy.peer_probe.enable` parameter. If it is set to 'yes' (which is the default value) then Sipwise C5 will consider probing of individual peering servers based on their settings.

Timeout of a single probing request can be defined through `kamailio.proxy.peer_probe.timeout` parameter. This is a value interpreted as seconds while Sipwise C5 will wait for a SIP response from the peering server. Default is 5 seconds.

The **probing interval** can be set through the `kamailio.proxy.peer_probe.interval` parameter. This is the time period in seconds that determines how often a probing request is sent to the peering servers. Default is 10 seconds.

The **SIP method** used for probing requests can be defined through `kamailio.proxy.peer_probe.method` parameter. Allowed values are: OPTIONS (default) and INFO.

TIP The system administrator, in most of the cases, will not need to modify the default configuration values other than that of timeout and interval.

If no available peering server is found, the call is rejected with the response code and reason configured in `kamailio.proxy.early_rejects.peering_unavailable.announce_code` and `kamailio.proxy.early_rejects.peering_unavailable.announce_reason`. If a sound file is configured within the *system sound set* assigned to the calling party, an announcement is played as early media before the rejection.

Individual Peering Server Configuration

When the peer probing feature is enabled on system-level, it is possible to add each individual peering server to the list of probed endpoints. You can change the probed status of a server in two ways:

Enable probing of a peering server via the admin web interface

1. Open the properties panel of a peering server: *Peerings* ▸ *select a peering group* ▸ *Details* ▸ *select a peering server* ▸ *Edit*
2. Tick the checkbox *Enable Probing*
3. *Save* changes

Copyright © Sipwise GmbH - All rights reserved

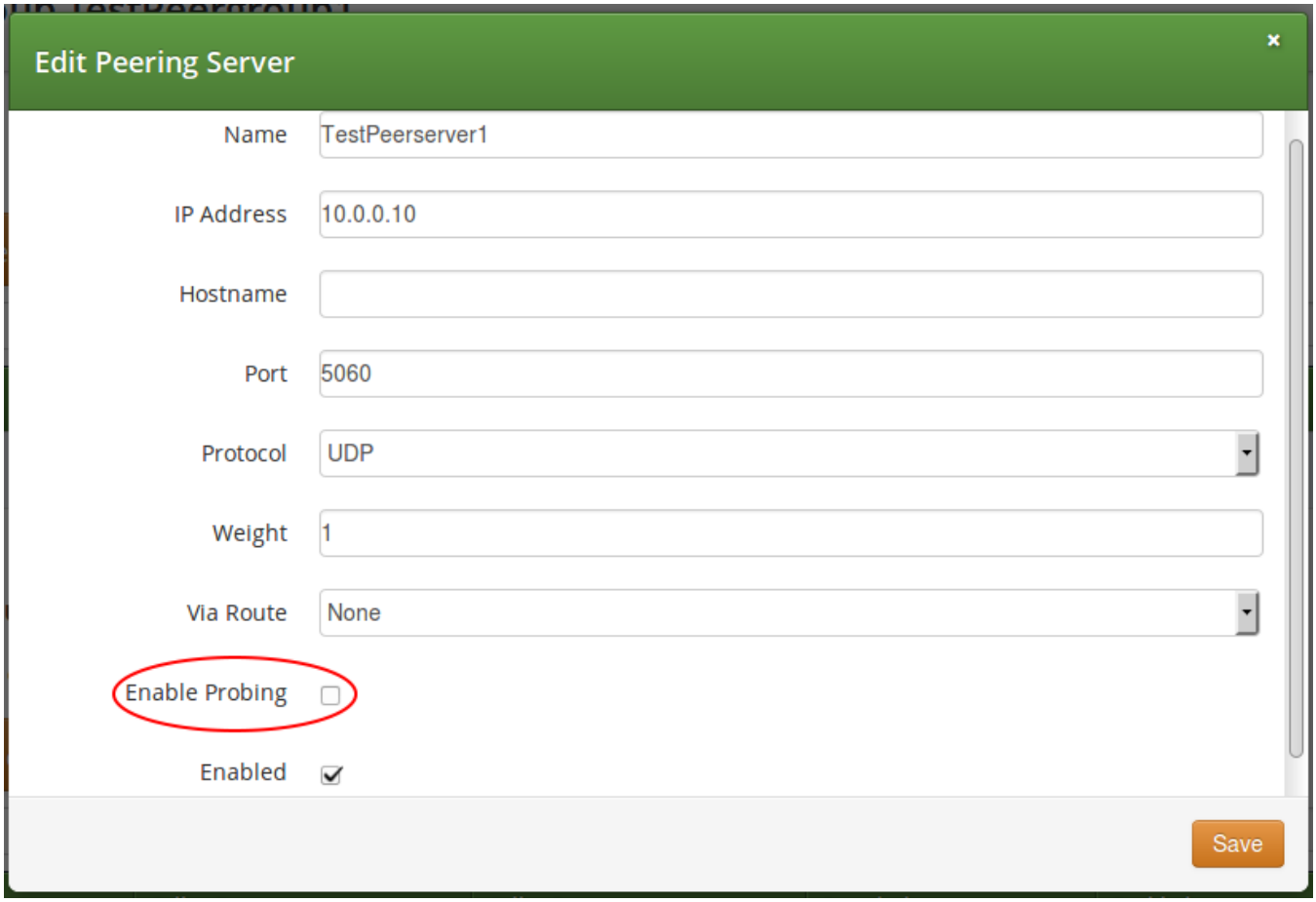


Figure 81. Enable Probing of Peering Server

Enable probing of a peering server via the REST API

- when you *create a new peering server* you will use an HTTP *POST* request and the target URL:

  https://<IP_of_NGCP>:1443/api/peeringservers

- when you *update an existing peering server* you will use an HTTP *PUT* or *PATCH* request and the target URL:

  https://<IP_of_NGCP>:1443/api/peeringservers/id

In all cases you have to set the 'probe' property to `true` in order to enable probing, and to `false` in order to disable probing. Default value is `false` and this property may be omitted in a create/update request, which ensures backward compatibility of the `/api/peeringservers` API resource.

7.13.3. Monitoring of Peer Probing

Peering server states, such as "reachable" / "unreachable", are continuously stored in a time-series database (Prometheus compatible database) by Sipwise C5 Proxy nodes. It is possible to **graphically represent the state of peering servers** on NGCP's admin web interface, just like other system variables (like CPU and memory usage, number of registered subscribers, etc.). However this is not available by default and must be configured by Sipwise.

Alarm conditions or state change events of peering servers are also reported by means of **SNMP traps**.

 Copyright © Sipwise GmbH - All rights reserved

An event trap is emitted each time the reachable state of one of the monitored peering servers changes. An alarm trap is emitted on problematic conditions arising or clearing.

The Sipwise MIB is extended by a table of peers per proxy, containing the peer ID and the peer name, along with the peer probe status. An external monitoring system can **poll the peers table via SNMP** to gather the peer status from each proxy's point of view.

The peer information for all nodes is stored in a table rooted at the OID `.1.3.6.1.4.1.34274.1.1.2.40.2.1` with the following OID path:

```
.iso.org.dod.internet.private.enterprises.sipwise.ngcp.ngcpObjects.ngcpM
onitor.ngcpMonitorPeering.psTable
```

The peer status is an indexed element of that table at the OID `.1.3.6.1.4.1.34274.1.1.2.40.2.1.7` with the following OID path:

```
.iso.org.dod.internet.private.enterprises.sipwise.ngcp.ngcpObjects.ngcpM
onitor.ngcpMonitorPeering.psTable.psEntry.psPeerStatus
```

Value of *psPeerStatus* can be:

- 0: unknown
- 1: administratively down
- 2: administratively up
- 3: probed, pending
- 4: probed, down
- 5: probed, up

7.13.4. Further Details for Advanced Users

TIP This subchapter of the handbook is targeted on advanced system operators and Sipwise engineers and is not necessary to read in order to properly manage peer probing feature of NGCP.

Behaviour of Kamailio Proxy Instances

Each *kamailio-proxy* instance on the proxy nodes performs the probing individually for performance reasons. Each proxy holds its result in its cache to avoid central storage and replication of the probing results.

Each proxy will send an SNMP event trap if it detects a state change or an SNMP alarm trap for problematic conditions arising or clearing for a peering server, because proxies might be geographically distributed along with their load-balancers and can therefore experience different probing results.

Each peering server is cross-checked against the hash table filled during outbound probing requests and is skipped by call routing logic, if a match is found.

Copyright © Sipwise GmbH - All rights reserved

On start or restart of the *kamailio-proxy* instance, the probing will start after the first interval, and NOT immediately after start. In the first probing interval the proxy will always try to send call traffic to peering servers until the first probing round is finished, and will only then start to skip unavailable peering servers.

Changes to Kamailio Proxy Configuration

A new configuration template: /etc/ngcp/config/templates/etc/kamailio/proxy/probe.cfg.tt2 is introduced to handle outbound probing requests.

Database Changes

A new DB column: provisioning.voip_peer_hosts.probe with type TINYINT(1) (boolean) is added to the DB schema.

A peer status change will populate the `kamailio.dispatcher` table, inserting the SIP URI in format `sip:$ip:$port;transport=$transport` in dispatcher group 100, which defines the probing group for peering servers.

Also the `kamailio.dispatcher.attrs` column is populated with a parameter `peerid=$id`. This ID is used during probing to load the peer preferences: `outbound_socket` and `lbrtp_set`, that are required to properly route the probing request.

7.14. Fax Server

There is a Fax Server included in Sipwise C5 . The following sections describe its architecture.

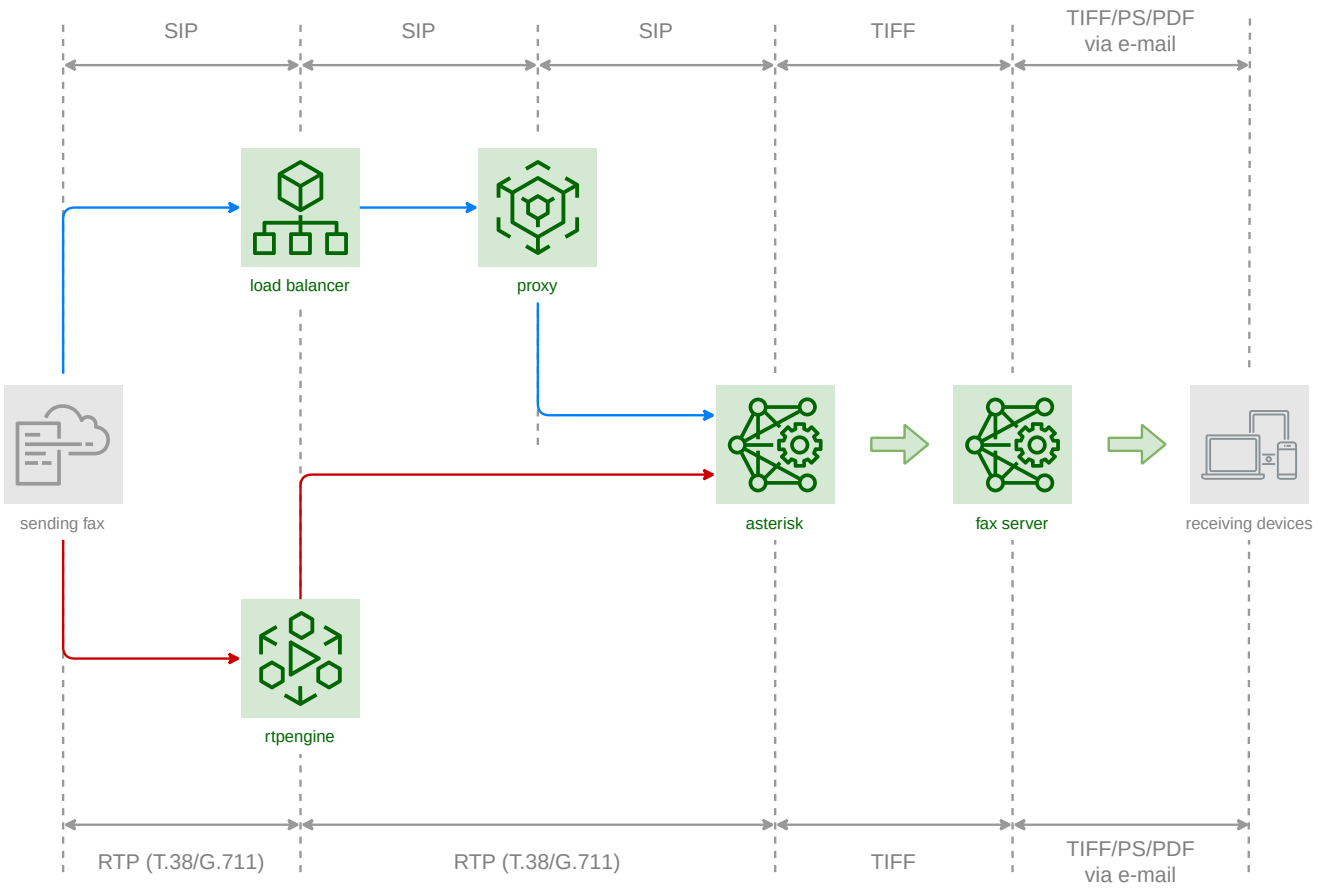
The Fax Server is included on the platform and requires no additional hardware. It supports both T38 and G711 codecs and provides a cost-effective paper-free office solution.

For the details of Fax Server configuration options, please see Faxserver Configuration chapter in this handbook.

7.14.1. Fax2Mail Architecture

To receive faxes via email, a phone call from a sender is connected to the fax application module (Asterisk + Sipwise C5 Fax Server) on Sipwise C5 . The received fax document is converted to the format the receiver has configured (either PS, PDF or TIFF) via the components outlined in the figure below. The email is delivered to one or more configured addresses.

 Copyright © Sipwise GmbH - All rights reserved



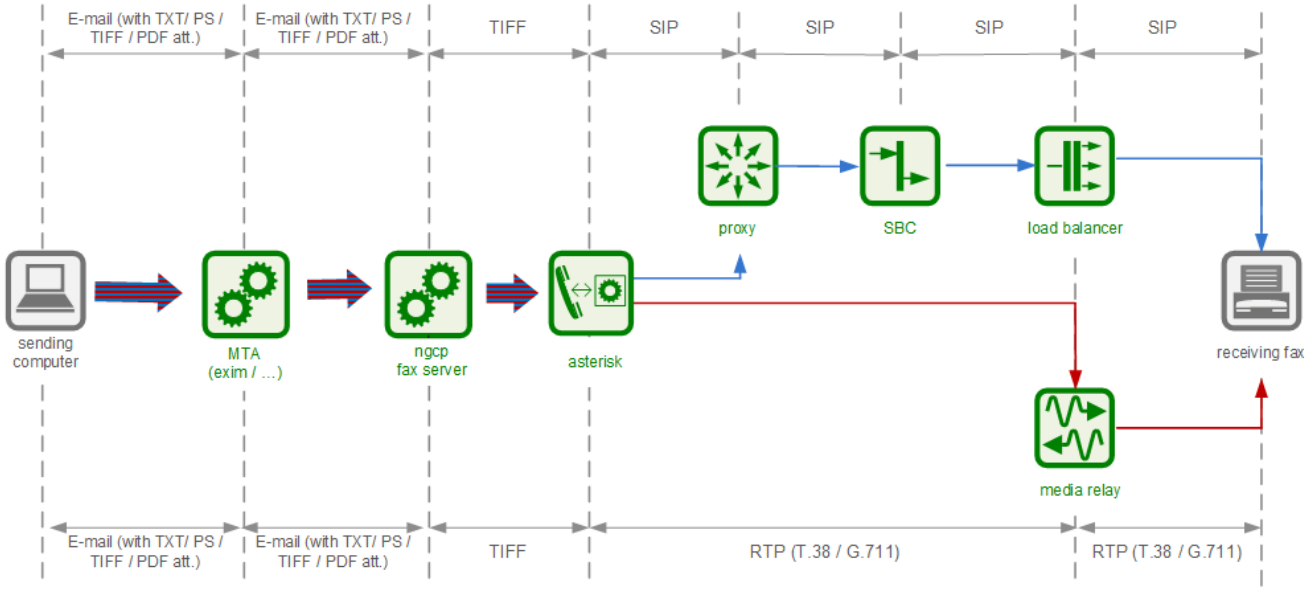
7.14.2. Sendfax and Mail2Fax Architecture

To send faxes via Sipwise C5 a sender can use any email client or an interface such as Webfax or REST API.

Currently, supported formats are TXT, PS, TIFF and PDF.

The document is sent to Sipwise C5 Fax Server instance on Sipwise C5 . Once successfully queued by the fax server, it is converted to an internal TIFF format and is sent via the components outlined in the below figure to the specified phone number. A fax device that can receive the document must be connected on the destination side.

Copyright © Sipwise GmbH - All rights reserved



7.15. Voicemail System

7.15.1. Accessing the IVR Menu

For a subscriber to manage his voicebox via IVR, there are three ways to access the voicebox. One is to call the URI voicebox@yourdomain from the subscriber itself, allowing password-less access to the IVR, as the authentication is already done on SIP level. The second is to call the URI voiceboxpass@yourdomain from any number, causing the system to prompt for a mailbox and the PIN. The third is to call the URI voiceboxpin@yourdomain where only the password/PIN will be requested. The PIN can be set in the *Voicemail and Voicebox* section of the *Subscriber Preferences*.

Mapping numbers and codes to IVR access

Since access might need to be provided from external networks like PSTN/Mobile, and since certain SIP phones do not support calling alphanumeric numbers to dial voicebox, you can map any number to the voicebox URIs using rewrite rules.

To do so, you can provision a match pattern e.g. `^(00|\+)12345$` with a replace pattern `voicebox`, `voiceboxpass` or `voiceboxpin` to map a number to either password-less for the first one or password-based IVR access for the other two. Create a new rewrite rule with the `Inbound` direction and the `Callee` field in the corresponding rewrite rule set.

For inbound calls from external networks, assign this rewrite rule set to the corresponding incoming peer. If you also need to map numbers for on-net calls, assign the rewrite rule set to subscribers or the whole SIP domain.

External IVR access

When reaching `voiceboxpass`, the subscriber is prompted for her mailbox number and a password. All numbers assigned to a subscriber are valid input (primary number and any alias number). By default, the required format is in E.164, so the subscriber needs to enter the full number including country code, for example 4912345 if she got assigned a German number.

While reaching for `voiceboxpin`, the subscriber is only prompted for the PIN.

 Copyright © Sipwise GmbH - All rights reserved

You can globally configure a rewrite rule in config.yml using asterisk.voicemail.normalize_match and asterisk.voicemail.normalize_replace, allowing you to customize the format a subscriber can enter, e.g. having `^0([1-9][0-9]+)$` as match part and `49$1` as replace part to accept German national format.

7.15.2. IVR Menu Structure

The following list shows you how the voicebox menu is structured.

- '1' Read voicemail messages
  - '3' Advanced options
    - '3' To Hear messages Envelope
    - '*' Return to the main menu
  - '4' Play previous message
  - '5' Repeat current message
  - '6' Play next message
  - '7' Delete current message
  - '9' Save message in a folder
    - '0' Save in new Messages
    - '1' Save in old Messages
    - '2' Save in Work Messages
    - '3' Save in Family Messages
    - '4' Save in Friends Messages
    - '#' Return to the main menu
- '2' Change folders
  - '0' Switch to new Messages
  - '1' Switch to old Messages
  - '2' Switch to Work Messages
  - '3' Switch to Family Messages
  - '4' Switch to Friends Messages
  - '#' Get Back
- '3' Advanced Options
  - '*' To return to the main menu
- '0' Mailbox options
  - '1' Record your unavailable message
    - '1' accept it
    - '2' Listen to it
    - '3' Rerecord it
  - '2' Record your busy message
    - '1' accept it

Copyright © Sipwise GmbH - All rights reserved

'2' Listen to it

'3' Rerecord it

'3' Record your name

'1' accept it

'2' Listen to it

'3' Rerecord it

'4' Record your temporary greetings

'1' accept it / or re-record if one already exist

'2' Listen to it / or delete if one already exist

'3' Rerecord it

'5' Change your password

'*' To return to the main menu

- '*' Help
- '#' Exit

7.15.3. Type Of Messages

A message/greeting is a short message that plays before the caller is allowed to record a message. The message is intended to let the caller know that you are not able to answer their call. It can also be used to convey other information like when you will be available, other methods to contact you, or other options that the caller can use to receive assistance.

The IVR menu has three types of greetings.

Unavailable Message

The standard voice mail greeting is the "unavailable" greeting. This is used if you don't answer the phone and so the call is directed to your voice mailbox.

- You can record a custom unavailable greeting.
- If you have not recorded your unavailable greeting but have recorded your name, the system will play a generic message like: "Recorded name is unavailable."
- If you have not recorded your unavailable greeting, the phone system will play a generic message like: "Digits-of-number-dialed is unavailable".

Busy Message

If you wish, you can record a custom greeting used when someone calls you and you are currently on the phone. This is called your "Busy" greeting.

- You can record a custom busy greeting.
- If you have not recorded your busy greeting but have recorded your name, the phone system will play a generic message: "Recorded name is busy."
- If you have not recorded your busy greeting and have not recorded your name (see below), the phone system will play a generic message: "Digits-of-number-dialed is busy."

 Copyright © Sipwise GmbH - All rights reserved

Temporary Greeting

You can also record a temporary greeting. If it exists, a temporary greeting will always be played instead of your "busy" or "unavailable" greetings. This could be used, for example, if you are going on vacation or will be out of the office for a while and want to inform people not to expect a return call anytime soon. Using a temporary greeting avoids having to change your normal unavailable greeting when you leave and when you come back.

7.15.4. Folders

The Voicemail system allows you to save and organize your messages into folders. There can be up to ten folders.

The Default Folder List

- 0 - New Messages
- 1 - Old Messages
- 2 - Work Messages
- 3 - Family Messages
- 4 - Friends Messages

When a caller leaves a message for you,the system will put the message into the "New Messages" folder. If you listen to the message, but do not delete the message or save the message to a different folder, it will automatically move the message to the "Old Messages" folder. When you first log into your mailbox, the Voicemail System will make the "New Messages" folder the current folder if you have any new messages. If you do not have any new messages the it will make the "Old Messages" folder the current folder.

7.15.5. Voicemail Languages Configuration

To add a new language or to change the pronunciation for an existing one, ensure that **mode=new** is defined in */etc/ngcp-config/templates/etc/asterisk/say.conf.tt2*. Adjust the configuration in the same file using the manual in the beginning. Then, as usual, make the new configuration active.

7.15.6. Flowcharts with Voice Prompts

This section shows flowcharts of calls to the voicemail system. Flowcharts contain the name of prompts as they are identified among *Asterisk* voice prompts.

Listening to New Messages

Copyright © Sipwise GmbH - All rights reserved

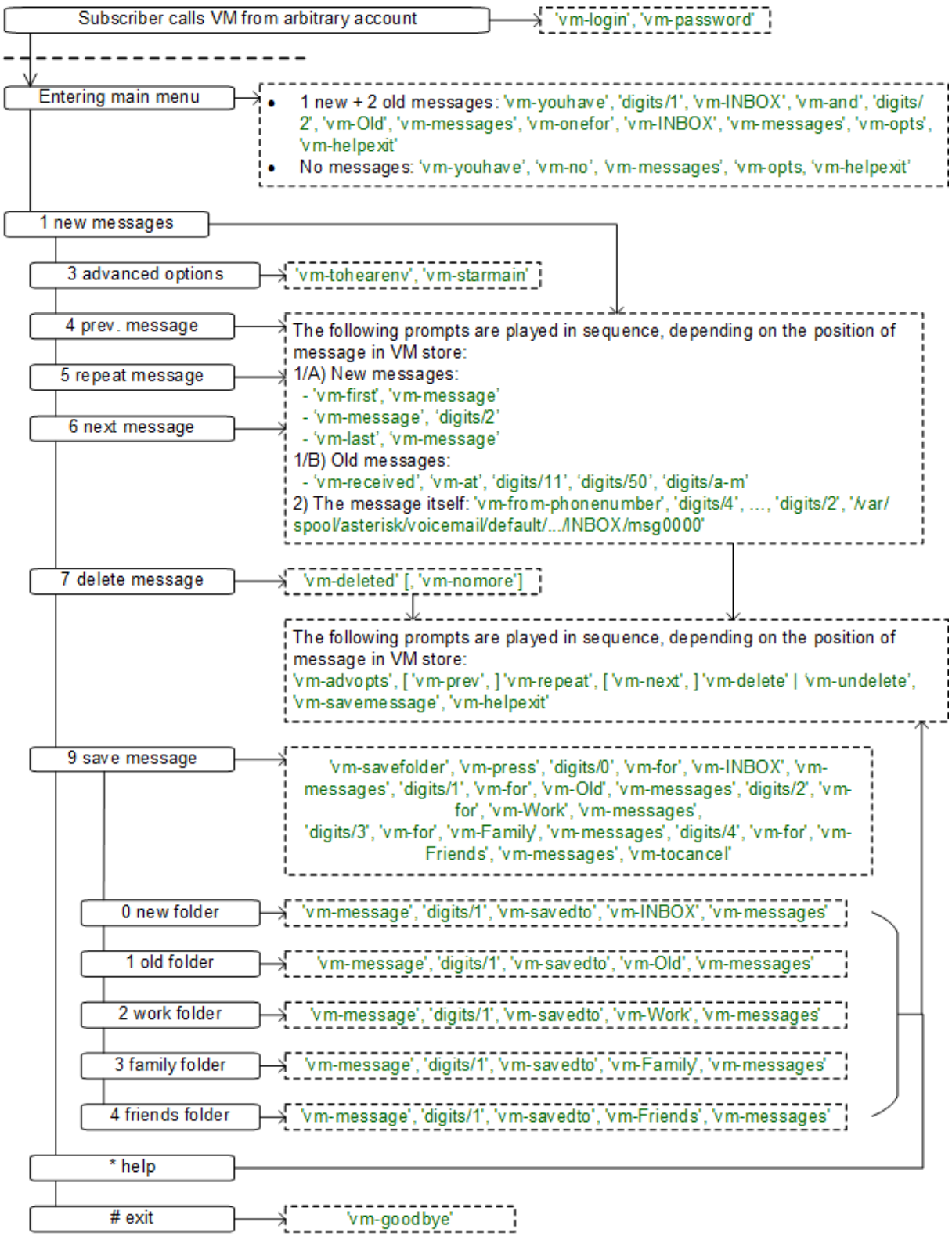


Figure 82. Flowchart of Listening to New Messages

Changing Voicemail Folders

Copyright © Sipwise GmbH - All rights reserved

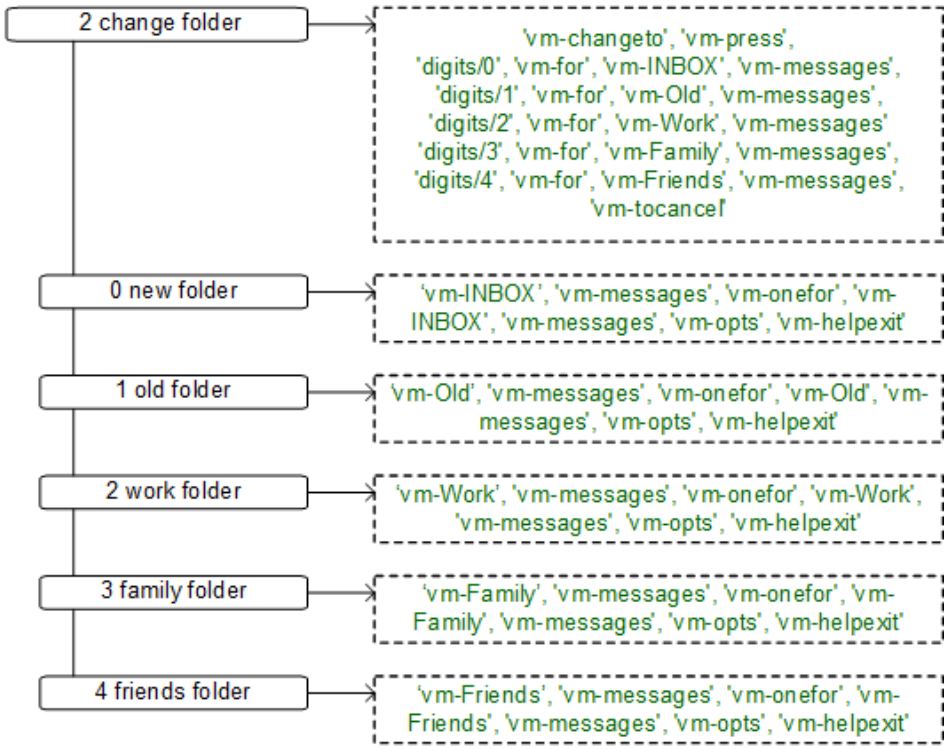


Figure 83. Flowchart of Changing Voicemail Folders

Mailbox Options

Copyright © Sipwise GmbH - All rights reserved

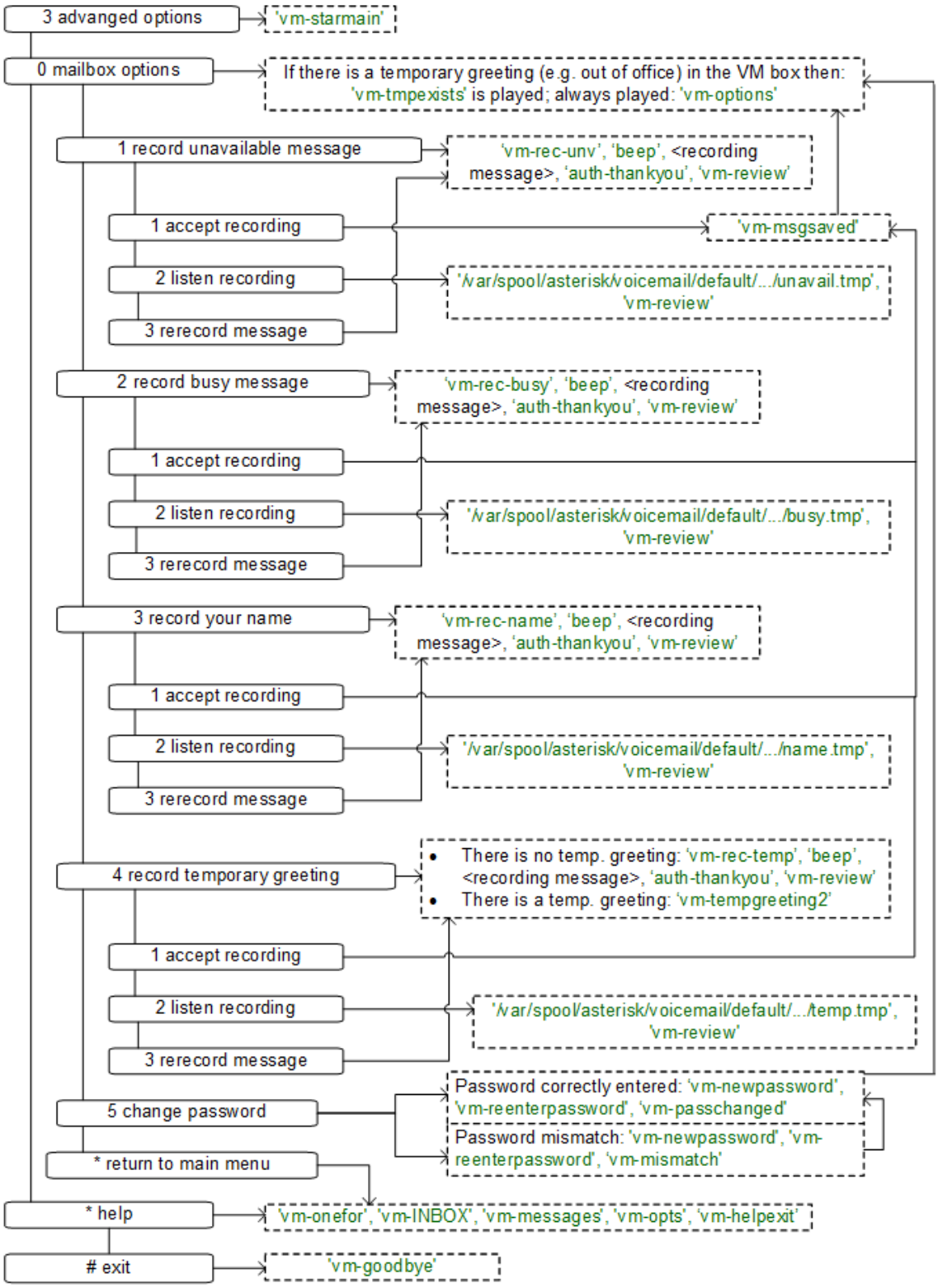


Figure 84. Flowchart of Changing Mailbox Options

Leaving a Message

Copyright © Sipwise GmbH - All rights reserved

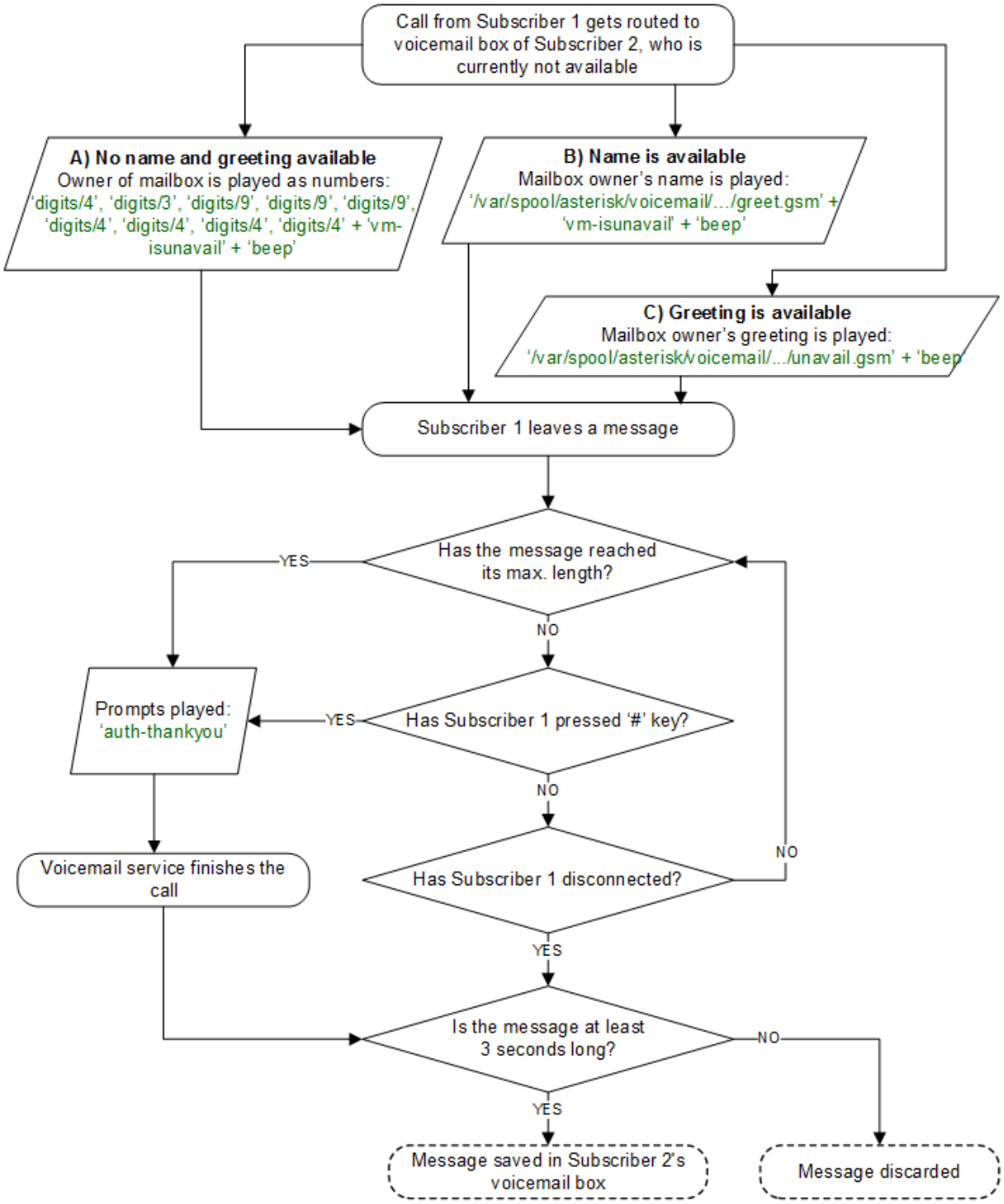
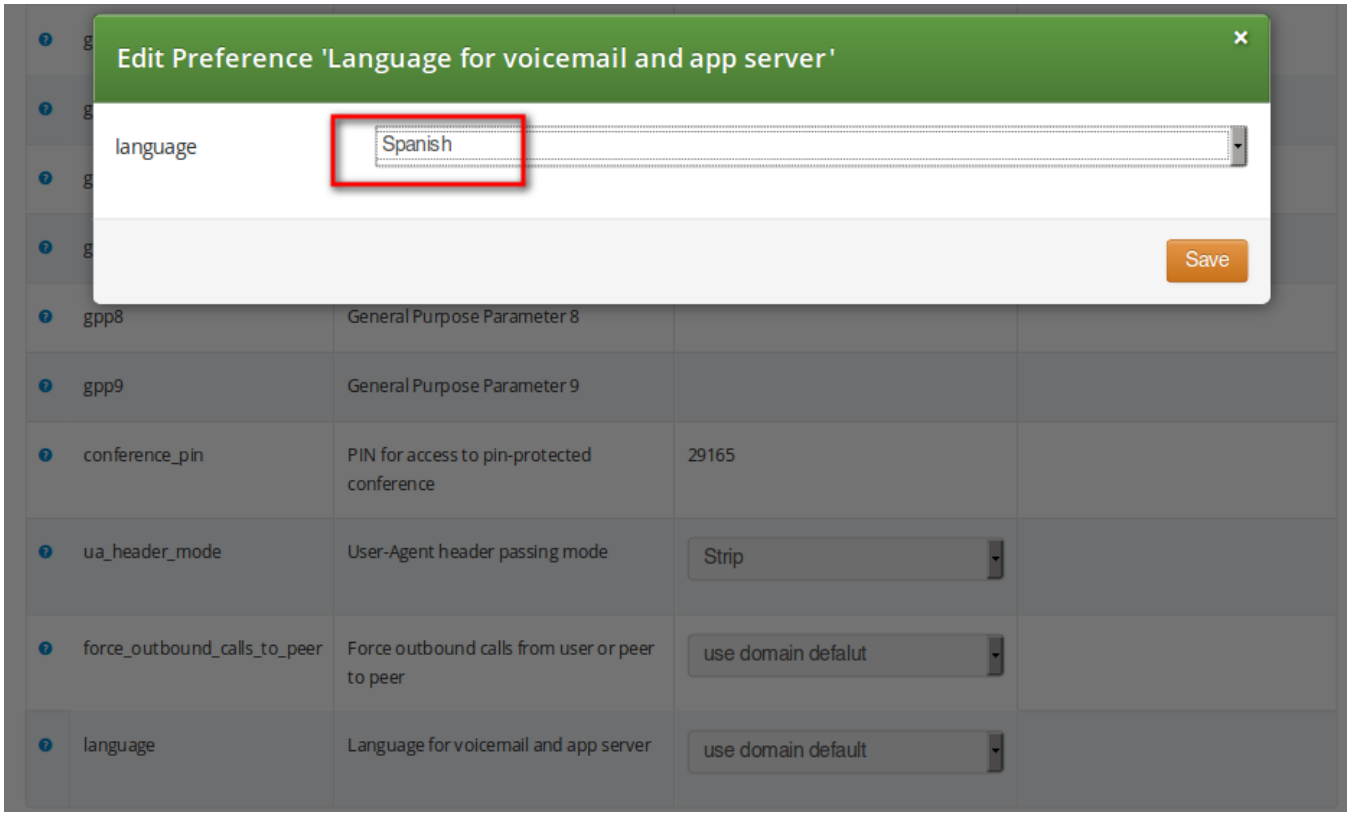


Figure 85. Flowchart of Leaving a Voice Message

7.16. Configuring Subscriber IVR Language

The language for the Voicemail system IVR or Vertical Service Codes (VSC) IVRs may be set using the subscriber or domain preference *language*.

Copyright © Sipwise GmbH - All rights reserved



The Sipwise C5 provides the pre-installed prompts for the Voicemail in the English, Spanish, French and Italian languages and the pre-installed prompts for the Vertical Service Codes IVRs in English only.

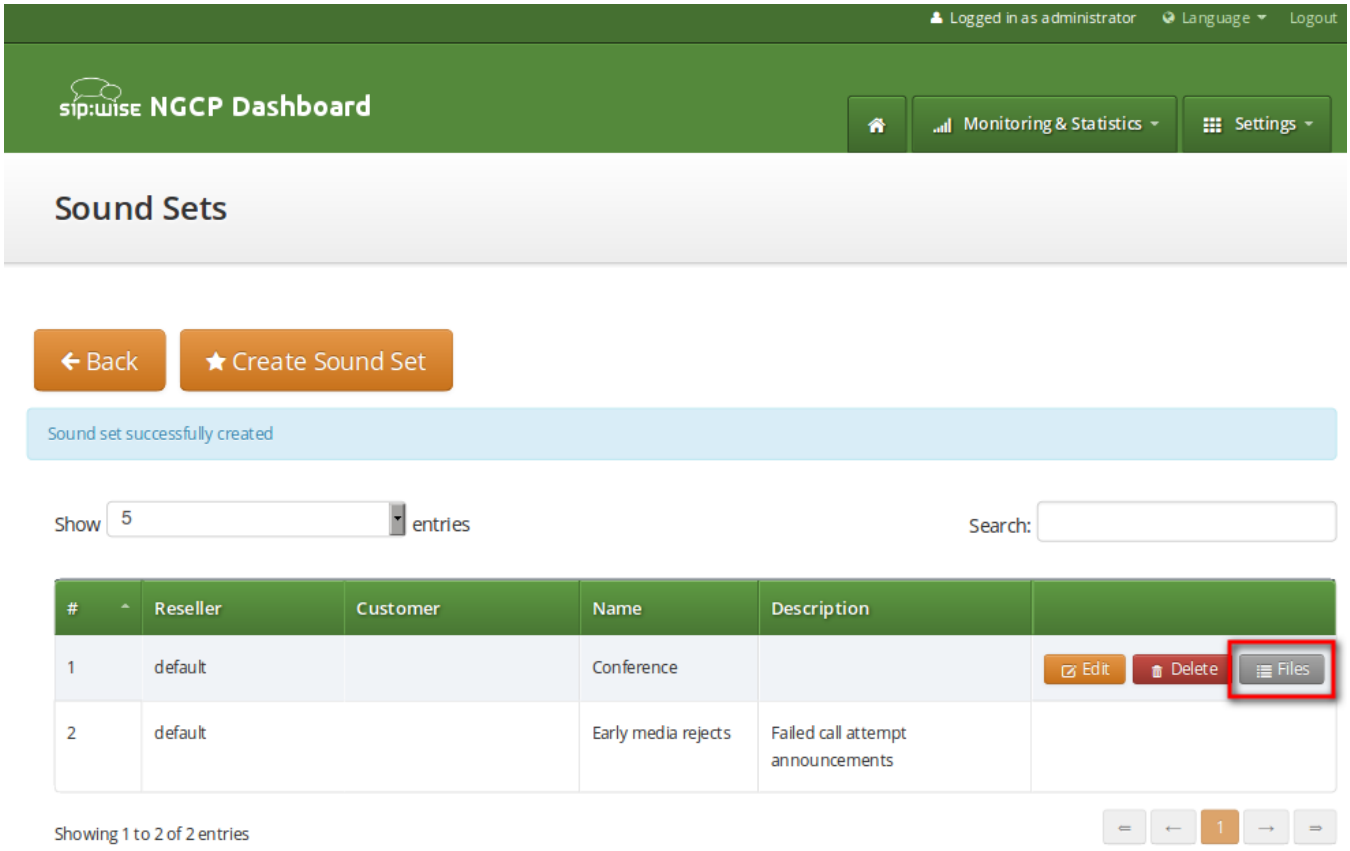
The other IVRs such as the Conference system and the error announcements use the Sound Sets configured in Sipwise C5 Panel and uploaded by the administrator in his language of choice.

7.17. Sound Sets

The Sipwise C5 provides the administrator with ability to upload the voice prompts such as conference prompts or call error announcements on the *Sound Sets page*. There is a preference *sound_set* in the *NAT and Media Flow Control* section on Domain and Subscriber levels to link subscribers to the sound set that they should hear (as usual the subscriber preference overrides the domain one). Additionally Sound Sets can be configured on Peer level in order to play a dedicated early reject prompt when an incoming call doesn't match any local subscriber. Sound Sets can be defined in *SettingsSound Sets*. To create a new Sound Set, click *Create Sound Set*. Then click the *Files* button.

NOTE As soon as the first audio files (for the sound sets) are uploaded, they will be stored using the SQL backend and will be not available in the filesystem (as usual files). But, as soon as these files are first time played for a caller, they will be stored (cached) in the '/ngcp-data/cache' directory. This is done so in order to reduce the time needed to retrieve them (on a routine basis). Anyway, this data will always be stored in the SQL backend as well for redundancy reasons. So your Sipwise C5 will always be able to retrieve them from SQL and cache them again if needed.

 Copyright © Sipwise GmbH - All rights reserved



NOTE You may use 8 or 16 bit mono WAV audio files for all of the voice prompts.

7.17.1. Sound_Set and Contract_Sound_Set Usage

Sound_Set and *Contract_Sound_Set* are used for different purposes:

- *Contract_Sound_Set* is a customer-specific sound set used for the Cloud PBX features. It can be assigned only to a PBX customer.
- *Sound_Set* contains general prompts, e.g. for the Conference, Music on Hold, Early Rejects, etc.

The Music on Hold and Digit prompts can be uploaded to both *Contract_Sound_Set* and *Sound_Set*. In this case, the one from the *Contract_Sound_Set* will take precedence.

7.17.2. Configuring Early Reject Sound Sets

The call error announcements are grouped under *Early Rejects* section. Unfold the section and click *Upload* next to the sound handles (Names) that you want to use. Choose a WAV file from your file system, and click the Loopplay setting if you want to play the file in a loop instead of only once. Click Save to upload the file.

Copyright © Sipwise GmbH - All rights reserved



The call error announcements are played to the user in early media hence the name "Early Reject". If you don't provide the sound files for any handles they will not be used and Sipwise C5 will fallback to sending the error response code back to the user.

The exact error status code and text are configurable in the /etc/ngcp-config/config.yml file, in kamailio.proxy.early_rejects section. Please look for the announcement handle listed in below table in order to find it in the configuration file.

Table 14. Early Reject Announcements

Handle	Description	Message played
block_in	This is what the calling party hears when a call is made from a number that is blocked by the incoming block list (*adm_block_in_list*, *block_in_list* customer/subscriber preferences)	Your call is blocked by the number you are trying to reach.
block_out	This is what the calling party hears when a call is made to a number that is blocked by the outgoing block list (*adm_block_out_list*, *block_out_list* customer/subscriber preferences)	Your call to the number you are trying to reach is blocked.

Copyright © Sipwise GmbH - All rights reserved

Handle	Description	Message played
block_ncos	This is what the calling party hears when a call is made to a number that is blocked by the NCOS level assigned to the subscriber or domain (the NCOS level chosen in *ncos* and *adm_ncos* preferences). *PLEASE NOTE:* It is not possible to configure the status code and text.	Your call to the number you are trying to reach is not permitted.
block_override_pin_wrong	Announcement played to calling party if it used wrong PIN code to override the outgoing user block list or the NCOS level for this call (the PIN set by *block_out_override_pin* and *adm_block_out_override_pin* preferences)	The PIN code you have entered is not correct.
callee_busy	Announcement played on incoming call to the subscriber which is currently busy (486 response from the UAS)	The number you are trying to reach is currently busy. Please try again later.
callee_offline	Announcement played on incoming call to the subscriber which is currently not registered	The number you are trying to reach is currently not available. Please try again later.
callee_tmp_unavailable	Announcement played on incoming call to the subscriber which is currently unavailable (408, other 4xx or no response code or 30x with malformed contact)	The number you are trying to reach is currently not available. Please try again later.
callee_unknown	Announcement that is played on call to unknown or invalid number (not associated with any of our subscribers/hunt groups)	The number you are trying to reach is not in use.
cf_loop	Announcement played when the called subscriber has the call forwarding configured to itself	The number you are trying to reach is forwarded to an invalid destination.
emergency_geo_unavailable	Announcement played when emergency destination is dialed but the destination is not provisioned for the location of the user. *PLEASE NOTE:* The configuration entry for this case in /etc/ngcp-config/config.yml file is emergency_invalid.	The emergency number you have dialed is not available in your region.

Copyright © Sipwise GmbH - All rights reserved

Handle	Description	Message played
emergency_unsupported	Announcement played when emergency destination is dialed but the emergency calls are administratively prohibited for this user or domain (*reject_emergency* preference is enabled)	You are not allowed to place emergency calls from this line. Please use a different phone.
error_please_try_later	Announcement played when the call is handled by 3rd party call control (PCC) and there was an error during call processing. *PLEASE NOTE:* This announcement may be configured in the sound set in voucher_recharge section.	An error has occurred. Please try again later.
invalid_speeddial	This is what the calling party hears when it calls an empty speed-dial slot	The speed dial slot you are trying to use is not available.
locked_in	Announcement played on incoming call to a subscriber that is locked for incoming calls	The number you are trying to reach is currently not permitted to receive calls.
locked_out	Announcement played on outgoing call to subscriber that is locked for outgoing calls	You are currently not allowed to place outbound calls.
max_calls_in	Announcement played on incoming call to a subscriber who has exceeded the *concurrent_max* or *concurrent_max_in* limit or whose customer has exceeded the *concurrent_max_per_account* or *concurrent_max_in_per_account* limit calls	The number you are trying to reach is currently busy. Please try again later.
max_calls_out	Announcement played on outgoing call to a subscriber who has exceeded the *concurrent_max* (total limit) or *concurrent_max_out* (limit on number of outbound calls) or whose customer has exceeded the *concurrent_max_per_account* or *concurrent_max_out_per_account* limit	All outgoing lines are currently in use. Please try again later.

Copyright © Sipwise GmbH - All rights reserved

Handle	Description	Message played
max_calls_peer	Announcement played on calls from the peering if that peer has reached the maximum number of concurrent calls (configured by admin in *concurrent_max* preference of peering server). *PLEASE NOTE:* There is no configuration option of the status code and text in config.yml file for this case.	The network you are trying to reach is currently busy. Please try again later.
no_credit	Announcement played when prepaid account has insufficient balance to make a call to this destination	You don't have sufficient credit balance for the number you are trying to reach.
peering_unavailable	Announcement played in case of outgoing off-net call when there is no peering rule matching this destination and/or source	The network you are trying to reach is not available.
reject_vsc	When the VSC (Vertical Service Code) service is disabled in domain or subscriber preferences (Access Restrictions / reject_vsc is set to TRUE) and a subscriber tries to make a call with VSC, an announcement is played.	N/A (custom message, no default)
relaying_denied	Announcement played on inbound call from trusted IP (e.g. external PBX) with non-local Request-URI domain	The network you are trying to reach is not available.
unauth_caller_ip	This is what the calling party hears when it tries to make a call from unauthorized IP address or network (*allowed_ips*, *man_allowed_ips* preferences)	You are not allowed to place calls from your current network location.
voicebox_unavailable	*PLEASE NOTE:* This announcement is already obsolete, as of Sipwise C5 version mr5.3	The voicemail of the number you are trying to reach is currently not available. Please try again later.

There are some early reject scenarios when either **no voice announcement is played, or a fixed announcement is played**. In either case a SIP error status message is sent from Sipwise C5 to the calling party. It is possible to configure the exact status code and text for such cases in the /etc/ngcp-config/config.yml file, in kamailio.proxy.early_rejects section. The below table gives an overview of those early reject cases.

Table 15. Additional Early Reject Reason Codes

Copyright © Sipwise GmbH - All rights reserved

Handle	Description
block_admin	Caller blocked by adm_block_in_list, adm_block_in_clir and callee blocked by adm_block_out_list (customer or subscriber preference)
block_callee	Callee blocked by subscriber preference block_out_list
block_caller	Caller blocked by subscriber preference block_in_list, block_in_clir
block_contract	Caller blocked by customer preference block_in_list, block_in_clir and callee blocked by customer preference block_out_list
callee_tmp_unavailable_gp	Callee is a PBX group with 0 members. Announcement callee_tmp_unavailable is played; status code and text can be configured.
callee_tmp_unavailable_tm	Callee is a PBX group and we have a timeout (i.e. no group member could be reached). Announcement callee_tmp_unavailable is played; status code and text can be configured.
emergency_invalid	*PLEASE NOTE:* This handle refers to the same early reject case as emergency_geo_unavailable, but is labeled differently in the configuration file.

7.17.3. Play an announcement on behalf of callee server failure in case of outbound calls

The Sipwise C5 makes it possible to play an announcement on behalf of callee server failure in case of outbound calls. The features can be activated on Subscribers and on Peers. For example: if subscriber A calls subscriber B and B refuses the call with code 404 without providing any announcement, Sipwise C5 can be configured to play a customized announcement to A on behalf of B.

To activate this feature, first create a system *Sound Set*, or use an already existing one, and then assign it to the callee subscriber. Upload in the *Sound Set* one or more announcements. Once the *Sound Set* is configured, the subscriber's preference *announce_error_codes_enable* must be enabled under *Subscriber Preferences NAT and Media Flow Control* menu. Last step is to list in the subscriber's preference *announce_error_codes_list* the announcements that will be played to the caller in case a particular error code is returned back from the callee. Each entry of the list has to be a string composed in the following way: <error_code>;<announcement_name>, where error_code is the SIP return code and announcement_name is name of the announcement taken from the sound_set list. Returning to the example above, to play *callee_unknown* message in case of *404* returned from the callee, the entry '404;callee_unknown' has to be added in *announce_error_codes_list* preference.

The same feature is available for peer as well.

 Copyright © Sipwise GmbH - All rights reserved

IMPORTANT

In case *announce_error_codes_enable* is enabled, it is important that the remote endpoint doesn't play any announcement for error codes listed in *announce_error_codes_list* otherwise the final result will be to have two announcements: one generated by the remote endpoint and one generated by Sipwise C5.

7.18. Conference System

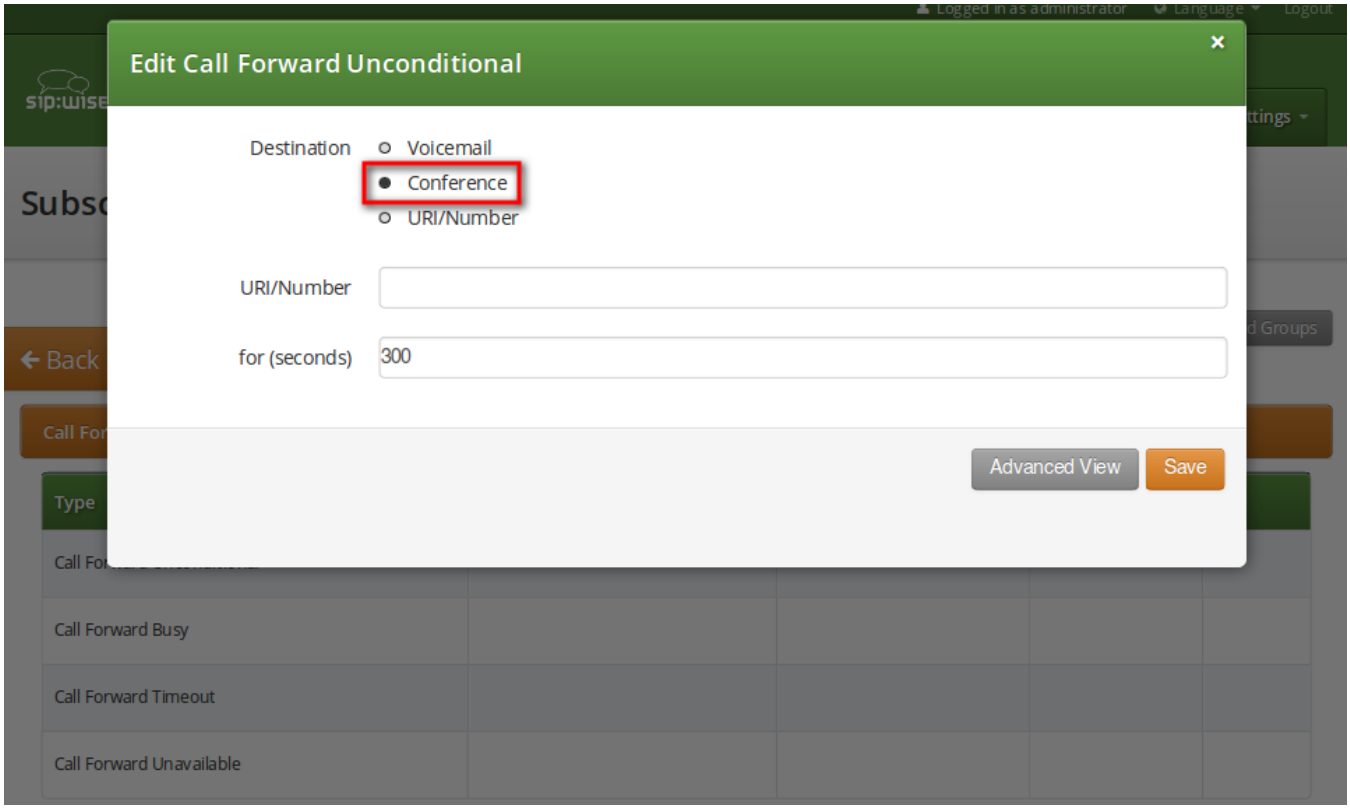
The Sipwise C5 provides the simple pin-protected conferencing service built using the SEMS DSM scripting language. Hence it is open for all kinds of modifications and extensions.

Template files for the sems conference scripts stored in */etc/ngcp-config/templates/etc/sems-b2b/*:

- IVR script: /etc/ngcp-config/templates/etc/sems-b2b/dsm/confpin.dsm.tt2
- Config: /etc/ngcp-config/templates/etc/sems-b2b/dsm/confpin.conf.tt2

7.18.1. Configuring Call Forward to Conference

Go to your *Subscriber Preferences* and click *Edit* on the Call Forward Type you want to set (e.g. *Call Forward Unconditional*).

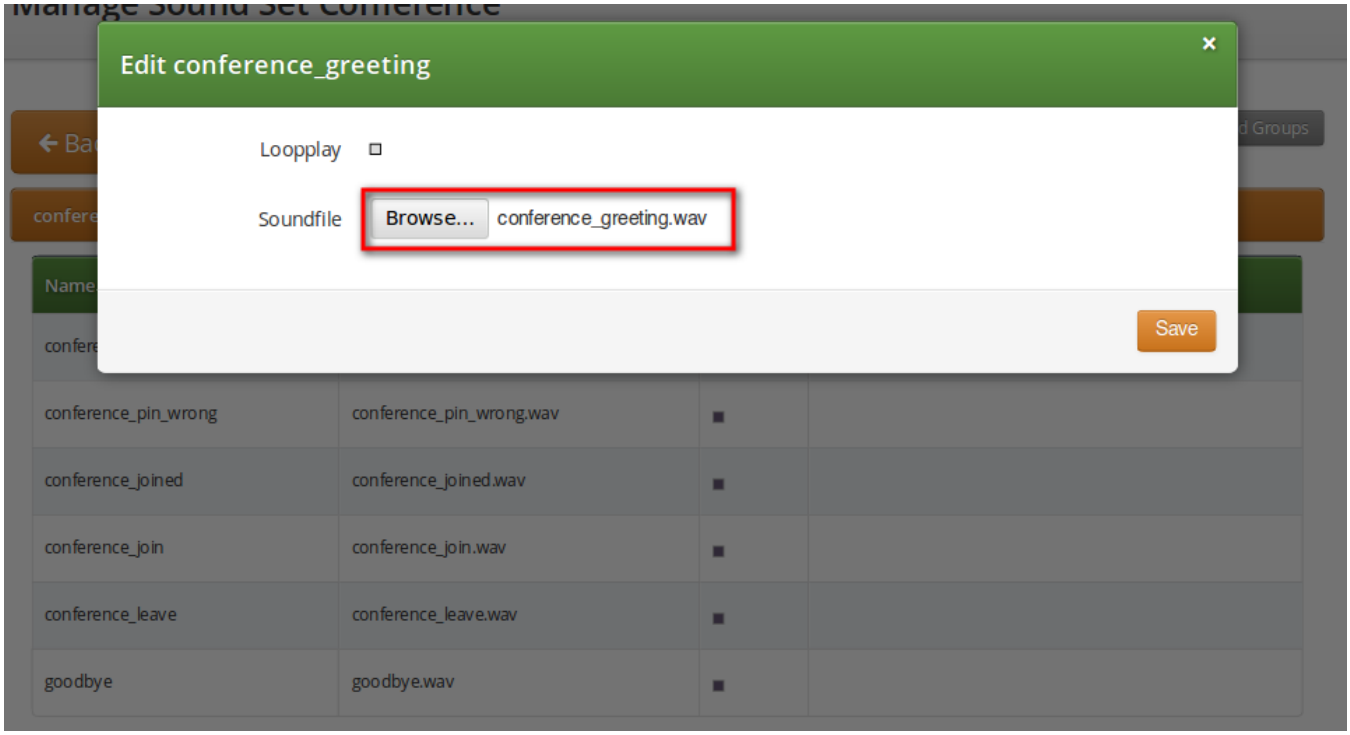


You should select *Conference* option in the *Destination* field and leave the *URI/Number* empty. The timeout defines for how long this destination should be tried to ring.

7.18.2. Configuring Conference Sound Sets

Sound Sets can be defined in *SettingsSound Sets*. To create a new Sound Set, click *Create Sound Set*. Then click the *Files* button.

Copyright © Sipwise GmbH - All rights reserved



Upload the following files:

Table 16. Conference Sound Sets

Handle	Message played
conference_greeting	Welcome to the conferencing service.
conference_pin	Please enter your PIN, followed by the pound key.
conference_pin_wrong	You have entered an invalid PIN number. Please try again.
conference_joined	You will be placed into the conference.
conference_first	You are the first person in the conference.
conference_join	A person has joined the conference.
conference_leave	A person has left the conference.
conference_max_participants	All conference lines are currently in use. Please try again later.
conference_waiting_music	...waiting music...
goodbye	Goodbye.

NOTE You may use 8 or 16 bit mono WAV audio files.

Then set the preference *sound_set* on the Domain or Subscriber level in order to assign the Sound Set you have just created to the subscriber (as usual the subscriber preference overrides the domain one).

Copyright © Sipwise GmbH - All rights reserved

7.18.3. Joining the Conference

There are 2 ways of joining a conference: with or without PIN code. The actual way of joining the conference depends on *Subscriber* settings. A subscriber who has activated the conference through call forwarding may set a PIN in order to protect the conference from unauthorized access. To activate the PIN one has to enter a value in *Subscriber Details Preferences Internals conference_pin* field.

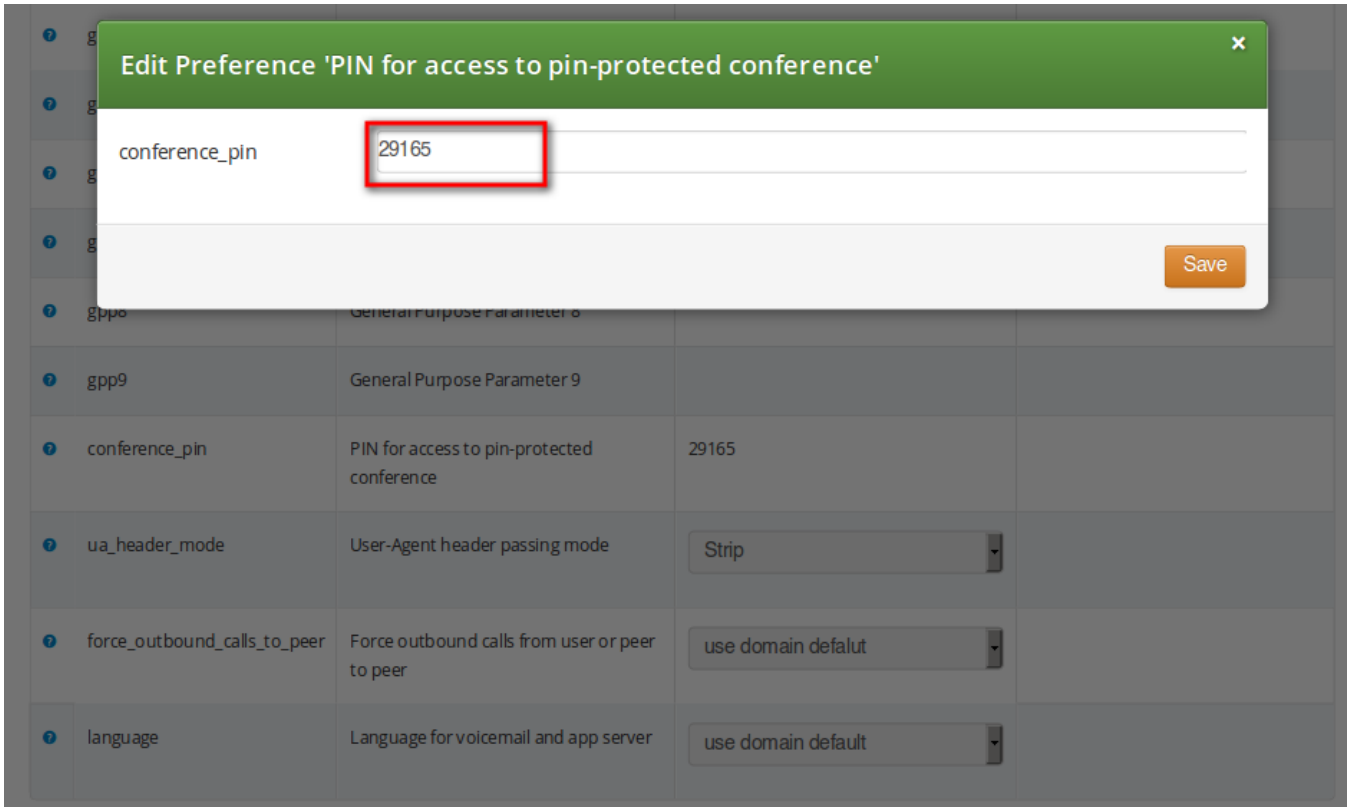


Figure 86. Setting Conference PIN

In case the PIN protection for the conference is activated, when someone calls the subscriber who has enabled the conference, the caller is prompted to enter the PIN of the conference. Upon the successful entry of the PIN the caller hears the announcement that he is going to be placed into the conference and at the same time this is announced to all participants already in the conference.

7.18.4. Conference Flowchart with Voice Prompts

The following 2 sections show flowcharts with voice prompts that are played to a caller when he dials the conference.

Conference Flowchart with PIN Validation

Copyright © Sipwise GmbH - All rights reserved

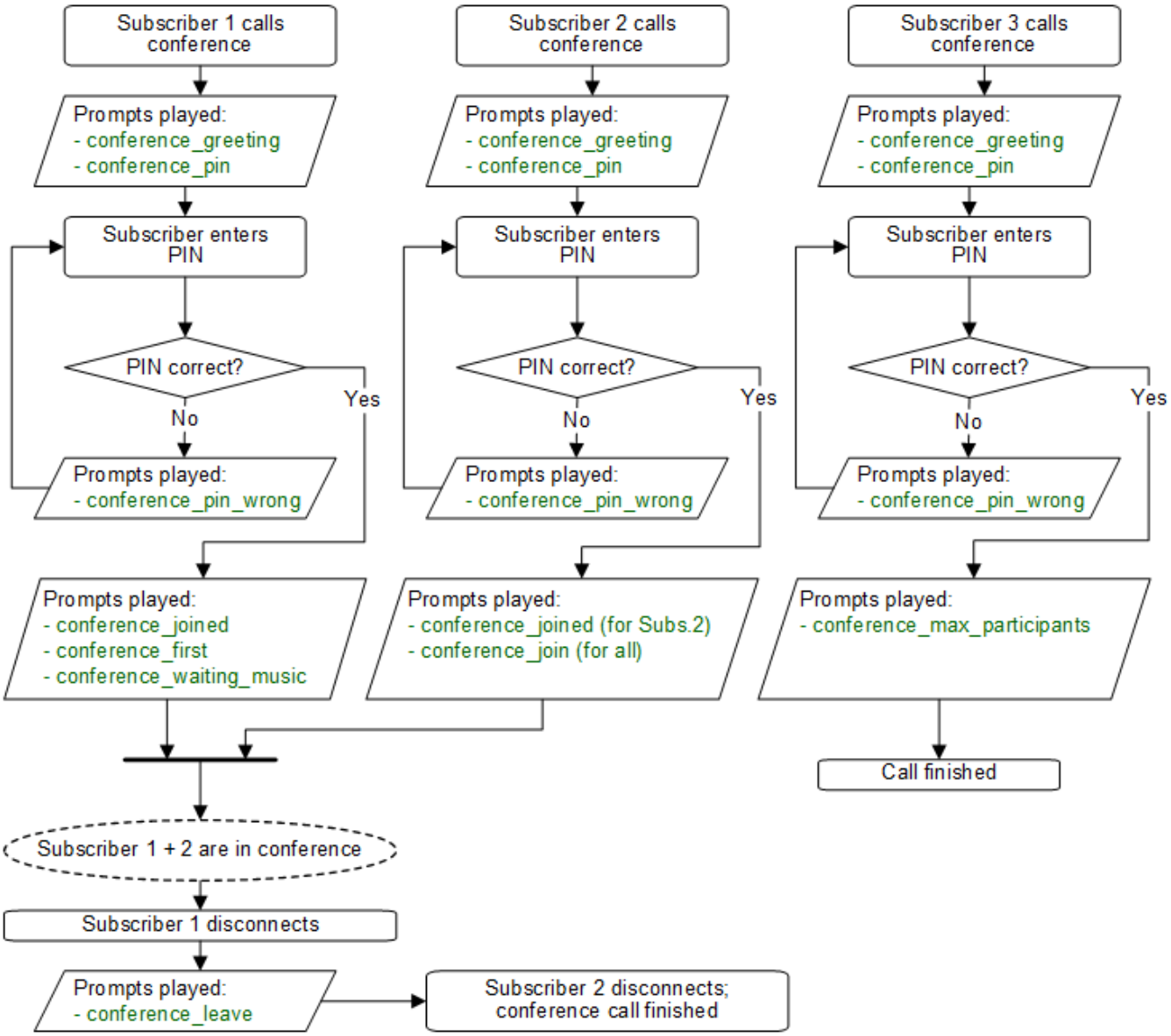


Figure 87. Flowchart of Conference with PIN Validation

Conference Flowchart without PIN

Copyright © Sipwise GmbH - All rights reserved

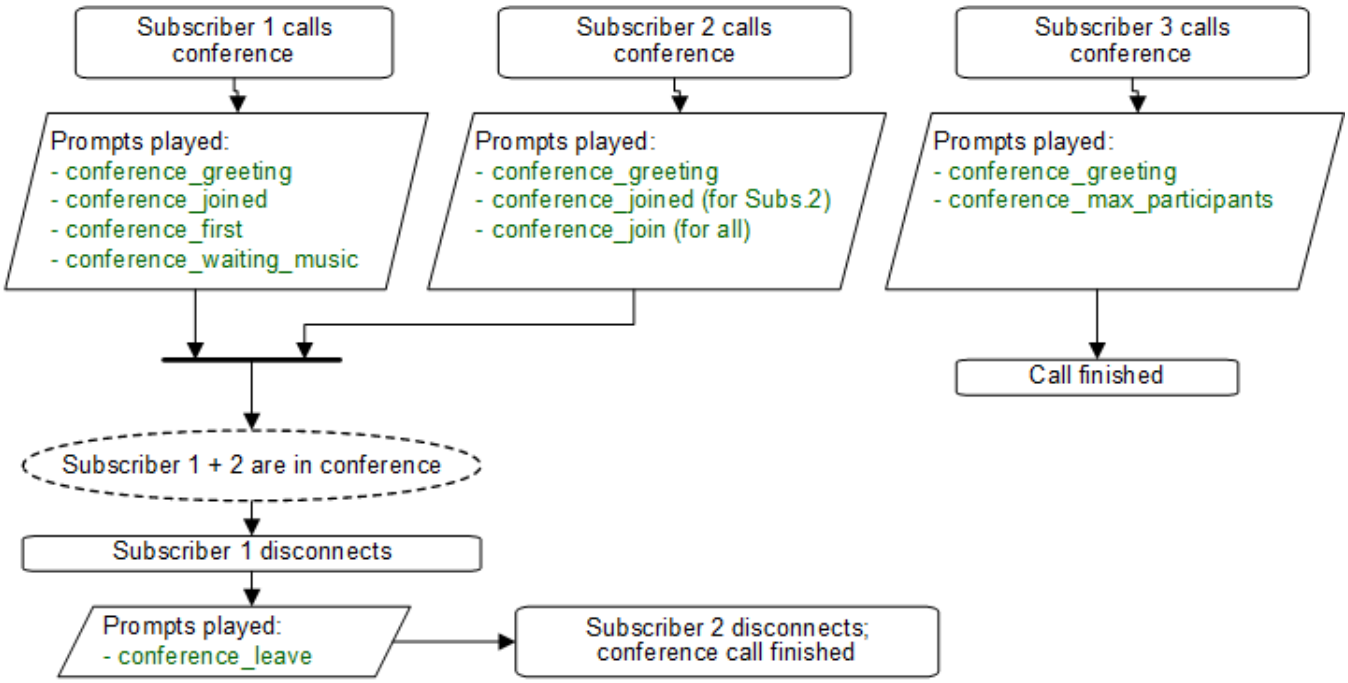


Figure 88. Flowchart of Conference without PIN

7.19. Malicious Call Identification (MCID)

MCID feature allows customers to report unwanted calls to the platform operator.

7.19.1. Setup

To enable the feature first edit config.yml and enable there apps: malicious_call: yes and kamailio: store_recentcalls: yes. The latter option enables kamailio to store recent calls per subscrbriber UUID in the redis DB (the amount of stored recent calls will not exceed the amount of provisionined subscribers).

Next step is to create a system sound set for the feature. In *SettingsSound Sets* either use your already existing *Sound Set* or create a new *Sound Set* and then assign it to your domain or subscribers. In the *Sound Set* there is a fileset *malicious_call_identificationmailicious_call_report* for that purpose.

Once the *Sound Set* is created the Subscriber's Preferences *Malicious Call Identification* must be enabled under *Subcriber Preferences Applications* menu. The same parameter can be set in the Customer's preferences to enable this feature for all its subscribers.

The final step is to create a new *Rewrite Rule* and to route calls to, for instance *123 -> MCID application. For that you create a *Calee Inbound* rewrite rule ^(*123)$ malicious_call

Finally you run ngcpcfg apply "Enabling MCID" to recreate the templates and automatically restart depended services.

7.19.2. Usage

As a subscriber, to report a malicious call you call to either *malicious_call* or to your custom number assigned for that purpose. Please note that you can report only your last received call. You will hear the media reply from the *Sound Set* you have previously configured.

Copyright © Sipwise GmbH - All rights reserved

To check reported malicious calls as the plafrom operator open *SettingsMalicious Calls* tab where you will see a list of registered calls. You can selectively delete records from the list and alternatively you can manage the reported calls by using the REST API.

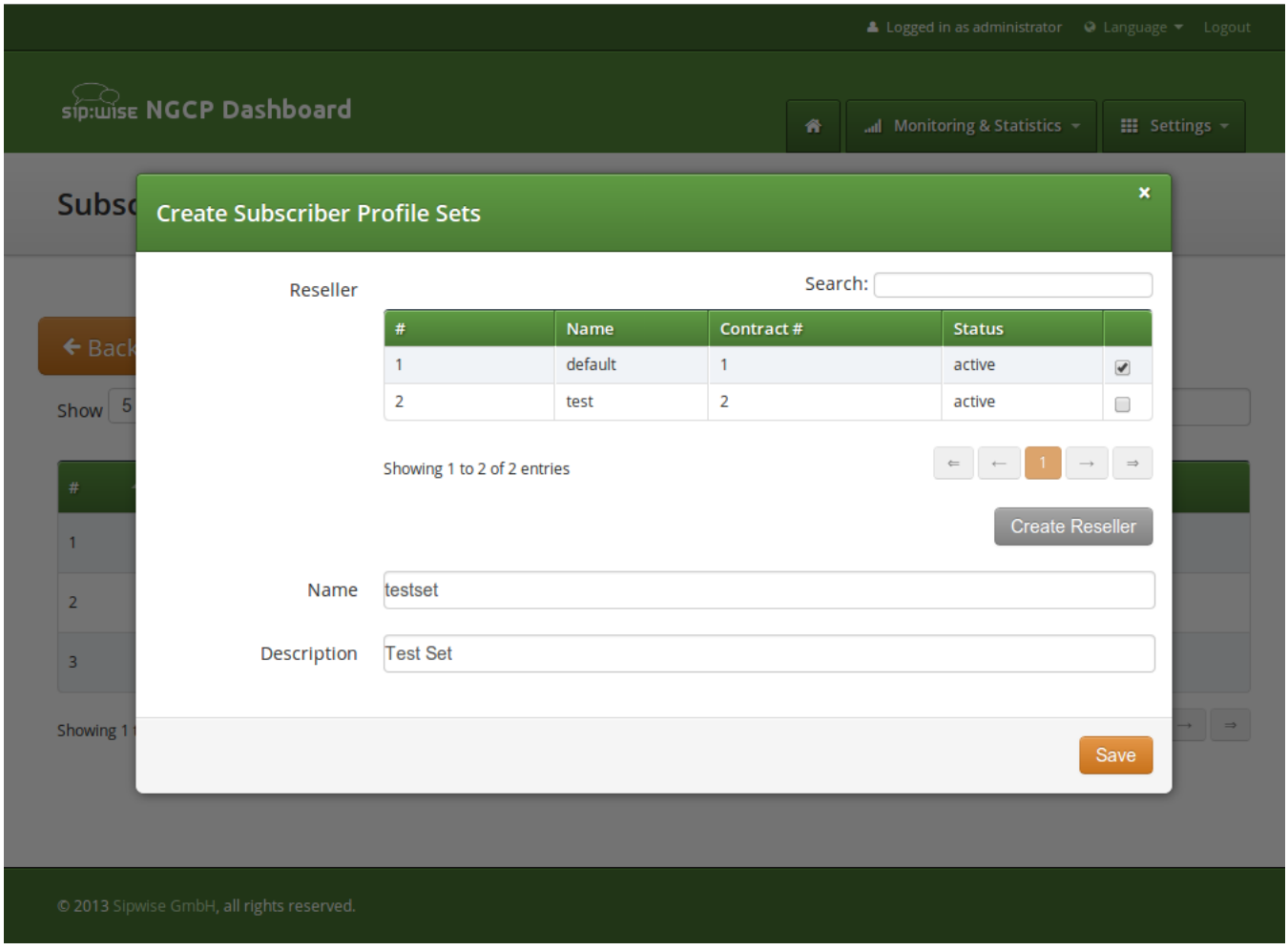
7.20. Subscriber Profiles

The preferences a subscriber can provision by himself via the CSC can be limited via profiles within profile sets assigned to subscribers.

7.20.1. Subscriber Profile Sets

Profile sets define containers for profiles. The idea is to define profile sets with different profiles by the administrator (or the reseller, if he is permitted to do so). Then, a subscriber with administrative privileges can re-assign profiles within his profile sets for the subscribers of his customer account.

Profile Sets can be defined in *SettingsSubscriber Profiles*. To create a new Profile Set, click *Create Subscriber Profile Set*.

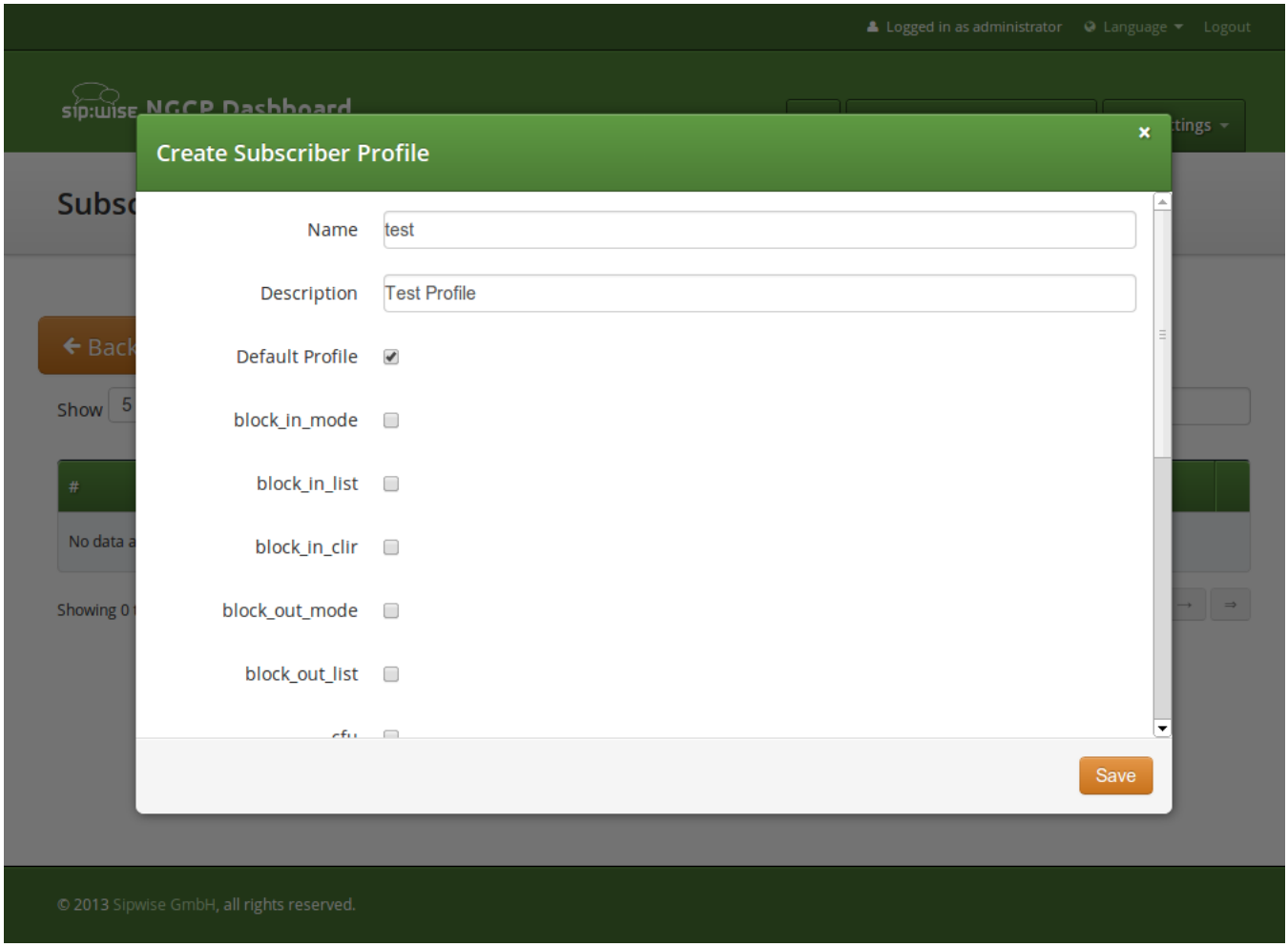


You need to provide a reseller, name and description.

To create Profiles within a Profile Set, hover over the Profile Set and click the *Profiles* button.

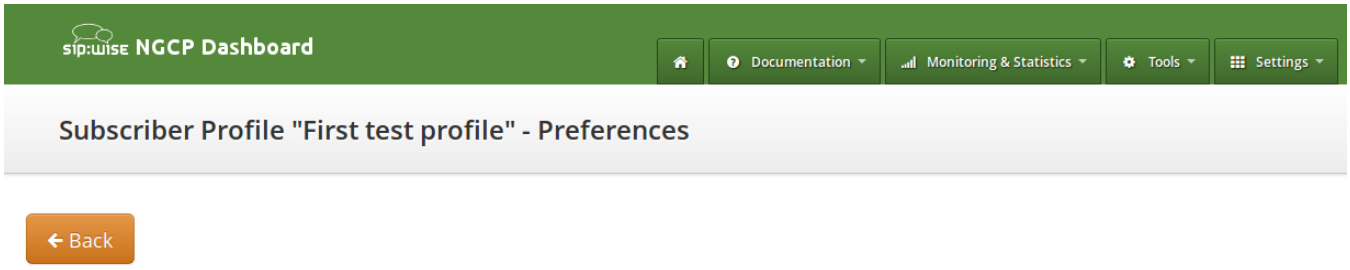
Profiles within a Profile Set can be created by clicking the *Create Subscriber Profile* button.

 Copyright © Sipwise GmbH - All rights reserved



Checking the *Default Profile* option causes this profile to get assigned automatically to all subscribers, who have the profile set assigned. Other options define the user preferences which should be made available to the subscriber.

NOTE When the platform administrator selects *Preferences* of the Subscriber Profile he will get an empty page like in the picture below, if none or only certain options are selected in the Subscriber Profile.



Some of the options, like ncos (NCOS level), will enable the definition of that preference within the Subscriber Profile Preferences. Thus all subscribers who have this profile assigned to will have the preference activated by default. The below picture shows the preferences linked to the sample Subscriber Profile:

Copyright © Sipwise GmbH - All rights reserved



7.21. SIP Loop Detection

In order to detect a SIP loop (incoming call as a response for a call request) Sipwise C5 checks the combination of *SIP-URI*, *To* and *From* headers.

This check can be enabled in config.yml by setting kamailio.proxy.loop_detection.enable: yes. The system tolerates kamailio.proxy.loop_detection.max loops within kamailio.proxy.loop_detection.expire seconds. Higher occurrence of loops will be reported with a SIP 482 "Loop Detected" error message

7.22. Call-Through Application

Call-through allows telephony client to dial into an IVR system and specify (in two-stage dialing fashion) a new destination number which is then dialed by Sipwise C5 to connect the client to the destination. As the call-through system needs to be protected from unauthorized use, a list of CLIs which are allowed to use the call-through system is stored in Sipwise C5 platform.

Table 17. Call-Through Mappings

Column	Description
uuid	The internal UUID of the call-through subscriber
auth_key	Authentication key (CLI)
source_uuid	The internal UUID of the subscriber that is authorized for outgoing call leg (same as uuid in call-through scenario)

7.22.1. Activation

The service is installed on every Sipwise C5 system, but is not activated by default. In order to activate the service set the value '1' to the *callthru_features* option under the *www_admin* section of the *config.yml* configuration file, then execute:

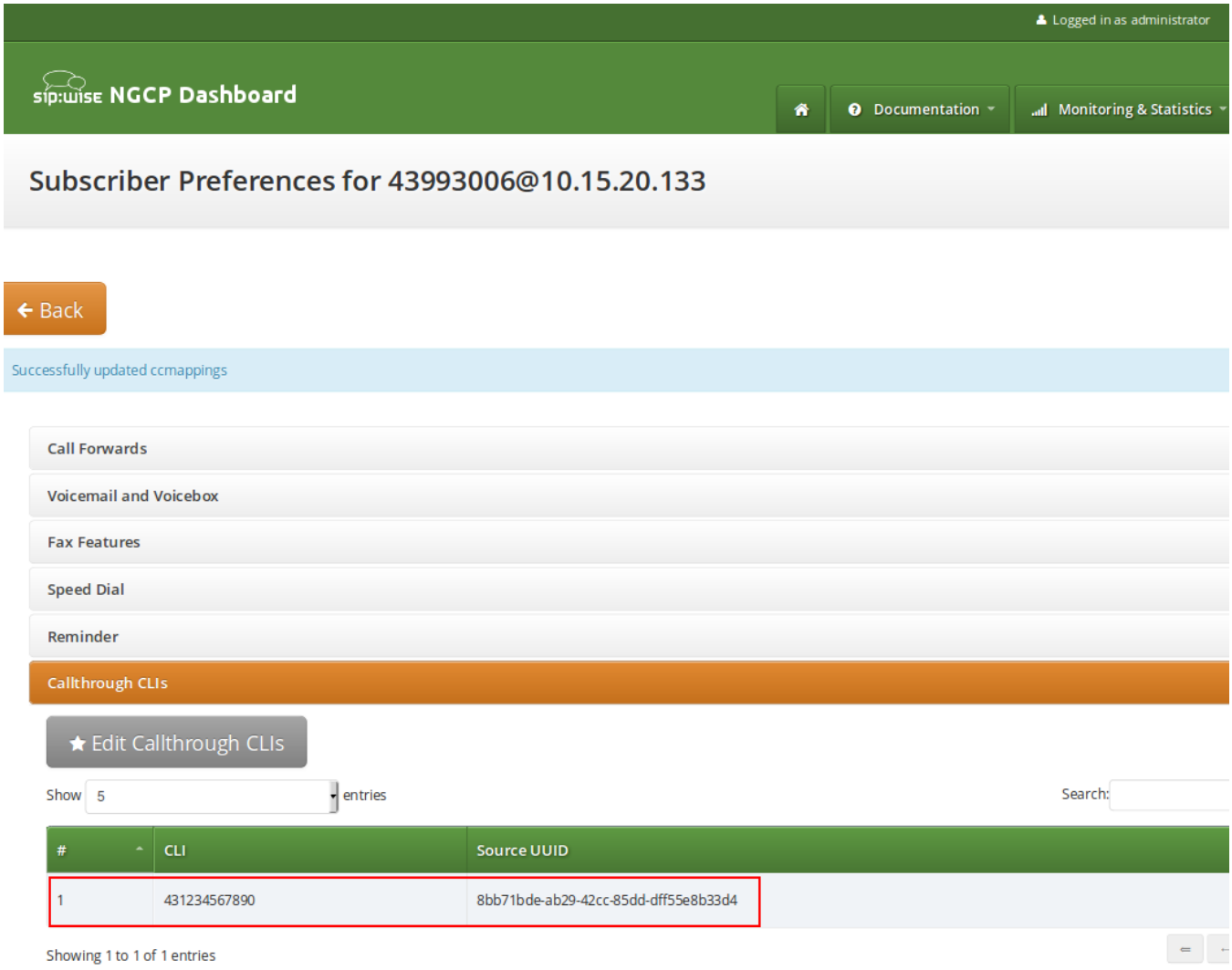
```
ngcpcfg set /etc/ngcp-config/config.yml www_admin.callthru_features='1'
ngcpcfg apply 'Enabled call-through'
ngcpcfg push all
```

 Copyright © Sipwise GmbH - All rights reserved

7.22.2. Administrative Configuration

Subscriber provisioning

In order to manage the call-through CLIs for subscriber, navigate to *SettingsSubscribers*, search for the subscriber you want to edit, press *Details* and then *Preferences*, scroll down to the *Callthrough CLIs* section and press *Edit Callthrough CLIs* button.

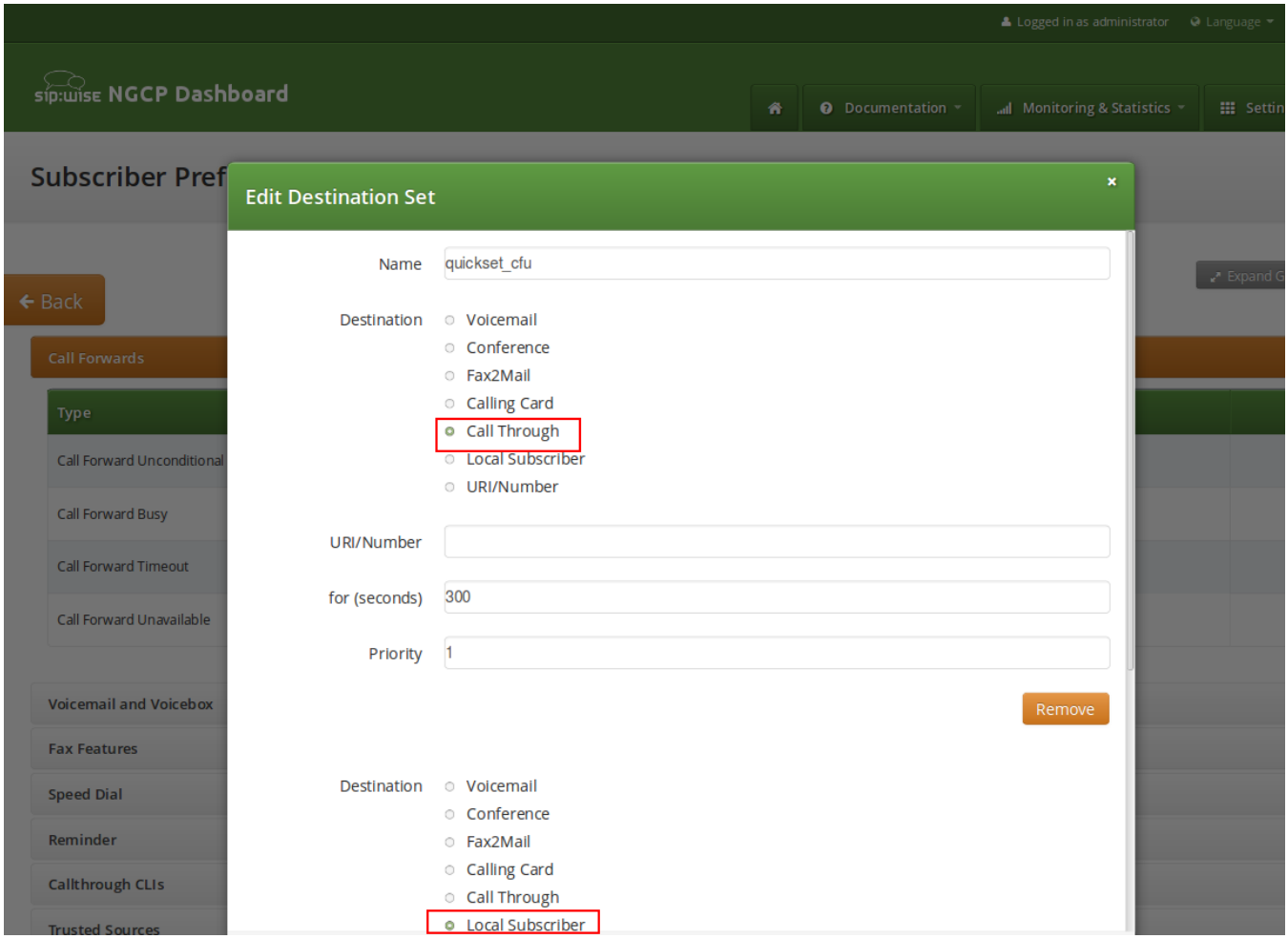


Using Sipwise C5 Panel the user then creates Call Forward to destination *Call Through*.

Forward to local user

If the subscriber has a Call Forward to the call-through application but caller's CLI is not in the authorized CLIs list for call-through, sems responds with error back to proxy and proxy advances to the next number in the Call Forward destinations set. User can enter special destination *Local Subscriber* as next target after *Call Through* in the destinations set in order to terminate the call to the subscriber as if the subscriber didn't exist. This way the user may reach the call-through application from his authorized CLI (e.g. mobile number) and all other callers would reach the SIP subscriber's registered phone as usual.

Copyright © Sipwise GmbH - All rights reserved



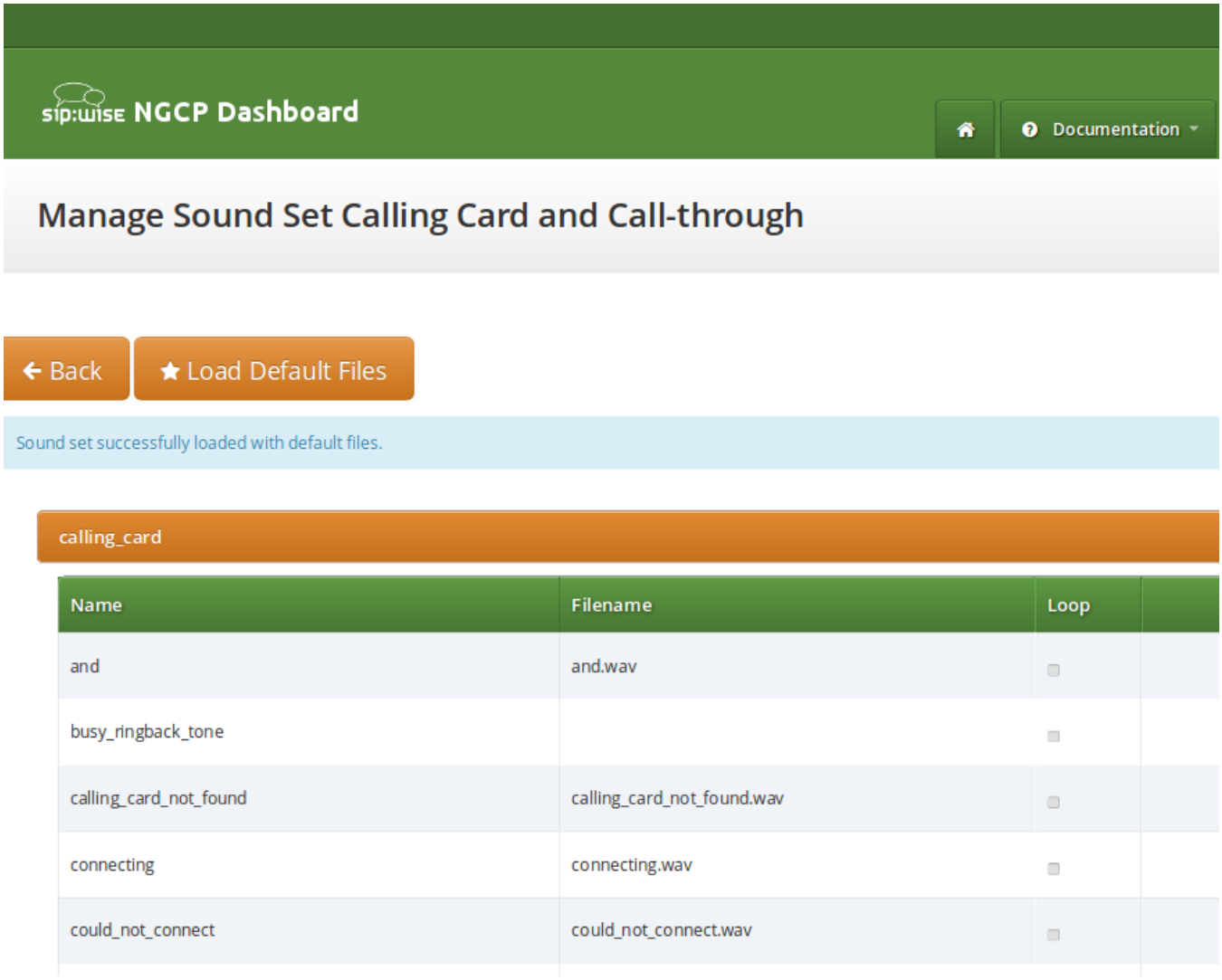
Sound Set provisioning

In order for the Callthrough application to work a Sound Set must be created and associated with the Domain or Subscriber.

Sound Sets can be defined in *SettingsSound Sets*. To create a new Sound Set, click *Create Sound Set*. Then click the *Files* button. Administrator can upload the default sounds in one of supported languages or uploaded by the administrator manually in his language of choice.

There is a preference *sound_set* on Domain and Subscriber levels to link subscribers to the sound set that they should hear (as usual the subscriber preference overrides the domain one).

Copyright © Sipwise GmbH - All rights reserved



NOTE You may use 8 or 16 bit mono WAV audio files for all of the voice prompts.

7.22.3. Call Flow

The call arrives at sems application server with Request-URI user callthrough.

Internal Header Parameters

The INVITE contains an extra SIP header P-App-Param with the following parameters:

Table 18. SIP Header parameters for call-through application

Name	Meaning
uuid	The internal UUID of the call-through subscriber
srcnumber	Caller's CLI for the authentication
outgoing_cli	New CLI to be used by sems application for the outgoing call leg

Copyright © Sipwise GmbH - All rights reserved

Caller authorization

Caller is authorized using mapping shown in table above: select source_uuid from provisioning.voip_cc_mappings where uuid=$uuid and auth_key=$srcnumber;

If the check fails return the configured error response code. Then proceed with the call setup as follows.

Outgoing call

Sems requests the user to enter destination and starts digit collection. Digit collection process is terminated after 5 seconds (configurable in sems config file) or by pressing the '#' key. User can start entering destination while the voice prompt is being played.

Sems sends INVITE to the proxy with Request-URI: sip:$number@$outboundproxy;sw_domain=$subscriber.domain

From: $outgoing_cli

On receiving the 401 or 407 response from the proxy the application authenticates using the digest credentials retrieved for the call-through subscriber from the voip_subscribers table:select s.username, s.password, d.domain from provisioning.voip_subscribers s, provisioning.voip_domains d where s.uuid=$source_uuid and s.domain_id=d.id;

If the call setup fails the application plays back the "could_not_connect" sound file. If successful the application acts transparently and does not provide any voice announcements or DTMF detection.

CLI configuration

The CLI on the outgoing call from the call-through module is set to the Network-Provided Number (NPN) of the call-through subscriber. There is nothing to configure.

7.23. Calling Card Application

Calling card application uses a similar concept to call-through except that authorization process operates on the PIN code entered by user using DTMF instead of the CLI. The Sipwise C5 maps incoming UUID of the pilot subscriber to the list of PINs for calling card application with their corresponding subscriber UUIDs for outbound call leg using table provisioning.voip_cc_mappings table {"uuid", "auth_key", "source_uuid"}

Table 19. Calling Cards

Column	Description
uuid	The internal UUID of the pilot subscriber
auth_key	Authentication key (PIN)
source_uuid	The internal UUID of the subscriber that is authorized for outgoing call leg

Copyright © Sipwise GmbH - All rights reserved

7.23.1. Administrative Configuration

Subscriber provisioning

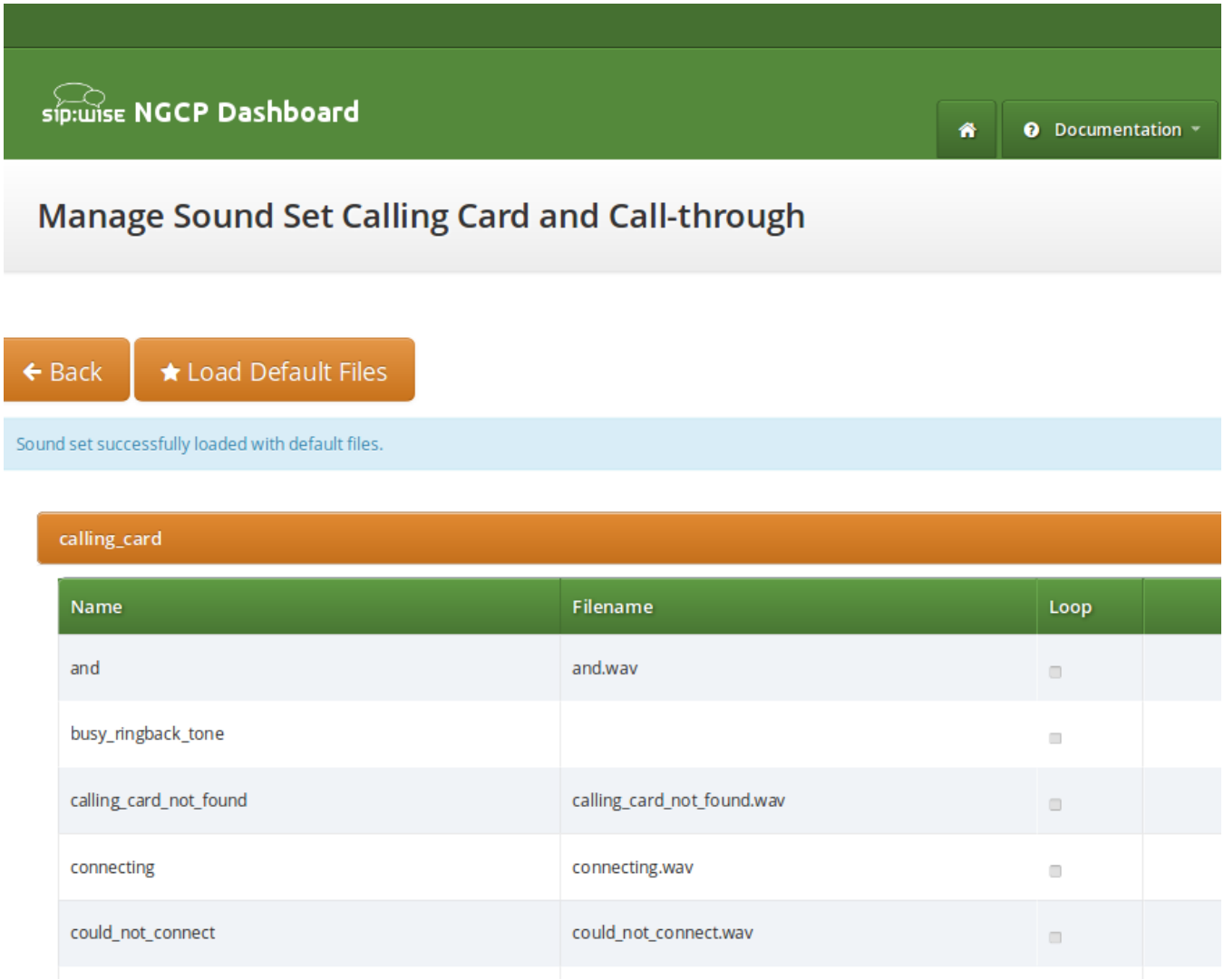
In order to use the calling cards service the user creates a Call Forward to destination *Calling Card* for the designated subscriber that will be used as access number for this service.

Sound Set provisioning

In order for the Calling Card application to work a Sound Set must be created and associated with the Domain or Subscriber.

Sound Sets can be defined in *SettingsSound Sets*. To create a new Sound Set, click *Create Sound Set*. Then click the *Files* button. Administrator can upload the default sounds in one of supported languages or uploaded by the administrator manually in his language of choice.

There is a preference *sound_set* on Domain and Subscriber levels to link subscribers to the sound set that they should hear (as usual the subscriber preference overrides the domain one).



NOTE You may use 8 or 16 bit mono WAV audio files for all of the voice prompts.

Copyright © Sipwise GmbH - All rights reserved

CLI configuration

The CLI on the outgoing call from the calling card app can be configured in one of the following ways using subscriber preferences:

1) Show original caller's CLI: the calling card subscriber shall have allowed_clis: * (any). Sems application sends the original caller's CLI in the From header, it is validated by the SIP proxy and sent to outside.

2) Show number of the pilot (calling card) subscriber: the calling card subscriber shall have an empty allowed_clis and desired number set as value of user_cli preference. The SIP proxy overrides the original caller's CLI in UPN with the value of the user_cli preference. The peer must have set outbound_from_user, outbound_from_display: User-Provided Number (UPN).

7.23.2. Call Flow

The call arrives at sems application server with Request-URI user callingcard.

Internal Header Parameters

The INVITE contains an extra SIP header P-App-Param with the following parameters:

Table 20. SIP Header parameters for calling card application

Name	Meaning
uuid	The internal UUID of the pilot subscriber
outgoing_cli	New CLI to be used by sems application for the outgoing call leg

Caller authorization

- Sems requests the user to enter PIN and starts digit collection. Digit collection process is terminated after 5 seconds (configurable in sems config file) or by pressing the '#' key. User can start entering destination while the voice prompt is being played.
- Sems checks that PIN is valid and belongs to the pilot subscriber using mapping as shown in the table. It fetches UUID of the subscriber to be used for outgoing call leg: select source_uuid from provisioning.voip_cc_mappings where uuid=$uuid and auth_key=$pin;
- If the check fails sems will request the user to re-enter PIN up to the configured number of times.
- If successful proceed with the call setup making call on behalf of subscriber determined by the source_uuid key as follows.

Outgoing call

Sems application plays back the available balance of the customer. Sems requests the user to enter destination and starts digit collection. Digit collection process is terminated after 5 seconds (configurable in sems config file) or by pressing the '#' key. User can start entering destination while the voice prompt is being played.

Sems sends INVITE to the proxy with Request-URI: sip:$number@$outboundproxy;sw_domain=$subscriber.domain

 Copyright © Sipwise GmbH - All rights reserved

From: $outgoing_cli

On receiving the 401 or 407 response from the proxy the application authenticates using the digest credentials retrieved for the subscriber for outgoing call leg from the voip_subscribers table: select s.username, s.password, d.domain from provisioning.voip_subscribers s, provisioning.voip_domains d where s.uuid=$source_uuid and s.domain_id=d.id;

Voucher recharge

During the destination collection phase in calling card application user can enter special code *1*<pin># (configurable in sems config file) to transfer balance from other calling card customer to the currently authorized customer. Sems transfers all remaining balance from that customer to the current customer.

Billing

The call via calling card application as well as call-through generates three CDRs:

- A to B: The incoming call from any source to the call-through subscriber.
- B to callingcard@app.local or callthrough@app.local: The call forward to the sems application.
- B to C: The outgoing call to the final destination. The three CDRs are handled by the billing process as usual, exported and shown in all call lists. .

Copyright © Sipwise GmbH - All rights reserved

7.24. Invoices and Invoice Templates

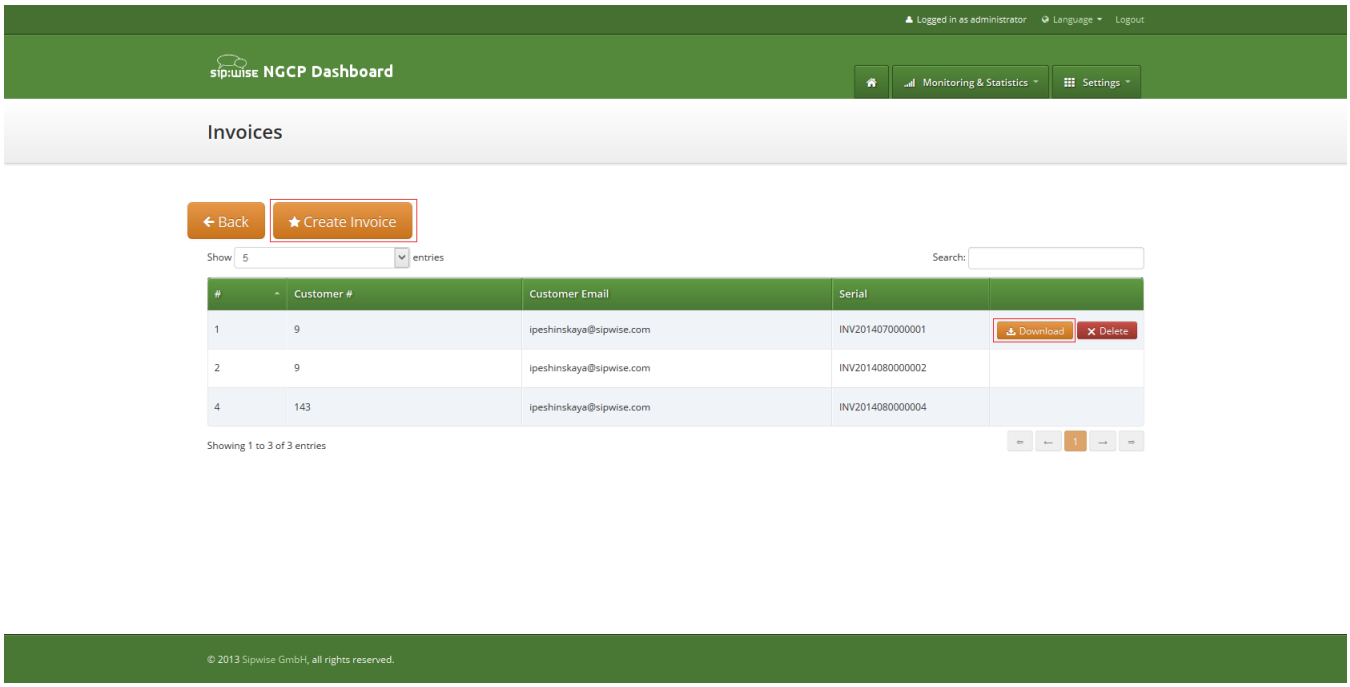
Content and vision of the invoices are customizable by invoice templates.

NOTE The Sipwise C5 generates invoices in pdf format.

7.24.1. Invoices Management

Invoices can be requested for generation, searched, downloaded and deleted on the administrative web interface. Navigate to *Settings* *Invoices* menu and you get a list of all invoices currently stored in the database.

TIP The system operator or a third party application can also generate, list, retrieve and delete invoices via the REST API. Please read further details here.

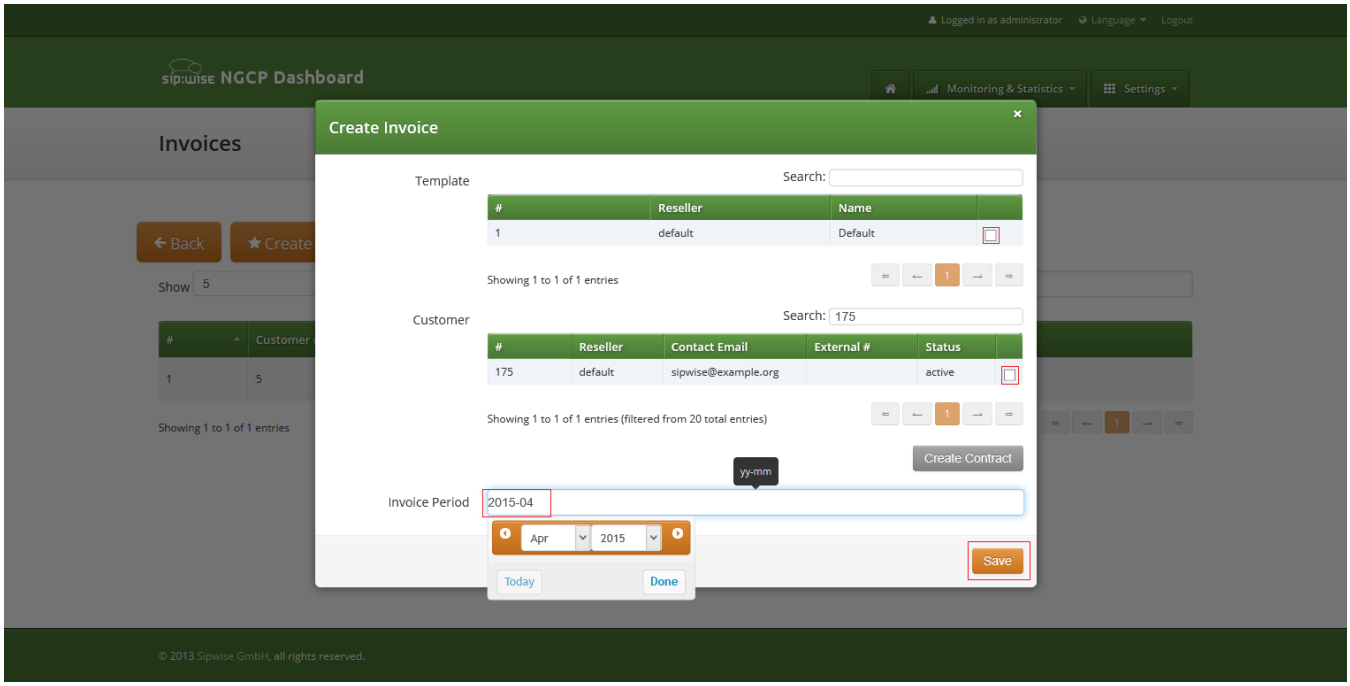


To request invoice generation for the particular customer and period press "Create invoice" button. On the invoice creation form following parameters are available for selection:

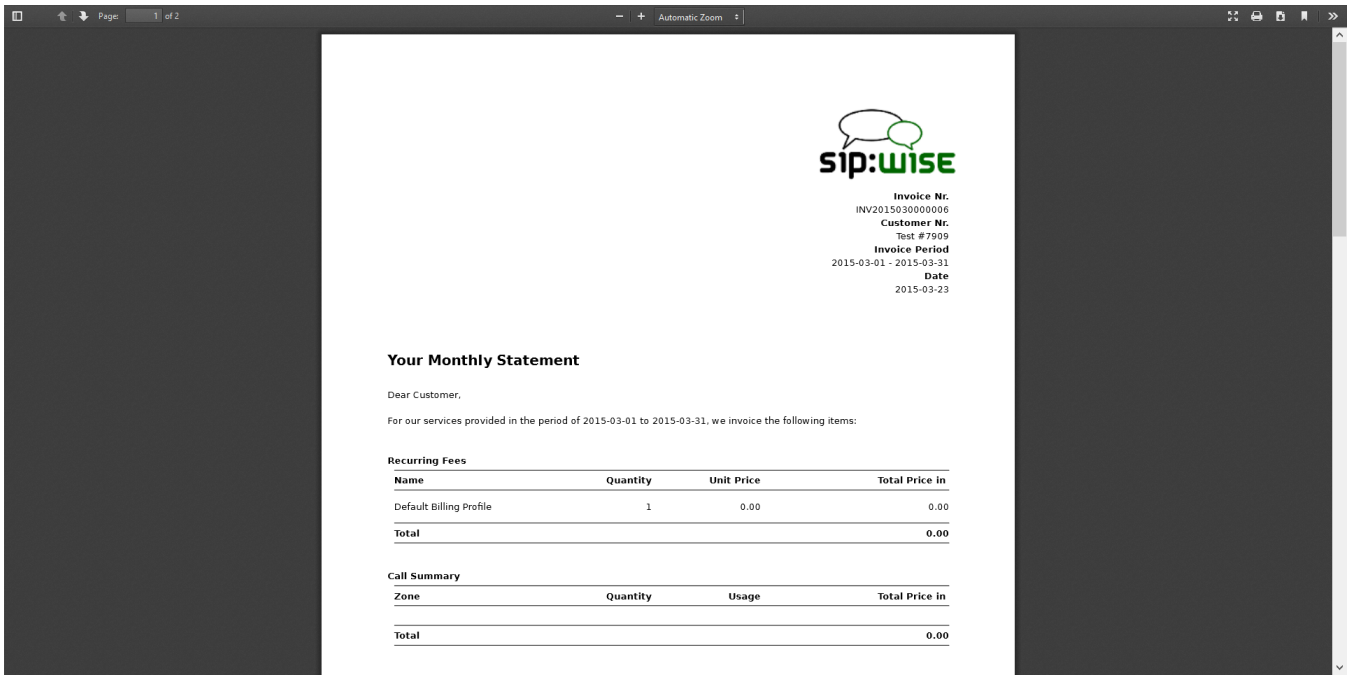
- **Template**: any of existent invoice template can be selected for the invoice generation.
- **Customer**: owner of the billing account, recipient of the invoice.
- **Invoice period**: billing period. Can be specified only as one calendar month. Calls with start time between first and last second of the period will be considered for the invoice

All form fields are mandatory.

 Copyright © Sipwise GmbH - All rights reserved



Generated invoice can be downloaded as pdf file.



To do it press button "Download" against invoice in the invoice management interface.

Respectively press on the button "Delete" to delete invoice.

7.24.2. Invoice Management via REST API

Besides managing invoices on the admin web interface of NGCP, the system administrator (or a third party system) has the opportunity to request generation and retrieval of invoices via the *REST API*.

The subsequent sections describe the available operations for invoice management with API requests in details. All operations work on the *Invoices* resource and use the `/api/invoices` base path. The authentication method is username/password in the examples given below, however it is

recommended to use a TLS client certificate for authentication on the REST API.

NOTE The full API documentation is always available at the location: https://<IP_of_NGCP_web_panel>:1443/api

Generate a New Invoice

The **prerequisite** for generating a new invoice is that the customer has to have **an invoice template** assigned to him.

The following example shows a CURL command that will request generation of an invoice:

- for customer with ID "79"
- for the time period of November 2017
- based on the invoice template with ID "1"

```
curl -i -X POST -H 'Connection: close' -H 'Content-Type:
application/json' \
  --user adminuser:adminpwd -k 'https://127.0.0.1:1443/api/invoices/'
\
  --data-binary '{ "customer_id" : "79", "template_id" : "1",
\
  "period_start": "2017-11-01 00:00:00", "period_end": "2017-11-30
23:59:59"  }'
```

Please note that in this operation we used the `/api/invoices` path (the *invoices* collection) and a *POST* request on it to create a new invoice item.

In case of a **successful operation**, Sipwise C5 will reply with **201 Created** HTTP status and send the ID of the invoice in *Location* header. In our example the new invoice item may be directly referred as `/api/invoices/3` (ID = 3).

```
HTTP/1.1 201 Created
Server: nginx
Date: Tue, 14 Nov 2017 13:38:40 GMT
Content-Length: 0
Connection: close
Location: /api/invoices/3
Set-Cookie: ngcp_panel_session=d5e4a8dd003fd7cac646653a6b5aefa703cf3e66;
path=/; expires=Tue, 14-Nov-2017 14:38:38 GMT; HttpOnly
X-Catalyst: 5.90114
Strict-Transport-Security: max-age=15768000
```

In case of a **failed operation**, e.g. when we request an invoicing period that is invalid for the customer, Sipwise C5 will reply with **422 Unprocessable Entity** or **500 Internal Server Error** HTTP status.

Copyright © Sipwise GmbH - All rights reserved

Download Invoice Data

You can download properties / data of a specific invoice by selecting the item by its ID, using an HTTP *GET* request.

```
curl -i -X GET -H 'Connection: close' --user adminuser:adminpwd -k \
'https://127.0.0.1:1443/api/invoices/3'
```

The above request will return a JSON data structure containing invoice properties:

Copyright © Sipwise GmbH - All rights reserved

```
HTTP/1.1 200 OK
Server: nginx
Date: Wed, 15 Nov 2017 12:13:04 GMT
Content-Type: application/hal+json;
profile="http://purl.org/sipwise/ngcp-api/"; charset=utf-8
Content-Length: 759
Connection: close
Link: </api/invoices/>; rel=collection
Link: <http://purl.org/sipwise/ngcp-api/>; rel=profile
Link: </api/invoices/3>; rel="item self"
Link: </api/invoices/3>; rel="item http://purl.org/sipwise/ngcp-
api/#rel-invoices"
Link: </api/customers/79>; rel="item http://purl.org/sipwise/ngcp-
api/#rel-customers"
Set-Cookie: ngcp_panel_session=219feccbee4fa936defd1ee511c84efe7b5a6d6a;
path=/; expires=Wed, 15-Nov-2017 13:13:03 GMT; HttpOnly
Strict-Transport-Security: max-age=15768000

{
   "_links" : {
      "collection" : {
         "href" : "/api/invoices/"
      },
      "curies" : {
         "href" : "http://purl.org/sipwise/ngcp-api/#rel-{rel}",
         "name" : "ngcp",
         "templated" : true
      },
      "ngcp:customers" : {
         "href" : "/api/customers/79"
      },
      "ngcp:invoices" : {
         "href" : "/api/invoices/3"
      },
      "profile" : {
         "href" : "http://purl.org/sipwise/ngcp-api/"
      },
      "self" : {
         "href" : "/api/invoices/3"
      }
   },
   "amount_net" : 0,
   "amount_total" : 0,
   "amount_vat" : 0,
   "id" : 3,
   "period_end" : "2017-11-30T23:59:59+00:00",
   "period_start" : "2017-11-01T00:00:00+00:00",
   "sent_date" : null,
   "serial" : "INV2017110000003"
}
```


Copyright © Sipwise GmbH - All rights reserved

It is also possible to query the complete *invoices* collection and use a filter (e.g. invoicing period, customer ID, etc.) to get the desired invoice item. In the example below we request all available invoices that belong to the customer with ID "79".

```
curl -i -X GET -H 'Connection: close' --user adminuser:adminpwd -k \
'https://127.0.0.1:1443/api/invoices/?customer_id=79'
```

The returned dataset is now slightly different because it is represented as an array of items, although in our example the array consist of only 1 item:

```
{
   "_embedded" : {
      "ngcp:invoices" : [
         {
            "_links" : {
               "collection" : {
                  "href" : "/api/invoices/"
               },
               "curies" : {
                  "href" : "http://purl.org/sipwise/ngcp-api/#rel-
{rel}",
                  "name" : "ngcp",
                  "templated" : true
               },
               "ngcp:customers" : {
                  "href" : "/api/customers/79"
               },
               "ngcp:invoices" : {
                  "href" : "/api/invoices/3"
               },
               "profile" : {
                  "href" : "http://purl.org/sipwise/ngcp-api/"
               },
               "self" : {
                  "href" : "/api/invoices/3"
               }
            },
            "amount_net" : 0,
            "amount_total" : 0,
            "amount_vat" : 0,
            "id" : 3,
            "period_end" : "2017-11-30T23:59:59+00:00",
            "period_start" : "2017-11-01T00:00:00+00:00",
            "sent_date" : null,
            "serial" : "INV2017110000003"
         }
      ]
   },
   "_links" : {
      "curies" : {
```

Copyright © Sipwise GmbH - All rights reserved

```
            "href" : "http://purl.org/sipwise/ngcp-api/#rel-{rel}",
            "name" : "ngcp",
            "templated" : true
         },
         "ngcp:invoices" : {
            "href" : "/api/invoices/3"
         },
         "profile" : {
            "href" : "http://purl.org/sipwise/ngcp-api/"
         },
         "self" : {
            "href" : "/api/invoices/?page=1&rows=10"
         }
      },
      "total_count" : 1
}
```

Download Invoice as PDF File

You can download a specific invoice as a PDF file in the following way:

- selecting the item by its ID (as in our example, but you can also use a filter and query the complete *invoices* collection)
- using an HTTP *GET* request
- adding **"Accept: application/pdf" header** to the request

```
curl -X GET -H 'Connection: close' -H 'Accept: application/pdf' \
  --user adminuser:adminpwd -k 'https://127.0.0.1:1443/api/invoices/3' >
result.pdf
```

Please note that in the example above we **do not add the "-i" option** that would also include the headers of the HTTP response in the output file. The output of the CURL command, i.e. the PDF file, is saved as "result.pdf" locally.

Delete an Invoice

In order to delete an invoice item you have to send a *DELETE* request on the specific item:

```
curl -i -X DELETE -H 'Connection: close' --user adminuser:adminpwd -k \
'https://127.0.0.1:1443/api/invoices/3'
```

In case of successful deletion Sipwise C5 should send HTTP status `204 No Content` as a response:

Copyright © Sipwise GmbH - All rights reserved

```
HTTP/1.1 204 No Content
Server: nginx
Date: Wed, 15 Nov 2017 13:42:42 GMT
Connection: close
Set-Cookie: ngcp_panel_session=10b66a6baf25a09739c2bb2377c70ecceee78387;
path=/; expires=Wed, 15-Nov-2017 14:42:42 GMT; HttpOnly
X-Catalyst: 5.90114
Strict-Transport-Security: max-age=15768000
```

7.24.3. Invoice Templates

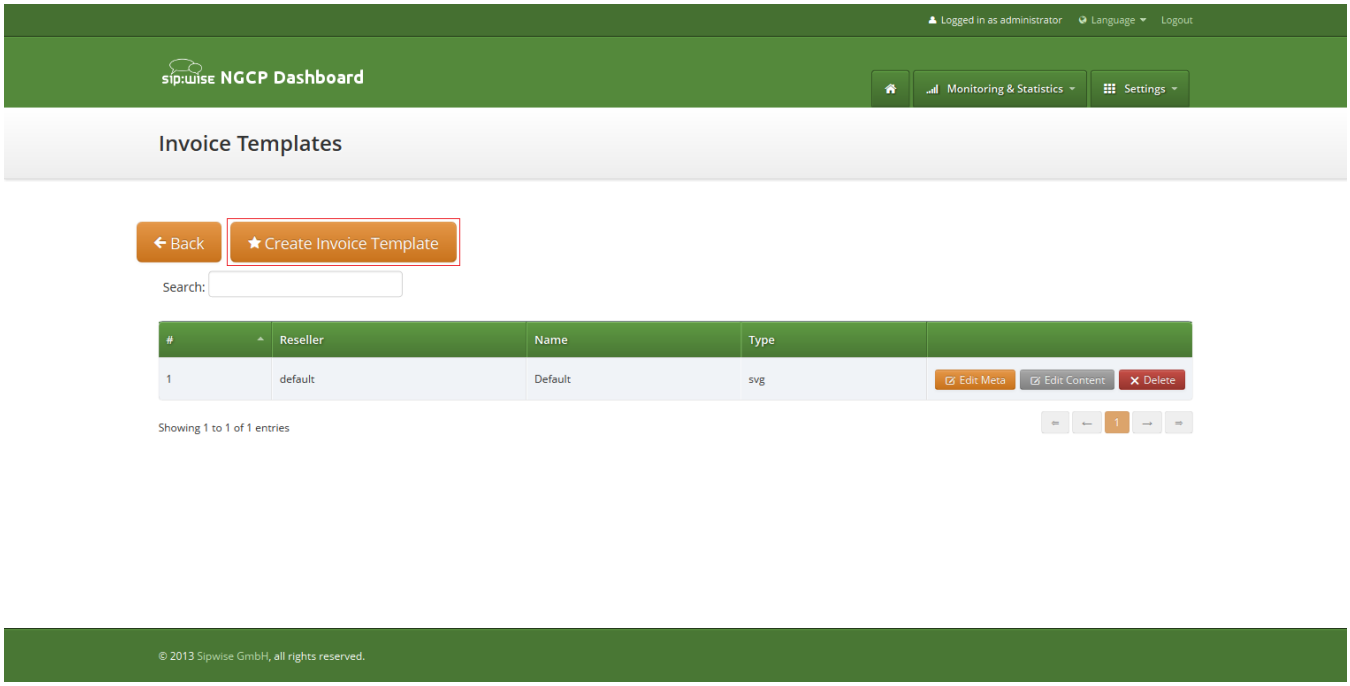
Invoice template defines structure and look of the generated invoices. The Sipwise C5 makes it possible to create some invoice templates. Multiple invoice templates can be used to send invoices to the different customers using different languages.

IMPORTANT At least one invoice template should be created to enable invoice generation. Each customer has to be associated to one of the existent invoice template, otherwise invoices will be not generated for this customer.

Customer can be linked to the invoice template in the customer interface.

Invoice Templates Management

Invoice templates can be searched, created, edited and deleted in the invoice templates management interface.



Invoice template creation is separated on two steps:

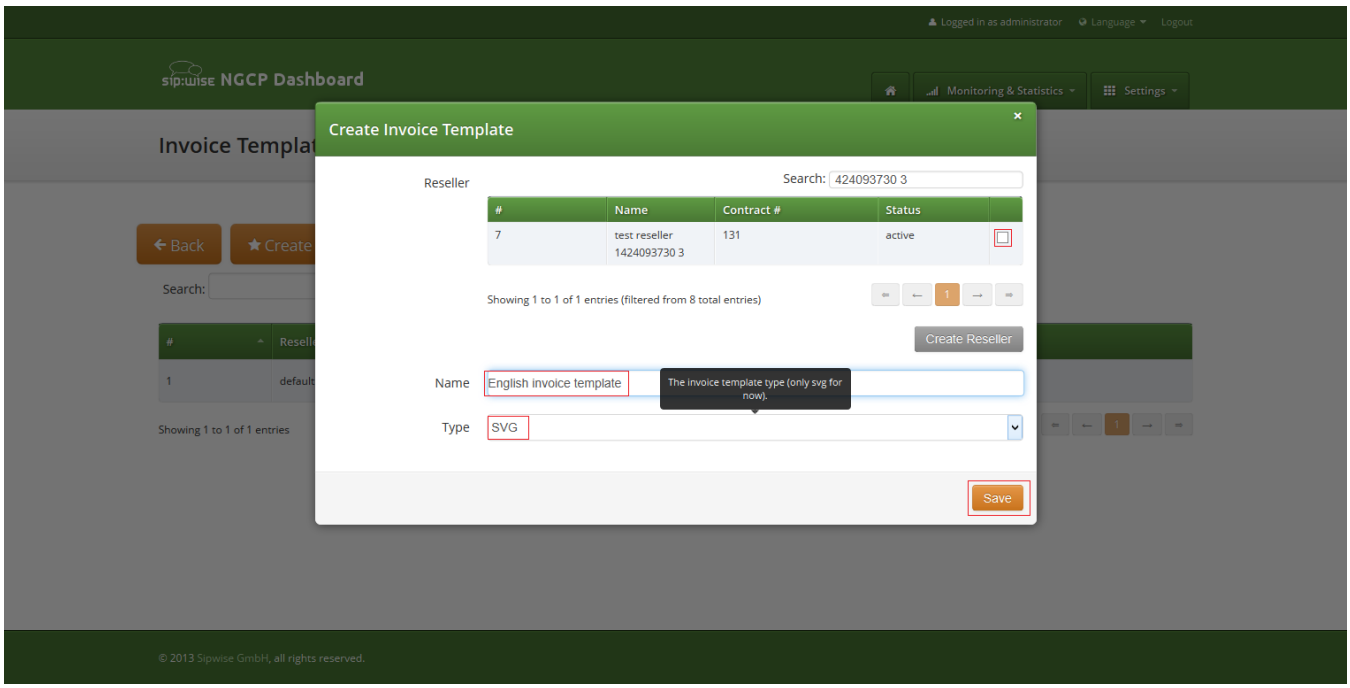
- Register new invoice template meta information.
- Edit content (template itself) of the invoice template.

To register new invoice template press "Create Invoice Template" button.

On the invoice template meta information form following parameters can be specified:

- **Reseller**: reseller who owns this invoice template. Please note, that it doesn't mean that the template will be used for the reseller customers by default. After creation, invoice template still need to be linked to the reseller customers.
- **Name**: unique invoice template name to differentiate invoice templates if there are some.
- **Type**: currently Sipwise C5 supports only svg format of the invoice templates.

All form fields are mandatory.



After registering new invoice template you can change invoice template structure in WYSIWYG SVG editor and preview result of the invoice generation based on the template.

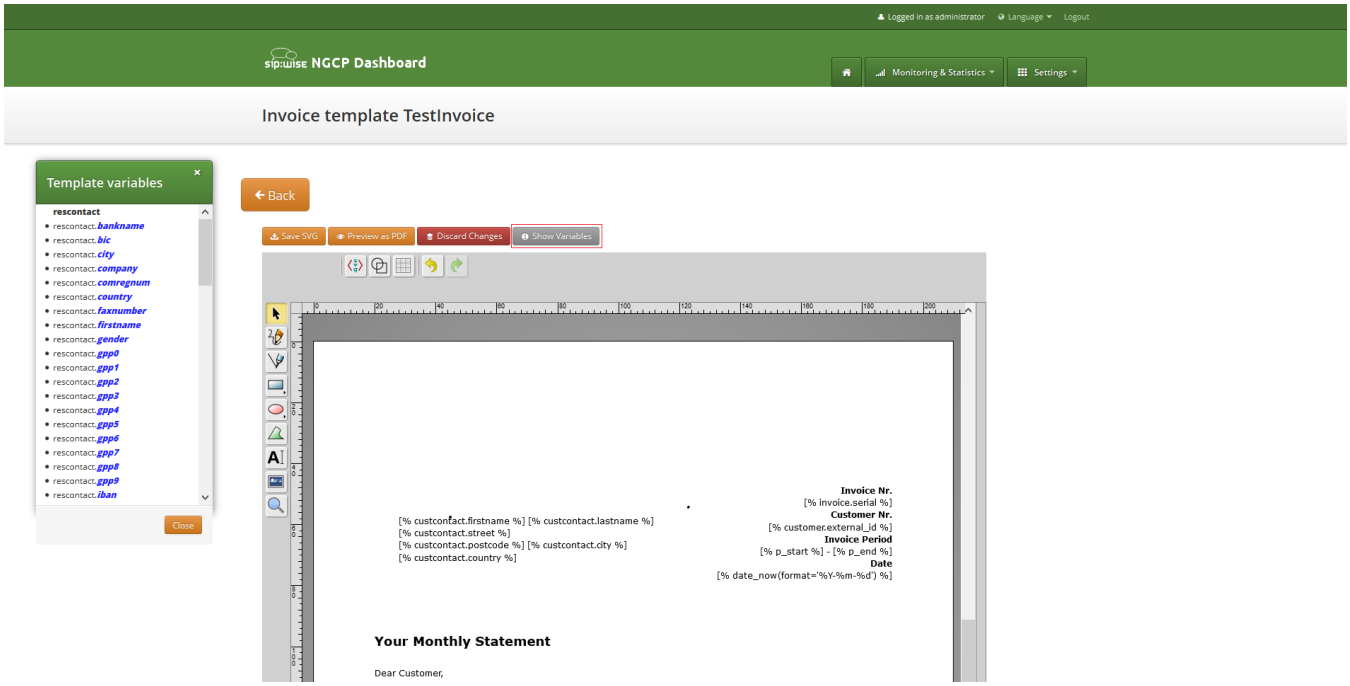
Invoice Template Content

Invoice template is a XML SVG source, which describes content, look and position of the text lines, images or other invoice template elements. The Sipwise C5 provides embedded WYSIWYG SVG editor svg-edit 2.6 to customize default template. The Sipwise C5 svg-edit has some changes in layers management, image edit, user interface, but this basic introduction still may be useful.

Template refers to the owner reseller contact ("rescontact"), customer contract ("customer"), customer contact ("custcontact"), billing profile ("billprof"), invoice ("invoice") data as variables in the "[%%]" mark-up with detailed information accessed as field name after point e.g. [%invoice.serial%]. During invoice generation all variables or other special tokens in the "[% %]" mark-ups will be replaced by their database values.

Press on "Show variables" button on invoice template content page to see full list of variables with the fields:

 Copyright © Sipwise GmbH - All rights reserved



You can add/change/remove embedded variables references directly in main svg-edit window. To edit text line in svg-edit main window double click on the text and place cursor on desired position in the text.

After implementation of the desired template changes, invoice template should be saved.

To return to Sipwise C5 invoice template **default** content you can press on the "Discard changes" button.

IMPORTANT "Discard changes" operation can't be undone.

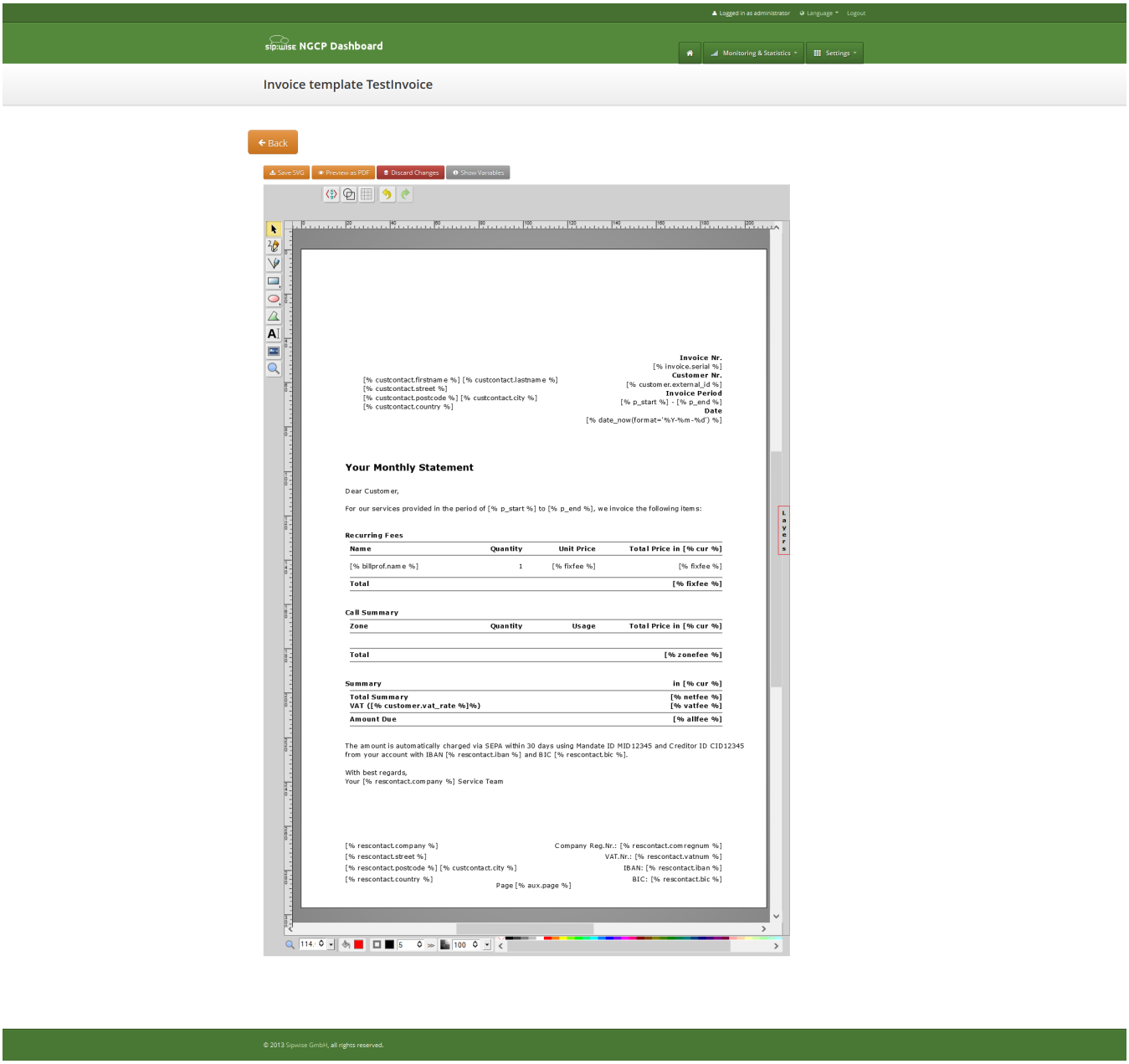
Layers

Default template contains three groups elements (<g/>), which can be thought of as pages, or in terms of svg-edit - layers. Layers are:

- **Background**: special layer, which will be repeated as background for every other page of the invoice.
- **Summary**: page with a invoice summary.
- **CallList**: page with calls made in a invoice period. Is invisible by default.

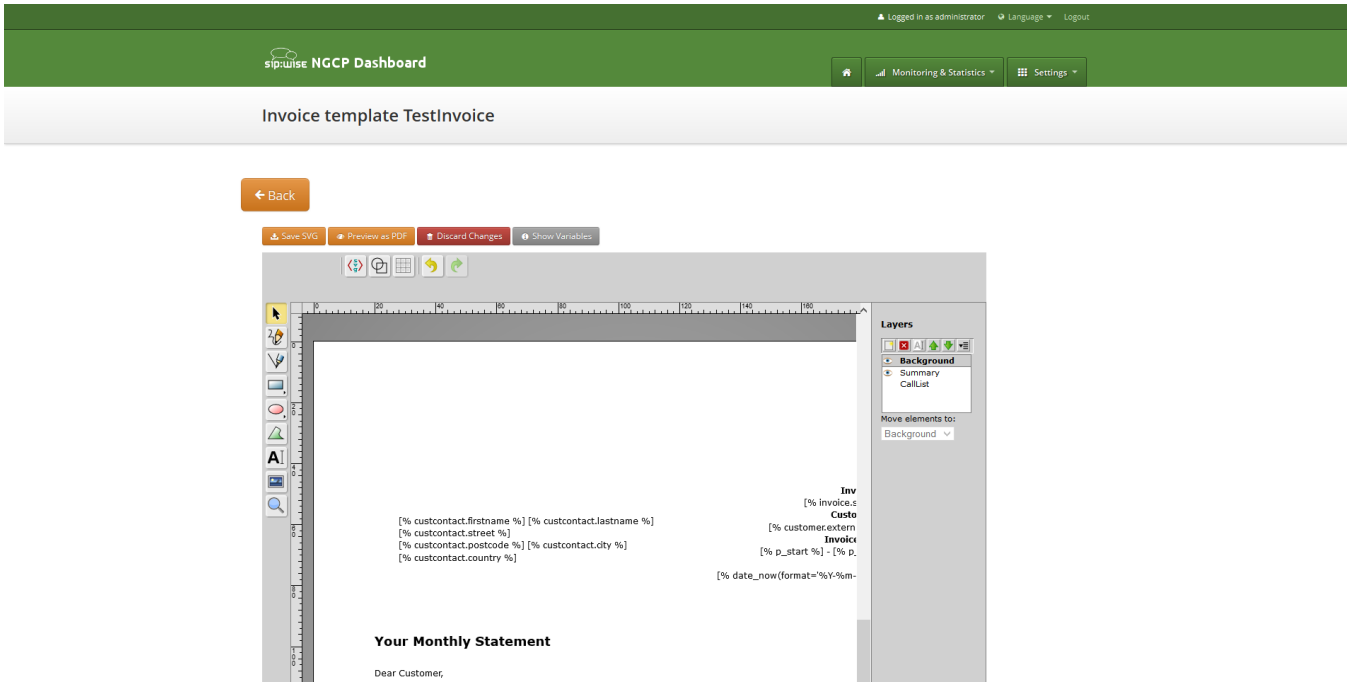
To see all invoice template layers, press on "Layers" vertical sign on right side of the svg-edit interface:

Copyright © Sipwise GmbH - All rights reserved



Side panel with layers list will be shown.

 Copyright © Sipwise GmbH - All rights reserved

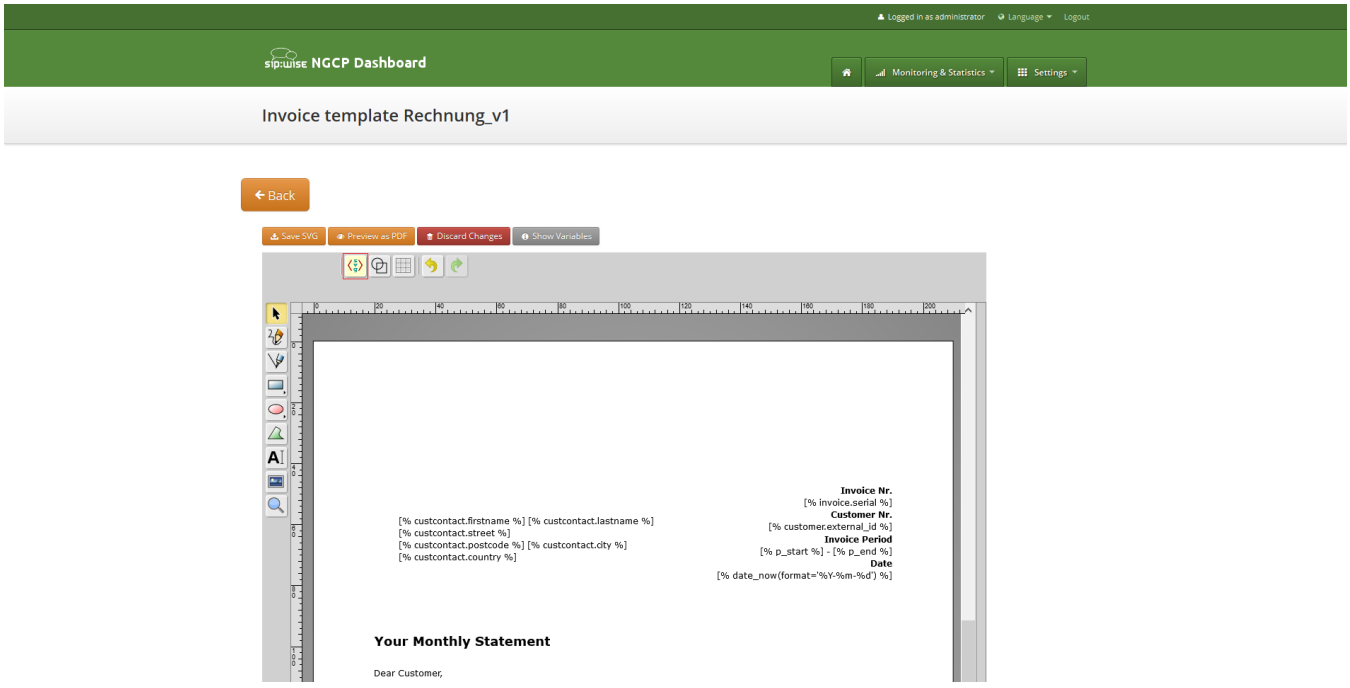


One of the layers is active, and its element can be edited in the main svg-edit window. Currently active layer's name is **bold** in the layers list. The layers may be visible or invisible. Visible layers have "eye" icon left of their names in the layers list.

To make a layer active, click on its name in the layers list. If the layer was invisible, its elements became visible on activation. Thus you can see mixed elements of some layers, then you can switch off visibility of other layers by click on their "eye" icons. It is good idea to keep visibility of the "Background" layer on, so look of the generated page will be seen.

Edit SVG XML source

Sometimes it may be convenient to edit svg source directly and svg-edit makes it possible to do it. After press on the <svg> icon in the top left corner of the svg-edit interface:



Copyright © Sipwise GmbH - All rights reserved

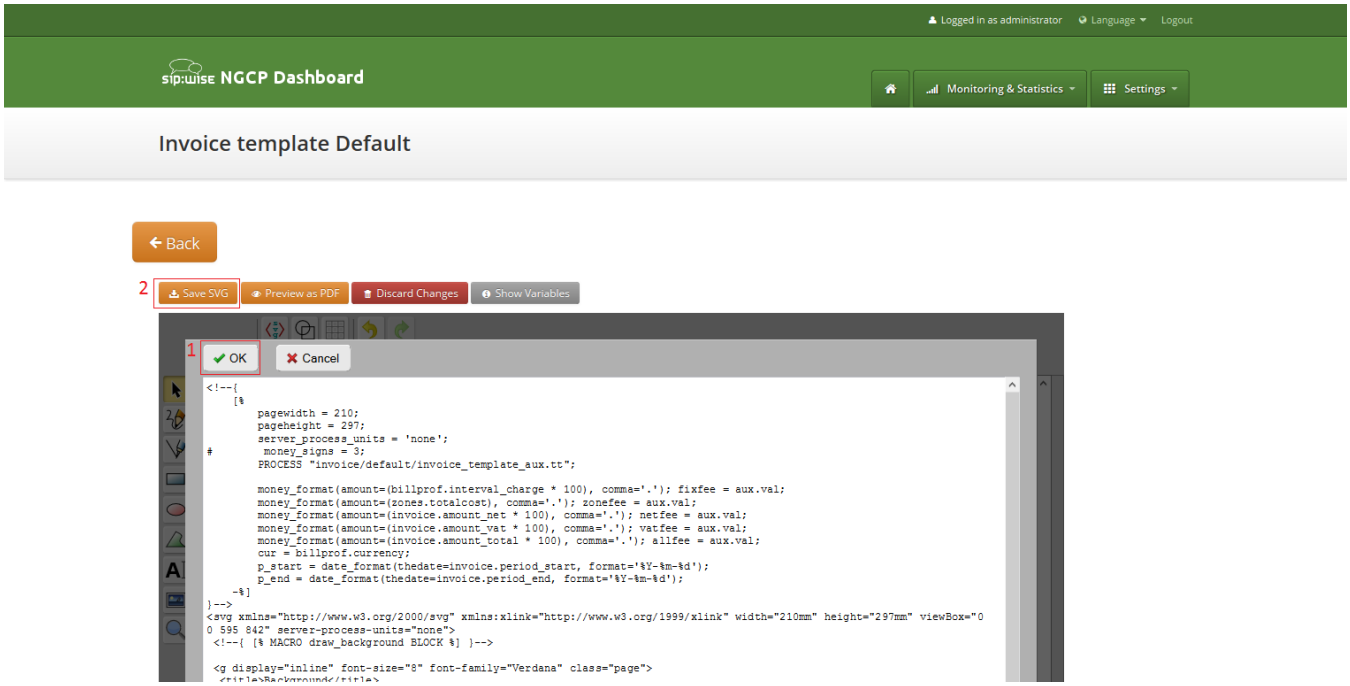
SVG XML source of the invoice template will be shown.

SVG source can be edited in place or copy-pasted as usual text.

NOTE Template keeps sizes and distances in pixels.

IMPORTANT When editing svg xml source, please change things inside special comment mark-up "<!--{ }-" very carefully. Otherwise invoice generation may be broken. Please be sure that document structure repeats default invoice template: has the same groups (<g/>) elements on the top level, text inside special comments mark-up "<!--{ }-" preserved or changed appropriately, svg xml structure is correct.

To save your changes in the svg xml source, first press "OK" button on the top left corner of the source page:



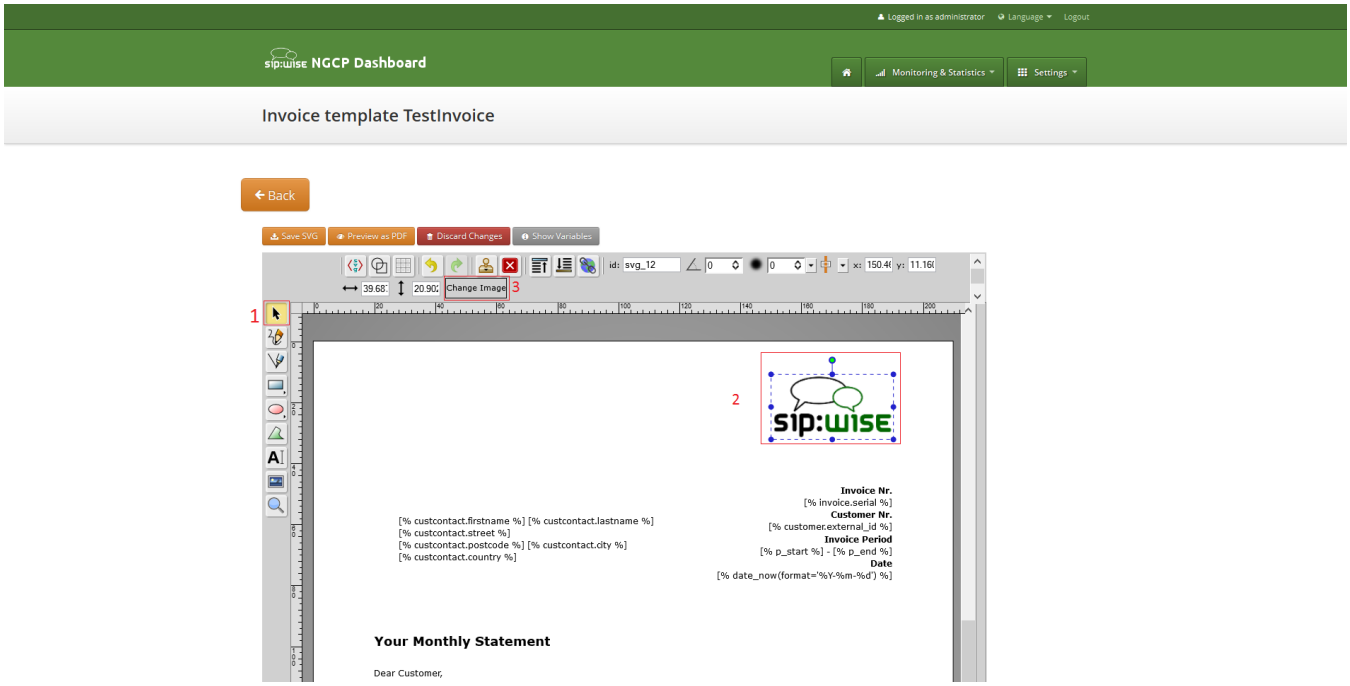
And then save invoice template changes.

NOTE You can copy and keep the svg source of your template as a file on the disk before start experimenting with the template. Later you will be able to return to this version replacing svg source.

Change logo image

- Make sure that "Select tool" is active.
- Select default logo, clicking on the logo image.
- Press "Change image" button, which should appear on the top toolbar.

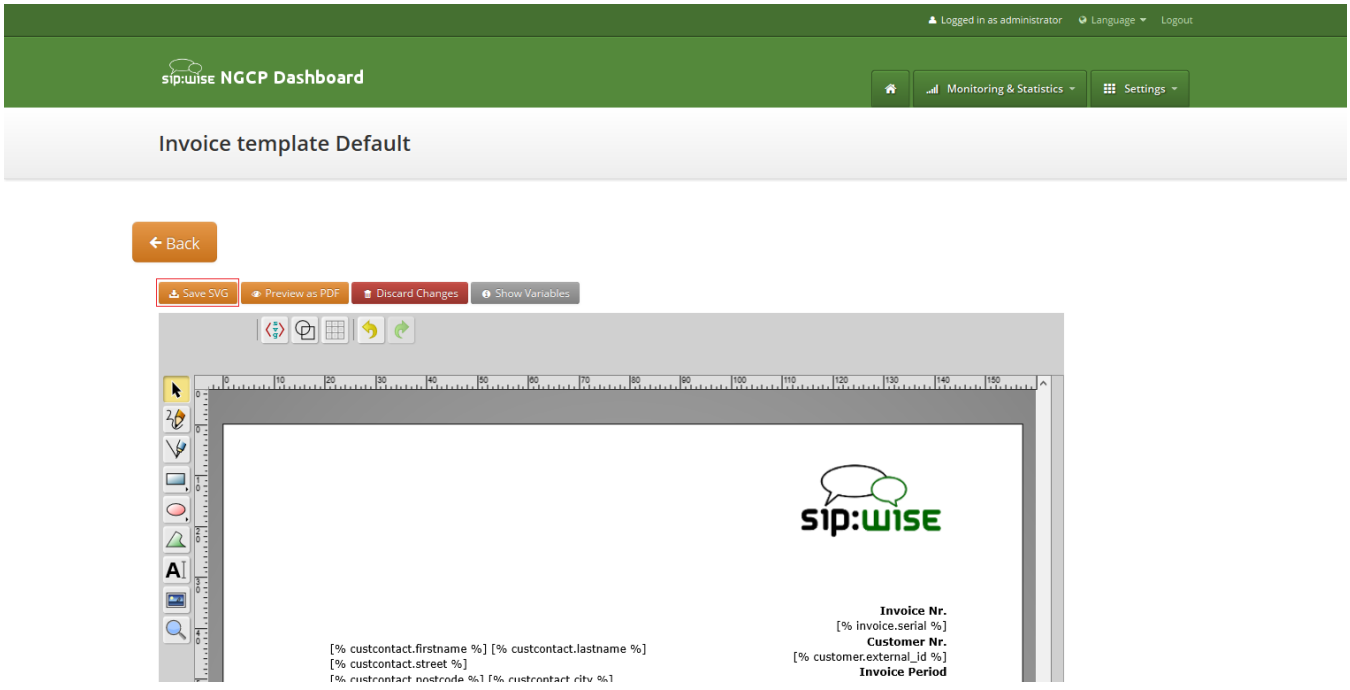
Copyright © Sipwise GmbH - All rights reserved



After image uploaded save invoice template changes.

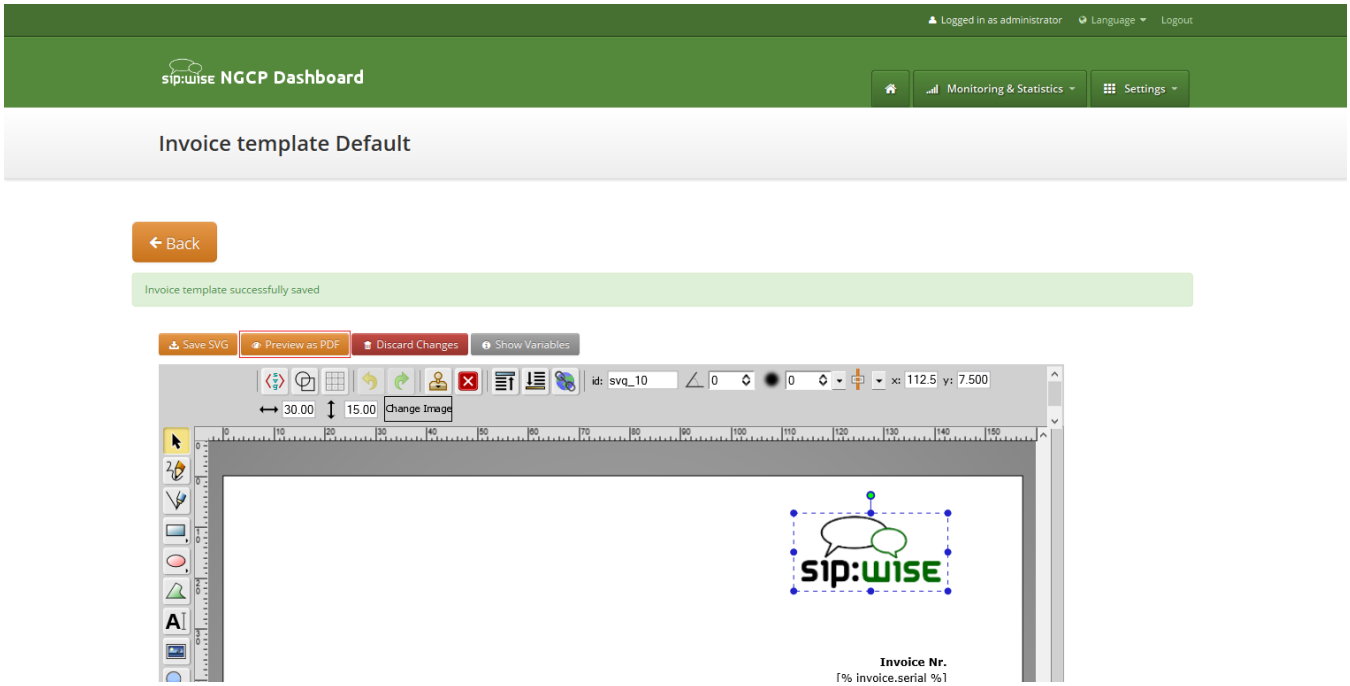
Save and preview invoice template content

To save invoice template content changes press button "Save SVG".



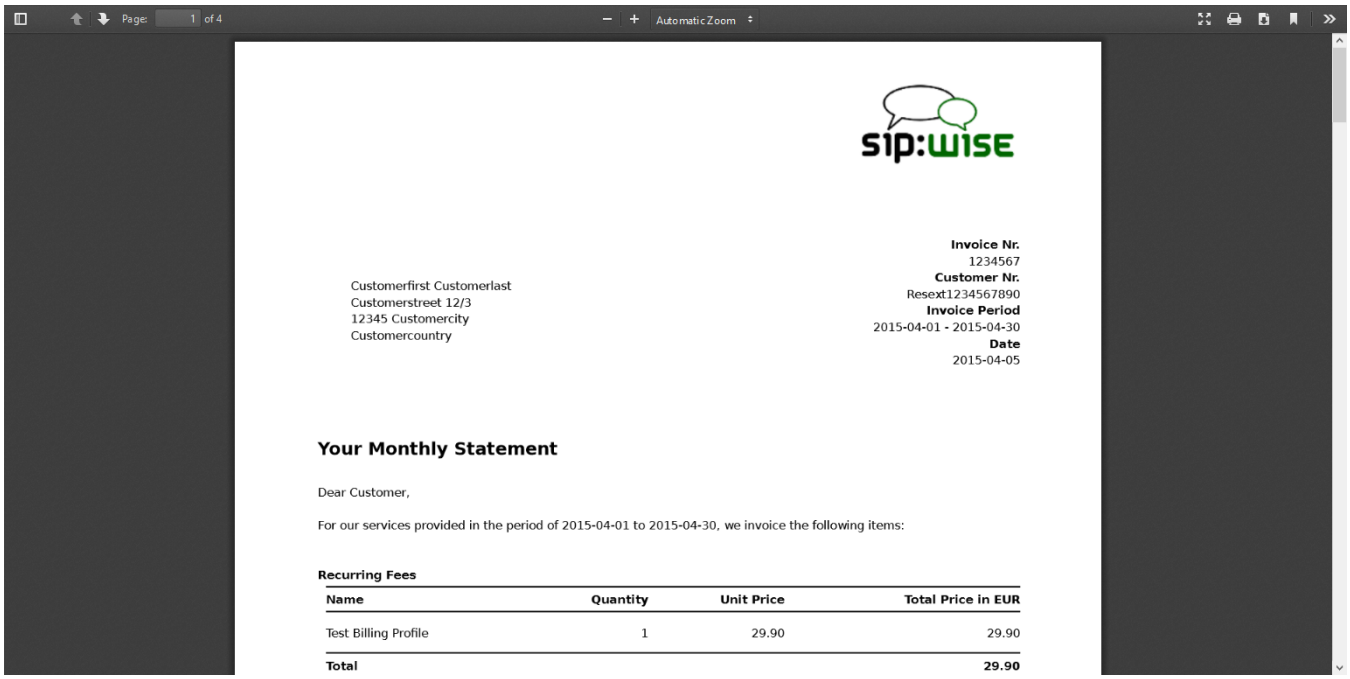
You will see message about successfully saved template. You can preview your invoice look in PDF format. Press on "Preview as PDF" button.

Copyright © Sipwise GmbH - All rights reserved



Invoice preview will be opened in the new window.

NOTE Example fake data will be used for preview generation.



 Copyright © Sipwise GmbH - All rights reserved

7.25. Email Reports and Notifications

7.25.1. Email events

The Sipwise C5 makes it possible to customize content of the emails sent on the following actions:

- Web password reset requested. Email will be sent to the subscriber, whom password was requested for resetting. If the subscriber doesn't have own email, letter will be sent to the customer, who owns the subscriber.
- Administrator password reset requested. Email will be sent to the administrator, whom password was requested for resetting. If the administrator doesn't have an email, an error message will appear when requesting the password reset.
- New subscriber created. Email will be sent to the newly created subscriber or to the customer, who owns new subscriber.
- Letter with the invoice. Letter will be sent to the customer.

7.25.2. Initial template values and template variables

Default email templates for each of the email events are inserted on the initial Sipwise C5 database creation. Content of the default template is described in the corresponding sections. Default email templates aren't linked to any reseller and can't be changed through Sipwise C5 Panel. They will be used to initialize default templates for the newly created reseller.

Each email template refers to the values from the database using special mark-ups "[%" and "%]". Each email template has fixed set of the variables. Variables can't be added or changed without changes in Sipwise C5 Panel code.

7.25.3. Subscriber password reset email template

Email will be sent after subscriber or subscriber administrator requested password reset for the subscriber account. Letter will be sent to the subscriber. If subscriber doesn't have own email, letter will be sent to the customer owning the subscriber.

Default content of the password reset email template is:

Template name	**passreset_default_email**
From	default@sipwise.com
Subject	Password reset email

Body	`Dear Customer,`
	
	`Please go to [%url%] to set your password and log into your self-care interface.`
	
	`Your faithful Sipwise system`
	
	`--`
	`This is an automatically generated message. Do not reply.`

Copyright © Sipwise GmbH - All rights reserved

Following variables will be provided to the email template:

- [%**url**%]: specially generated url where subscriber can define his new password.
- [%**subscriber**%]: username@domain of the subscriber, which password was requested for reset.

7.25.4. Administrator password reset email template

Email will be sent after administrator requested password reset for it's account. Letter will be sent to the administrator. If the administrator doesn't have an email, an error message will appear when requesting the password reset.

Default content of the password reset email template is:

Template name	**admin_passreset_default_email**
From	default@sipwise.com
Subject	Password reset email

Body	`Dear Customer,`
	
	`Please go to [%url%] to set your password and log into your admin interface.`
	
	`Your faithful Sipwise system`
	
	`--`
	`This is an automatically generated message. Do not reply.`

Following variables will be provided to the email template:

- [%**url**%]: specially generated url where administrator can define his new password.
- [%**admin**%]: username@domain of the administrator, which password was requested for reset.

7.25.5. New subscriber notification email template

Email will be sent on the new subscriber creation. Letter will be sent to the newly created subscriber if it has an email. Otherwise, letter will be sent to the customer who owns the subscriber.

NOTE By default email content template is addressed to the customer. Please consider this when create the subscriber with an email.

Template name	**subscriber_default_email**
From	default@sipwise.com
Subject	Subscriber created

Copyright © Sipwise GmbH - All rights reserved

Body	Dear Customer, A new subscriber [%subscriber%] has been created for you. Your faithful Sipwise system -- This is an automatically generated message. Do not reply.

Following variables will be provided to the email template:

- [%**url**%]: specially generated url where subscriber can define his new password.
- [%**subscriber**%]: username@domain of the subscriber, which password was requested for reset.

7.25.6. Invoice email template

Template name	**invoice_default_email**
From	default@sipwise.com
Subject	Invoice #[%invoice.serial%] from [%invoice.period_start_obj.ymd%] to [%invoice.period_end_obj.ymd%]
Body	Dear Customer, Please find your invoice #[%invoice.serial%] for [%invoice.period_start_obj.month_name%], [%invoice.period_start_obj.year%] in attachment of this letter. Your faithful Sipwise system -- This is an automatically generated message. Do not reply.

Variables passed to the email template:

- [%**invoice**%]: container variable for the invoice information.

Copyright © Sipwise GmbH - All rights reserved

NOTE

Invoice fields

- [%invoice.**serial**%]
- [%invoice.**amount_net**%]
- [%invoice.**amount_vat**%]
- [%invoice.**amount_total**%]
- [%invoice.**period_start_obj**%]
- [%invoice.**period_end_obj**%]

The fields [%invoice.period_start_obj%] and [%invoice.period_end_obj%] provide methods of the perl package DateTime for the invoice start date and end date. Further information about DateTime can be obtained from the package documentation:

man DateTime

- [%**provider**%]: container variable for the reseller contact. All database contact values will be available.
- [%**client**%]: container variable for the customer contact.

Copyright © Sipwise GmbH - All rights reserved

Contact fields example for the "provider". Replace "provider" to client to access proper "customer" contact fields.

- [%provider.**gender**%]
- [%provider.**firstname**%]
- [%provider.**lastname**%]
- [%provider.**comregnum**%]
- [%provider.**company**%]
- [%provider.**street**%]
- [%provider.**postcode**%]
- [%provider.**city**%]

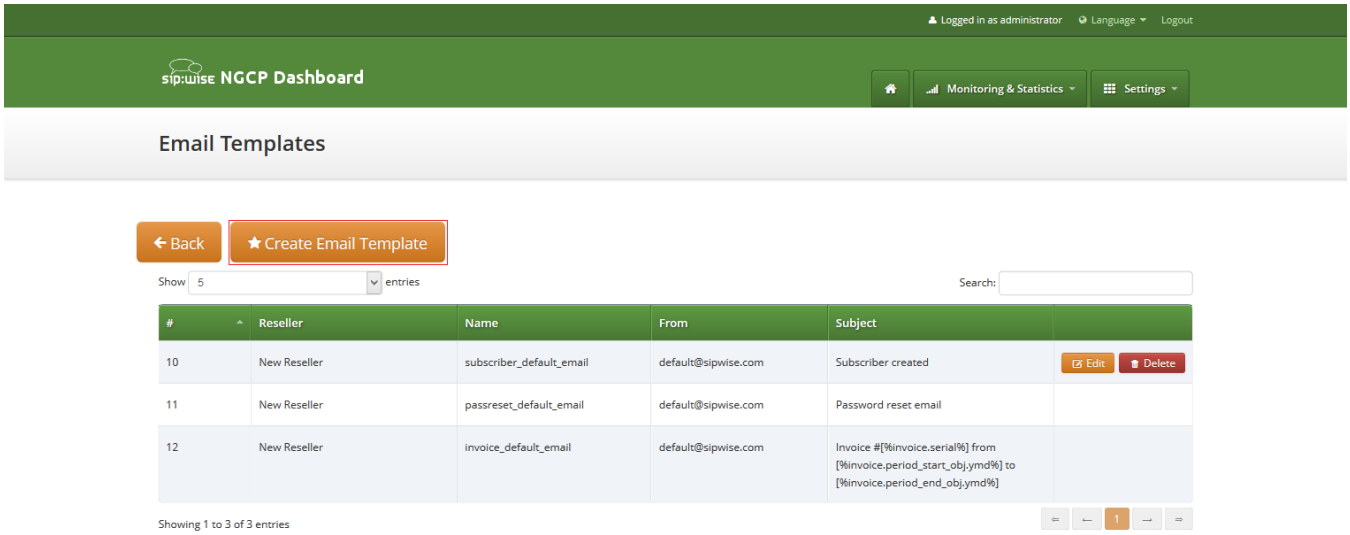
NOTE

- [%provider.**country**%]
- [%provider.**phonenumber**%]
- [%provider.**mobilenumber**%]
- [%provider.**email**%]
- [%provider.**newsletter**%]
- [%provider.**faxnumber**%]
- [%provider.**iban**%]
- [%provider.**bic**%]
- [%provider.**vatnum**%]
- [%provider.**bankname**%]
- [%provider.**gpp0** - provider.**gpp9**%]

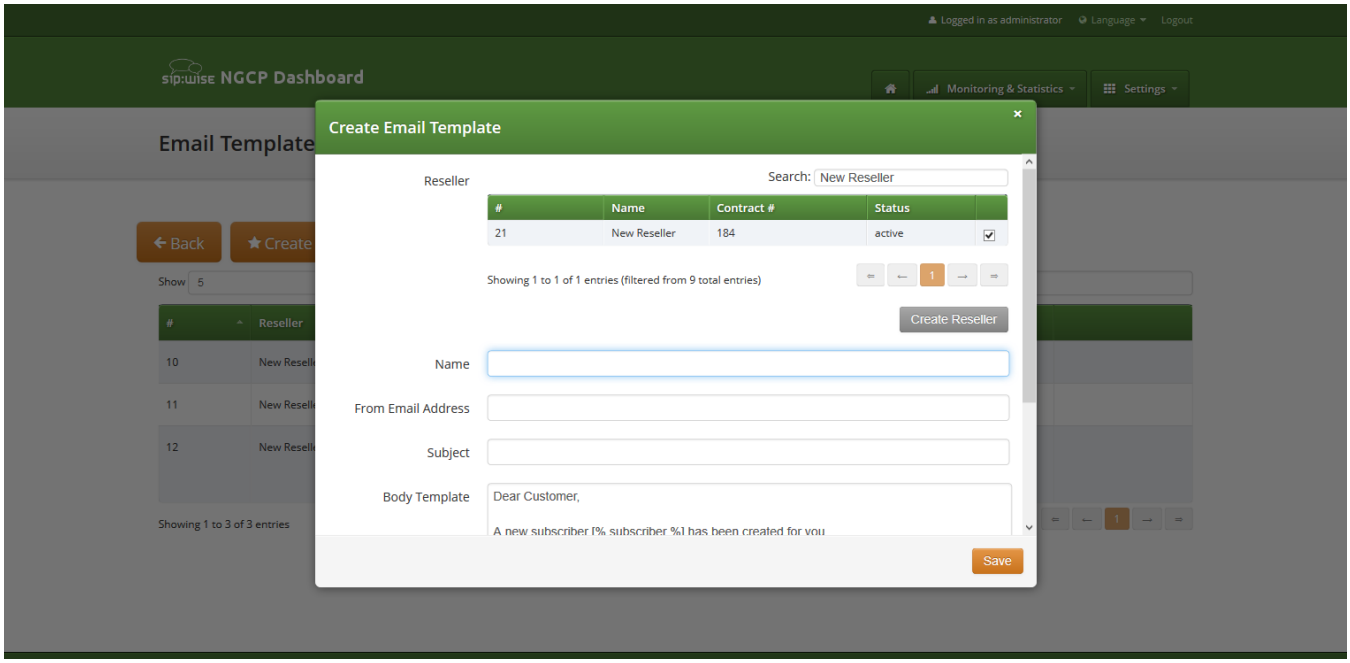
7.25.7. Email templates management

Email templates linked to the resellers can be customized in the email templates management interface. For the administrative account email templates of all the resellers will be shown. Respectively for the reseller account only owned email templates will be shown.

Copyright © Sipwise GmbH - All rights reserved



To create new email template press button "Create Email Template".



On the email template form all fields are mandatory:

- **Reseller**: reseller who owns this email template.
- **Name**: currently only email template with the following names will be considered by Sipwise C5 on the appropriate event :

  admin_passreset_default_email;

  passreset_default_email;

  subscriber_default_email;

  invoice_default_email;
- **From Email Address**: email address which will be used in the From field in the letter sent by Sipwise

 Copyright © Sipwise GmbH - All rights reserved

C5 .

- **Subject**: Template of the email subject. Subject will be processed with the same template variables as the email body.
- **Body**: Email text template. Will be processed with appropriate template variables.

Copyright © Sipwise GmbH - All rights reserved

7.26. The Vertical Service Code Interface

Vertical Service Codes (VSC) are codes a user can dial on his phone to provision specific features for his subscriber account. The format is **<code>**<value> to activate a specific feature, and <code> or <code># to deactivate it. The *code* parameter is a two-digit code, e.g. 72. The *value* parameter is the value being set for the corresponding feature.

IMPORTANT The *value* user input is normalized using the Rewrite Rules Sets assigned to domain as described in Configuring Rewrite Rule Sets.

By default, the following codes are configured for setting features. The examples below assume that there is a domain rewrite rule normalizing the number format *0<ac><sn>* to *<cc><ac><sn>* using *43* as country code.

- **72** - enable *Call Forward Unconditional* e.g. to 431000 by dialing *72*01000, and disable it by dialing #72.
- **90** - enable *Call Forward on Busy* e.g. to 431000 by dialing *90*01000, and disable it by dialing #90.
- **92** - enable *Call Forward on Timeout* e.g. after 30 seconds of ringing to 431000 by dialing *92*30*01000, and disable it by dialing #92.
- **93** - enable *Call Forward on Not Available* e.g. to 431000 by dialing *93*01000, and disable it by dialing #93.
- **96** - disable at once all the *Call Forwards* previously configured using VSC, e.g. disable all the CFs by dialing #96.
- **50** - set *Speed Dial Slot*, e.g. set slot 1 to 431000 by dialing *50*101000, which then can be used by dialing *1. There is no code to disable a speed dial slot. When a slot is no longer necessary, it can be ultimately removed using the web interface or can be ignored, because it is not impacting the calls from and to this subscriber.
- **55** - set *One-Shot Reminder Call* e.g. to 08:30 by dialing *55*0830.
- **31** - enable/disable *Calling Line Identification Restriction* for one call, e.g. to call 431000 anonymously make sure to have subscriber's caller preferences 'clir' disabled and dial *31*01000. If 'clir' preference is enable, this will disable *Calling Line Identification Restriction* for this particular call.
- **32** - enable *Block Incoming Anonymous Calls* by dialing **32**, and disable it by dialing #32.
- **80** - call using *Call Block Override PIN*, number should be prefixed with a block override PIN configured in admin panel to disable the outgoing user/admin block list and NCOS level for a call. For example, when override PIN is set to 7890, dial *80*789001000 to call 431000 bypassing block lists.
- **95** - allow to redial the last dialed number by the subscriber. Note: the feature has to be enabled for the subscriber/domain using preference *last_number_redial*.
- **71** - allow to hear a voice announcement of the last caller's number, after the announcement dial the key defined in *semsvsccallback_last_caller_confirmation_key* to return the call. Note: it is not possible return a call if the caller's number is unavailable.
- **74** - allow to return the call to the last caller's number who called you without hearing the last call ID announcement. Note: it is not possible return a call if the caller's number is unavailable.
- **20** - allow to remove the records of the recent calls to and from you.

Copyright © Sipwise GmbH - All rights reserved

IMPORTANT

In order to use the feature codes related to recent calls (**95**, **71**, **74**, **20**) the *kamailioproxystore_recentcalls* preference in */etc/ngcp-config/config.yml* has to be set to `yes`. Additionally, to handle caller numbers coming from Peers (e.g. PSTN), set the *kamailioproxyforeign_domain_via_peer* preference to `yes` in */etc/ngcp-config/config.yml* file.

7.26.1. Vertical Service Codes for PBX customers

Subscribers under the same PBX customer can enjoy some PBX-specific features by means of special VSCs.

Sipwise C5 provides the following PBX-specific VSCs:

- **97** - *Call Parking:* during a conversation the subscriber can park the call with his phone to a "parking slot" and later on continue the conversation from another phone. To do that, a destination must be dialled as follows: `*97*3`; this will park the call to slot no. 3.

  PLEASE NOTE:

  Cisco IP phones provide a softkey for Call Parking, that means the subscriber must only dial the parking slot number after pressing "Park" softkey on the phone.

  Other IP phones can perform Call Parking as a *blind transfer*, where the destination of the transfer must be dialled in the format described above.

  Both the caller and the callee can park the call.
- **98** - *Call Unparking:* if a call has been parked, a subscriber may continue the conversation from any extension (phone) under the same PBX customer. To do that, the subscriber must dial the following sequence: `*98*3`; this will pick up the call that was parked at slot no. 3.
- **99** - *Directed Call Pickup:* if a subscriber's phone is ringing (e.g. extension 23) and another subscriber wants to answer the call instead of the original callee, he may pick up the call by dialling `*99*23` on his phone.

7.26.2. Configuration of Vertical Service Codes

You can change any of the codes (but not the format) in */etc/ngcp-config/config.yml* in the section *semsvsc*. After the changes, execute *ngcpcfg apply "changed VSC codes"*.

CAUTION

If you have the EMTAs under your control, make sure that the specified VSCs don't overlap with EMTA-internal VSCs, because the VSC calls must be sent to Sipwise C5 via SIP like normal telephone calls.

7.26.3. Voice Prompts for Vertical Service Code Configuration

Table 21. VSC Voice Prompts

Prompt Handle	Related VSC	Message
vsc_error	any	An error has occurred. Please try again later.
vsc_invalid	wrong code	Invalid feature code.

Copyright © Sipwise GmbH - All rights reserved

Prompt Handle	Related VSC	Message
reject_vsc	any	Vertical service codes are disabled for this line.
vsc_cfu_on	72 (Call Forward Unconditional)	Your unconditional call forward has successfully been activated.
vsc_cfu_off	72 (Call Forward Unconditional)	Your unconditional call forward has successfully been deactivated.
vsc_cfb_on	90 (Call Forward Busy)	Your call forward on busy has successfully been activated.
vsc_cfb_off	90 (Call Forward Busy)	Your call forward on busy has successfully been deactivated.
vsc_cft_on	92 (Call Forward on Timeout)	Your call forward on ring timeout has successfully been activated.
vsc_cft_off	92 (Call Forward on Timeout)	Your call forward on ring timeout has successfully been deactivated.
vsc_cfna_on	93 (Call Forward on Not Available)	Your call forward while not reachable has successfully been activated.
vsc_cfna_off	93 (Call Forward on Not Available)	Your call forward while not reachable has successfully been deactivated.
vsc_speeddial	50 (Speed Dial Slot)	Your speed dial slot has successfully been stored.
vsc_reminder_on	55 (One-Shot Reminder Call)	Your reminder has successfully been activated.
vsc_reminder_off	55 (One-Shot Reminder Call)	Your reminder has successfully been deactivated.
vsc_blockinclir_on	32 (Block Incoming Anonymous Calls)	Your rejection of anonymous calls has successfully been activated.
vsc_blockinclir_off	32 (Block Incoming Anonymous Calls)	Your rejection of anonymous calls has successfully been deactivated.

Copyright © Sipwise GmbH - All rights reserved

7.27. XMPP and Instant Messaging

Instant Messaging (IM) based on XMPP comes with Sipwise C5 out of the box. Sipwise C5 uses prosody as internal XMPP server. Each subscriber created on the platform have assigned a XMPP user, reachable already - out of the box - by using the same SIP credentials. You can easily open an XMPP client (e.g. Pidgin) and login with your SIP username@domain and your SIP password. Then, using the XMPP client options, you can create your buddy list by adding your buddies in the format user@domain.

Copyright © Sipwise GmbH - All rights reserved

7.28. Call Recording

7.28.1. Introduction to Call Recording Function

Sipwise C5 provides an opportunity to record call media content and store that in files.

Some characteristics of the Call Recording:

- Call Recording function can store both unidirectional (originating either from caller, or from callee) or bidirectional (combined) streams from calls, resulting in 1, 2 or 3 physical files as output
- The location and format of the files is configurable.
- File storage is planned to occur on an NFS shared folder.
- Activation of call recording may happen generally for a *Domain / Peer / Subscriber* through Sipwise C5 admin web interface.

IMPORTANT NGCP's Call Recording function is not meant for individual call interception purpose! Sipwise provides its Lawful Interception solution for that use case.

- Querying or deletion of existing recordings may happen through the REST API.
- Listing recordings of a subscriber is possible on NGCP's admin web interface.

The Call Recording function is implemented using NGCP's ***rtpengine*** module.

NOTE There are 2 *rtpengine* daemons employed when call recording is enabled and active. The *main rtpengine* takes care of forwarding media packets between caller and callee, as usual, while the *secondary rtpengine* (recording) daemon is responsible for storing call data streams in the file system.

Call Recording is disabled by default. Enabling and configuration of Call Recording takes place in 2 steps:

1. Enabling the feature on Sipwise C5 by setting configuration parameters in the main config.yml configuration file.
2. Activating the feature for a *Domain / Peer / Subscriber.*

7.28.2. Information on Files and Directories

NGCP's Call Recording function uses an **NFS shared folder** to save recorded streams.

IMPORTANT Since call data amount may be huge (depending on the number and duration of calls), it is *strongly not recommended* to store recorded streams on NGCP's local disks. However if you *have to* store recorded streams as files in the local filesystem, please contact Sipwise Support team in order to get the proper configuration of Call Recording function.

The NFS share gets mounted during startup of the recording daemon. If the NFS share cannot be mounted for some reason, the recording daemon will not start.

Filenames have the format: <call_ID>-<random>-<SSRC>.<extension>, where:

Copyright © Sipwise GmbH - All rights reserved

- call_ID: SIP Call-ID of the call being recorded
- random: is a string of random characters, unique for each recorded call. It's purpose is to avoid possible filename collisions if a Call-ID ever gets reused.
- SSRC: is the RTP SSRC for unidirectional recordings, or "mix" for the bidirectional (combined) audio.
- extension: is either "mp3" or "wav", depending on the configuration (rtpengine.recording.output_format)

There might be 1, 2 or 3 files produced as recorded streams. The **number of files** depends on the configuration:

1. `rtpengine.recording.output_mixed = 'yes'` (combined stream required)

   `rtpengine.recording.output_single = 'no'` (unidirectional streams not required)

2. `rtpengine.recording.output_mixed = 'no'` (combined stream not required)

   `rtpengine.recording.output_single = 'yes'` (unidirectional streams required)

3. `rtpengine.recording.output_mixed = 'yes'` (combined stream required)

   `rtpengine.recording.output_single = 'yes'` (unidirectional streams required)

7.28.3. Configuration

The Call Recording function can be enabled and configured on Sipwise C5 by changing the following configuration parameters in config.yml file:

```
rtpengine:
  ...
  recording:
    enable: no
    mp3_bitrate: '48000'
    nfs_host: 192.168.1.1
    nfs_remote_path: /var/recordings
    output_dir: /var/lib/rtpengine-recording
    output_format: wav
    output_mixed: yes
    output_single: yes
    resample: no
    resample_to: '16000'
    spool_dir: /var/spool/rtpengine
```

Enabling Call Recording

Enabling the function requires changing the value of rtpengine.recording.enable parameter to "yes". In order to make the new configuration active, it's necessary to do:

```
ngcpcfg apply 'Activated call recording'
```

Copyright © Sipwise GmbH - All rights reserved

Description of configuration parameters:

- enable: when set to “yes” Call Recording function is enabled; default: “no”
- mp3_bitrate: the bitrate used when recording happens in MP3 format; default: "48000"
- nfs_host: IP address of the NFS host that provides storage space for recorded streams; default: "192.168.1.1"
- nfs_remote_path: the remote path (folder) where files of recorded streams are stored on the NFS share; default: "/var/recordings"
- output_dir: is the local mount point for the NFS share, and thus where the final audio files will be written; default: "/var/lib/rtpengine-recording"

CAUTION	Normally you don’t need to change the default setting. If you do change the value, please be aware that recorded files will be written by *root* user in that directory.

- output_format: possible values are “wav” (Wave) or “mp3” (MP3); default: “wav”
- output_mixed: “yes” means that there is a file that contains a mixed stream of caller and callee voice data; default: "yes"
- output_single: “yes” means that there is a separate file for each stream direction, i.e. for the streams originating from caller and callee; default: "yes"
- resample: when set to “yes” the call data stream will be resampled before storing it in the file; default: “no”
- resample_to: the sample rate used for resampling output; default: "16000"
- spool_dir: is the place for temporary metadata files that are used by the recording daemon and the main rtpengine daemon for their communication; default: "/var/spool/rtpengine"

CAUTION	You should not change the default setting unless you have a good reason to do so! Sipwise has thoroughly tested the Call Recording function with the default setting.

If Call Recording is enabled you can see 2 *rtpengine* processes running when checking Sipwise C5 system state with *ngcp-service* tool:

```
root@sp1:/etc/ngcp-config# ngcp-service summary
Ok Service                          Managed    Started   Status
-- ------------------------------ ---------- --------- ---------
…
   kamailio-lb                      managed    by-ha     active
   ngcp-voisniff                    managed    by-ha     active
   rtpengine                        managed    by-ha     active
   rtpengine-recording              managed    by-ha     active
…
```


Copyright © Sipwise GmbH - All rights reserved

Activating Call Recording

Activating Call Recording for e.g. a *Subscriber:* please use NGCP's admin web interface for this purpose. On the web interface one has to navigate as follows: *Settings Subscribers select subscriber Details Preferences NAT and Media Flow Control*. Afterwards the record_call option has to be enabled by pressing the *Edit* button and ticking the checkbox.

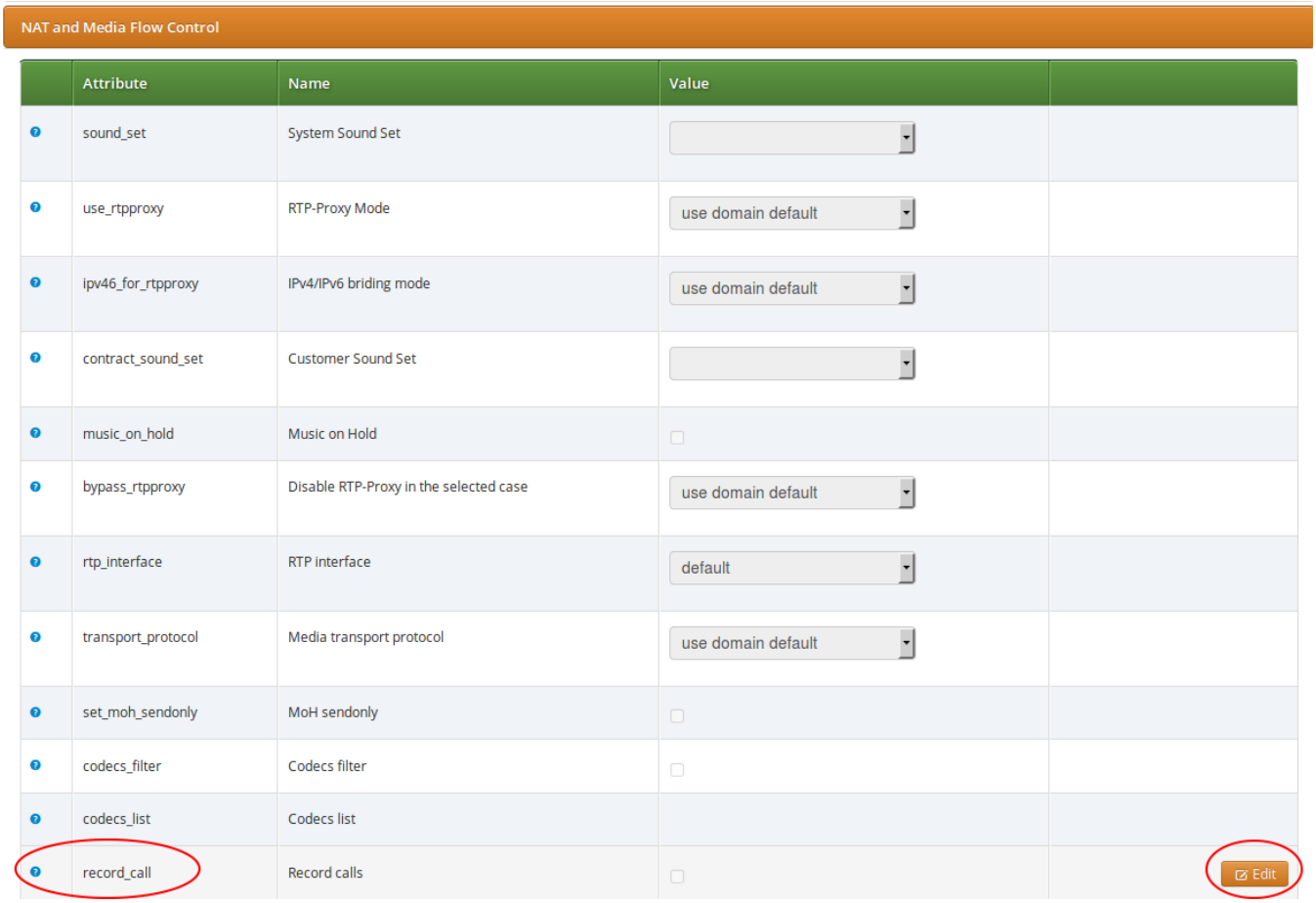


Figure 89. Activating Call Recording

NOTE The call recording function may be activated for a single *Subscriber, a Domain and a Peer server* in the same way: *Preferences NAT and Media Flow Control record_call*. When activating call recording for a *Domain or Peer* this effectively activates the function for all subscribers that belong to the selected domain, and for all calls with a local endpoint going through the selected peer server, respectively.

Once the call recording is active it's also possible to play a pre recording announcement to the caller before the call is established. In order to do this set *Preferences Applications play_announce_before_recording* to "Always" or "External/Internal calls only" depending if you want to play the message only for calls coming from PSTN or only for calls coming from local subscribers. In order to play the message it's mandatory to upload an audio file into *early_media announce_before_recording* inside the corresponding Sound Set. The play_announce_before_recording preference can be activated hierarchically at domain, customer and subscriber level.

It is possible to **list existing call recordings** of a *Subscriber* through the admin web interface of NGCP. In order to do so, please navigate to: *Settings Subscribers select subscriber Details Call Recordings*

Copyright © Sipwise GmbH - All rights reserved

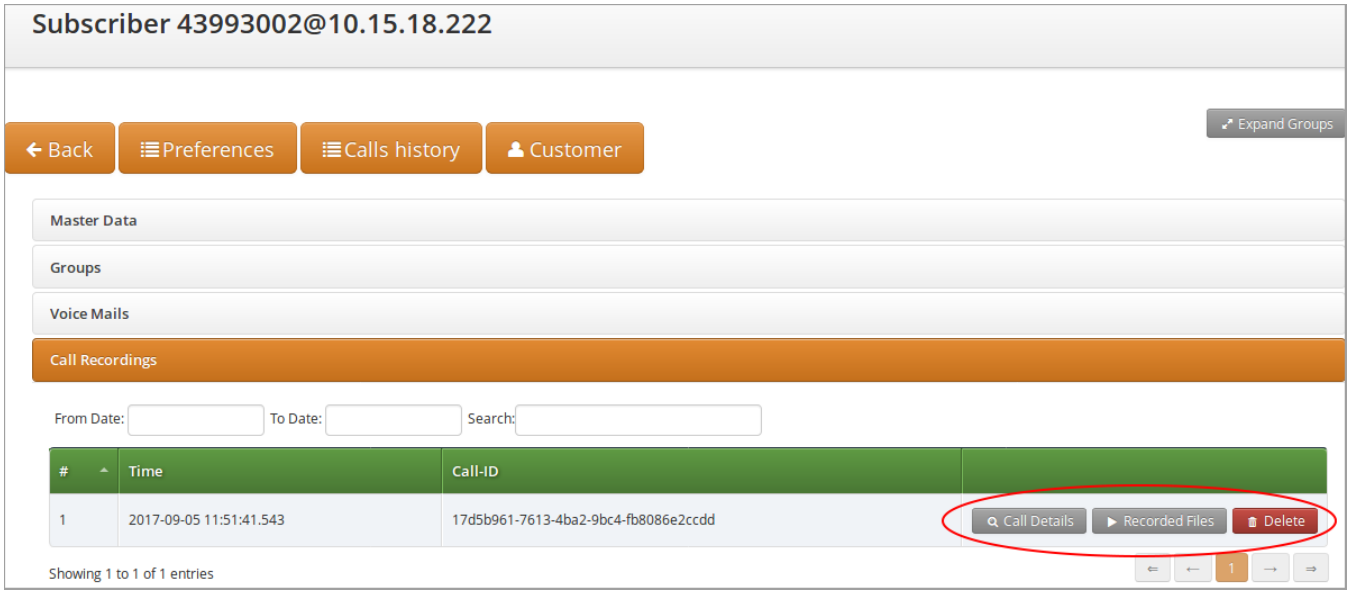


Figure 90. Listing Call Recordings

If you select an item in the list, besides the main properties such as the time of call and the SIP Call-ID, you can retrieve the details of the related call (press the *Call Details* button), get the list of recorded files (press the *Recorded Files* button) or *Delete* the recorded call.

When selecting *Call Details* you will see the most important accounting data of the call. Furthermore you can see the SIP *Call Flow* or the complete *Call Details* if you press the respective buttons.

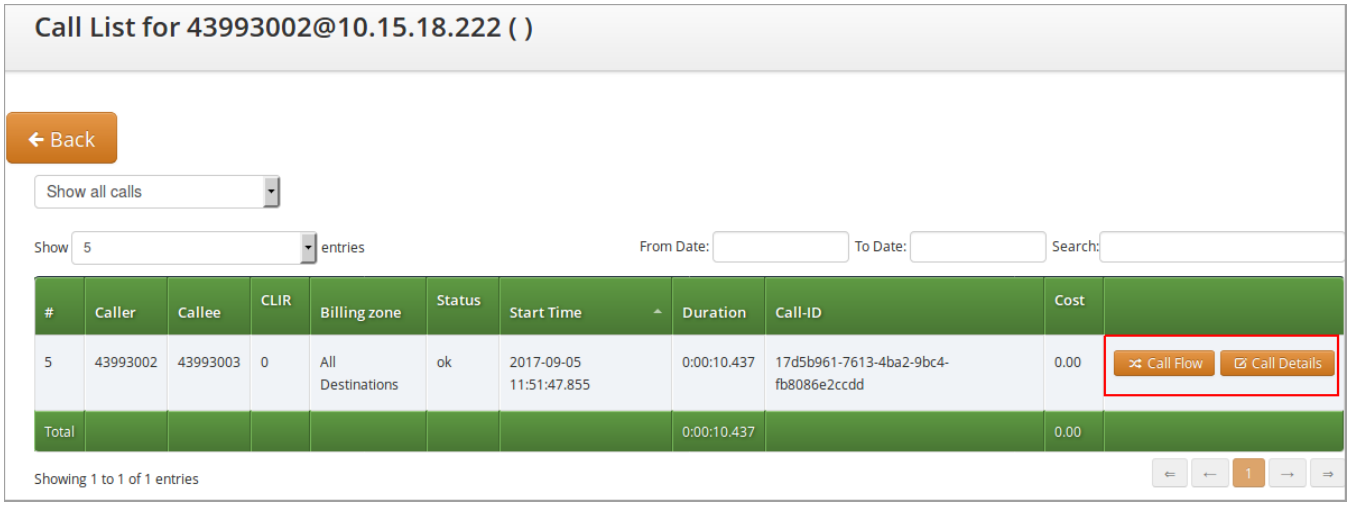


Figure 91. Listing Call Details for a Recording

When navigating to *Recorded Files* of a call you will be presented with a list of files. For each file item:

- type of stream is shown, that can be either "mixed" (combined voice data), or "single" (voice data of caller or callee)
- file format is shown, that can be either "wav", or "mp3"
- you can download the file by pressing the *Play* button

 Copyright © Sipwise GmbH - All rights reserved

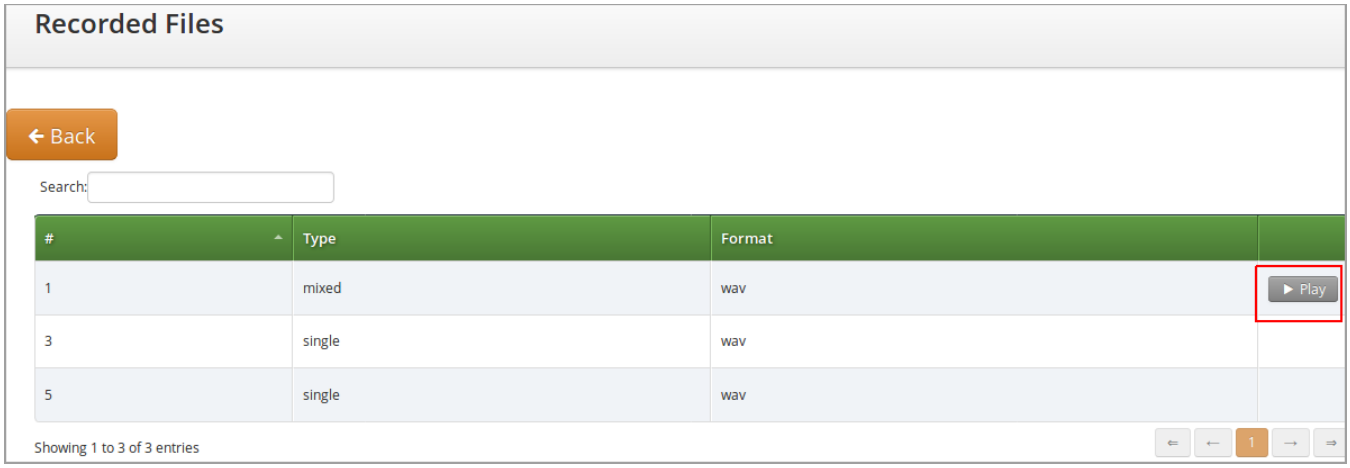


Figure 92. Listing Files for a Recording

7.28.4. REST API

The Sipwise C5 REST API provides methods for querying and deletion of existing recording data. The full documentation of the available API methods is available on the admin web interface of the NGCP, as usual.

The following API methods are provided for managing Call Recordings:

- CallRecordings:

  Provides information about the calls recorded in the system; can also be used to delete a recording entry

  accessible by the path: /api/callrecordings (collection) or /api/callrecordings/id (single item)

  Supported HTTP methods: OPTIONS, GET, DELETE

- CallRecordingStreams:

  Provides information about recorded streams, such as start time, end time, format, mixed/single type, etc.; can also be used to delete a recorded stream

  accessible by the path: /api/callrecordingstreams (collection) or /api/callrecordingstreams/id (single item)

  Supported HTTP methods: OPTIONS, GET, DELETE

- CallRecordingFiles:

  Provides information about recorded streams, such as start time, end time, format, mixed/single type, etc.; additionally returns the file content too

  accessible by the path: /api/callrecordingfiles (collection) or /api/callrecordingfiles/id (single item)

  Supported HTTP methods: OPTIONS, GET

7.28.5. Pre-Recording Announcement

Many country regulations require that an informative announcement is played to the caller before the call is actually recorded. The Sipwise C5 allows you to configure your own custom announcement with few simple steps.

Copyright © Sipwise GmbH - All rights reserved

First create a system sound set for the feature. In *Settings Sound Sets* either use your already existing *Sound Set* or create a new *Sound Set* and then assign it to your domain or subscribers. In the *Sound Set* there is an announcement *early_media announce_before_recording* for that purpose.

Once the *Sound Set* is created the subscriber's preference *play_announce_before_recording* of the callee must be enabled under *Subscriber Preferences Applications* menu. The same parameter can be set in the Domain's or Customer's preferences to enable this feature for all its subscribers.

NOTE The announcement will be played to caller before the call is routed to the callee.

IMPORTANT In case of **CFU** or **CFNA** with Pre-Recording Announcement feature enabled on both the forwarder and the final callee, only the Announcement of final callee will be played to caller. In case of **CFB** and **CFT**, instead, the announcement of the forwarder will be played first, then the announcement of final callee will be played after the call forward is executed.

Copyright © Sipwise GmbH - All rights reserved

7.29. Media Transcoding and Transrating

7.29.1. Overview

Sipwise C5 offers the capability to convert RTP media between several supported codecs, a feature known as transcoding. While this feature is always available on Sipwise C5, it's engaged only when a subscriber, peer, or domain is explicitly configured for it. By default, Sipwise C5 lets RTP endpoints negotiate the codec to use among themselves without interfering.

IMPORTANT	Media transcoding is a relatively CPU-intensive feature. As such, each individual node of a Sipwise C5 performing media transcoding can only support a limited number of concurrent calls for which transcoding is active.

7.29.2. Supported Codecs

The following audio codecs, which are commonly found in use by SIP/RTP clients, are currently supported for transcoding.

- G.711 (μ-Law and a-Law)
- G.722
- G.723.1
- G.729
- GSM
- AMR (narrowband and wideband, the latter also known as AMR-WB)
- Opus
- Speex
- DTMF event packets (**telephone-event**)
- Comfort noise

Some codecs operate at different sampling rates than other codecs. If transcoding happens between two such codecs, the audio will be resampled as necessary. Similarly, if transcoding happens between a mono (1-channel) and a stereo (2-channel) codec, the audio will be up-mixed and down-mixed as necessary.

7.29.3. Configuration

Transcoding can be engaged for individual subscribers, peers, or domains on their respective preferences page in the Sipwise C5 admin web interface.

Copyright © Sipwise GmbH - All rights reserved

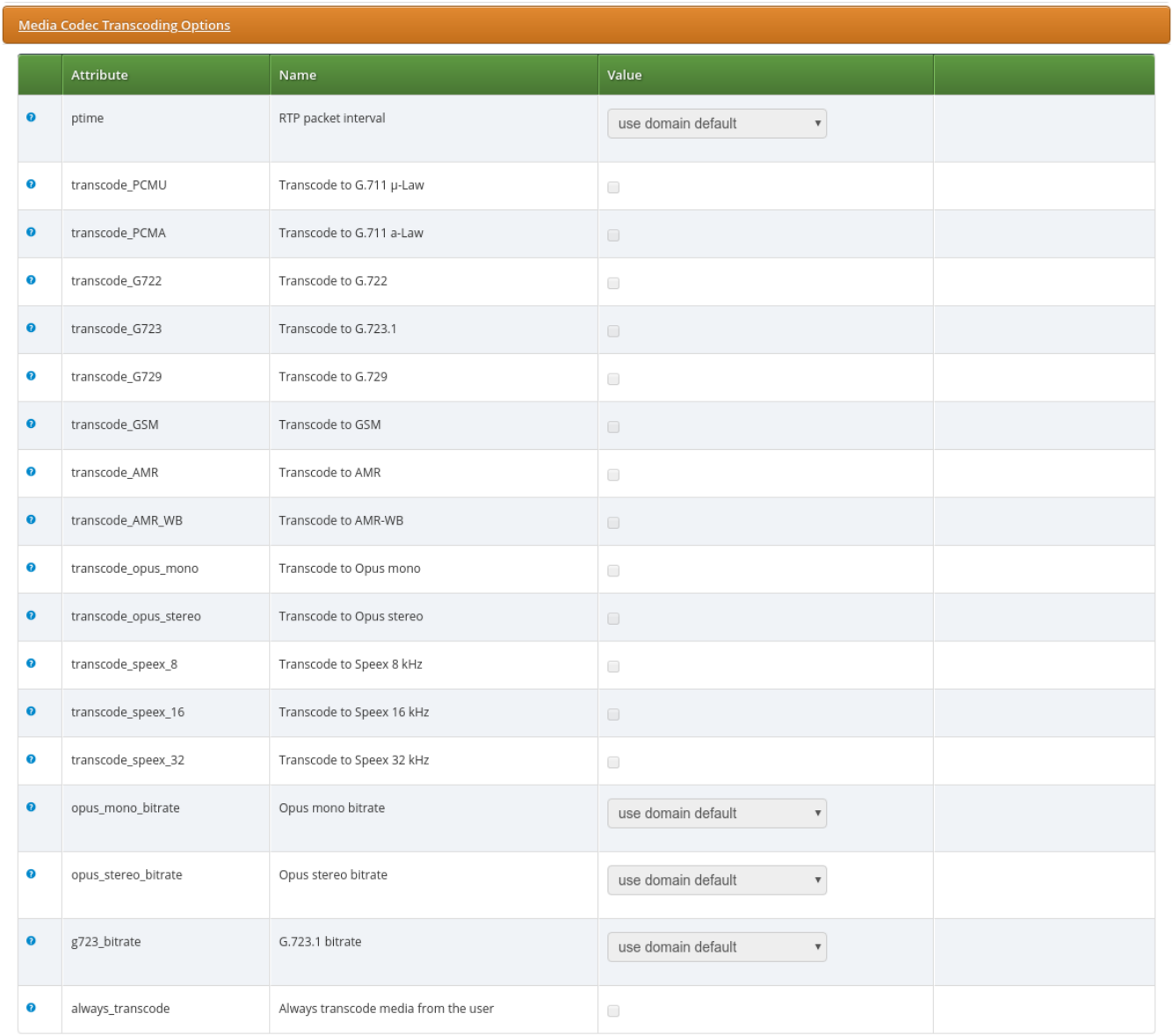


Figure 93. Transcoding Configuration

Setting any of the transcoding options for a domain makes it affect all the subscribers in this domain.

Individual options are described below.

ptime

Packetisation time in milliseconds. Normally Sipwise C5 lets the RTP endpoints select and negotiate the packetisation time they want to use. Setting this option to anything other than unchanged will engage transrating via the transcoding engine towards this subscriber or peer even if none of the other transcoding options are set, in which case the media will be transrated (repacketised).

For example, setting this to 40 ms would mean that each RTP packet sent towards this subscriber or peer would contain 40 milliseconds worth of audio, even if the other side of the call sends media that is packetised differently. It would also make Sipwise C5 indicate towards this subscriber or peer that it would prefer to receive audio in 40 millisecond packets (through the a=ptime SDP attribute).

 Copyright © Sipwise GmbH - All rights reserved

transcode_...

Enabling one of these options adds the selected codecs to the list of codecs offered to this subscriber or peer, even if the original list of offered codecs did not include it. If this additional codec ends up being accepted by this subscriber or peer, then it will be transcoding to the first supported codec that was originally offered.

For example, if a calling RTP client A indicates support for PCMA (G.711 a-Law) as well as G.722, and calls a subscriber B that is configured for transcoding to G.729, then subscriber B would be offered PCMA, G.722, and G.729 by Sipwise C5. If subscriber B then accepts G.729 and starts sending G.729, Sipwise C5 would engage its transcoding engine and transcode the audio to PCMA (because PCMA and not G.722 was the codec preferred by A) before forwarding it to A. Vice versa, PCMA arriving from A would be transcoded to G.729 before being sent to B. (If B were to reject G.729 and instead starts to send PCMA or G.722, no transcoding would happen.)

Notes on individual codecs:

- **AMR** is available in both narrowband (AMR operating at 8 kHz) and wideband (AMR-WB operating at 16 kHz) variants. These are distinct codecs and can be configured for transcoding separately or together.
- **Opus** always operates at 48 kHz, but is supported in both mono and stereo (1 and 2 audio channels respectively). Both can be offered at the same time if so desired.
- **Speex** is supported at sampling rates of 8, 16, and 32 kHz. These can be configured separately for transcoding, or together.
- **DTMF** is not an actual audio codec, but rather represents transcoding between DTMF event packets and in-band DTMF audio tones. This is described in more detail below.
- **CN** refers to comfort noise payloads. More information about this is below.

..._bitrate

Some codecs (Opus and G.723.1 in particular) can be configured for different bitrates, which would impact the amount of network bandwidth they use, as well as the audio quality produced. For Opus, different bitrates can be selected for their mono and stereo instances. Selecting a bitrate has no effect if transcoding to the respective codec is not engaged.

AMR-specific Options

AMR and AMR-WB support dynamic mode switching (adapting the bitrate) during runtime. The configurable bitrate therefore is only the initial bitrate at which the encoder is started, and is further constrained by the **mode-set** option.

The **mode-set** is a list of comma-separated AMR modes, e.g. **0,1,2,3**. If a **mode-set** is received from a remote peer, then only the modes (bitrates) given in the list will be used for the encoder. If a specific bitrate is also configured, then this bitrate will be used as the highest possible starting bitrate. If this bitrate is not allowed by the **mode-set**, then the next lower bitrate will be used. If no lower bitrates are permitted, then the next higher bitrate will be used.

For outgoing AMR offers, a **mode-set** can be configured in the preferences page, and it will be honoured just like a **mode-set** that is received from a remote peer. A remote AMR peer may support only a specific subset of AMR modes, which would need to be configured in the **mode-set**. The Sipwise C5 supports encoding and decoding all possible AMR modes.

Copyright © Sipwise GmbH - All rights reserved

If no **mode-set** is received nor configured, then all bitrates are permitted.

AMR has two basic payload variants: **bandwidth-efficient** and **octet-aligned**. A remote AMR peer may support only one of them, in which case the correct one must be configured in the preferences. The default is **bandwidth-efficient**. The Sipwise C5 supports sending and receiving both and will honour the request made by a remote AMR peer for one or the other.

Further options that may be required by a remote AMR peer to restrict permissible mode changes are: **mode-change-period** restricts mode changes to be made only every other packet if set to **2** (the default is **1**); **mode-change-capability** advertises the capability to restrict the modes changes as such without actually restricting them; and **mode-change-neighbor** restricts mode changes to be made only to neighbouring modes if enabled (the default is disabled).

AMR supports requesting specific encoder modes from the remote peer via **Codec Mode Requests** (CMR). The Sipwise C5 will honour a CMR received from the remote peer and switch encoder mode accordingly. Then optionally, if no further CMRs are received from the peer, the Sipwise C5 can then try to increase encoder bitrate again to improve audio quality. This can be enabled by setting the **mode change interval** preference to non-zero. It specifies an interval in milliseconds at which to switch to a higher encoder bitrate if no CMR was received during that time, and if a higher bitrate is available and allowed by the **mode-set**.

In the opposite direction, the Sipwise C5 can optionally request higher bitrates from the remote peer by sending CMRs out. This can be enabled by setting the **CMR interval** preference to non-zero. It specifies an interval in milliseconds at which the Sipwise C5 will request a higher bitrate from the remote encoder if a higher bitrate is available and it was not being used during that time.

always_transcode

Setting this flag instructs Sipwise C5 to always engage transcoding to the first (preferred) codec indicated by an RTP endpoint, even if another codec is available that is supported by both parties to a call. Enabling this flag can potentially engage the transcoding engine for a call even if none of the other transcoding options are set.

For example: Subscriber A is calling subscriber B. Subscriber A is indicating support for PCMA and G.722. Subscriber B answers the call, rejects PCMA but accepts G.722, and starts sending G.722 to A. Normally Sipwise C5 would not get involved and would let G.722 pass between A and B. But if subscriber B has the always_transcode flag set, Sipwise C5 would now start transcoding the G.722 sent by B into PCMA before forwarding it to A, because PCMA was indicated as the preferred codec by A. Vice versa, PCMA arriving from A would be transcoded into G.722 and then forwarded to B.

DTMF transcoding

Sipwise C5 supports transcoding between DTMF event packets (using the RTP telephone-event type payload) and DTMF tones carried in-band in the audio stream. DTMF transcoding is supported in both directions: transcoding DTMF event packets to DTMF tones, and DTMF tones in an audio stream and transcoding them to DTMF event packets.

Support for DTMF transcoding can be enabled in one of two ways:

- Enabling the setting transcode_dtmf for a subscriber, peer, or domain. This is useful if the subscriber, peer, or domain requires support for DTMF event packets, but the calling entity might only support DTMF tones carried in-band in the audio stream.

 Copyright © Sipwise GmbH - All rights reserved

- Enabling the setting always_transcode for a subscriber, peer, or domain. This is useful for the reverse case: if the subscriber, peer, or domain might only support DTMF tones carried in-band in the audio stream, but the calling entity requires support for DTMF event packets.

Enabling DTMF transcoding for any call requires that all audio passes through the transcoding engine, as well as a DSP for detecting DTMF tones in one direction. This carries an additional performance impact with it, and so DTMF transcoding should only be enabled when really necessary.

DTMF conversion of INFO messages

Sipwise C5 supports the conversion of SIP INFO messages with application/dtmf-relay payload to DTMF event or PCM DTMF inband tone, depending on whether the destination participant supports the telephone-event RTP payload type or not. DTMF conversion of INFO messages works only in one way direction: DTMF events or PCM DTMF inband tones are not converted back to DTMF INFO messages.

The preference kamailio.proxy.allow_info_method has to be set to yes in config.yml in order to make the DTMF INFO message conversion working.

Enabling DTMF conversion of INFO messages for any call requires that all audio passes through the transcoding engine, as well as a DSP for detecting DTMF tones in one direction. This carries an additional performance impact with it, and so it should only be enabled when really necessary.

NOTE At the moment this feature is for internal use only. External generated INFO messages will be not converted but passed through.

Comfort Noise

RTP supports a special payload format for carrying comfort noise (CN) audio without having to encode it as actual audio. Support for CN is optional and negotiation is normally left up to the endpoints. If CN transcoding is enabled, Sipwise C5 can transcode CN payloads to audio noise and vice versa.

When active and when CN is supported by one endpoint but not the other, any received CN payloads are converted to audio noise and this noise is then inserted into the audio stream. In the reverse direction, Sipwise C5 can optionally detect silence audio frames and then convert these to predefined CN payloads.

To enable silence detection, the setting rtpengine.silence_detect in config.yml must be set to a non-zero value. This value denotes the silence detection threshold in percent and therefore must be between 0.0 and 100.0. Any audio samples that are lower than the configured threshold will be as silence, and if all audio samples in an audio frame are detected as silence, then that frame will be replaced by a CN frame. To only replace audio frames that contain complete silence with CN frames, a very low but non-zero value can be used here, such as 0.1%.

NOTE Certain audio codecs always leave a residual DC offset in the audio samples, even when encoding complete silence. G.711 a-Law (PCMA) is one such example, for which the minimum silence detection threshold is 0.013%.

When a silence audio frame is replaced by a CN frame, the generated CN payload is not based on the audio that is being replaced, but rather taken from the static and predefined setting rtpengine.cn_payload in config.yml. This is a list of comfort noise parameters according to RFC 3389. The first value in the list is mandatory and denotes the volume of the noise payload in negative dBov (meaning larger values result in quieter audio, with zero being the loudest), while all subsequent values

Copyright © Sipwise GmbH - All rights reserved

are optional and denote spectral information about the noise. The default is the single value 32, which describes noise with a volume of -32 dBov without any spectral information.

7.29.4. T.38 transcoding

In addition to transcoding between audio codecs, Sipwise C5 supports transcoding between T.38 fax transmissions (over UDPTL transport) and T.30 fax data over regular audio channels. The audio codec commonly used to carry T.30 data is G.711, but any other audio codec that is supported for transcoding can also be used, provided it offers a high enough bitrate and audio quality.

Two settings to control T.38 transcoding are available for subscribers, peers, and domains:

- The setting t38_decode instructs Sipwise C5 to accept an offered T.38 session towards the subscriber, peer, or domain, and translate it into a regular audio call carrying T.30 fax data. By default, G.711 (both μ-Law and a-Law) will be offered. If any other codecs are selected through the transcode_... options, then only those codecs will be offered and the G.711 default will be omitted. Non-T.38 offers are not affected by this settings.
- The setting t38_force forces any audio call towards this subscriber, peer, or domain to be translated to a T.38 session, regardless of whether the audio media actually carries T.30 fax data or not. This is useful if it is known that the remote destination is a fax endpoint supporting T.38.

7.29.5. DTX jitter buffer

DTX is short for **discontinuous transmission** and describes an event during which a remote RTP client intermittently stops sending RTP. Some codecs explicitly support this as a feature (e.g. AMR and AMR-WB) while in other instances a sending RTP client may erroneously produce gaps in the RTP stream. If a receiving RTP client isn't expecting such a behaviour, the resulting audio output may have unpleasant characteristics, such as stuttering or dropping the audio to complete silence.

By default the Sipwise C5 will not attempt to correct such a DTX behaviour and will simply pass through any DTX gaps untouched. However, for any call with active audio transcoding, the Sipwise C5 can be configured to enable a DTX jitter buffer in order to fill in DTX gaps with comfort noise to produce a steady output RTP stream.

Configuration

The master switch to enable the DTX jitter buffer is found in config.yml under the setting rtpengine.dtx_delay. This value denotes the processing delay in milliseconds (i.e. the length of the jitter buffer) and should be set to just slightly higher than the highest expected ingress jitter. The default value of zero disables the DTX buffer.

When a DTX event is detected, the gap will be filled in using an RFC 3389 compliant comfort noise generator. The parameters for this generator are set under rtpengine.dtx_cn_params. The default is just a single value of 35, specifying white noise at -35 dBov. Higher numbers lead to quieter noise. Subsequent numbers (i.e. from the second number onward) are optional spectral parameters.

The maximum duration for which comfort noise should be generated can be set using rtpengine.max_dtx in seconds. The default is unlimited (zero).

If an incoming RTP stream experiences a timer drift (i.e. the RTP clock runs slower or faster than expected), then the timer handling the DTX buffer will be shifted forward or backward by rtpengine.dtx_shift milliseconds (default 5). Timer drifts can lead to DTX buffer overflows, which are

 Copyright © Sipwise GmbH - All rights reserved

detected if more than rtpengine.dtx_buffer packets with an RTP delay of more than rtpengine.dtx_lag milliseconds are held in the buffer (defaults 10 and 100, respectively).

AMR and AMR-WB explicitly support DTX and provide their own comfort noise format (called SID) to fill in DTX gaps. If rtpengine.amr_dtx is set to native (the default) then the native AMR SID/CN generator will be used to produce the noise to fill in DTX gaps. If this setting is set to CN then the generic CN generator (which is used for other codecs) is used instead.

Copyright © Sipwise GmbH - All rights reserved

7.30. Announcement Before Call Setup

This feature allows a callee to play a custom announcement to the caller every time it receives a call. The announcement is played in early media mode, therefore it can be used as a simple business welcome message or to inform the caller about a different cost of the call before it will be actually charged.

The configuration of the announcement is similar to the activation of Pre-Recording Announcement and it requires few simple steps.

First create a system sound set for the feature. In *Settings Sound Sets* either use your already existing *Sound Set* or create a new *Sound Set* and then assign it to your domain or subscribers. In the *Sound Set* there is an announcement *early_media announce_before_call_setup* for that purpose.

Once the *Sound Set* is created the subscriber's preference *play_announce_before_call_setup* must be enabled under *Subscriber Preferences Applications* menu. The same parameter can be set in the Domain's or Customer's preferences to enable this feature for all its subscribers.

NOTE The announcement will be played to caller before the call is routed to the callee.

IMPORTANT Differently from *Pre-Recording Announcement*, in all **Call Forward** cases with *Announcement Before Call Setup* feature enabled on both the forwarder and the final callee, only the announcement of the forwarder will be played to caller.

This feature and *Pre-Recording Announcement* can be activated at the same time. In this case the *Announcement Before Call Setup* will be played as first.

Copyright © Sipwise GmbH - All rights reserved

7.31. Emulated Ringback Tone

An usual problem linked to the activation of pre-call announcements, like for example Announcement Before Call Setup, Pre-Recording Announcement, Announcement Before Call Forward, is the impossibility of the caller device or provider to generate again the ringback tones after the early media announcement is terminated. This usually happens even if a new '180 Ringing' message is sent back to the caller.

To overcome this problem the Sipwise C5 makes it possible to play a pre-recorded ringback tone to the caller party in early media mode. This feature can be activated on the caller Subscriber or Peer where the tone has to be played. The emulated ringback is played in a loop and it is stopped if one of the following conditions happens:

- a '183 Progress' message is sent back by the callee, this because the callee most likely would like to play an announcement in early media mode
- a '200 OK' message is sent back by the callee
- the call fails

The configuration of the announcement is similar to the activation of Pre-Recording Announcement and it requires few simple steps.

First create a system sound set for the feature. In *Settings* *Sound Sets* either use your already existing *Sound Set* or create a new *Sound Set* and then assign it to your domain or subscriber. In the *Sound Set* there is an announcement *early_media* *ringback_tone* for that purpose.

Once the *Sound Set* is created the caller subscriber's preference *play_emulated_ringback_tone* must be enabled under *Subscriber* *Preferences* *Applications* menu. The same parameter can be set in the Domain's or Customer's preferences to enable this feature for all its subscribers. It can also be enabled on Peer's preferences to activate the feature when the call is coming from the peer.

By default the callee's soundset is used to generate the emulated ringback tone. If you prefer to use the ringback tone uploaded in the caller soundset you have to set config.yml preference *kamailio.proxy.play_ringback_tone_of_caller* to 'yes'.

The same feature can be used to play back to the caller a music on hold or other announcements while the callee endpoint is ringing. To do that just upload the media you prefer instead of a ringback tone.

IMPORTANT The callee has to send a '180 Ringing' message to trigger the emulated ring back tone to start.

NOTE The generation of the emulated ringback tone is currently limited to two specific scenarios: in the one specified above after a pre-call announcement and in a TPCC initiated call where the caller is already in conversation.

Copyright © Sipwise GmbH - All rights reserved

7.32. Store Recent Calls and Redial

Sipwise C5 allows to store the number of the last incoming and outgoing calls of each subscriber. To enable the feature edit config.yml and enable there kamailio: store_recentcalls: yes. Only the very last incoming and outgoing call is stored.

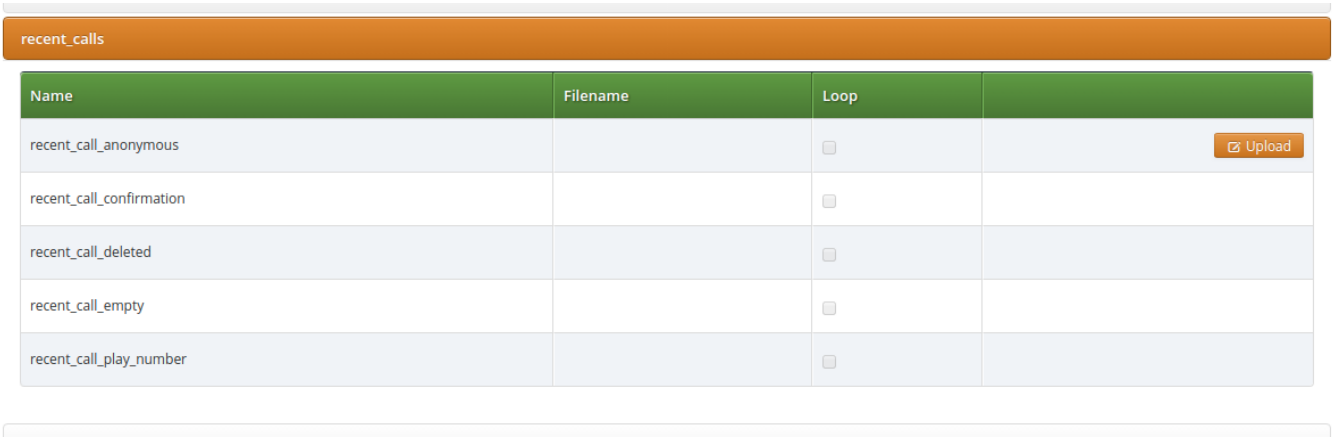
Each subscriber can interact with his own personal stored records using Vertical Service Codes (see VSC). In particular a subscriber can:

- redial last dialed number (this feature has to be enabled for the each subscriber/domain using preference *last_number_redial*)
- hear a voice announcement of the last caller's number, then press the key defined in *semsvsc callback_last_caller_confirmation_key* preference of */etc/ngcp-config/config.yml* to return the call
- return the call to the last caller's number without hearing the number announcement
- delete the personal stored records of any recent calls to and from him

NOTE it is not possible return a call if the caller's number is unavailable (e.g. anonymous calls).

7.32.1. Configuring Recent Calls Sound Sets

Sound Sets can be defined in *SettingsSound Sets*. To create a new Sound Set, click *Create Sound Set*. Then click the *Files* button.



Upload the following files:

Table 22. Recent Calls Sound Sets

Handle	Message played
recent_call_play_number	The last call you received was from ....
recent_call_confirmation	To call back this number press key ....
recent_call_anonymous	The last caller didn't share his number, you can not return the call to this person.
recent_call_empty	Your recent call history is empty.
recent_call_deleted	Your recent call history has been successfully deleted.

Copyright © Sipwise GmbH - All rights reserved

NOTE You may use 8 or 16 bit mono WAV audio files.

Then set the preference *sound_set* on the Domain or Subscriber level in order to assign the Sound Set (as usual the subscriber preference overrides the domain one).

7.32.2. Advanced configuration

By default the expiration time for the most recent incoming and outgoing call per subscriber are 3600 seconds (1 hour) and 86400 seconds (1 day) respectively. If you wish to prolong or shorten the expiration time open constants.yml and set there recentcalls: expire: 3600 and recentcalls: out_expire: 86400 to a new value, then issue ngcpcfg apply "recentcalls expire modification" afterwards.

Copyright © Sipwise GmbH - All rights reserved

7.33. Time sets management

7.33.1. Time sets specifications and data description

The Sipwise C5 provides administrative WEB and API interface to manage time sets.

Supported fields, input and output format are based on iCalendar EVENT specification.

Not all iCalendar and EVENT properties are supported, but those that are used for time points and periods definition or stated mandatory by specification:

- CALENDAR supported properties:

  NAME
- EVENT supported properties:

  SUMMARY

  DTSTART

  DTEND

  RRULE

Important to mention that current implementation does not support these EVENT properties:

- DTSTAMP (UID is used in generated calendar ics file, both UID and DTSTAMP are ignored during uploading calendar file);
- DURATION (DTEND is used);
- RDATE
- EXDATE
- PRIORITY

Main EVENT property, that is used for time points and periods definition is RRULE. Current time sets implementation supports all properties described in the RRULE specification except WKST.

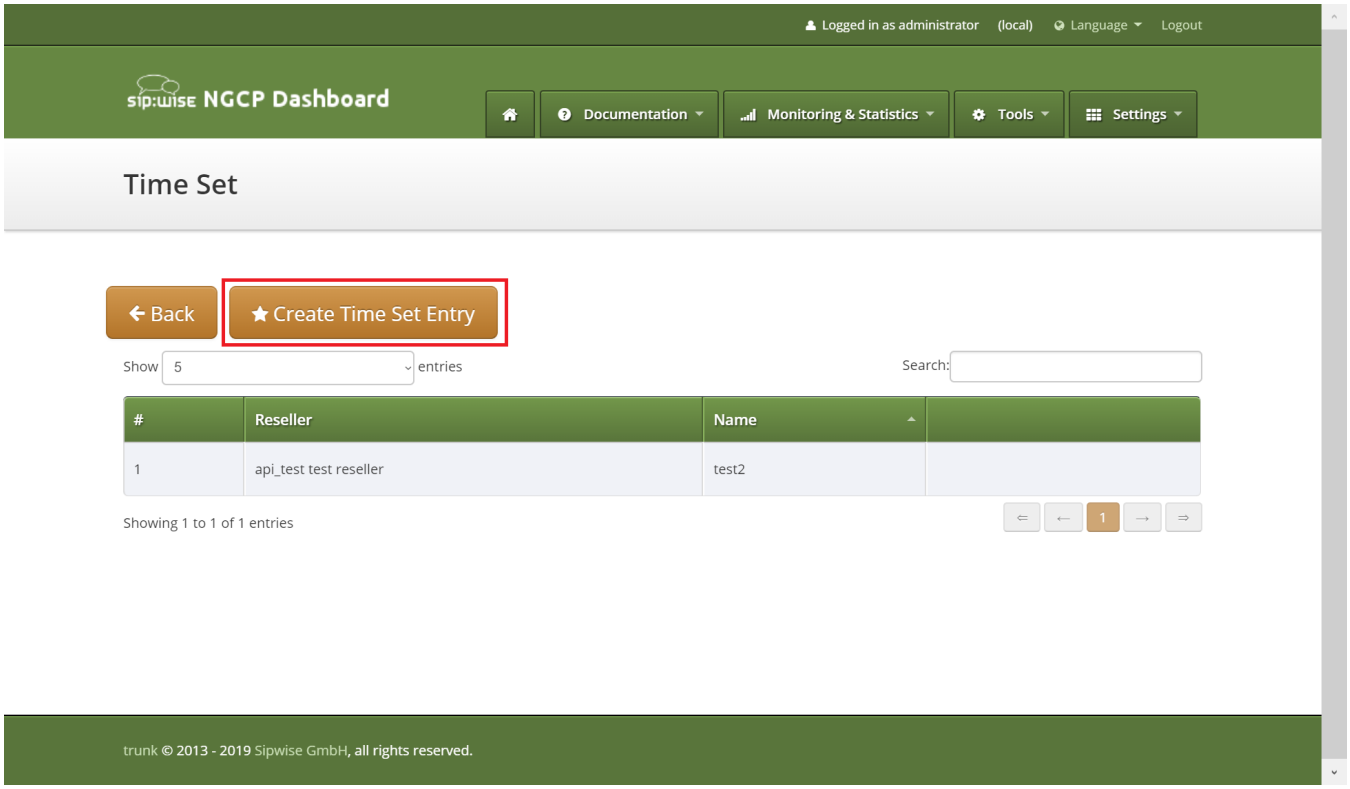
Default value for week start is MO (Monday).

7.33.2. Web interface for the time sets

Time sets management section is provided in two variants. One is main time sets management section and other is a chapter on reseller details page. Variants have minor differences. Functionality will be explained using time sets dedicated interface. Differences will be explained below.

Time set can be created using creation form. On time sets management interface press button "Create Time Set Entry":

Copyright © Sipwise GmbH - All rights reserved

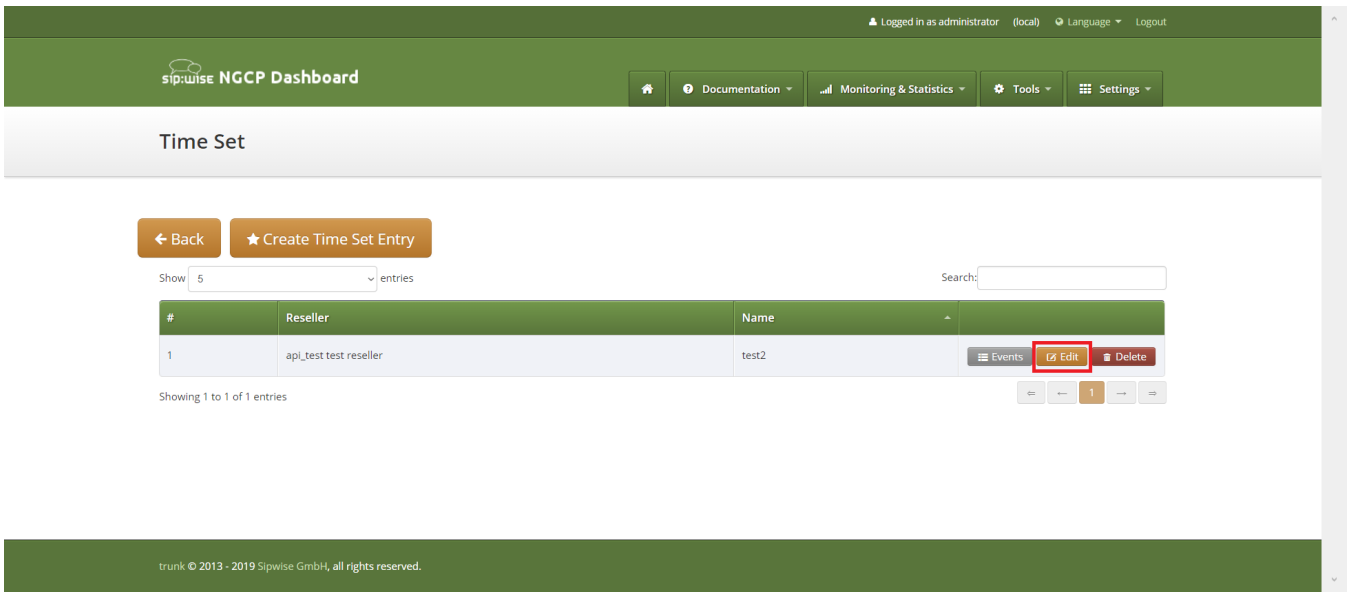


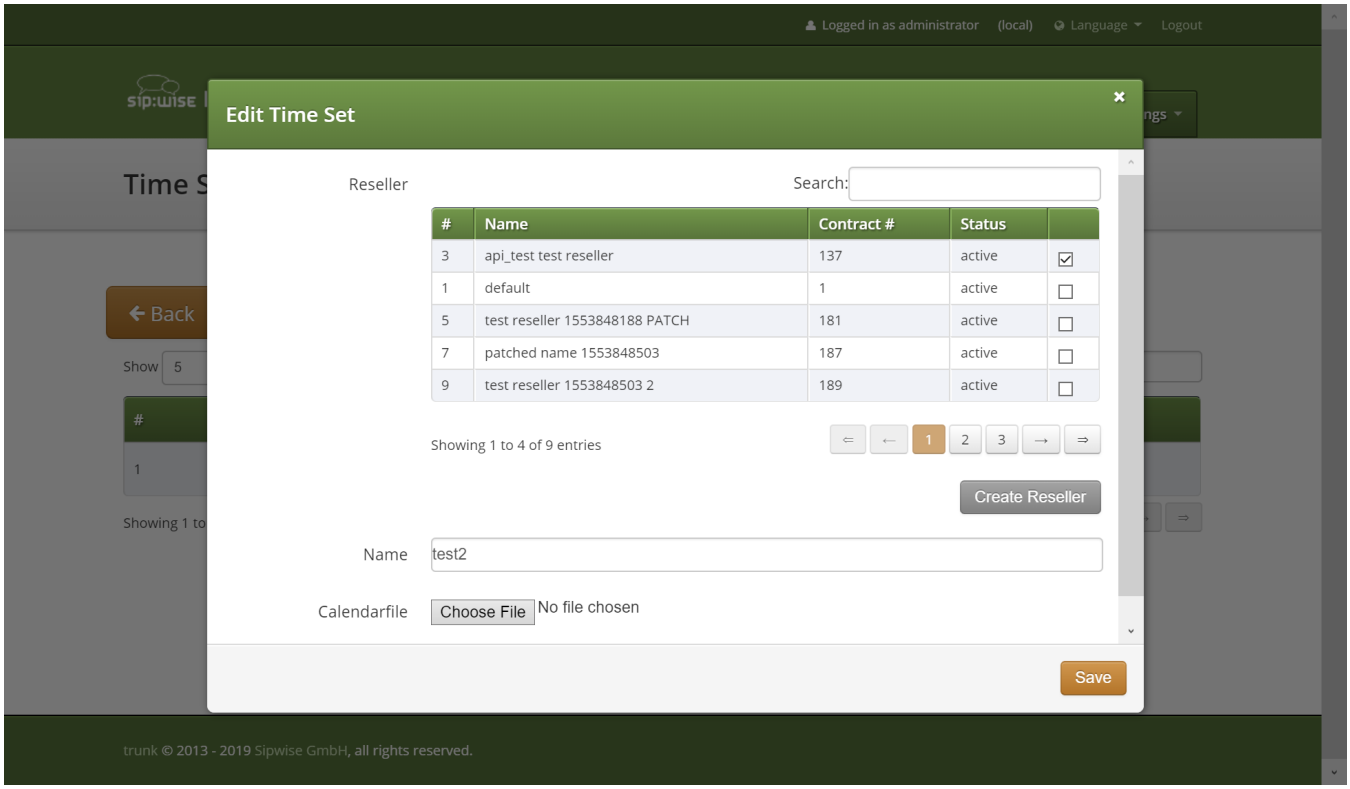
"Reseller" field is mandatory.

"Name" field should be defined, if iCalendar (ics) file is not going to be uploaded or file doesn't have NAME property for the CALENDAR entry.

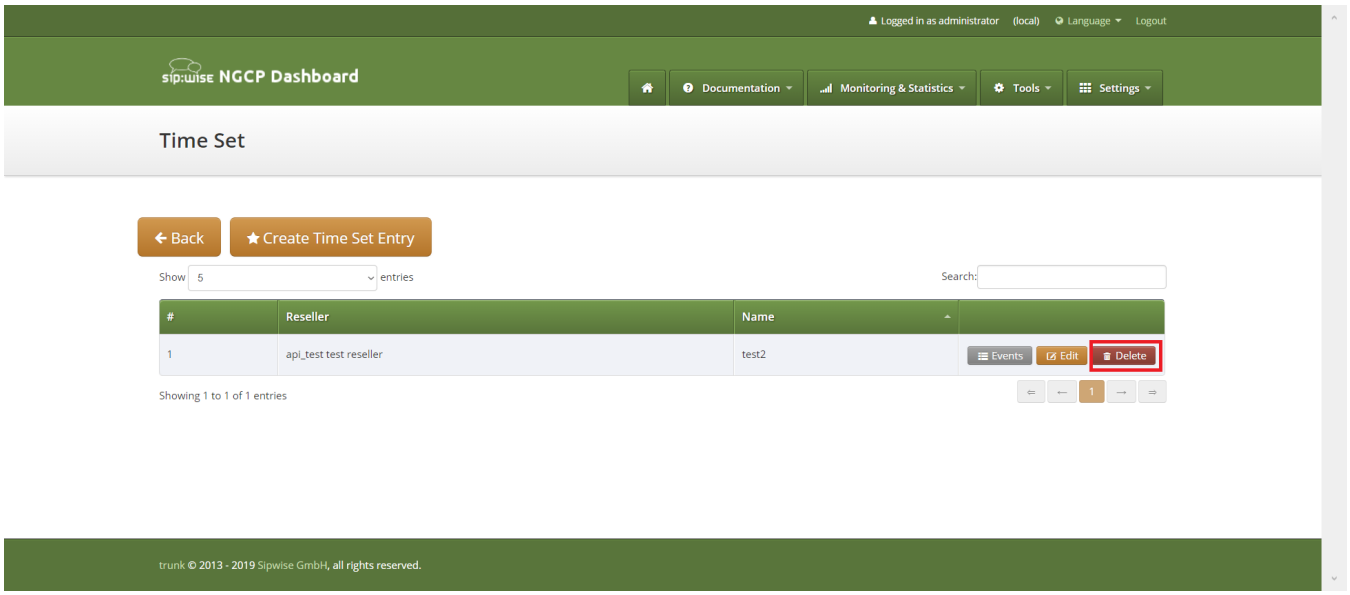
If both NAME in the uploaded iCalendar (ics) file and form field "Name" aren't empty then value from the form field will be taken.

Created time set can be modified:

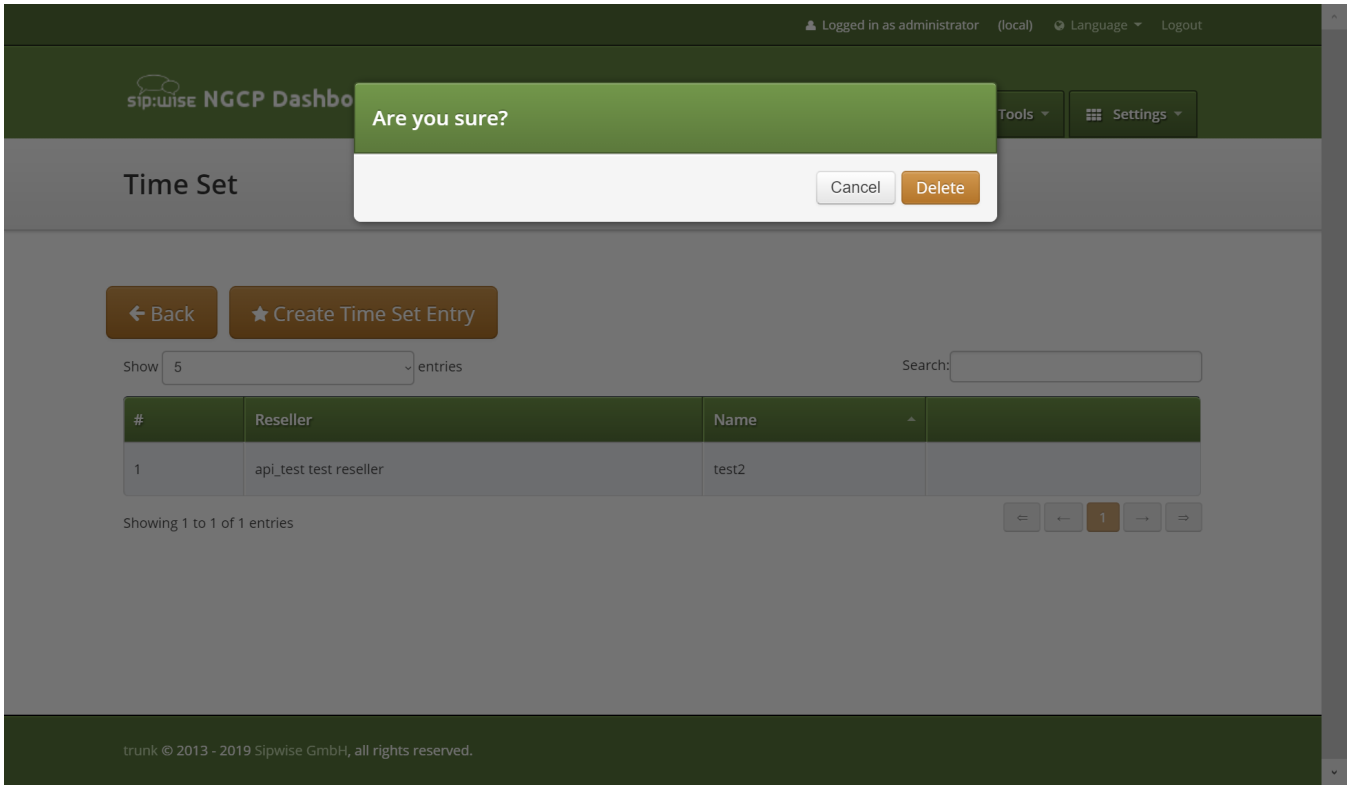




or deleted:



 Copyright © Sipwise GmbH - All rights reserved

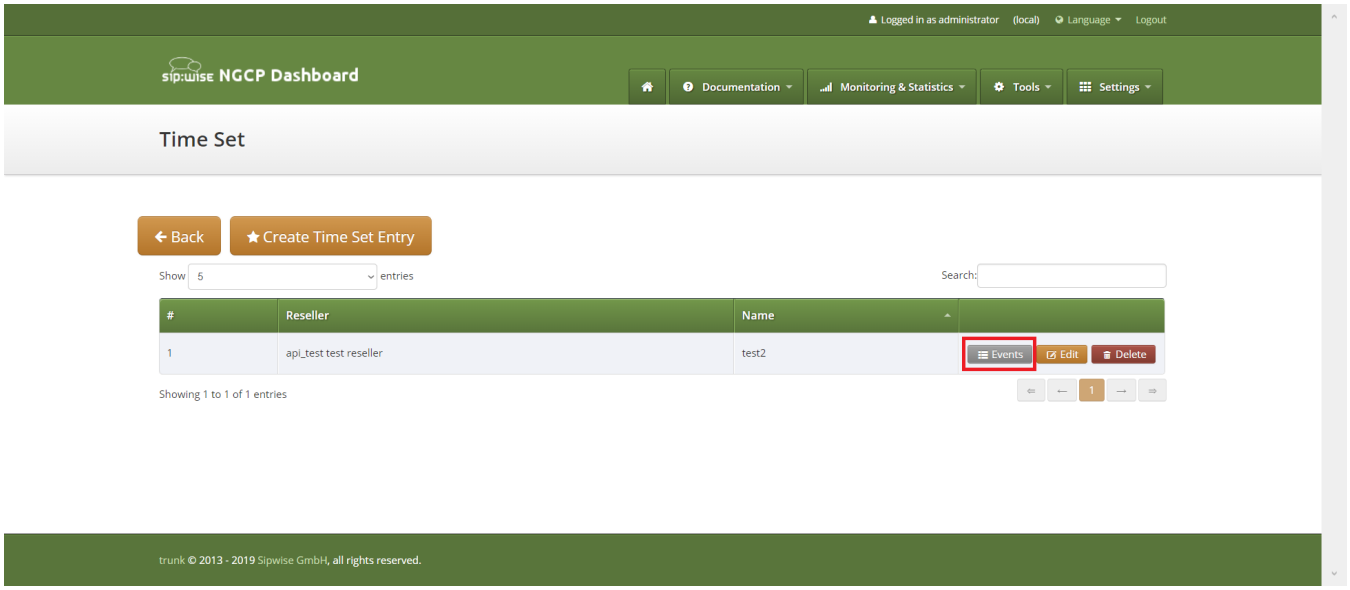


NOTE If calendar ics file will be uploaded to edit time set, all presented events will be deleted and events from the uploaded file will be added after it.

7.33.3. Web interface for the time set events

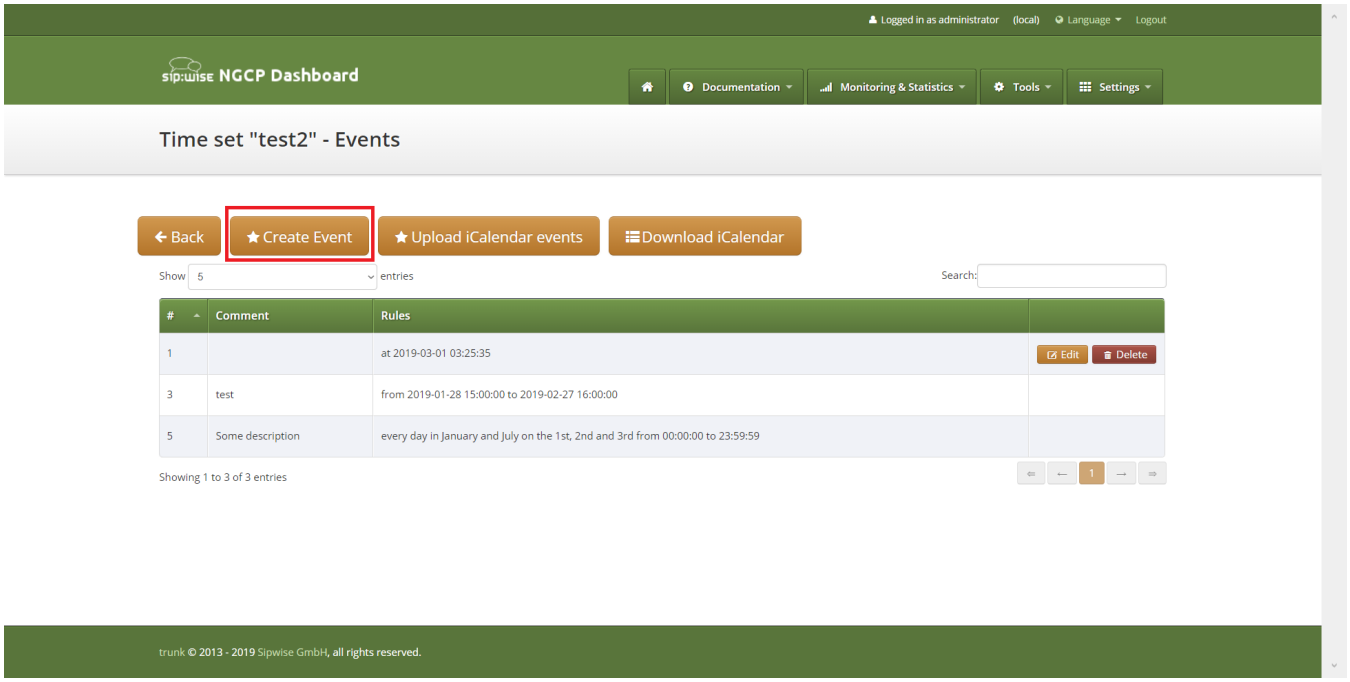
Time set can contain set of events. Each time set event will be used to generate CALENDAR EVENT entry in the generated iCalendar file. So all fields in the time set event forms represent properties of the iCalendar EVENT component.

To manage time set events press "Events" button against proper time set.

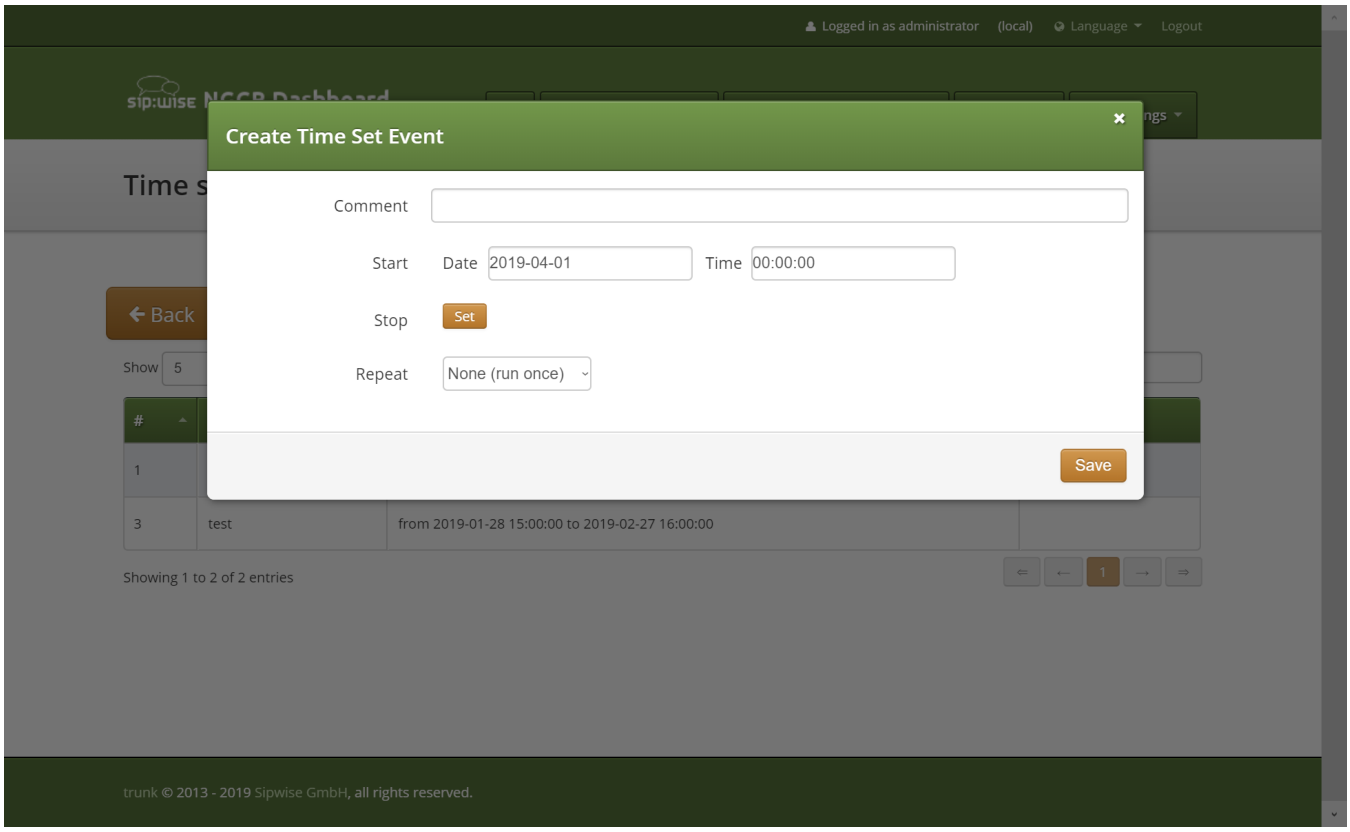


Events management section will appear:

To create event press "Create Event" button.



Form to create event will be shown:



Time set event form fields explanation

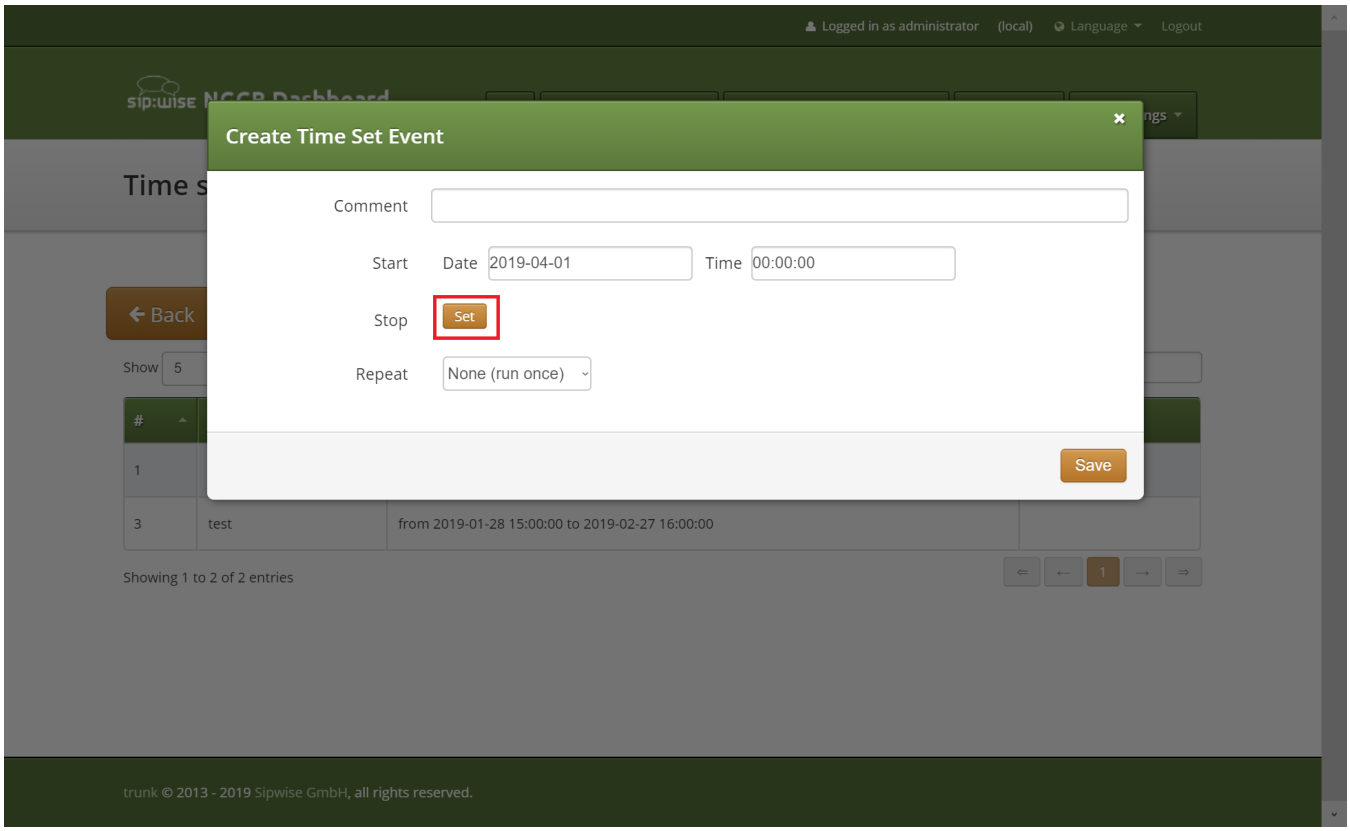
"Start" field reflects DTSTART property of the EVENT. "Start" is mandatory and by default is set to the start of the current day. "Start" value format is datetime.

"Stop" field reflects DTEND property of the EVENT. For the events within recurrence "Stop" will define

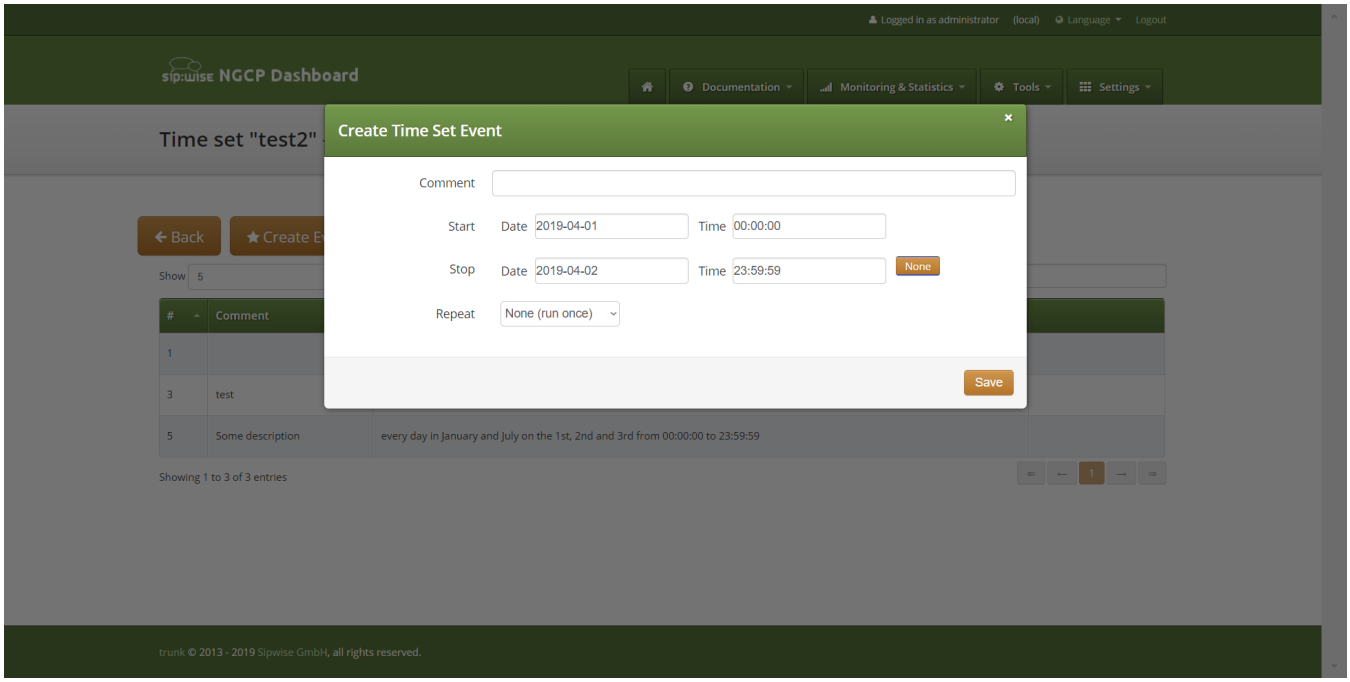
 Copyright © Sipwise GmbH - All rights reserved

duration of each iteration.

To specify "Stop" datetime, press button "Set".



Fields to enter DTEND date and time will appear:



To return "Stop" field to the empty value press button "None". Value in the form fields will be preserved, but newly created EVENT will have empty DTEND property.

Other fields in the form are optional. Most of them aren't visible by default and will be shown if

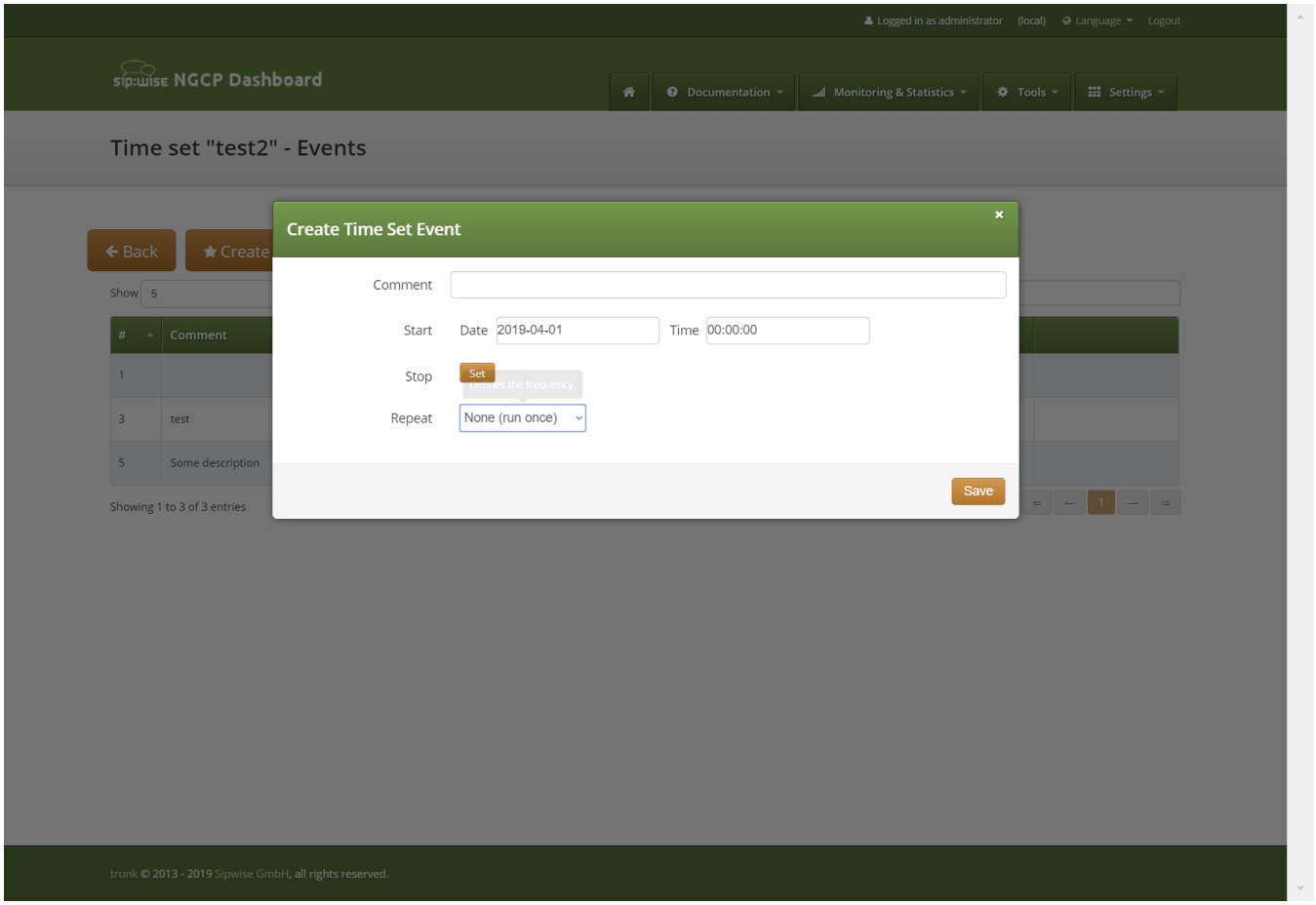
Copyright © Sipwise GmbH - All rights reserved

requested by user or required by data into other fields.

RRULE property of the EVENT is a recurrence rule and defines set of the iterations for the EVENT.

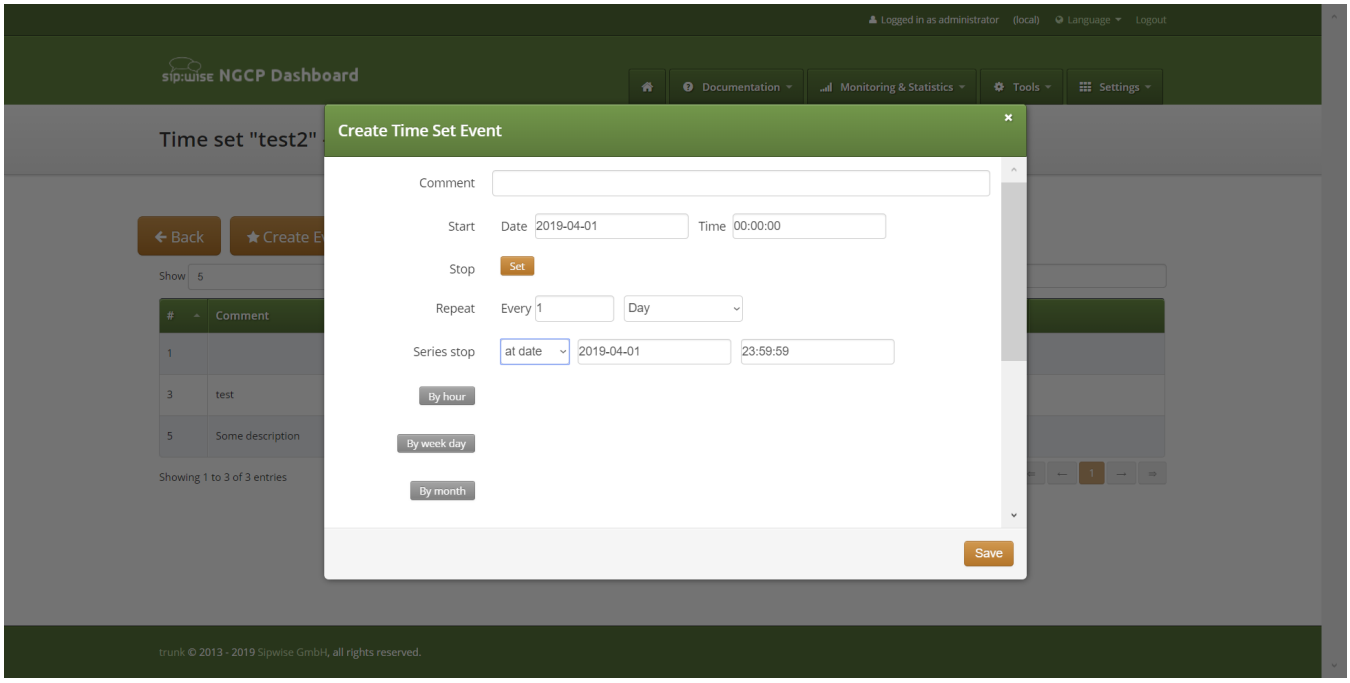
To customize recurrence rule for the EVENT select proper repetition unit for the "Repeat" form field. Input field for the recurrence interval will appear left to the frequency select.

According to the selected unit, FREQ property of the EVENT RRULE will be set to one of the: SECONDLY, MINUTELY, HOURLY, DAYLY, WEEKLY, MONTHLY, YEARLY.

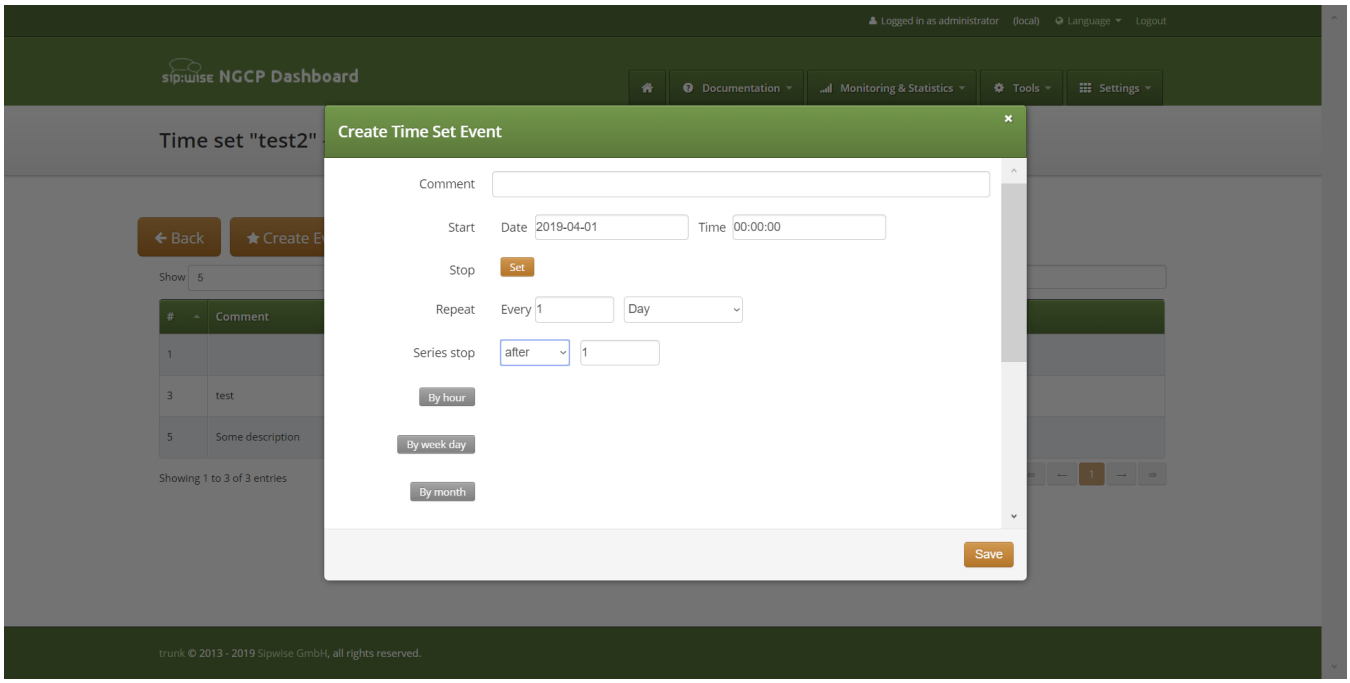


To specify end of the EVENT iterations, select "Series stop" value. For the "at date" option will be shown input for the date and time that will define UNTIL property of the EVENT RRULE.

 Copyright © Sipwise GmbH - All rights reserved

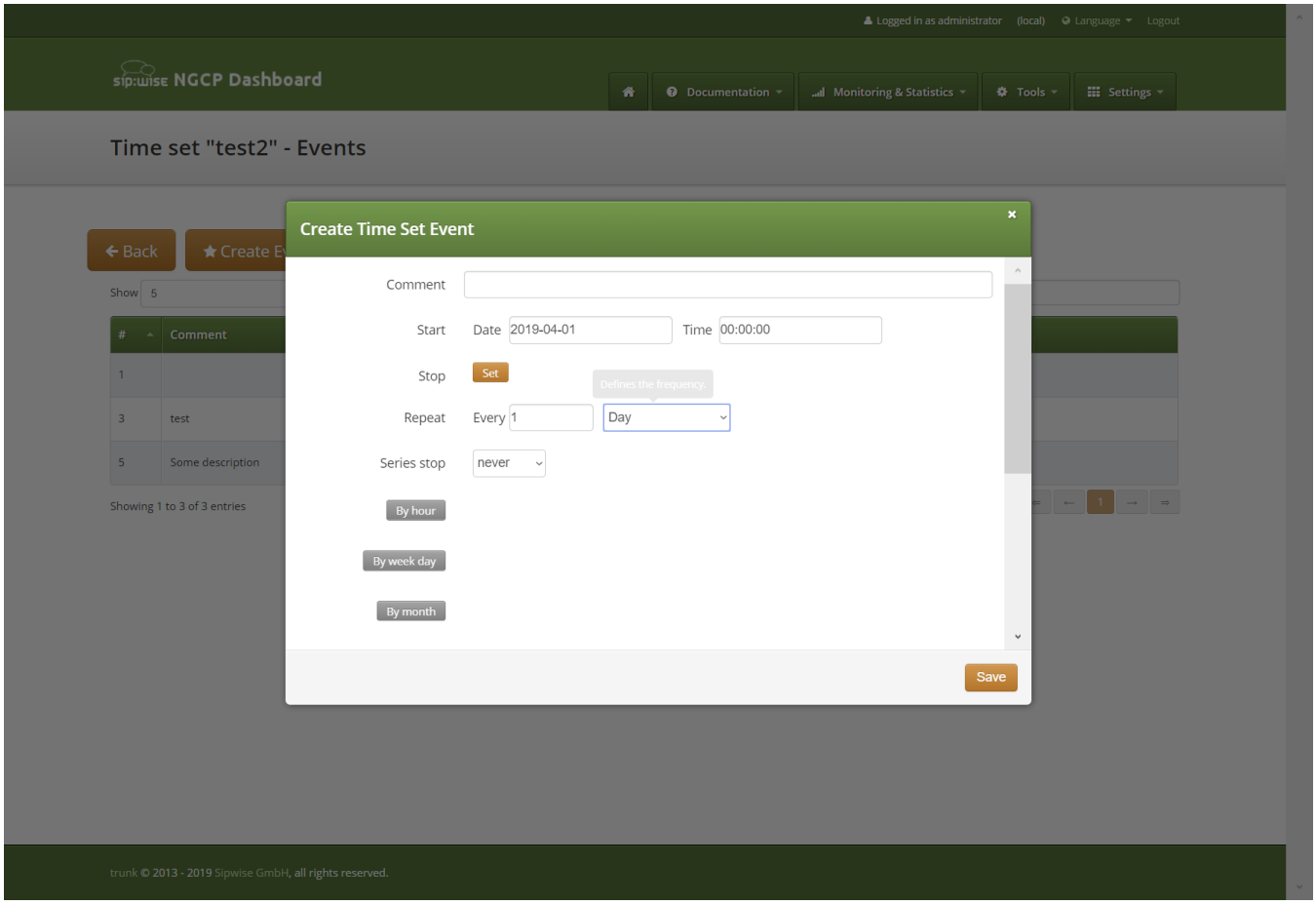


"after" option, respectively, will put entered value to the COUNT property of the EVENT RRULE.



Form fields "By hour", "By week day", "By month", "By month day", "By set position", "By week number", "By year day", "By second" and "By minute" aren't shown by default. To enter value to any of these fields press according button on the left. Button with field name is grayed off when corresponding EVENT property is empty.

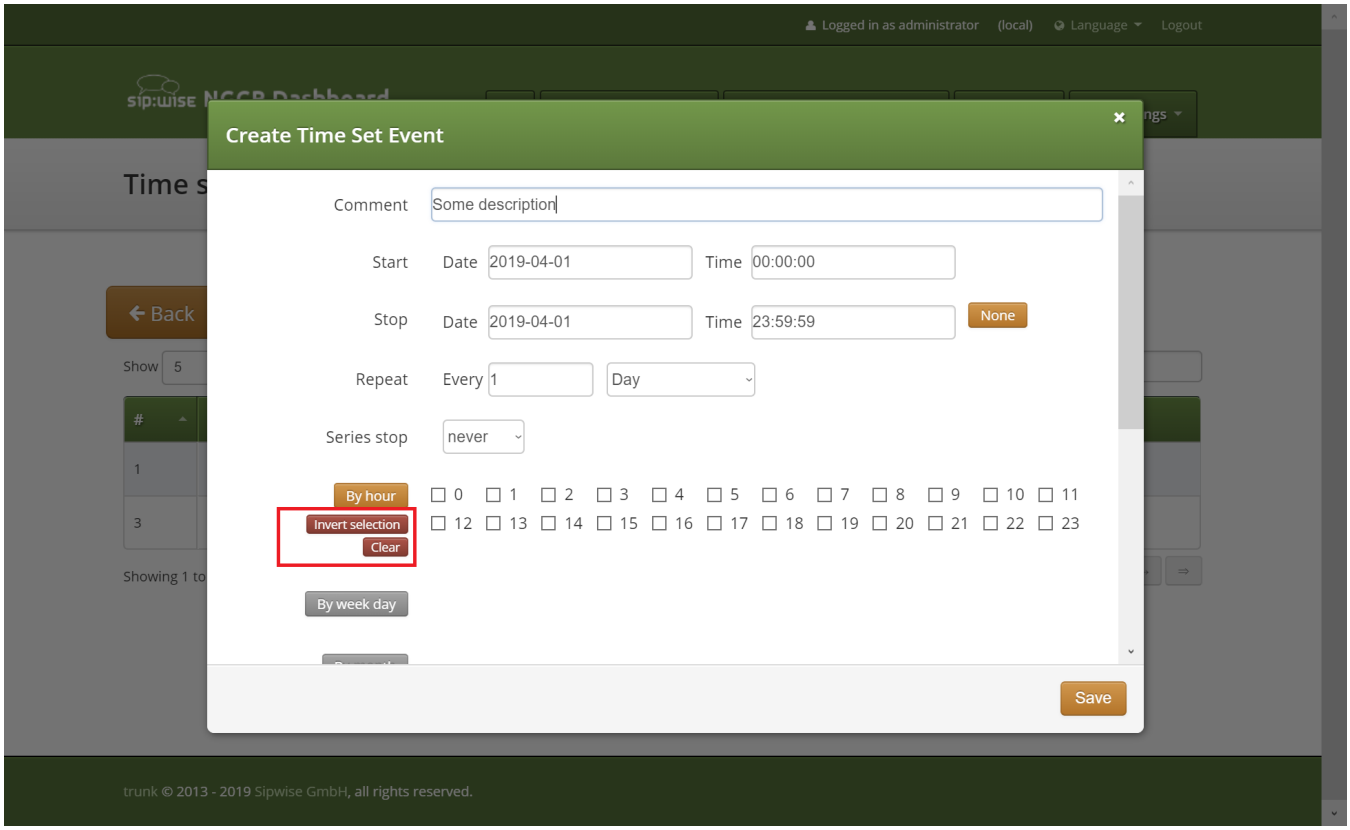
Copyright © Sipwise GmbH - All rights reserved



When gray button with field name is pressed, field input control appears on the right. In the same time button with field name becomes orange, indicating that field value will be saved for the EVENT.

Fields with checkboxes controls have auxiliary button "Invert selection". When button "Invert selection" is pressed currently empty checkboxes become selected and currently selected checkboxes become empty.

 Copyright © Sipwise GmbH - All rights reserved



When form data will be saved, checkboxes values will be saved as coma separated numbers.

BYxxx RRULE properties expand or limit behavior of the FREQ according to the table in the RRULE specification.

Field "By week day" has two variants of the input: checkbox for each week day and text input. Text input can be used, if "By week day" value is more complex than list of week days, separated by coma, for example for FREQ MONTHLY value "2TH,-3FR" in the "By week day" will mean second Thursday from the month start and third Friday from the month end in every month. Such value can't be presented as checkboxes selection.

Fields "By set position" and "By year day" are text inputs. Value format for these fields is set of the [+/-]NUMBER values, separated by comma.

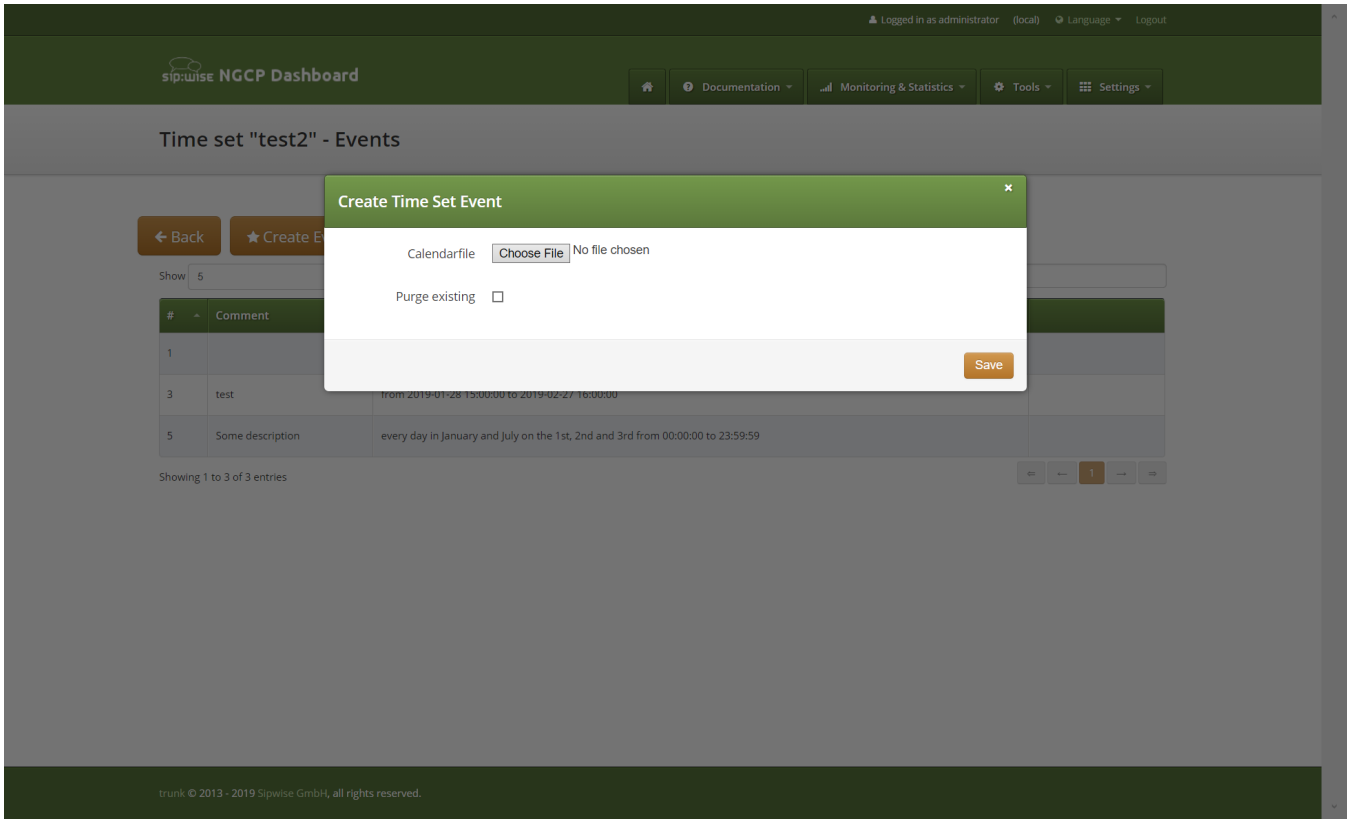
For the "By year day" minus sign in front of year day number means that this day should be taken by number from the end of the year.

For the "By set position" minus sign in front of the position of the iteration means that the iteration should be taken by number from the end of the generated iterations sequence.

After new event created, event will appear in time set event list. It will have column with rrule text description, buttons to request event edit form or event deletion.

In events list section all events can be redefined uploading ics iCalendar file:

Copyright © Sipwise GmbH - All rights reserved



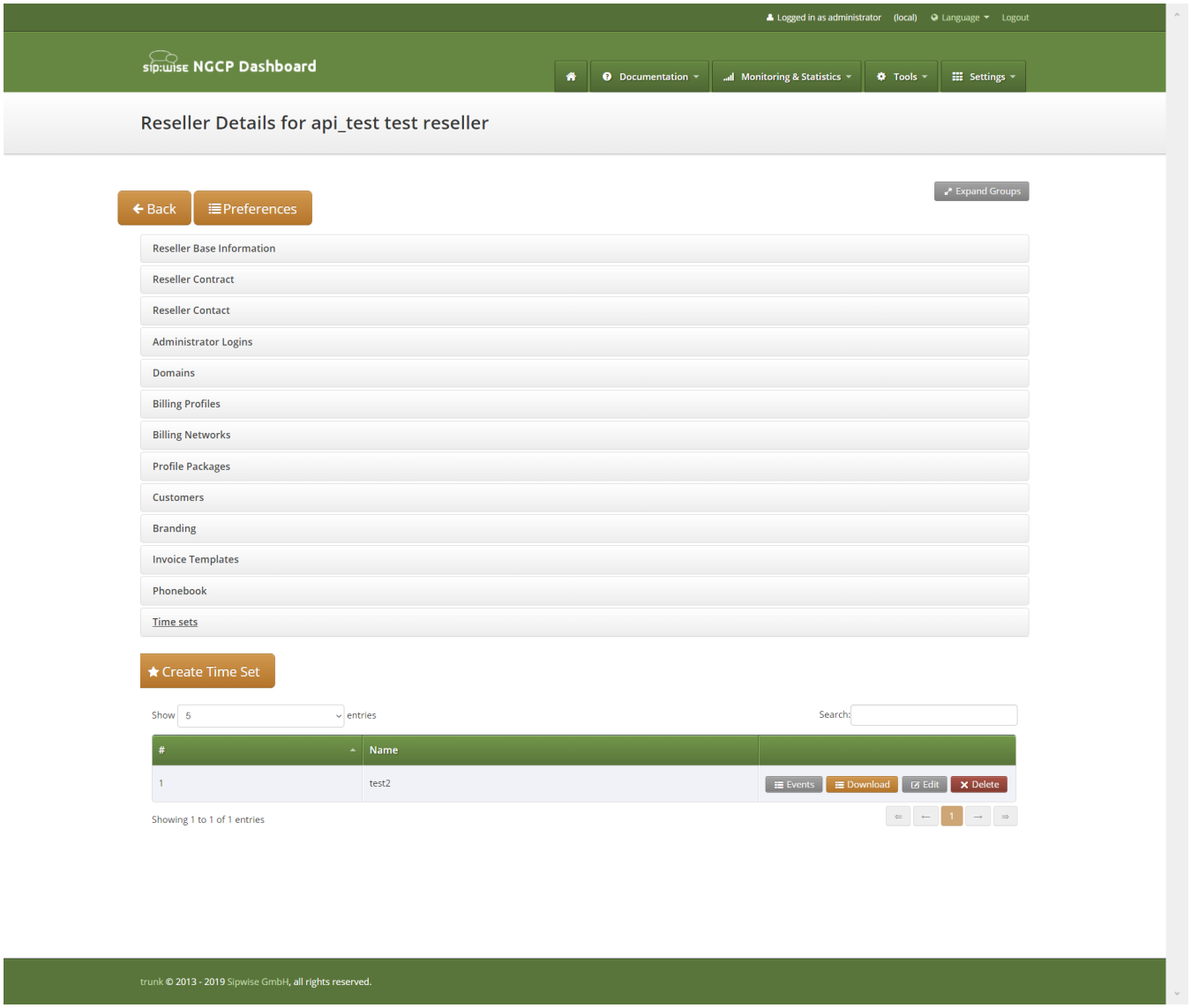
If "Purge existing" option is selected, all existing time set events will be removed before creation of the events from the uploaded file.

To download iCalendar ics file of the time set, press "Download iCalendar".

7.33.4. Web interface for time set related to reseller

Reseller details page provides list of the time sets connected to the reseller and allows create, edit, delete and download time set and has link to the time set events section:

 Copyright © Sipwise GmbH - All rights reserved



In difference to the main time set interface, iCalendar ics file for the time set can be downloaded from the time set list pressing "Download" button.

Creation form doesn't have "Reseller" field and is processed in context of the current reseller.

7.33.5. REST API

Time sets management is possible using API REST entry point /api/timesets/.

Time sets API has possibility to get and return information both as "application/json" data and as "text/calendar" file.

To create time set with events full specification of the all fields in json format can be used:

curl --request POST --user administrator:administrator --header 'Prefer: return=representation' --header 'Content-Type: application/json' 'https://127.0.0.1:1443/api/timesets/' --data '{"reseller_id":"3", "name":"api_test_timeset_name1","times":[{"start":"1971-01-01 00:00:01","until":"1997-01-01 23:59:59","end":"2020-12-31 23:59:59"}]}'

Also time set and events can be uploaded as ics iCalendar file:

Copyright © Sipwise GmbH - All rights reserved

curl --request POST --user administrator:administrator --header 'Prefer: return=representation' --header 'Content-Type: multipart/form-data' 'https://127.0.0.1:1443/api/timesets/' --form 'json={"reseller_id":3,"name":"unique_name"}' --form 'calendarfile=@/path/to/calendar.ics'

Output of the GET request to the time set item can be text/calendar:

curl --request GET --user administrator:administrator --header 'Accept: text/calendar' 'https://127.0.0.1:1443/api/timesets/12' \> /path/to/download/calendar.ics

or application/json:

curl --request GET --user administrator:administrator --header 'Accept: application/json' 'https://127.0.0.1:1443/api/timesets/12'

By default API will send response in text/calendar format.

Output will be generated iCalendar including time set events:

```
curl --request GET --user administrator:administrator
'https://127.0.0.1:1443/api/timesets/12'
BEGIN:VCALENDAR
PRODID:-//Mozilla.org/NONSGML Mozilla Calendar V1.1//EN
NAME:api_test_timeset_name2
VERSION:2.0

BEGIN:VEVENT
UID:sipwise19@sipwise15
SUMMARY:unique_name event 19
DTSTART:19710101T000001
DTEND:20201231T235959
END:VEVENT
END:VCALENDAR+
```

 Copyright © Sipwise GmbH - All rights reserved

7.34. Announcement To Callee

This feature allows a caller to play a custom announcement to the callee every time it performs a call. The announcement is played to the callee immediately after it answers the call (200OK is received by Sipwise C5), therefore it can be used to provide some information about the caller. Meanwhile the callee is listening the announcement, the caller is still in ringing status. The parties will be put in connection right after the end of the announcement.

The configuration of the announcement is similar to the activation of Pre-Recording Announcement and it requires few simple steps.

First create a system sound set for the feature. In *Settings Sound Sets* either use your already existing *Sound Set* or create a new *Sound Set* and then assign it to your domain or subscribers. In the *Sound Set* there is an announcement *early_media announce_to_callee* for that purpose.

Once the *Sound Set* is created the caller subscriber's preference *play_announce_to_callee* must be enabled under *Subscriber Preferences Applications* menu. The same parameter can be set in the Domain's or Customer's preferences to enable this feature for all its subscribers.

NOTE The announcement length is limited to 30 seconds.

IMPORTANT Differently from *Pre-Recording Announcement*, in all **Call Forward** cases with *Announcement To Callee* feature enabled on both the caller and the forwarder, only the announcement of the forwarder will be played to callee.

Copyright © Sipwise GmbH - All rights reserved

7.35. Header Manipulations

7.35.1. Overview

Header Manipulations feature enables a flexible way to modify headers of SIP messages when it is being processed by the SIP proxy. That helps with scenarios where based on specific header conditions, certain header changes must take place (e.g.: If "From" does not match expected number or a format and there is no Diversion header then the P-Asserted-Identity header must be modified (or added) with the current subscriber's network provided CLI number). Another example is when there is a faulty User-Agent sending a malformed Contact entry then, based on the User-Agent header value and Contact header format, it is becomes possible to detect such scenario, and fix the Contact header on the fly without directly modifying the proxy logic.

Header Manipulations (also called in the UI/API as **Header Rules**) consist of:

- **Sets** - belong to a reseller, contain **Rules** and can be assigned to `Domains`, `Subscribers` and `Peer Hosts`.
- **Rules** - belong to **Sets** and contains **Conditions** and **Actions**
- **Conditions** - contain header expressions to be evaluated, if any **Condition** returns `False` then the whole **Rule** is immediately ignored.
- **Actions** - contain actions that are applied to the headers if all **Conditions** of the **Rule** are evaluated as `True`

7.35.2. Sets

Set is a topmost level entry and used in `Domain` / `Peer` / `Subscriber` based header manipulation scenarios. Only one Set can be assigned per `Domain` / `Peer` / `Subscriber` but the same Set can be assigned to any of them simultaneously.

- Reseller (for platform administrators): reseller_id the Set belongs to
- Name: Set name
- Description: custom description

7.35.3. Rules

Rule should have at least one Condition and Action to be taken into account. All Conditions of the Rule must match, otherwise the Rule is skipped.

- Name: Rule name
- Description: custom description
- Priority: a number that defines priority of the Rule. Smaller numbers have higher priority. All Rules within the same Set are evaluated by priority and as such it is important to have them ordered as expected
- Direction: defines when the Rule is used

    Inbound: applied when a SIP message is received by the proxy from the load balanced, the Set to be applied is picked from the `caller` preferences

    Outbound: applied when a SIP message is about to leave the proxy, where Set to be applied is

 Copyright © Sipwise GmbH - All rights reserved

picked from the `callee` preferences

Local: applied when a SIP message is going to be routed to a local, where Set to be applied is picked from the `caller` preferences

Peer: applied when a SIP message is going to be routed to a peering, where Set to be applied is picked from the `caller` preferences

Call Forward Inbound: applied when a SIP message is coming via the call forwarding, where Set to be applied is picked from the `caller` preferences

Call Forward Outbound: applied when a call forwarding is triggered, where Set to be applied is picked from the `callee` preferences

Reply: applied when a SIP reply message is received by the proxy, where Set to be applied is picked from the `caller` preferences

- Stopper: can be either `True` or `False`. When set to `True` and the Rule is successful then no further Rules are processed within the given Set
- Enabled: can be either `True` or `False` and defines whether to include the Rule into processing within the given Set

7.35.4. Conditions

Condition contains one or many header validation expressions. Conditions do not modify any header data, only evaluate it. Conditions also operate with internal proxy runtime variables ($avp,$xavp) and they can be used to compare for instance a header value with an $avp value (those are for expert use only).

- Match: what to evaluate

  header: header value

  preference: subscriber or peering preference

  avp: $avp variable
- Part: if the header (or $avp) value is a SIP-URI then it is possible to pick which part of it should be taken for the evaluation:

  full: the entire value

  username: SIP username

  domain: SIP domain

  port: SIP port
- Name: header or $avp name
- Expression: expression that is used to compare the extracted value

  is: strict string comparison

  contains: if the value contains the matching part

  matches: same as contains but also accepts `*` to be used as none to many characters and `?` as any single character

  regex: a regular expression
- Not: if set to `True` then the Condition returns `True` if the evaluation fails
- Type: user value type

Copyright © Sipwise GmbH - All rights reserved

input: raw string

preference: preference name, in this case the value(s) is taken from the preference. If the preference contains more than one value then all of them are evaluated until first match

avp: $avp name, in this case the value(s) is taken from the $avp. If the $avp contains more than one value then all of them are evaluated until first match

- Value: one or many values, **NOTE:** supports inline $avp transformations, e.g: if `$avp(s:source_cli) = 456` then `123$avp(s:source_cli)789` is evaluated as `123456789`
- Enabled: skipped if set to `False`
- Rewrite Rule Set: if the header value needs to be normalised before evaluation then an existing Rewrite Rule Set can be used for that, it is mandatory to select a header rules part when the option is used
- Rewrite Rule: Rewrite Rules to use from the selected Rewrite Rule Set

Inbound for Caller

Inbound for Callee

Outbound for Caller

Outbound for Callee

7.35.5. Actions

Action contains one or many changes that are applied to headers if the Rule is successful, it is also possible to modify the internal proxy $avp runtime values (expert use only). Unlike Conditions, all Actions are applied regardless if some cannot be applied (e.g.: a header to remove does not exist).

- Priority: a number that defines priority of the Action. Smaller numbers have higher priority. The order is important when for instance you copy data between headers or need to add a header if it does not exist yet
- Header: header or $avp name to apply actions to
- Header Part: if the header (or $avp) value is a SIP-URI then it is possible to pick which part of the value needs to be changed

full: the entire value

username: SIP username

domain: SIP domain

port: SIP port

- Type: type of action

add: add a new header. If the header already exists the action is skipped

set: replace value of an existing header. If the header does not exist the action is skipped

remove: remove an existing header

header: copy a value (or a part) from an existing header to this header

preference: copy a value (or a part) from an existing preference to this header

rsub: apply regex substring to the current header. format: `match;replace` (e.g: value=`1234567` rsub=`^12([0-9]45)67$;$1 result=345`)

Copyright © Sipwise GmbH - All rights reserved

- Value Part: if the value is a SIP-URI then it is possible to pick which part of the value should be used

  full: the entire value

  username: SIP username

  domain: SIP domain

  port: SIP port

- Value: value to apply or a header name if Type=header, or a preference name if Type=preference. **NOTE:** support inline $avp transformations, e.g: if `$avp(s:source_cli)=456` then `123$avp(s:source_cli)789` is evaluated as `123456789`
- Rewrite Rule Set: if the header value needs to be normalised after evaluation then an existing Rewrite Rule Set can be used for that, it is mandatory to select a header rules part when the option is used
- Rewrite Rule: Rewrite Rules to use from the selected Rewrite Rule Set

  Inbound for Caller

  Inbound for Callee

  Outbound for Caller

  Outbound for Callee

7.35.6. Special Headers

There are special header names, they can be use in conditions and set in actions. The names are case insensitive.

- @Request-URI: retrieve or modify RURI of the SIP message
- @Reply-Status: retrieve or modify the reply status (e.g: 486). Can only be used in the reply direction. **NOTE:**: reply status 1xx and 2xx cannot be changed as well as cannot be set
- @Reply-Reason: retrieve or modify the reply reason (e.g: Request Terminated). Can only be used in the reply direction

7.35.7. Usage

To start using Header Manipulations a Set needs to be created. Then one or more Rules should be created with a Direction depending on when you need modify the headers. If there are headers to adjust before any proxy logic processing takes place, then Direction `inbound` should be used. If there are headers to adjust right before the SIP message leaves the proxy then Direction `outbound` should be used. Once you created a Rule it is time to add at least one Condition and one Action, otherwise the Rule is skipped. Condition is an expression that you use to check if one or more headers of the SIP message (or an $avp in expert cases) exist and their values match the expression, otherwise the Rule is skipped and next one with the same Direction and by Priority is evaluated. If all Conditions of the Rule are evaluated as `True` then all Actions of the Rule are applied. If the Stopper flag is 'True' then the processing ends, otherwise next Rule in line is evaluated.

Set can be assigned to a domain in the `Domain preferences` and is automatically inherited by all subscribers of the domain. It can be also applied to the `Peering preferences`. It is possible to override the Set per subscriber in the `Subscriber preferences`.

It is also possible to have `Subscriber only Rules`, they are created in the admin UI via the

Copyright © Sipwise GmbH - All rights reserved

`Subscriber preferences` and in the API they are created via `/api/headerrulesets`. Internally it is a Set but with a defined subscriber_id. It is only possible to have one Set like this per subscriber. It is automatically created when used from the admin UI and you only work on the Rules level. This is useful when there is something specific that needs to be modified in the headers for particular subscriber(s). When Rules are applied in the logic the Domain/Subscriber Set is applied first and then per subscriber defined Rules if defined.

7.35.8. Usage Examples

Inbound Call Add Diversion header

Goal: if `From` username starts with 43, add `Diversion` header if it does not exist and skip if already exists

```
Rule:
  Name: add_diversion
  Description: Add Diversion
  Priority: 1
  Direction: inbound
  Stopper: 0
  Enabled: 1

  Conditions:
    -
      Match: header
      Part: username
      Name: From
      Expression: matches
      Not: 0
      Type: input
      Values:
        43*
      Enabled: 1
      Rewrite Rule Set:
      Rewrite Rule:

  Actions:
    -
      Priority: 1
      Header: Diversion
      Header Part: full
      Type: add
      Value Part: full
      Value: sip:431001@sipwise.com
      Rewrite Rule Set:
      Rewrite Rule:
```

Outbound Add or Replace X-Test header

Goal: if `From` username equals to subscriber preference `cli`, add `X-Test` header if it does not exist or

Copyright © Sipwise GmbH - All rights reserved

replace it if exists

```
Rule:
  Name: add_replace_x_test
  Description: Add or Replace X-Test
  Priority: 1
  Direction: outbound
  Stopper: 0
  Enabled: 1

  Conditions:
    -
      Match: header
      Part: username
      Name: From
      Expression: is
      Not: 0
      Type: preference
      Values:
        cli
      Enabled: 1
      Rewrite Rule Set:
      Rewrite Rule:

  Actions:
    -
      Priority: 1
      Header: X-Test
      Header Part: full
      Type: add
      Value Part: full
      Value: sip:430001@sipwise.com
      Rewrite Rule Set:
      Rewrite Rule:
    -
      Priority: 2
      Header: X-Test
      Header Part: full
      Type: set
      Value Part: full
      Value: sip:430001@sipwise.com
      Rewrite Rule Set:
      Rewrite Rule:
```

Remove P-Asserted-Identiy, Replace Diversion, Add X-Test

Goal: if a call is terminated to a local subscriber and `P-Asserted-Identity` exists, and its domain part contains sipwise.com or sipwise.local, Replace Diversion with the value from P-Asserted-Identity, remove P-Asserted-Identity and add X-Test with a value from the `cli` preference

Copyright © Sipwise GmbH - All rights reserved

```
Rule:
  Name: local_pai_diversion_x_test
  Description: Local remove PAI, replace Diversion, add X-Test
  Priority: 1
  Direction: local
  Stopper: 0
  Enabled: 1

  Conditions:
    -
      Match: header
      Part: domain
      Name: P-Asserted-Identity
      Expression: contains
      Not: 0
      Type: input
      Values:
        sipwise.com
        sipwise.local
      Enabled: 1
      Rewrite Rule Set:
      Rewrite Rule:

  Actions:
    -
      Priority: 1
      Header: Diversion
      Header Part: full
      Type: header
      Value Part: full
      Value: P-Asserted-Identity
      Rewrite Rule Set:
      Rewrite Rule:
    -
      Priority: 2
      Header: P-Asserted-Identity
      Header Part: full
      Type: remove
      Value Part: full
      Value:
      Rewrite Rule Set:
      Rewrite Rule:
    -
      Priority: 3
      Header: X-Test
      Header Part: full
      Type: preference
      Value Part: full
      Value: cli
      Rewrite Rule Set:
      Rewrite Rule:
```


Copyright © Sipwise GmbH - All rights reserved

7.36. Phonebook

7.36.1. Overview

Phonebook enables on the NGCP platform a configurable list of entries (namenumber) per Reseller, Customer or Subscriber that are provisioned on the subscribers' devices (phones) and presented there as an external phonebook.

7.36.2. Inheritance

- Entries that are defined on Reseller level are automatically included into the subscriber's phonebook unless overridden on Customer or Subscriber levels.
- Entries that are defined on Customer level override matching entries from Reseller level and automatically included into the subscriber's phonebook unless overridden on Subscriber level.
- Entries that are defined on Subscriber level override matching entries from Reseller and Customer levels and automatically included into the subscriber's phonebook.

```
Overrides uses the number field for matching.
```

7.36.3. Reseller Phonebook

To manage Reseller Phonebook you must be an admin user, there is a Phonebook entry in Settings dropdown list.



Figure 94. Create Reseller Phonebook Entry

- Name: Enter a name for the entry (e.g.: Emergency Number or Alice)

Copyright © Sipwise GmbH - All rights reserved

- Number: E164 number. The format is free but E164 is preferred so that the entries can be overridden when needed on Customer or Subscriber levels.

7.36.4. Customer Phonebook

Customer Phonebook is located as one of the available Customer detail entries.

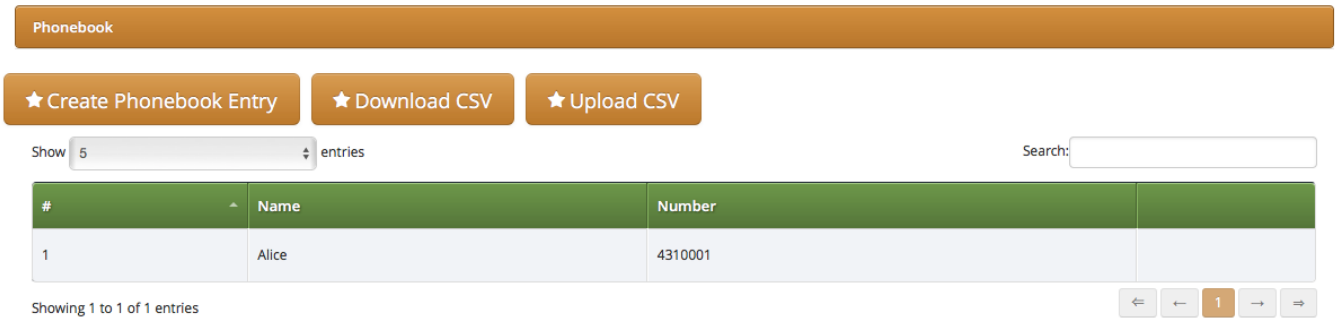


Figure 95. Customer Phonebook Entries

In the given example there is Alice number defined in the phonebook, so it is shared across all subscirbers of the customer, plus numbers from Reseller Phonebook.

7.36.5. Subscriber Phonebook

Subscriber Phonebook is located as one of the available Subscriber detail entries.

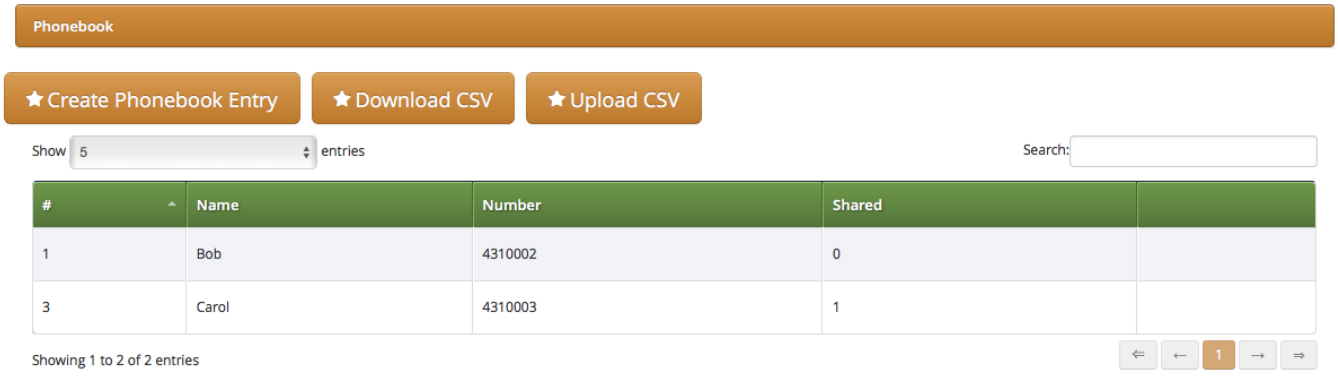


Figure 96. Subscriber Phonebook Entries

In the given example there is Bob and Carol numbers defined in the phonebook.

There is a new field on Subscriber level, 'Shared', it defines whether the number is private to the subscriber, if 'Shared=0', or shared across other subscribers of the same customer if 'Shared=1'.

Therefore, Bob is private to the subscriber and Carol entry is shared.

7.36.6. Using CSV Upload and Download

Clicking 'Download CSV' automatically downloads all entries of the level into a csv file. The downloaded entries are exactly ones in the phonebook and inheritance is not used in this case.

Clicking 'Upload CSV' opens a dialog.

 Copyright © Sipwise GmbH - All rights reserved

- Upload phonebook: choose a local .csv file with expected fields '"Name"','Number', and 'Shared' if uploaded into Subscriber level. It is a good practice to quote name to resolve situations with spaces. Commas must be escapted with \.

```
phonebook_upload.csv
"Shaw\, Alice",4310001
```

```
phonebook_upload_subscriber.csv
"Shaw\, Alice",4310001,0
"Bob",4310002,1
```

- Purge existing: if checked, deletes all existing entries before uploading new ones, otherwise the entries are merged by the Number field, where new entries override old entries on match.

7.36.7. Manually enabling Phonebook in a PBX device

Currently supported devices:

- Cisco SPA
- Panasonic
- Yealink
  1. In your provisioned PBX device, fetch its MAC address.
  2. Open Directory/Remote Phonebook and enter one of the supported URLs depending on the device model:
- Cisco SPA: https://$host:$port:/pbx/directory/spa/$mac
- Panasonic: https://$host:$port:/pbx/directory/panasonic/userid=$mac
- Yealink: https://$host:$port:/pbx/directory/yealink/userid=$mac

Where host is the NGCP host and port is the API configured port.

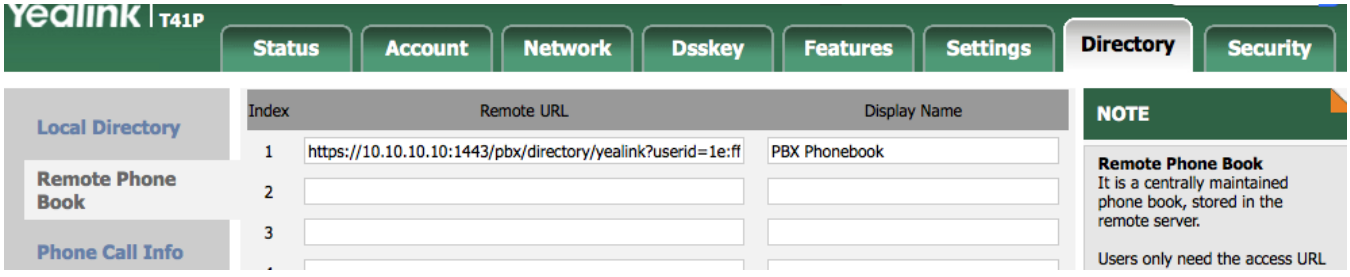


Figure 97. Phonebook Device Provisioning

Copyright © Sipwise GmbH - All rights reserved

7.37. Subscriber Location Mappings

7.37.1. Overview

Subscriber Location Mappings enables a possibility for incoming calls on a subscriber to be also routed and handled on additional "locations".

A location can represent:

- A SIP-URI of an NGCP subscriber, which may have one or more endpoints registered to it.
- Remote peering destinations.

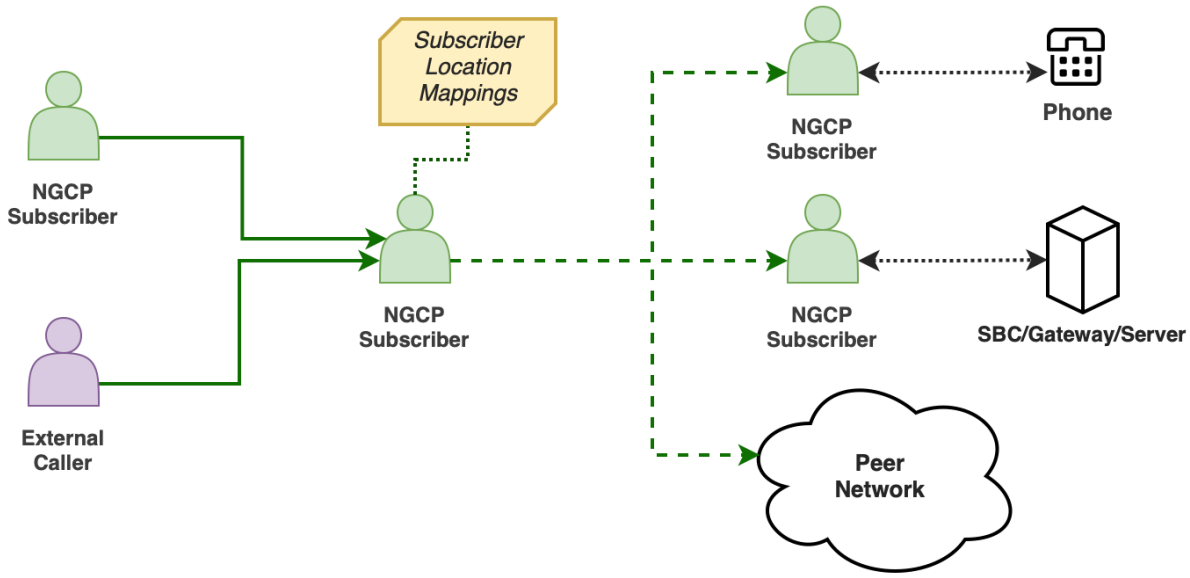


Figure 98. Subscriber Location Mappings - Overall View.

The feature can be set within subscriber preferences, via Admin Panel or REST API.

Its main advantages are as follows:

(a) Definition of Several Call Branches.

By the definition of one ore more location entries on a subscriber, it is possible to generate separate call branches for each destination when an incoming call is received on that subscriber.

(b) Set Triggering Conditions and Mode for Each Call Branch.

In order to trigger a call branch, conditions can be defined for both caller and callee numbers via regular expressions, as well as the definition of the call branch behavior to be applied.

(c) Number Manipulations.

Number manipulations can be applied on the *username* part of the *Request-URI* (RURI) or the `To` field in a SIP INVITE message.

Example Scenarios

There may be different scenarios in which the feature can be considered. Some examples, as follows:

Copyright © Sipwise GmbH - All rights reserved

Table 23. Scenario Examples.

Scenario	Description
NGCP Subscriber	• Incoming calls on an NGCP subscriber are (also) delivered to another NGCP subscriber. • Two cases are possible: re-routed or multiplied calls.
External Gateway	• There is a Session Border Controller (SBC) with a registration in NGCP that is intended to receive calls for NGCP subscribers. • Incoming calls on an NGCP subscriber are also routed to the SBC. • The SBC may route the calls to other PBX.
Remote Peer	• Incoming calls on an NGCP subscriber are (also) delivered to a remote peer according to the outbound peering logic in place. • The remote peer will receive the call with source equals to the original caller.

The following sections will provide more details about the feature.

7.37.2. Fields

Location mappings fields are summarized in the following table.

Table 24. Location Mapping Fields.

Field	Description
Location SIP-URI	This field has two possible value options: • A SIP-URI of an NGCP subscriber (i.e. *sip:user@domain*). Mandatory value for the `Add`, `Replace` and `Offline` modes. See "*Mode*" below. *NOTE: This entry does not refer to a SIP-URI with location binding address.* • A fictitious or an empty name for the `Forward` mode (The value is ignored during the branch creation). See "*Mode*" below.
Caller Pattern	A regular expression pattern to match the `From` field of the INVITE (After Inbound Rewrite Rules). If left empty, it defaults to the `.+` pattern.
Callee Pattern	A regular expression pattern to match the `R-URI`/`To` of the INVITE (After Inbound Rewrite Rules). If left empty, it defaults to the `.+` pattern.

Copyright © Sipwise GmbH - All rights reserved

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Mode</td><td>Defines the behavior of the matched entry during the call termination:
<table>
<tr><td><code>Add</code></td><td>It appends the entry into the destinations list.
The defined Location SIP-URI must have registered endpoints.
Multiple <code>Add</code> entries matched are added into the call branching process.</td></tr>
<tr><td><code>Replace</code></td><td>It replaces the subscriber' SIP-URI as destination by this entry.
The defined Location SIP-URI must have registered endpoints.</td></tr>
<tr><td><code>Offline</code></td><td>It appends the entry into the destinations list ONLY if the subscriber is offline (i.e. there are no registration records for the subscriber).</td></tr>
<tr><td><code>Forward</code></td><td>It appends the entry into the destinations list for a remote peer.
If this mode is set, is not possible to call to another local subscriber.</td></tr>
</table>
"Mode" field is mandatory.</td></tr>
<tr><td>RURI/To Username</td><td>Replaces the username part of the <code>RURI/To</code> in a SIP INVITE with the value defined on this field. Useful for cases when the remote username is different than the one used for the location mapping.</td></tr>
<tr><td>External ID</td><td>An arbitrary string name (e.g. an identifier of a 3rd party provisioning/billing system). This is an optional attribute, which can be kept as empty.</td></tr>
<tr><td>Enabled</td><td>Enable/Disable the entry.</td></tr>
</table>

About The 'Forward' Mode

The location mappings "*Forward*" mode works differently when compared to the others:

- This mode does not require the *Location SIP-URI* field, since it applies outbound peering logic to select the correct peer and automatically calculates the final destination.
- It creates a new parallel branch meant for peers only and it will behave like a Call Forward Unconditional (CFU). (NOTE: If you need to generate an internal call to local subscribers, use the `Add` mode instead.)
- The remote peer will receive the original caller as the source of the call (NOTE: User Provided Redirecting Number is intentionally disabled for this particular mode). Also, two different Call Detail Records (CDR) will be created: the first one from 'A' to 'B' (in which the CDR `call_type` field equals

 Copyright © Sipwise GmbH - All rights reserved

to '*call*'), and the second one from the original callee subscriber to the peer (with `call_type` = '*cfu*').

- To prevent the call to be looped back, the new parallel branch is terminated if the inbound peer belongs to the same peer group of the outbound peer. However, it is possible to disable this check by activating the `allow_lm_forward_loop` peer preference ("*Internals*" section) of the outgoing peer.

7.37.3. Provisioning via Admin Panel

Subscriber Location Mappings can be accessed at *Subscriber Preferences Location Mappings* section. The interface will offer the option to create a new entry, or edit any of the existing ones.

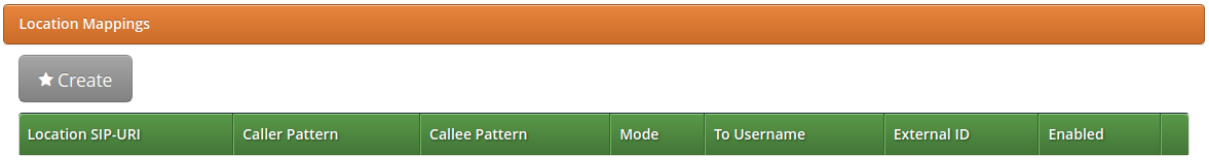


Figure 99. Location Mappings Section.

For creation, press on the "*Create*" button, which will pop-up a new window.

As shown in the example below, data can be entered and saved to create the new entry.

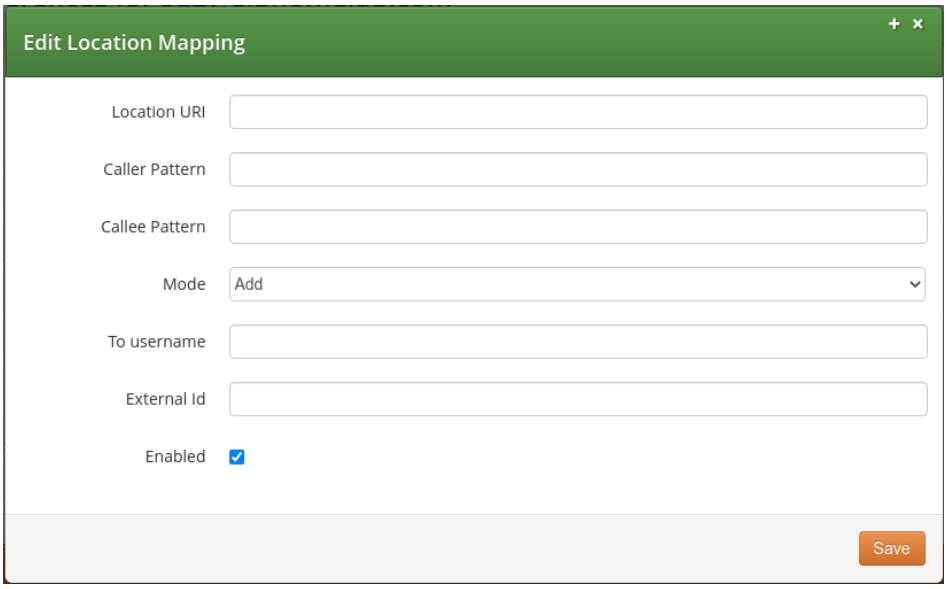


Figure 100. Subscriber Location Mappings: Create Entry.

Parameters are the ones described on the "*Fields*" section.

Practical Example

Let us assume that an administrator will use Admin Panel to provision some location mappings on a selected subscriber. The following *Subscriber Details* are considered:

Table 25. Practical Example - Subscriber Details.

Copyright © Sipwise GmbH - All rights reserved

Field	Value
SIP URI	sip:su1@siphomelab.com
Primary Number	4312521523
Alias Numbers	4312521523001 4312521523002

Then, on *Subscriber Preferences*, within the *Location Mappings* section, the following entries are defined:

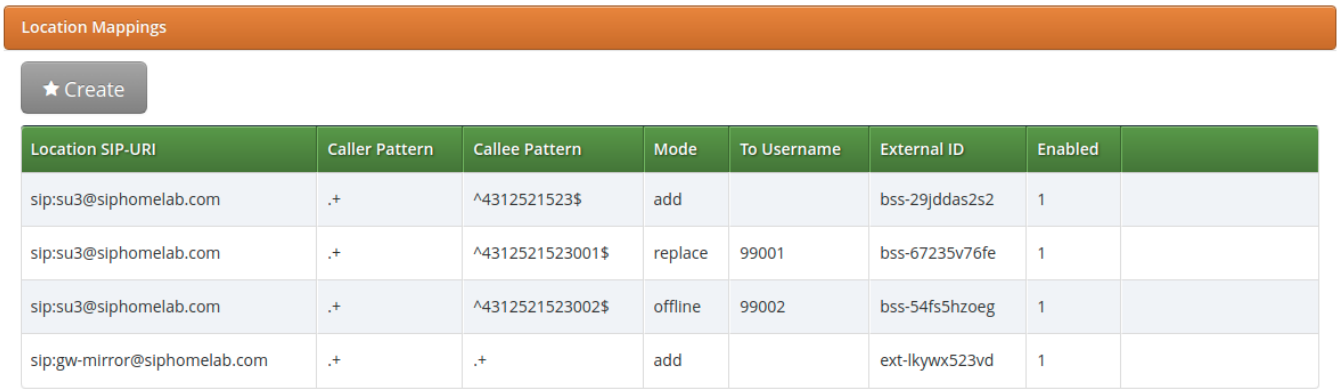


Figure 101. Practical Example - Subscriber Location Mappings.

In the provided example, the logic is as following:

Table 26. Practical Example - Logic & Result.

Logic	Result
• An incoming call to the subscriber number 4312521523 will be additionally sent to the devices registered with *sip:su3@siphomelab.com*.	• The call will ring on the registered devices of *sip:su1@siphomelab.com*. • The call will ring on the registered devices of *sip:su3@siphomelab.com*. • The call will ring on the registered devices of *sip:gw-mirror@siphomelab.com*. • When a device takes the call, the call will be cancelled on the rest of devices.
• An incoming call to the subscriber number 4312521523001 will replace the current subscriber SIP-URI as destination (i.e. the call will not be terminated at the devices registered with subscriber *sip:su1@siphomelab.com*). • Instead, the call will be sent to the registered devices with location entry *sip:su3@siphomelab.com* with the "*username*" part of the INVITE replaced with number 990001.	• The call will NOT ring on the registered devices of *sip:su1@siphomelab.com*. • The call will ring on the registered devices of *sip:su3@siphomelab.com* with the number 990001. • The call will ring on the registered devices of *sip:gw-mirror@siphomelab.com*. • When a device takes the call, the call will be cancelled on the rest of devices.

 Copyright © Sipwise GmbH - All rights reserved

Logic	Result
• An incoming call to the subscriber number `4312521523002` will be additionally sent to the devices registered with *sip:su3@siphomelab.com* only if there are NO active registrations under this subscriber (i.e. *sip:su1@siphomelab.com* is offline). • Also, the "*username*" part of the INVITE will be replaced with number `990002`.	• If *sip:su1@siphomelab.com* is offline, then the call will ring on the registered devices of *sip:su3@siphomelab.com* with the number `990002`. • The call will ring on the registered devices of *sip:gw-mirror@siphomelab.com*. • When a device takes the call, the call will be cancelled on the rest of devices.
• Any incoming call to the subscriber *sip:su1@siphomelab.com* will be additionally sent to the devices registered with *sip:gw-mirror@siphomelab.com*.	• When a device takes the call, the call will be cancelled on the rest of devices.

Copyright © Sipwise GmbH - All rights reserved

7.38. STIR/SHAKEN

7.38.1. Overview

What does STIR/SHAKEN stand for?

STIR - Secure Telephony Identity Revisited (RFC8224, RFC8225 and RFC8226)

SHAKEN - Secure Handling of Asserted Information using tokens (RFC8588)

This technology gives an opportunity to make sure that the calling side (caller's number) is not spoofed, hence can help during the common work/inter-cooperation between ITSPs, in terms of call spoofing detection. Therefore when it is massively used by ITSPs, it decreases the amount of robo-calls.

NOTE ITSP - Internet Telephony Service Provider. Is a general abbreviation for VoIP service providers, which can use IP-based technologies/protocols such as SIP protocol or H.323 stack of protocols to provide telephony services.

The Sipwise C5 is currently compliant with the list of general RFC standards:

- RFC8224 - Authenticated Identity Management in the Session Initiation Protocol (SIP) (this RFC obsoletes RFC 4474) ;
- RFC8588 - Personal Assertion Token (PaSSporT) Extension for signature-based handling of Asserted information using toKENs (SHAKEN) ;

Why do you need STIR/SHAKEN?

The fact of the matter is that as the VoIP world is growing and extending proposed technologies, new security mechanisms come into play, which can give you a good protection level on different layers of VoIP. STIR/SHAKEN is one of them and it works as a superstructure over the VoIP system. It can be considered as the Application layer of TCP/IP stack (in common with SIP).

This standard has been developed by IETF, in order to propose a new way to detect spoofed calls and to help ITSPs to protect their customers. Hence decrease the amount of fraud events/bot calls. The strong side of this technology is that it not only works within the scope of your VoIP system, but it is also dependent on other intermediary/termination ITSPs. Therefore it can bring a good quality improvement for a protection from undesired calls (bot-calling).

NOTE The SIP header declared for this purpose is called literally "Identity:" .

IMPORTANT The "Identity-Info:" header is deprecated by RFC 8224 and now is stored as the ";info=" parameter within the "Identity:" header.

7.38.2. The work of this mechanism

First of all, it's worth to mention that essentially, all logical parts of STIR/SHAKEN fall into these sub-categories:

- Originating service provider - a system which performs an authentication (IP based, digest over MD5 etc.) of the calling party (SIP). It has to support all the needed RFC regulations to add/manipulate the Identity header. This entity does all the work to prepare the "Identity" header before sending a

 Copyright © Sipwise GmbH - All rights reserved

call out ;

- Terminating service provider - a remote VoIP system (call termination ITSP), which verifies if the call is compliant with the STIR/SHAKEN technology, then verifies it before to accept and send to the edge subscriber. As well as with the Originating service provider, it has to support all the needed functionality to manipulate the Identity header. This entity does all the work on the Identity header of the received request, such as: decryption of the JWT token stored inside the "Identity" header and an attestation level verification ;
- Authentication service - is a superstructure/standalone authority used by the originating ITSP to obtain certificates/keys needed to encrypt the "Identity" header's value (JWT token). It is only used for work with certificates and keys ;
- Verification service - is a superstructure/standalone authority used by a terminating ITSP to obtain certificates/keys needed to decrypt the "Identity" header's value (JWT token). The same as with the Authentication service, it's only used for work with certificates and keys ;

The Sipwise C5 system role will be:

- Originating service provider - when Sipwise C5 sends a call out to PSTN networks
- Terminating service provider - when Sipwise C5 is a recipient of the call coming from PSTN networks

NOTE This technology assumes that there is some centralized authority or a list of them, which will be used both by the originating and the terminating service providers.

How does this mechanism work? First glance on that

- Firstly a SIP call is originated and obtained by certain ITSP (let's call it ITSP A) ;
- ITSP A verifies the call source and its number, in order to define how to confirm validity (full, partial or gateway attestation) ;
- ITSP A creates a SIP Identity header that contains the information on the calling number, called number, attestation level and call origination, along with the certificate (important). All information in the Identity header gets encrypted into the JWT token, with the following list of parameters ;
- The call (INVITE request) is sent out with the SIP Identity header (which has a reference to the certificate) to certain destination ITSP (let's call it ITSP B) ;
- ITSP B verifies the identity of the header and the certificate itself ;
- ITSP B makes sure the attestation level is complaint with local security standards (A, B or C) ;
- ITSP B sends a call to the edge subscriber (with possibly added parameters to the PAI header - verstat) ;

To sum up, essentially this technology gives a way for the destination user/service provider to verify that the original caller (calling side) is the one that it pretends to be.

Possible attestation levels:

- Full attestation (A) - SP authenticates the calling party and confirms it is authorized to use this particular number. An example - SIP registration (we say that originating ITSP recognizes the entire phone number as being registered with the originating subscriber) ;
- Partial attestation (B) - SP verifies the call origin, but cannot fully confirm that the source of the call

Copyright © Sipwise GmbH - All rights reserved

is authorized to use this number. An instance - a calling party (number) from behind a remote PBX (you give a block of numbers to the customer and trust this customer entirely, but you cannot verify directly individuals that use those numbers) ;

- Gateway attestation (C) - SP authenticates the call's origin, but there is no way to verify the source. An instance - a call received from an international gateway (you have a legal interconnection, but you don't know who is sitting behind and using source numbers, which you let to pass through your network) ;

NOTE In most case scenarios, the attestation level which will appear in calls involving Sipwise C5 - will be "A". Unless you don't use a subscriber preference "trusted source" or/and a domain preference "unauth_inbound_calls".

7.38.3. Identity header constitution

- a JSON Web token - JWT, that in addition consists of the following parts (divided by the dot symbol '.'):

  header - stores: an encryption algorithm, used extension (usually 'shaken'), token type (usually 'passport'), location of the cert used to sign the token

  payload - stores: an attestation level, calling/called number, a timestamp when token was created (epoch), an origination identifier (most likely UUID)

  signature - some encoded string (binary data) in Base64 URL

- three parameters (at least for now only three of them are supported):

  ';info=' - is a substitution of the deprecated Identity-Info header (will usually have a reference to a public key, which will be used to decrypt then a signature encrypted previously with a private key)

  ';alg=' - specifies the use of a cryptographic algorithm for the signature part of the JWS (for e.g. 'ES256')

  ';ppt=' - can extend the payload of PASSporT with additional JWT claims (requires that a relying party supports a particular extended claim, for e.g. 'shaken')

TIP Signature - is some encoded string (binary data) in Base64 URL, where its syntax is: *Base64URL(ES256 (Base64URL(JWT header).Base64URL(JWT payload)))* Firstly the Base64 URL is built, then also the signature with the private key is built, and a URL to the public key certificate corresponding to that key is added (as a value in the ";info=" parameter).

NOTE The Receiving party will have to download the certificate and perform a signature verification. In case the check passes, it will trust the value stored in the attestation level parameter. And of course the value of caller/callee ("orig" / "dest") has to match the actual data (in SIP headers) to be sure it's not an attack and is not a call which reuses a value from another INVITE not related to the current one.

7.38.4. Implementation of STIR/SHAKEN in Sipwise C5

Currently the Sipwise C5 is using the Kamailio project's *Secsipid.so* module to provide all, or almost all required STIR/SHAKEN related features.

 Copyright © Sipwise GmbH - All rights reserved

NOTE The secsipid module relies on libsecsipid from the Secsipidx project. Link to the SecSIPIDx project at Github: https://github.com/asipto/secsipidx

Secsipid - is a first implementation of STIR/SHAKEN IETF extensions (RFC8224, RFC8588) for asserting caller identity in Kamailio. The module is being developed further, and is reliable enough to be included in the Sipwise C5.

The configuration of the STIR/SHAKEN is done with */etc/ngcp-config/config.yml* (at least in the current implementation of Sipwise C5), with the section: *kamailio.proxy.stir* . To see how it's being configured, please take a look at kamailio (**Appendix "Configuration Overview" "config.yml Overview"**).

Domain preferences:

- 'stir_pub_url' - Public key HTTP URL
- 'stir_pub_url' - Enable STIR Identity validation

A quick start with STIR/SHAKEN:

1. Make sure which authorities you would like to get in contact with, in order to grab the needed certificates/keys ;
2. Your certificates should be prepared for work with a needed domain (your SIP domain), then you upload the given private key(s) to your Sipwise C5 system ;
3. Basically a general parameter (in the *config.yml*) you need to take care of is 'domains', and of course the given keys to it. The rest can be left untouched, unless you have some preferences for your domain(s) ;
4. Don't forget to define the related domain preferences (via the web panel) 'stir_pub_url' and 'stir_check' to enable SHAKEN for your domain(s);
5. After you're done with the configuration part, you need to apply changes with: ngcpcfg apply "added stir-shaken support" ;

IMPORTANT Applying the changes will imply a restart of the Proxy service, therefore it's strongly recommended to apply the changes outside of business hours.

IMPORTANT Remember that the kamailio user needs read permissions on the private-key files, otherwise it will not be able to consume them.

TIP The CA authority which provides you the needed certificates, should be something well known and widely used by other operators in your area.

NOTE Keep in mind that you can have multiple domains and hence keys for them stored in the Sipwise C5. This depends on the amount of SIP domains you actually have, which use STIR/SHAKEN.

NOTE The fact that you enabled support for STIR/SHAKEN on your platform, does not mean all of the present domains will have to use it. Checks/manipulations will be only enabled for those domains that are defined in *kamailio.proxy.stir.domains* of */etc/ngcp-config/config.yml*

Copyright © Sipwise GmbH - All rights reserved

Outbound calls and STIR/SHAKEN

For now the Sipwise C5 supports 'A' and 'B' attestation levels when sending outbound calls:

- 'A' - corresponds to a normally registered subscriber at the Sipwise C5 platform ;
- 'B' - will be set in case the subscriber uses the preference "trusted source" and/or is affected by the domain preference "unauth_inbound_calls" ;

Inbound calls and STIR/SHAKEN

As soon as you enable STIR/SHAKEN for a certain domain served by the Sipwise C5 system, calls coming to this domain will be challenged according to RFC standards dedicated to processing of the Identity header.

For now the Sipwise C5 supports the following list of 4XX response codes:

- 428 - indicates an absence of the Identity header or wrong extension added into the ';ppt=' parameter ;
- 436 - indicates an inability to acquire the credentials needed by the verification service for validating the signature in an Identity header field ;
- 438 - indicates that no Identity header field with a valid and supported PASSporT object has been received ;

Normally, in case of the incoming call to a SIP domain with enabled STIR/SHAKEN, the "Identity" header's content will be decrypted, verified and the call will be sent further according to the logic of the Sipwise C5 system setup.

 Copyright © Sipwise GmbH - All rights reserved

7.39. Fileshare

7.39.1. Overview

Fileshare is a REST API endpoint, /api/fileshare, that provides NGCP users with a way to share data among other NGCP users or external users.

This feature is primarily used by the NGCP SIP::App and comes as a replacement for the ComX based fileshare component which is no longer supported by NGCP.

This feature can be provided to clients directly or integrated into own applicaton services.

7.39.2. Example Scenarios



Figure 102. Fileshare File Upload

Copyright © Sipwise GmbH - All rights reserved



Figure 103. Fileshare File Download

Copyright © Sipwise GmbH - All rights reserved



Figure 104. Fileshare Uploaded Files

Copyright © Sipwise GmbH - All rights reserved

7.39.3. Configuration

Fileshare is disabled by default and requires basic configuration in the config.yml

```
config.yml

fileshare:
  enable: no
  limits:
    quota: 10737418240
    upload_size: 10485760
    user_files: 10
    user_quota: 20971520
  public_links: no
  ttl: 86400
```

- fileshare.enable: disabled by default, when enabled, rest_api.enable is required to be on as /api/fileshare uses API v2.
- fileshare.limits.quota: allowed database quota threshold in bytes, when the table size reaches the quota threshold all subsequent file uploads are denied.
- fileshare.limits.upload_size: single file upload size in bytes.
- fileshare.limits.user_files: amount of uploaded files per user (subscriber).
- fileshare.limits.user_quota: per subscriber uploaded files quota in bytes, calculated by the exact size of all uploaded files per user (subscriber)
- fileshare.public_links: defines whether uploaded files can be downloaded by anybody without authentication (the exact URL must be provided to the end user).
- fileshare.ttl: uploaded file expiration time in seconds, default is 86400 (1 day) client, e.g: /api/fileshare/58f417bb-4d28-47cf-9673-0a9453b79cc5), access to the uploaded files collection via /api/fileshare always requires authentication.

7.39.4. Approach

File Upload

To upload a file the client must use POST and the file attached into the form named "file"

An example of file upload using 'curl':

```
curl -X POST -F 'file=@/tmp/sample_image.png' https://ngcp-
api/api/fileshare
```

NOTE one of the available authorization methods is required for the file upload (Basic Authentication or JWT)

There is an optional upload body parameter 'ttl', that can be provided to override the default ttl for the uploaded file.

Copyright © Sipwise GmbH - All rights reserved

An example of file upload with ttl using 'curl':

```
curl -X POST -F 'file=@/tmp/sample_image.png' -F ttl=3600 https://ngcp-
api/api/fileshare
```

Once a file is successfully uploaded a JSON response is returned with the following data:

```
{
  "id": "b994894d-882b-4672-a152-ff1e68ab7ee0",
  "name": "sample_image.png",
  "mime_type": "image/png",
  "ttl": "3600",
  "created_at": "2021-12-14T20:58:13.483Z",
  "expires_at": "2021-12-14T21:58:13.483Z",
  "size": 198791,
  "_links": {
    "self": {
      "href": "/api/fileshare/b994894d-882b-4672-a152-ff1e68ab7ee0"
    },
    "collection": {
      "href": "/api/fileshare"
    }
  }
}
```

- The upload name is taken from the provided filename.
- The mime type is automatically determinted by the server.

File Download

To download a file a full URL must be provided, e.g.:

```
curl -X GET https://ngcp-api/api/fileshare/b994894d-882b-4672-a152-
ff1e68ab7ee0
```

If fileshare.public_links is enabled in the config.yml, then there is no authorization required for the link to work, otherwise an authorization is required for the link to be used (any authenticated NGCP subscriber (or admin) user can use the link).

The response is a data stream.

Files Collection

As a subscriber (or an admin) you can retrieve all your uploaded files list as:

```
curl -X GET https://ngcp-api/api/fileshare/
```

Copyright © Sipwise GmbH - All rights reserved

The response is a JSON HAL collection.


```
{
  "_links": {
    "self": {
      "href": "/api/fileshare?page=1&rows=10"
    },
    "ngcp:fileshare": [
      {
        "href": "/api/fileshare/1ec51a35-92e7-46be-aa98-db69f11483a7"
      },
      {
        "href": "/api/fileshare/58f417bb-4d28-47cf-9673-0a9453b79cc5"
      }
    ],
    "collection": {
      "href": "/api/fileshare"
    }
  },
  "total_count": 2,
  "_embedded": {
    "ngcp:fileshare": [
      {
        "id": "1ec51a35-92e7-46be-aa98-db69f11483a7",
        "name": "doc.pdf",
        "mime_type": "application/pdf",
        "ttl": 86400,
        "created_at": "2021-12-11T11:46:39.000Z",
        "expires_at": "2021-12-12T11:46:39.000Z",
        "size": 10
      },
      {
        "id": "58f417bb-4d28-47cf-9673-0a9453b79cc5",
        "name": "sample_text.txt",
        "mime_type": "text/plain",
        "ttl": 3600,
        "created_at": "2021-12-14T18:17:15.000Z",
        "expires_at": "2021-12-14T19:17:15.000Z",
        "size": 10
      }
    ]
  }
```


NOTE It is also possible to receive the response in the OpenAPI format by providing an additional header, **Accept: application/json**

Copyright © Sipwise GmbH - All rights reserved

```
{
  "id": "1ec51a35-92e7-46be-aa98-db69f11483a7",
  "name": "doc.pdf",
  "mime_type": "application/pdf",
  "ttl": 86400,
  "created_at": "2021-12-11T11:46:39.000Z",
  "expires_at": "2021-12-12T11:46:39.000Z",
  "size": 10
},
{
  "id": "58f417bb-4d28-47cf-9673-0a9453b79cc5",
  "name": "sample_text.txt",
  "mime_type": "text/plain",
  "ttl": 3600,
  "created_at": "2021-12-14T18:17:15.000Z",
  "expires_at": "2021-12-14T19:17:15.000Z",
  "size": 10
}
```


File Removal

Files are removed automatically by the server based on their expiration time or manually per request.

```
curl -X DELETE https://ngcp-api/api/fileshare/b994894d-882b-4672-a152-
ff1e68ab7ee0
```

Copyright © Sipwise GmbH - All rights reserved

7.40. Batch Provisioning

7.40.1. Overview

Batch provisioning of subscribers can ensure that all details of the required subscriber settings are correctly stored in the database. For the easy and convenient operation, NGCP provides the possibility to enter or upload the necessary, variable subscriber data and execute the provisioning automatically. This can be achieved through the administrative web interface, command line interface, or also via REST API.

The advantages are as follows:

(a) Standardizing complex subscriber setup procedures.

Batch provisioning allows to define provisioning templates, in which specific subscriber settings can be set. A template allows to define different types of input fields, parameters for internal calculations, and the setting of subscriber-related parameters.

(b) Scripting languages.

Provisioning templates allows the definition of code snippets for parameter calculations. Code can be based on JavaScript or Perl programming languages.

(c) Transactional processing (Rollback behavior).

Batch provisioning is based on transactional processing. If provisioning fails at a specific point, it will rollback what was created up to that point, in order to avoid residual data in the database.

(d) Processing speed.

Batch provisioning allows to process a large number of subscriber data with a considerable gain in processing speed.

The following sections will provide more details about these features.

7.40.2. Template Structure and Main Fields

Provisioning templates are commonly defined on the YAML language. The list of currently supported main keys are shown and described as follows.

Copyright © Sipwise GmbH - All rights reserved

Currently supported main template fields.

```
fields:
  #Structure that contains several input fields.
contract_contact:
  #Structure that contains several provisioning fields
  #for the contact of the customer.
contract:
  #Structure that contains several provisioning fields
  #for the customer.
contract_preferences:
  #Structure that contains several provisioning fields
  #for customer preferences.
subscriber:
  #Structure that contains several provisioning fields
  #for subscriber details.
subscriber_preferences:
  #Structure that contains several provisioning fields
  #for subscriber preferences.
registrations:
  #Structure that contains several provisioning fields
  #for permanent registrations.
trusted_sources:
  #Structure that contains several provisioning fields
  #for trusted sources.
cf_mappings:
  #A structure that contains several provisioning fields
  #for call forward mappings.
```

The 'fields' key

The fields key allows to define input hierarchical structures by providing additional key-value pair parameters. The supported attribute names are the ones defined in the Comprehensive Perl Archive Network (CPAN) `HTML::FormHandler::Field` Modules.

Parameters must contain a `name` and data `type`. For the latter, some basic type values are: *Integer*, *Float*, *Text*, *Boolean*, *Checkbox* and *Select*. Additional attributes can be appended as well.

The following example aims to show how to declare different parameters within the `fields` key. Let us assume that an administrator wants to:

- Declare a parameter named "my_field1", which will be a mandatory input text labeled as "*This is field 1 (Text)*".
- Declare a parameter named "my_field2", which will be an optional integer input labeled as "*This is field 2 (Integer)*".
- Declare a parameter named "my_field3", which will be optional Boolean input labeled as "*This is field 3 (Boolean)*".
- Declare a parameter named "my_field4", which will be a mandatory dropdown list labeled as "*This is field 4 (Select)*" with two sub-options labeled "*optA-label*", "*optB-label*" with the static values "*optA-value*", "*optB-value*", respectively.

Copyright © Sipwise GmbH - All rights reserved

The resulting YAML declaration will be:

Example Form of Four Different Field Types.

```
fields:
  - name: my_field1
    label: "This is Field 1 (Text)"
    type: Text
    required: 1
  - name: my_field2
    label: "This is Field 2 (Integer)"
    type: Integer
  - name: my_field3
    label: "This is Field 3 (Boolean)"
    type: Boolean
  - name: my_field4
    label: "This is Field 4 (Select)"
    type: Select
    options:
      - label: optA-label
        value: optA-value
      - label: optB-label
        value: optB-value
    required: 1
```

The resulting web form will be shown as the picture below.

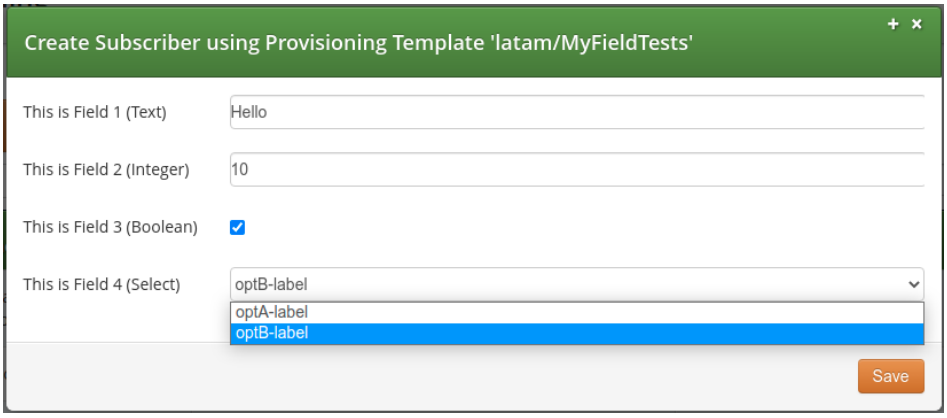


Figure 105. Resulting Form.

The 'purge' attribute name

The `purge` attribute name is a reserved word that can be declared within the `fields` tag, as a *Boolean* type. When set to `1`, and if there is an existing subscriber with the same data to be provisioned (e.g. same username, number or alias), then the existing subscriber will be terminated before the new one is created. On the opposite, if set to `0`, that row will be skipped.

 Copyright © Sipwise GmbH - All rights reserved

NOTE The engine will not terminate anything other than a subscriber (Including its child items such as preferences, trusted sources, call forwards, etc.). Thus, the contract or contact is not terminated/deleted.

The 'calculated' reserved parameter type

There is a special attribute type named calculated, which is not visible on form elements. It is declared for internal use to: (a) Store static values on attributes; (b) Store the output of computed operations coming from code snippets. As defined in `HTML::FormHandler`, the former requires declaring the `value` attribute name to store the data. The latter requires declaring the `value_code` attribute in order to store the data in the `value` attribute. Next sections will describe further details about this.

Rest of Template Keys

The rest of the template keys can contain any of the NGCP settings, as detailed in the table below.

Table 27. Rest of Template Keys.

Field Name	Sub-fields
`contract_contact`	Available field names can be taken from the `/api/customercontacts/` API properties (Please, check API documentation), or from the fields belonging to the `billing.contacts` SQL table.
`contract`	Available field names can be taken from the `/api/customers/` API properties (Please, check API documentation), or from the fields belonging to the `billing.contracts` SQL table.
`contract_preferences`	Available field names can be taken from the `/api/customerpreferences/` API properties (Please, check API documentation).
`subscriber`	Available field names can be taken from the `/api/subscribers/` API properties (Please, check API documentation).
`subscriber_preferences`	Available field names can be taken from the `/api/subscriberpreferences/` API properties (Please, check API documentation).
`registrations`	Available field names can be taken from the `/api/subscriberregistrations/` API properties (Please, check API documentation).
`trusted_sources`	Available field names can be taken from the `/api/trustedsources/` API properties (Please, check API documentation).
`cf_mappings`	Available field names can be taken from the `/api/cfmappings/` API properties (Please, check API documentation).

The 'identifier' sub-field and data lookup

The `identifier` field is a comma-separated list composed by one or more JSON/SQL properties. It can be specified within any of the template keys of the table above. It is a string list that the batch processing engine uses for data lookup in the database.

Copyright © Sipwise GmbH - All rights reserved

For instance, let us assume that the following declaration has been defined:

Reference Example For The `identifier` Attribute.

```
contract_contact:
  identifier: "firstname, lastname, status"
  reseller: default
  firstname_code: "function() { return row.first_name; }"
  lastname_code: "function() { return row.last_name; }"
  status: "active"
contract:
  identifier: contact_id
  contact_id_code: "function() { return contract_contact.id; }"
  product: "Basic SIP Account"
  billing_profile: "Default Billing Profile"
```

Firstly, a *Contract* refers to a customer (i.e. a PBX). When the engine processes a CSV file (or a Web form), it will search for a contract contact with the names specified in the `firstname` and `lastname` fields (e.g. "John, Doe"), and with the status field equals to "*active*".

If the contact entry does not yet exist, it will be created. At this stage, there are two situations:

(a) Contact does not yet exist: A new one is created, with some new `contact_id` value.

(b) Contact already exists: It is looked up and returns some existing `contact_id` value.

Regarding the `contract` tag, its identifier is the `contact_id` field. Again, the engine will try to look up for a contract with the `contact_id` previously obtained:

- When the contract contact was just created (Case (a) above), then there will neither be an existing contract. Hence, the contract is also created.
- In case (b), there is already a contact (e.g. "John Doe", which is active), then possibly also already a contract linked to it. Hence, the engine will not create another contract, but it looks up the existing one.
- Finally, the engine proceeds with creating subscribers and their corresponding settings.

Further sections will describe extra details on the template declaration.

7.40.3. Storing Data

When using internal attributes (i.e. defined by the calculated type), it is possible to store static data or code-calculated data (i.e. obtained from code snippets). Static data can be stored in any suitable attribute name according the template section in which will be declared. Notwithstanding, in order to store the output from a code snippet, that attribute name MUST be declared along with the `_code` suffix concatenated to it. Some examples:

- Within the `fields` key, the `value` attribute name needs to be used to store static data. Hence, the `value_code` attribute name must be declared in the template if a code snippet is put in place to calculate such data. When the data is computed, it will be stored in the `value` attribute.
- Within the `contract` key, `contact_id` is one of the several available attributes for a customer that

 Copyright © Sipwise GmbH - All rights reserved

can be declared in that section. If the value of this attribute needs to be computed, then the `contact_id_code` attribute name needs to be declared, along with the corresponding code snippet. Then, when the data is computed, it will be stored in the `contact_id` attribute.

7.40.4. Code Snippets

Code snippets allows to use programming logic to calculate attribute values. The supported programming languages are JavaScript and Perl.

The following directives must be declared to use code snippets in templates, for both JavaScript and Perl, respectively:

Code Snippet Declaration: JavaScript and Perl.

```
"function() {
  //JavaScript code logic
  ...
  return ... ; //Output to be stored in a *_code attribute name.
}"
```

```
"sub {
  #Perl code logic
  ...
  return ... ; #Output to be stored in a *_code attribute name.
}"
```

Native JavaScript/Perl directives can be used as part of the code logic, such as string concatenations, number stripping, etc. Furthermore, there are reserved variable names and special functions that can be used in a code. These are detailed in the following sections.

Reserved Variable Names

Reserved variable names can be used in the definition of code snippets, summarized as follows.

Table 28. Reserved Variable Names.

Variable Name	Type	Description
`row`	*Class*	It represents the CSV file row (or the input form when filling out via Admin Panel) according the `fields` parameters defined in the template.
`contract_contact`	*Class*	Settings for the contact of the customer. It contains information such as the name, the postal and email addresses, among others.
`contract`	*Class*	Settings for customer. It refers to the `billing.contract` SQL table.
`contract_preferences`	*Class*	Settings for customer preferences.

Copyright © Sipwise GmbH - All rights reserved

`subscriber`	*Class*	Settings for subscriber details.
`subscriber_preferences`	*Class*	Settings for subscriber preferences.
`registrations`	*Class*	Settings for permanent registrations.
`trusted_sources`	*Class*	Settings for trusted sources.
`cf_mappings`	*Class*	Settings for call forward mappings.
`reseller`	*String*	Reseller name. The entity name that represents a collection of settings to provide telecommunication services to subscribers on the NGCP.
`billing_profile`	*String*	Billing profile name.
`profile_package`	*String*	Billing profile package name.
`domain`	*String*	SIP domain name. Particularly, it refers to the record for the `billing.domain` SQL table.
`provisioning_domain`	*String*	SIP domain name. Particularly, it refers to the `provisioning.voip_domain` SQL table.
`product`	*String*	The product type for the PBX customer. Either "*Basic SIP Account*" or "*Cloud PBX Account*".
`now`	*String*	It represents the transaction timestamp.

Special Functions

Apart from using native JavaScript/Perl functions, there are special utility functions available, summarized as follows.

Table 29. Special Functions.

Function Name	Arguments	Description

Copyright © Sipwise GmbH - All rights reserved

`split_number()`	*String*	It can split a number (entered as a text) into country code (`cc`), area code (`ac`) and subscriber number (`sn`). It MUST be used along with the definition of the `cc_ac_map` attribute name within the `fields` tag, as follows:
		
		`fields:`
		`- name: cc_ac_map`
		`  type: calculated`
		`  value:`
		`  - CC1:`
		`  - AC1:`
		`  - AC2:`
		`  ...`
		`  - CC2:`
		`  - AC1:`
		`  - AC2:`
		`  ...`
		
		Where:
		
		• `CC`: *Country Code number (e.g. 43, 56, etc.)*
		• `AC`: *Area Code number (e.g. 1, 2142, etc.)*

`debug()`	*String*	It prints 'debug' log information in the /var/log/ngcp/panel.log and /var/log/ngcp/panel-debug.log files when used via Admin Panel, or in the output of the `ngcp-provisioning-templates` script (depending on the `--log-level` option specified).
`info()`	*String*	It prints 'info' log information in the /var/log/ngcp/panel.log and /var/log/ngcp/panel-debug.log files when used via Admin Panel, or in the output of the `ngcp-provisioning-templates` script (depending on the `--log-level` option specified).
`warn()`	*String*	It prints 'warning' log information in the /var/log/ngcp/panel.log and /var/log/ngcp/panel-debug.log files when used via Admin Panel, or in the output of the `ngcp-provisioning-templates` script (depending on the `--log-level` option specified).

Copyright © Sipwise GmbH - All rights reserved

`error()`	*String*	It prints 'error' log information in the /var/log/ngcp/panel.log and /var/log/ngcp/panel-debug.log files when used via Admin Panel, or in the output of the `ngcp-provisioning-templates` script (depending on the `--log-level` option specified).

IMPORTANT

For Perl, the following core functions are disabled to prevent harmful code injection: *binmode(), close(), closedir(), dbmclose(), dbmopen(), eof(), fileno(), flock(), format(), getc(), read(), readdir(), rewinddir(), say(), seek(), seekdir(), select(), syscall(), sysread(), sysseek(), syswrite(), tell(), telldir(), truncate(), write(), print(), printf(), chdir(), chmod(), chown(), chroot(), fcntl(), glob(), ioctl(), link(), lstat(), mkdir(), open(), opendir(), readlink(), rename(), rmdir(), stat(), symlink(), sysopen(), umask(), unlink(), utime(), alarm(), exec(), fork(), getpgrp(), getppid(), getpriority(), kill(), pipe(), setpgrp(), setpriority(), sleep(), system(), times(), wait(), waitpid(), accept(), bind(), connect(), getpeername(), getsockname(), getsockopt(), listen(), recv(), send(), setsockopt(), shutdown(), socket(), socketpair(), msgctl(), msgget(), msgrcv(), msgsnd(), semctl(), semget(), semop(), shmctl(), shmget(), shmread(), shmwrite(), endgrent(), endhostent(), endnetent(), endpwent(), getgrent(), getgrgid(), getgrnam(), getlogin(), getpwent(), getpwnam(), getpwuid(), setgrent(), setpwent(), endprotoent(), endservent(), gethostbyaddr(), gethostbyname(), gethostent(), getnetbyaddr(), getnetbyname(), getnetent(), getprotobyname(), getprotobynumber(), getprotoent(), getservbyname(), getservbyport(), getservent(), sethostent(), setnetent(), setprotoent(), setservent(), exit(), goto().*

Finally, let us consider the following template section example:

Code-Snippet Example.

```
fields:
  - name: cc
    label: "Country Code:"
    type: Text
    required: 1
  - name: ac
    label: "Area Code:"
    type: Text
    required: 1
  - name: sn
    label: "Subscriber Number:"
    type: Text
    required: 1
  - name: sip_username
    type: calculated
    value_code: "function() {
                    return row.cc.concat(row.ac).concat(row.sn);
                }"
```

Here, the text input fields `cc`, `ac` and `sn` along with the internal field `sip_username` are declared. It is

Copyright © Sipwise GmbH - All rights reserved

required for this case that `sip_username` must be a string concatenation of the `cc`, `ac` and `sn` fields, respectively. Therefore, `sip_username` type is declared as `calculated` and its value must be declared through the `value_code` attribute name. This attribute shows a code snippet which contains a JavaScript function that performs the string concatenation. The `row` variable refers to the CSV file row, or the form input when filling out the data through Admin Panel. Finally, when the data is computed, it will be stored in the `value` attribute.

7.40.5. Batch Provisioning Via Admin Panel

Batch provisioning can be enabled as a global, system-wide configuration in the main configuration file (config.yml) through the `www_admin.batch_provisioning_features` property set to `1`. The feature is available for all administrative users of the platform, that covers users with generic administrator role and users with "customer care" role. These users will be referred as "admin users" later in this document.

When set, the Batch Provisioning page can be accessed through the *Tools Batch Provisioning* menu entry. The interface will offer the option to create a new provisioning template, or edit an existing one.

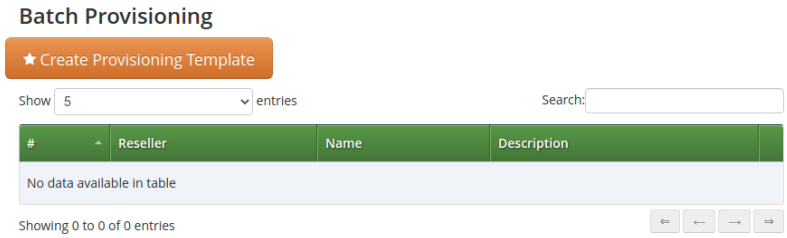


Figure 106. Main Batch Provisioning Page.

For creation, press on the '*Create Provisioning Template*' button. This will pop-up a new window, as shown below.

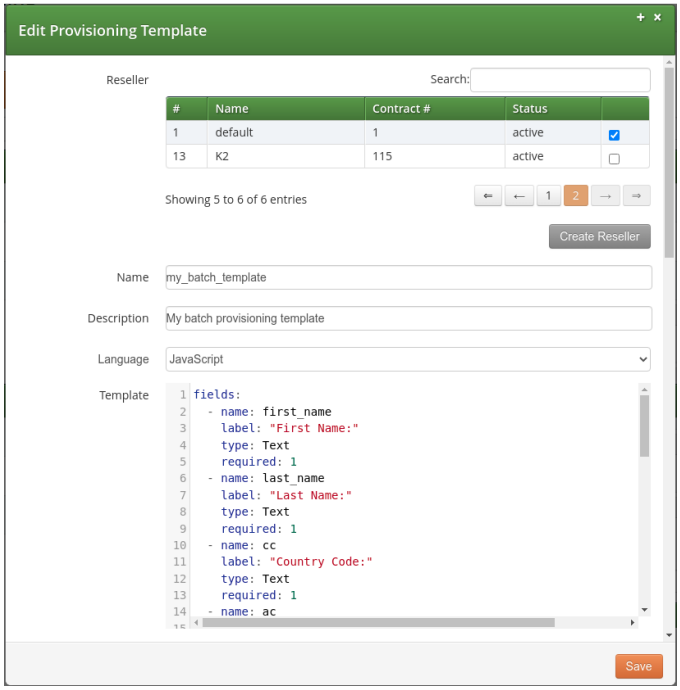


Figure 107. Create Provisioning Template.

Copyright © Sipwise GmbH - All rights reserved

The parameters are as follows:

- **Reseller**: The reseller this template belongs to.
- **Name**: A free form string used to identify the provisioning template in the Admin Panel. This may be edited at any time.
- **Description**: Information about the provisioning template.
- **Language**: Scripting language used in the provisioning template for the 'calculated' fields. Either *JavaScript* (default) or *Perl*.
- **Template**: The provisioning template to be used. By default, there is a built-in template that has been created to set basic options. Please, check appendix for further details.

Once a template has been defined, there are two options available for data input:

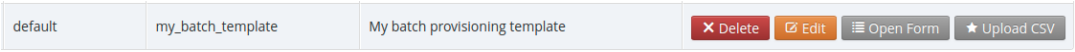


Figure 108. Provisioning Options.

Manual Entry For A Single Subscriber

This option is available when clicking on the '*Open Form*' button. A new window will be displayed to enter the variable data of the subscriber in several form fields:

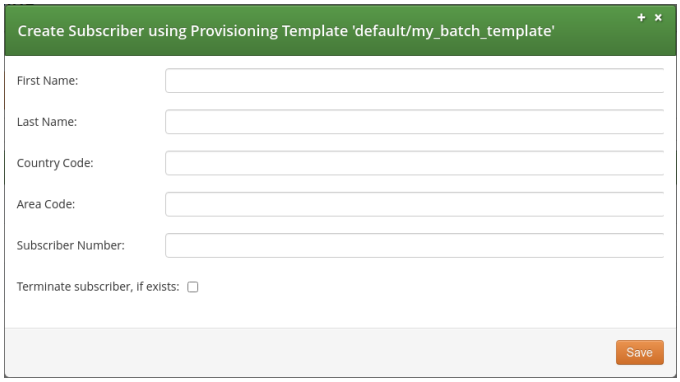


Figure 109. Single Subscriber Provisioning.

The fields displayed will depend of what is defined on the provisioning template. If the default built-in template is in use, then parameters are as follows:

- **First Name**: The given name of the customer in which the subscriber will belong to.
- **Last Name**: The surname of the customer in which the subscriber will belong to.
- **Country Code (CC)**: Country code of the subscriber telephone number.
- **Area Code (AC)**: Area code of the subscriber telephone number.
- **Subscriber Number (SN)**: Telephone number of the subscriber.
- **Terminate subscriber, if exists**: If ticked, it will terminate any existing subscriber with that name.

Bulk Entry For More Subscribers

This option is available when clicking on the '*Upload CSV*' button, which allows to upload a CSV file that

 Copyright © Sipwise GmbH - All rights reserved

contains a line of data for each subscriber. A new window will be displayed to enter the data:

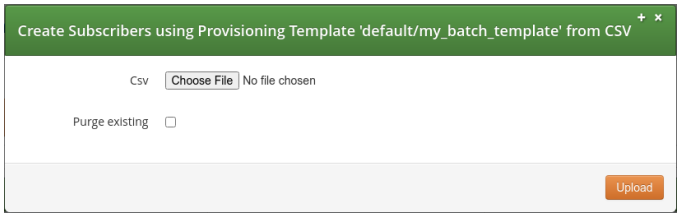


Figure 110. Bulk Subscriber Provisioning.

The parameters are as follows:

- **CSV**: The CSV file to upload. The file has to follow the format defined in the template.
- **Purge Existing**: If ticked, it will terminate any existing subscribers with that name. This checkbox is equivalent to specify the value 1 in each row of the CSV file.

7.40.6. Batch Provisioning via CLI

NGCP offers the `ngcp-provisioning-template` command line tool for batch provisioning, which allows to run a 'provisioning template' from database or from config.yml file. This will produce a subscriber setup including required billing contact, contract, preferences, etc. from an input form defined by that template.

Usage:

For templates defined in *config.yml* file:

```
ngcp-provisioning-template "provisioning-template-name" [options]
```

For templates defined in database:

```
ngcp-provisioning-template "reseller-name/provisioning-template-name"
[options]
```

NOTE For templates defined in database, different resellers could each have a provisioning template with the same name.

The form fields can be passed as command line options, as described below.

Table 30. Usage of The ngcp-provisioning-template Command.

Option	Description
`--help`	Prints a brief help message and exits.
`--db-host=db-host-IP-address`	The host of the ngcp database to connect to. If omitted, the database connection settings of ngcp-panel will be used.

Copyright © Sipwise GmbH - All rights reserved

Option	Description
`--db-port=db-host-port`	The port of the ngcp database to connect to. Only relevant if `--db-host` is specified.
`--db-user=db-username`	The database user for the ngcp database to connect to. Only relevant if `--db-host` is specified.
`--db-password=db-password`	The the database user password (if any) for the ngcp database to connect to. Only relevant if `--db-host` is specified.
`--file=csv-filename`	Specify a CSV filename to process. Each row represents form values for one subscriber to create.
`--[input-attribute -name]=[attribute-value]`	Provide an input attribute name and its value, as defined within the `fields` tag in the template (Attribute must not be internal, but form visible). Only relevant if no `--file` is specified.
`--purge`	Terminate an existing subscriber with duplicate number/aliases first.
`--log-level`	Verbosity level of printed messages while processing (`error`, `warn`, `info`, `debug`).

By default, database connection parameters are read from the `/etc/ngcp-panel/provisioning.conf` configuration file. For developing and/or testing reasons, it is possible though to use different database connection parameters, which can be specified with the `--db-*` arguments.

If no CSV file is specified with the `--file` option, each of the non-internal input parameters defined in the template (within the fields tag) can be provided as part of the command line options. For instance, if `first_name` input variable is declared in the template, then such option can be entered as `--first_name` (e.g. `--first_name=John`).

Batch provisioning is based on transactional processing. If provisioning fails at a specific point, it will rollback what was created up to that point, in order to avoid residual data in the database.

TIP When developing a template, it is best to use the `ngcp-provisioning-template` script with debug enabled, since it is possible to check log messages directly on the command output.

Some examples for single subscriber use, and CSV-file use.

Single subscriber (Debug level on):

Copyright © Sipwise GmbH - All rights reserved

```
ngcp-provisioning-template "default/My Template" \
  --log-level=debug \
  --first_name=Peter \
  --last_name=White \
  --cc=43 \
  --ac=1 \
  --sn=2521523 \
  --purge
```

CSV File (Debug level on):

```
ngcp-provisioning-template "default/My Template" \
  --log-level=debug \
  --file=bulk.csv \
  --purge
```

7.40.7. Batch Provisioning Via API

Batch provisioning is also available in REST API through the `/api/provisioningtemplates/` path. Please, check API documentation for details about available HTTP methods, properties and query parameters.

The following sub-sections aim to highlight common use cases. Let us assume the following template base data will be used.

Table 31. Example Of API Provisioning Data.

Field	Value
Reseller	default
Reseller ID	1
SIP Domain	mydomain.com
Product	Basic SIP Account
Template Name	*My API Provisioning Template*
Template Language	js (JavaScript)
Template Content	Same as Built-in Template
API IP <Address:Port>	127.0.0.1:1443
API Key File	./apiclient.pem

Template Creation

For template creation, the HTTP `POST` method is available. Template metadata can be specified in JSON notation using the `name`, `description`, `reseller_id`, and `lang` attributes. Template definition can be provided in JSON notation within the `template` attribute, as shown below.

Copyright © Sipwise GmbH - All rights reserved

Example of Template Creation via API.

```
curl -ki --cert ./apiclient.pem -X POST \
  -H 'Connection: close' \
  -H 'Content-Type: application/json' \
  "https://127.0.0.1:1443/api/provisioningtemplates/" \
  --data-binary '{
    "description" : "My API-Created Template",
    "name" : "My API Provisioning Template",
    "reseller_id" : 1,
    "lang" : "js",
    "template" : {
      "fields" : [
        {
           "name" : "first_name",
           "label" : "First Name:",
           "type" : "Text",
           "required" : "1"
        },
        {
          "name" : "last_name",
          "label" : "Last Name:",
          "type" : "Text",
          "required" : "1"
        },
        {
          "label" : "Country Code:",
          "name" : "cc",
          "type" : "Text",
          "required" : "1"
        },
        {
          "name" : "ac",
          "label" : "Area Code:",
          "type" : "Text",
          "required" : "1"
        },
        {
          "name" : "sn",
          "label" : "Subscriber Number:",
          "type" : "Text",
          "required" : "1"
        },
        {
          "name" : "sip_username",
          "type" : "calculated",
          "value_code" : "function() { return
row.cc.concat(row.ac).concat(row.sn); }"
        },
        {
          "name" : "purge",
```

 Copyright © Sipwise GmbH - All rights reserved

```
          "label" : "Terminate subscriber, if exists:",
          "type" : "Boolean"
        }
      ],
      "contract_contact" : {
        "identifier" : "firstname, lastname, status",
        "reseller" : "default",
        "firstname_code" : "function() { return row.first_name; }",
        "lastname_code" : "function() { return row.last_name; }",
        "status" : "active"
      },
      "contract" : {
        "product" : "Basic SIP Account",
        "billing_profile" : "Default Billing Profile",
        "identifier" : "contact_id",
        "contact_id_code" : "function() { return contract_contact.id; }"
      },
      "subscriber" : {
        "domain" : "mydomain.com",
        "primary_number" : {
          "cc_code" : "function() { return row.cc; }",
          "ac_code" : "function() { return row.ac; }",
          "sn_code" : "function() { return row.sn; }"
        },
        "username_code" : "function() { return row.sip_username; }",
        "password_code" : "function() { return row.sip_password; }"
      },
      "subscriber_preferences" : {
        "gpp0" : "test"
      }
    }
  }'
```

Template Request

For template request, the HTTP `GET` method is available. A specific template can be fetched according the id value of the template, which corresponds to the concatenation of reseller name, the `/` symbol and the template name. Based on the previous example creation, the corresponding value is "*default/My API Provisioning Template*".

Example of Template Request via API.

```
curl -ki --cert ./apiclient.pem -X GET \
  -H 'Connection: close' \
  "https://127.0.0.1:1443/api/provisioningtemplates/default/My API
Provisioning Template/"
```

NOTE On batch provisioning templates, the `id` parameter is a *String* type (Rather than an *Integer* type, as commonly used on other API commands).

Copyright © Sipwise GmbH - All rights reserved

`GET` method will output the template definition within the template attribute, which output format can be controlled by using the `?format=[value]` directive. The available values are: `yml`, `yaml` or `json`. Example:

```
curl -ki --cert ./apiclient.pem -X GET \
  -H 'Connection: close' \
  "https://127.0.0.1:1443/api/provisioningtemplates/default/My API
Provisioning Template/?format=json"
```

Additionally, as templates can be stored on database or in config.yml file, it is possible to use the Boolean editable parameter to list: (a) Database-stored templates only (`true`); or (b) Templates defined on config.yml file only (`false`). Example:

```
curl -ki --cert ./apiclient.pem -X GET \
  -H 'Connection: close' \
  "https://127.0.0.1:1443/api/provisioningtemplates/?editable=false"
```

NOTE Templates defined on config.yml will be displayed on Admin Panel, but they cannot be edited.

Template Update

For template update, the HTTP `PUT` and `PATCH` methods are available.

For `PUT`, it is possible to set the whole template at once only. Let us assume that the `gpp0` attribute (located within the `subscriber_preferences` tag) will be modified to other value, meanwhile the rest of template properties will remain the same.

Example of Template Update Via API (PUT).

```
curl -ki --cert ./apiclient.pem -X PUT \
  -H 'Connection: close' \
  -H 'Content-Type: application/json' \
  "https://127.0.0.1:1443/api/provisioningtemplates/default/My API
Provisioning Template/" \
  --data-binary '{
    "description" : "My API-Created Template",
    "name" : "My API Provisioning Template",
    "reseller_id" : 1,
    "lang" : "js",
    "template" : {
      "fields" : [
        {
          "name" : "first_name",
          "label" : "First Name:",
          "type" : "Text",
          "required" : "1"
        },
```


Copyright © Sipwise GmbH - All rights reserved

```
        {
          "name" : "last_name",
          "label" : "Last Name:",
          "type" : "Text",
          "required" : "1"
        },
        {
          "label" : "Country Code:",
          "name" : "cc",
          "type" : "Text",
          "required" : "1"
        },
        {
          "name" : "ac",
          "label" : "Area Code:",
          "type" : "Text",
          "required" : "1"
        },
        {
          "name" : "sn",
          "label" : "Subscriber Number:",
          "type" : "Text",
          "required" : "1"
        },
        {
          "name" : "sip_username",
          "type" : "calculated",
          "value_code" : "function() { return
row.cc.concat(row.ac).concat(row.sn); }"
        },
        {
          "name" : "purge",
          "label" : "Terminate subscriber, if exists:",
          "type" : "Boolean"
        }
      ],
      "contract_contact" : {
        "identifier" : "firstname, lastname, active",
        "reseller" : "default",
        "firstname_code" : "function() { return row.first_name; }",
        "lastname_code" : "function() { return row.last_name; }",
        "status" : "active"
      },
      "contract" : {
        "product" : "Basic SIP Account",
        "billing_profile" : "Default Billing Profile",
        "identifier" : "contact_id",
        "contact_id_code" : "function() { return contract_contact.id; }"
      },
      "subscriber" : {
        "domain" : "mydomain.com",
        "primary_number" : {
```

Copyright © Sipwise GmbH - All rights reserved

```
          "cc_code" : "function() { return row.cc; }",
          "ac_code" : "function() { return row.ac; }",
          "sn_code" : "function() { return row.sn; }"
        },
        "username_code" : "function() { return row.sip_username; }",
        "password_code" : "function() { return row.sip_password; }"
      },
      "subscriber_preferences" : {
        "gpp0" : "My new gpp0 value"
      }
    }
  }'
```

For `PATCH`, it is possible to modify main template attributes at once only. Sub-attributes belonging to a section will need to be replaced as a whole. Let us assume that the `description` attribute and the `gpp0` attribute (located within the `template.subscriber_preferences` section) will be updated to different values, respectively.

Example of Template Update Via API (PATCH).

```
curl -ki --cert ./apiclient.pem -X PATCH \
  -H 'Connection: close' \
  -H 'Content-Type: application/json-patch+json' \
  "https://127.0.0.1:1443/api/provisioningtemplates/default/My API
Provisioning Template/" \
  --data-binary '[
    { "op" : "replace", "path" : "/description", "value" : "My new
description for API-Created Template" },
    { "op" : "replace", "path" : "/template", "value" : {
      "fields" : [
        {
          "name" : "first_name",
          "label" : "First Name:",
          "type" : "Text",
          "required" : "1"
        },
        {
          "name" : "last_name",
          "label" : "Last Name:",
          "type" : "Text",
          "required" : "1"
        },
        {
          "label" : "Country Code:",
          "name" : "cc",
          "type" : "Text",
          "required" : "1"
        },
        {
          "name" : "ac",
          "label" : "Area Code:",
```


Copyright © Sipwise GmbH - All rights reserved

```
          "type" : "Text",
          "required" : "1"
        },
        {
          "name" : "sn",
          "label" : "Subscriber Number:",
          "type" : "Text",
          "required" : "1"
        },
        {
          "name" : "sip_username",
          "type" : "calculated",
          "value_code" : "function() { return
row.cc.concat(row.ac).concat(row.sn); }"
        },
        {
          "name" : "purge",
          "label" : "Terminate subscriber, if exists:",
          "type" : "Boolean"
        }
      ],
      "contract_contact" : {
        "identifier" : "firstname, lastname, status",
        "reseller" : "default",
        "firstname_code" : "function() { return row.first_name; }",
        "lastname_code" : "function() { return row.last_name; }",
        "status" : "active"
      },
      "contract" : {
        "product" : "Basic SIP Account",
        "billing_profile" : "Default Billing Profile",
        "identifier" : "contact_id",
        "contact_id_code" : "function() { return contract_contact.id; }"
      },
      "subscriber" : {
        "domain" : "mydomain.com",
        "primary_number" : {
          "cc_code" : "function() { return row.cc; }",
          "ac_code" : "function() { return row.ac; }",
          "sn_code" : "function() { return row.sn; }"
        },
        "username_code" : "function() { return row.sip_username; }",
        "password_code" : "function() { return row.sip_password; }"
      },
      "subscriber_preferences" : {
        "gpp0" : "My new gpp0 value"
      }
     }
    }
  ]'
```

Copyright © Sipwise GmbH - All rights reserved

Template Deletion

For template deletion, the HTTP `DELETE` method is available. It is required to provide the `id` value (*String*) of the template. Based on the examples, the value is "*default/My API Provisioning Template*".

Example of Template Deletion Via API.

```
curl -ki --cert ./apiclient.pem -X DELETE \
  -H 'Connection: close' \
  "https://127.0.0.1:1443/api/provisioningtemplates/default/My API
Provisioning Template/"
```

API Data Provisioning: Single Subscriber

A `POST` can be requested to provision a single subscriber by using JSON. Let us assume that the following single subscriber data will be provisioned:

Table 32. Example Of Single Subscriber Provisioning Via API.

Customer Contact Info		Subscriber			Purge?
First Name	Surname	CC	AC	SN	
Mark	Brown	56	2	33445511	yes

Then, the corresponding API command will be:

Example of Single Subscriber Provisioning Via API.

```
curl -ki --cert ./apiclient.pem -X POST \
  -H 'Connection: close' \
  -H 'Content-Type: application/json' \
  "https://127.0.0.1:1443/api/provisioningtemplates/default/My API
Provisioning Template/" \
  --data-binary '{
    "first_name" : "Mark" ,
    "last_name" : "Brown" ,
    "cc" : "56" ,
    "ac" : "2",
    "sn" : "33445511" ,
    "purge" : true
  }'
```

API Data Provisioning: Batch of Subscribers

To bulk-upload subscribers, it is required to specify the Content-Type as `text/csv` and POST the CSV file in the request body. Let us assume that the following subscriber data will be provisioned in the bulk.csv file:

Table 33. Example Of Bulk Subscriber Provisioning Via API.

Copyright © Sipwise GmbH - All rights reserved

Customer Contact Info		Subscriber(s)			Purge?
First Name	Surname	CC	AC	SN	
James	Smith	56	2	33445511	yes
Richard	Stone	56	2	33445522	yes
John	Doe	43	123	1001 1002 1003 1004 1005	yes

The CSV file will contain:

```
James, Smith, 56, 2, 33445511, 1
Richard, Stone, 56, 2, 33445522, 1
John, Doe, 43, 123, 1001, 1
John, Doe, 43, 123, 1002, 1
John, Doe, 43, 123, 1003, 1
John, Doe, 43, 123, 1004, 1
John, Doe, 43, 123, 1005, 1
```

Then, the corresponding API command will be:

Example of Bulk Subscriber Provisioning Via API.

```
curl -ki --cert ./apiclient.pem -X POST \
  -H 'Connection: close' \
  -H 'Content-Type: text/csv' \
  "https://127.0.0.1:1443/api/provisioningtemplates/default/My API
Provisioning Template/" \
  --data-binary '@./bulk.csv'
```

Copyright © Sipwise GmbH - All rights reserved

7.41. Call List Suppressions

7.41.1. Overview

Call List Suppressions allows to filter specific numbers from Call History and Invoices. The feature aims to support country regulations such as gene violation or abuse of women, in which calls made to specific service numbers should not be made visible in any way to the aggressor.

The feature can be set via Admin Panel.

Its main advantages are as follows:

(a) Filter Calls.

Matched call numbers (against a regular expression) will not appear at all.

(b) Obfuscate Calls.

Matched call numbers (against a regular expression) will be replaced by a given label.

The following sections will provide more details about the feature.

7.41.2. Provisioning via Admin Panel

Call List Suppressions can be accessed at *Settings Call List Suppressions* section. The interface will offer the option to create a new entry, edit any of the existing ones, or upload/download via CSV.

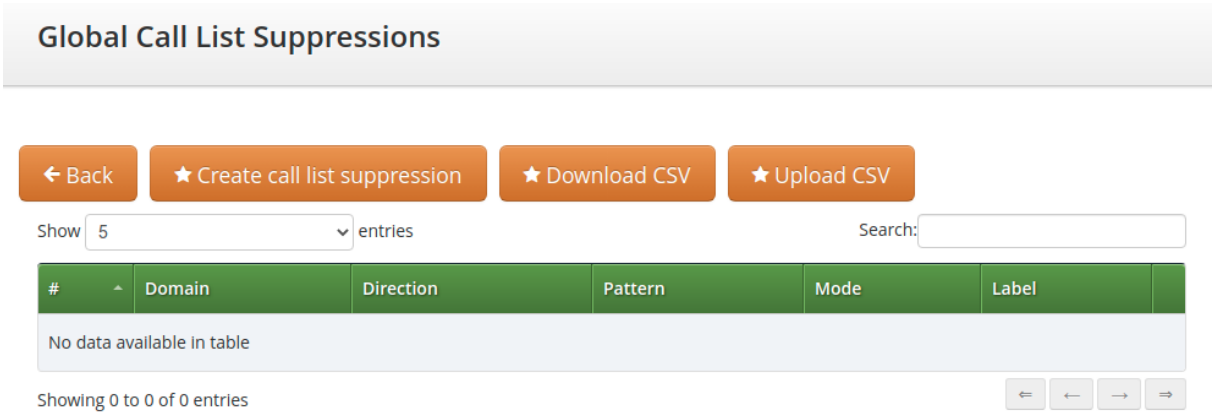


Figure 111. Call List Supressions.

7.41.3. Single Entry

For creation, press on the "*Create call list supression*" button, which will pop-up a new window, as shown below.

Copyright © Sipwise GmbH - All rights reserved

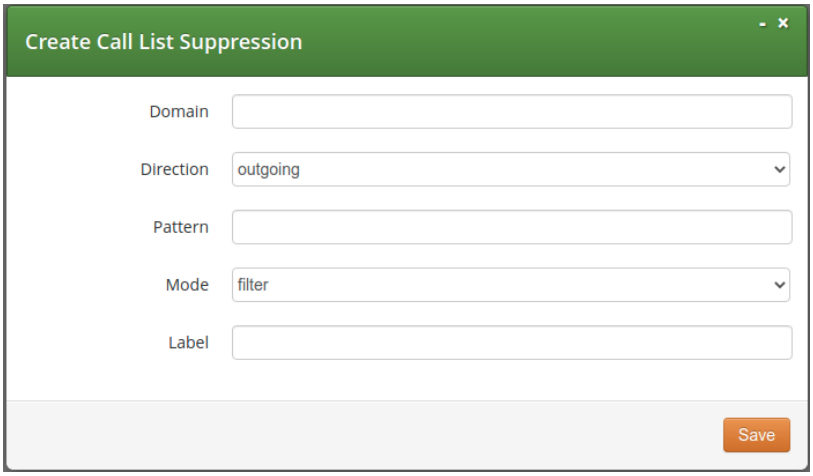


Figure 112. Subscriber Location Mappings: Create Entry.

Parameters are as follows:

- **Domain**: The domain of subscribers in which this call list suppression applies to. An empty domain means to apply it to subscribers of any domain.
- **Direction**: The direction (`outgoing` or `incoming`) of calls this call list suppression applies to.
- **Pattern**: A regular expression the dialed number (CDR `destination_user_in` field) has to match in case of `outgoing` direction, or the inbound number (CDR `source_cli` field) in case of incoming direction.
- **Mode**: It defines the suppression mode. For subscriber and subscriber admins, The `filter` option means that matching calls will not appear at all, while the `obfuscate` option means that the number is replaced by the given label. Also, the mode can be disabled by selecting the `disabled` option.
- **Label**: A string name to be defined for this list. Admin and reseller users see it for `filter` mode suppressions. For the `obfuscate mode`, this label will be used as a replacement string.

7.41.4. Bulk Entry

This option is available when clicking on the "*Upload CSV*" button, which allows to upload a CSV file that contains a line of data for each list. A new window will be displayed to enter the data:

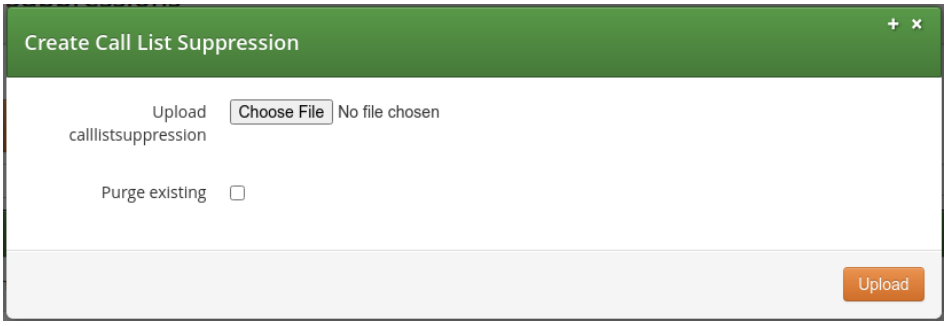


Figure 113. Bulk Provisioning.

The parameters are as follows:

- **CSV**: The CSV file to upload. The file has to follow the same order as defined for single entries.

Copyright © Sipwise GmbH - All rights reserved

- **Purge Existing**: If ticked, it will delete any existing list before provisioning the new one.

7.41.5. Practical Example

Let us assume that an administrator will use Admin Panel to provision two call list supression on a selected number. The following details are considered:

Table 34. Practical Example - Input Data.

Domain	Direction	Pattern	Mode	Label
voipdomain.com	`outgoing`	^571016$	`filter`	*cls-a*
voipdomain.com	`outgoing`	^571017$	`obfuscate`	*cls-b*

In the provided example, the logic is as following:

Table 35. Practical Example - Logic & Result.

Logic	Result
• An outgoing call to the destination number 571016 is made.	• The call will not be shown in the call history (*Conversations* section) of the Customer Self-Care interface. • An NGCP administrator will be able to see the call in the call history via Admin Panel, with the label "*filtered: cls-a*". • The call will not be shown on NGCP-generated invoices.
• An outgoing call to the destination number 571017 is made.	• The call will be shown in the call history (*Conversations* section) of the Customer Self-Care interface as "*Call to cls-b*". • An NGCP administrator will be able to see the call in the call history, with the label "*obfuscated: cls-b*". • The call will be shown as "*cls-b*" in NGCP-generated invoices.

Copyright © Sipwise GmbH - All rights reserved

7.42. Message Body Filtering

In some cases it is required to filter out from message bodies some specific multipart/mixed body because not supported by the endpoints. This can be simply achieved using the config.yml option *kamailio.lb.filter_content_type*. The filter will be executed on:

- all the INVITE requests that contains body
- all the replies with return code 200 and 183 that contains body

To configure it, proceed following the next steps:

1. set the *kamailio.lb.filter_content_type.enable* option to 'yes'.
2. set the *kamailio.lb.filter_content_type.action* to 'filter' or 'drop'. If one of the listed content-type is found, the first option will strip out all the bodies expect for application/sdp from the message, the second one will simply drop the message.
3. list in *kamailio.lb.filter_content_type.content_type_list* the content_types and the direction (request, reply or all) where the action has to be applied.
4. apply the changes executing *ngcpcfg apply 'enabled filter_content_type'*.

NOTE The action 'drop' works only for 'requests' and not for 'replies'.

For example, to filter out from all the reply messages the 'application/isup' content-type and from all the messages the 'application/vnd.etsi.cug+xml', you have to configure the following:

```
kamailio:
  lb:
    filter_content_type:
      enable: yes
      action: filter (or drop)
      content_type_list:
      - content_type: application/isup
        direction: reply
      - content_type: application/vnd.etsi.cug+xml
        direction: all
```

NOTE *kamailio.lb.filter_content_type* definition substitutes the already existing *kamailio.lb.remove_isup_body_from_replies*, extending its functionality to other content types. The migration from the previous setting to the new one is automatically done during the upgrade.

Copyright © Sipwise GmbH - All rights reserved

Chapter 8. Extra Modules

8.1. Cloud PBX

The Sipwise C5 comes with a commercial Cloud PBX module to provide B2B features for small and medium sized enterprises. The following chapters describe the configuration of the PBX features.

8.1.1. PBX Device Provisioning

How it works

A device gets provisioned with the following steps:

- Your customer creates a PBX device for a supported model and inputs a device's MAC address.
- Sipwise C5 sends the provided MAC address to the device vendor (e.g. api.eds.al-enterprise.com, rps.yealink.com or sraps.snom.com).
- When the corresponding device is connected to the network, the device fetches the provisioning URL from the vendor site.
- The device downloads its specific configuration and the firmware from Sipwise C5.
- The phone updates the firmware and automatically sets the SIP proxy server, username and password and other SIP parameters received from Sipwise C5.

PBX device provisioning requires appropriate device models, firmwares, configurations and profiles to be added to the system.

A *device model* defines a specific hardware device, like the vendor, the model name, the number of keys and their capabilities. For example, a Cisco SPA504G has 4 keys, which can be used for private lines, shared lines (SLA) and busy lamp field (BLF). If you have an additional attendant console, you get 32 more buttons, which can only do BLF. The list of supported devices can be found in List of available pre-configured devices.

A *device firmware* is used to update a potentially outdated factory firmware on a device. The default firmwares included in Sipwise C5 were tested with the provided device configurations and hence guarantee that all the supported features work as expected. That is why we recommend using the default firmwares and device configurations provided by Sipwise.

To make device provisioning easy-to-use for end-users, they do not have to care about firmwares or configurations mentioned above. Instead, you provide a *device profile* for every supported device model and associate such a device profile with a specific device configuration and firmware. When a customer employee with administrative rights provisions PBX devices for the company, they select the corresponding device profiles and specifies MAC addresses if necessary. Sipwise C5 will take care of the rest.

Sipwise C5 is supplied with a set of supported device models, their firmwares, configurations and profile. You can enable them and your customers will be able to use PBX device provisioning immediately.

To perform basic configuration and upload the set for a specific vendor, device model(s) or for all supported devices, execute the steps described in the following section.

 Copyright © Sipwise GmbH - All rights reserved

Initial device provisioning configuration

Execute the following initial steps before your customers can easily and securely provision their PBX devices:

1. Set the certificates and the keys for your HTTPs FQDN
2. Upload the required device models/firmwares/configurations/profiles

Set the certificates and the key for your web domain

You can create new ones or use the existing certificate and the key for your web FQDN.

- Put the required files into the */etc/ngcp-config/shared-files/ssl* folder.
- Specify the paths to the files and the FQDN in the following config.yml parameters:

  server_certfile

  server_keyfile

  Specify the FQDN in **autoprov.server.host**

  Optionally, enable **autoprov.server.nginx_debug**

The final configuration should look similar to this one:

```
autoprov:
  hardphone:
    skip_vendor_redirect: no
    skip_vendor_ssl_verify: []
    skip_vendor_ssl_verify_hostname: []
  server:
    bootstrap_port: '1445'
    cisco_port: '1447'
    client_ca_certfile: /etc/ngcp-config/shared-files/ssl/client-auth-
ca.crt
    host: portal.yourdomain.com
    nginx_debug: yes
    port: '1444'
    server_ca_certfile: /etc/ngcp-config/shared-files/ssl/my_server.crt
    server_certfile: /etc/ngcp-config/shared-files/ssl/certificate.pem
    server_certfile_cisco: /etc/ngcp-config/shared-
files/ssl/certificate_cisco.pem
    server_keyfile: /etc/ngcp-config/shared-files/ssl/private_key.pem
    server_keyfile_cisco: /etc/ngcp-config/shared-
files/ssl/private_key_cisco.pem
    ssl_enabled: yes
    ssl_mac_check: no
  softphone:
    config_lockdown: '0'
    webauth: '0'
```

- Apply and push the changes

Copyright © Sipwise GmbH - All rights reserved

```
ngcpcfg apply 'PBX device provisioning configuration'
ngcpcfg push all
```

Upload the required device items

To upload device models/firmwares/configurations/profiles for devices with ZTP support, you need to obtain credentials from the corresponding vendor or its local distributor in advance. These credentials are required to send information about your devices and their provisioning URLs to the corresponding ZTP/RPS systems.

The script *ngcp-insert-pbx-devices* will insert the specified items into the database. For example, to upload items for all supported Yealink devices for the default reseller, execute the script with the following parameters on your management node (standby on PRO; web01a/db01a on CARRIER):

```
ngcp-insert-pbx-devices --api-user youruser --api-pass yourpassword
--yealink-user user --yealink-password password
```

TIP Execute *ngcp-insert-pbx-devices --help* to find other useful parameters, e.g. *--device -models*, *--resellers* and others.

8.1.2. Preparing PBX Rewrite Rules

In a PBX environment, the dial-plans usually looks different than for normal SIP subscribers. PBX subscribers should be able to directly dial internal extensions (e.g. 100) instead of the full number to reach another PBX subscriber in the same PBX segment. Therefore, we need to define specific *Rewrite Rules* to make this work.

The PBX dial plans are different from country to country. In the Central European area, you can directly dial an extension (e.g. 100), and if you want to dial an international number like 0049 1 23456, you have to dial a break-out digit first (e.g. 0), so the number to be dialed is 0 0049 1 23456. Other countries are used to other break-out codes (e.g. 9), which then results in 9 0049 1 23456. If you dial a national number like 01 23456, then the number to actually be dialled is 9 01 23456.

Since all numbers must be normalized to E.164 format via inbound rewrite rules, the rules need to be set up accordingly.

As with normal SIP accounts, you can use variables like ${caller_cc} and ${caller_ac}. For PBX subscribers, there is an additional variable ${caller_cloud_pbx_base_cli}, which is the primary number of the administrative subscriber for a PBX customer. When you create this *pilot subscriber*, you can specify the primary number and a list of alias numbers. If the customer creates his own extensions by himself, he only has to provide the extension number, which will be appended to the primary number of the pilot subscriber, and which in turn forms the primary number of the extension subscriber. So if the pilot subscriber's primary number is 49 1 23456, and he creates a new extension subscriber with extension 100, the primary number of the extension will be 49 1 23456100. If another extension of this customer dials 100, you can prefix that with ${caller_cloud_pbx_base_cli}, so the 100 gets normalized to 49 1 23456100 and the correct extension subscriber can be matched. You will learn more about number assignment for PBX subscribers in the following sections, when the creation of them is shown.

Let's assume that the break-out code for the example customers created below is 0, so we have to

 Copyright © Sipwise GmbH - All rights reserved

create a *Rewrite Rule Set* with the following rules.

Inbound Rewrite Rules for Caller

PBX devices usually send their SIP URI (e.g. myuser@mydomain.org) as initiating CLI, so nothing special is to be done here. If you have PBX devices which only send their extension number (e.g. 100) as CLI, you need to prefix that with the ${caller_cloud_pbx_base_cli} variable, like this:

- **Match Pattern**: `^([1-9][0-9]+)$`
- **Replacement Pattern**: `${caller_cloud_pbx_base_cli}\1`
- **Description**: `extension to e164`
- **Direction**: `Inbound`
- **Field**: `Caller`

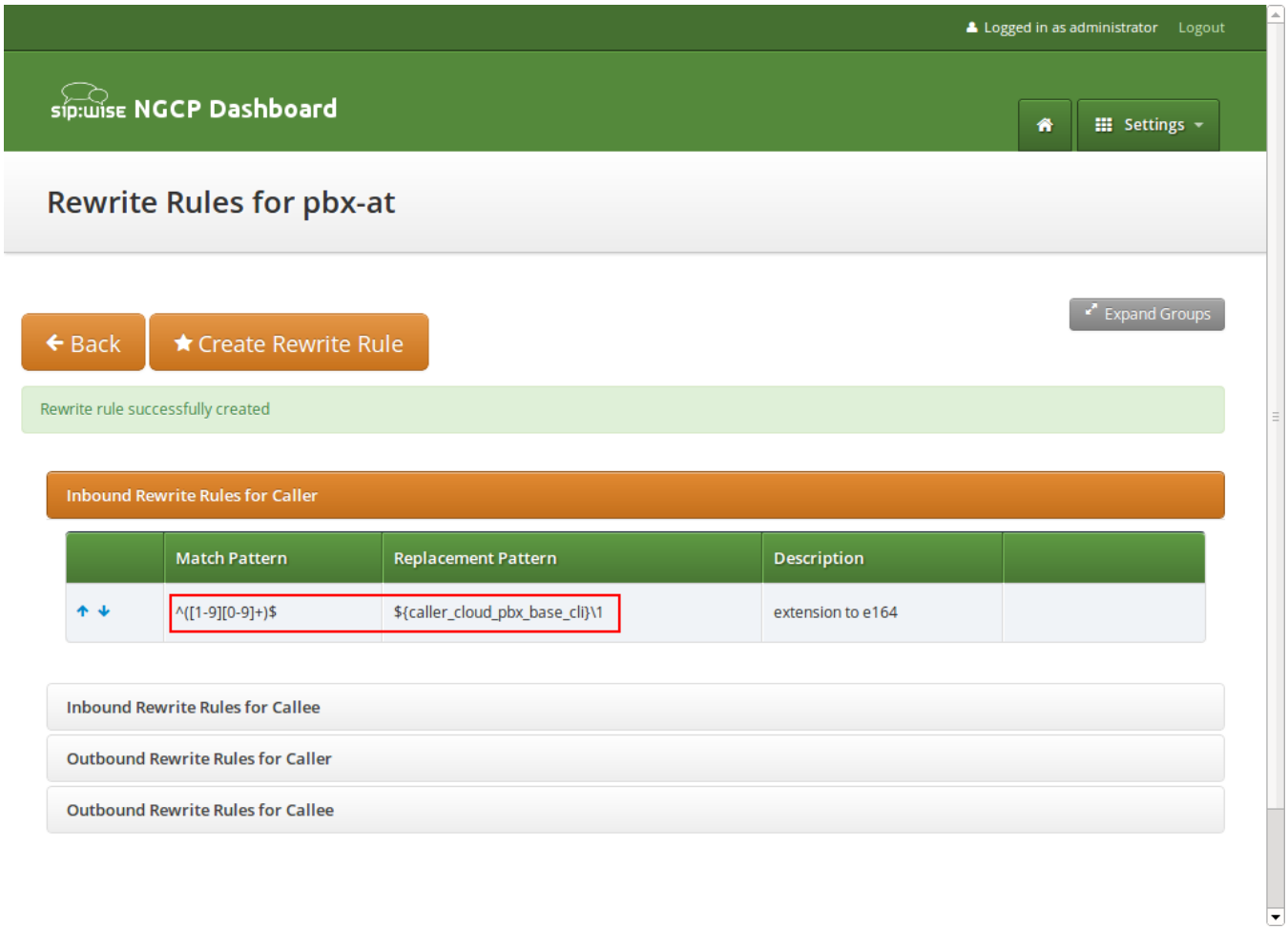


Figure 114. Inbound Rewrite Rule for Caller

Inbound Rewrite Rules for Callee

These rules are the most important ones, as they define which number formats the PBX subscribers can dial. For the break-out code of 0, the following rules are necessary e.g. for German dialplans to allow pbx internal extension dialing, local area calls without area codes, national calls with area code, and international calls with country codes.

Copyright © Sipwise GmbH - All rights reserved

PBX internal extension dialin

- **Match Pattern**: `^([1-9][0-9]+)$`
- **Replacement Pattern**: `${caller_cloud_pbx_base_cli}\1`
- **Description**: `extension to e164`
- **Direction**: `Inbound`
- **Field**: `Callee`

Local dialing without area code (use break-out code 0)

- **Match Pattern**: `^0([1-9][0-9]+)$`
- **Replacement Pattern**: `${caller_cc}${caller_ac}\1`
- **Description**: `local to e164`
- **Direction**: `Inbound`
- **Field**: `Callee`

National dialing (use break-out code 0 and prefix area code by 0)

- **Match Pattern**: `^00([1-9][0-9]+)$`
- **Replacement Pattern**: `${caller_cc}\1`
- **Description**: `national to e164`
- **Direction**: `Inbound`
- **Field**: `Callee`

International dialing (use break-out code 0 and prefix country code by 00)

- **Match Pattern**: `^000([1-9][0-9]+)$`
- **Replacement Pattern**: `\1`
- **Description**: `international to e164`
- **Direction**: `Inbound`
- **Field**: `Callee`

Copyright © Sipwise GmbH - All rights reserved

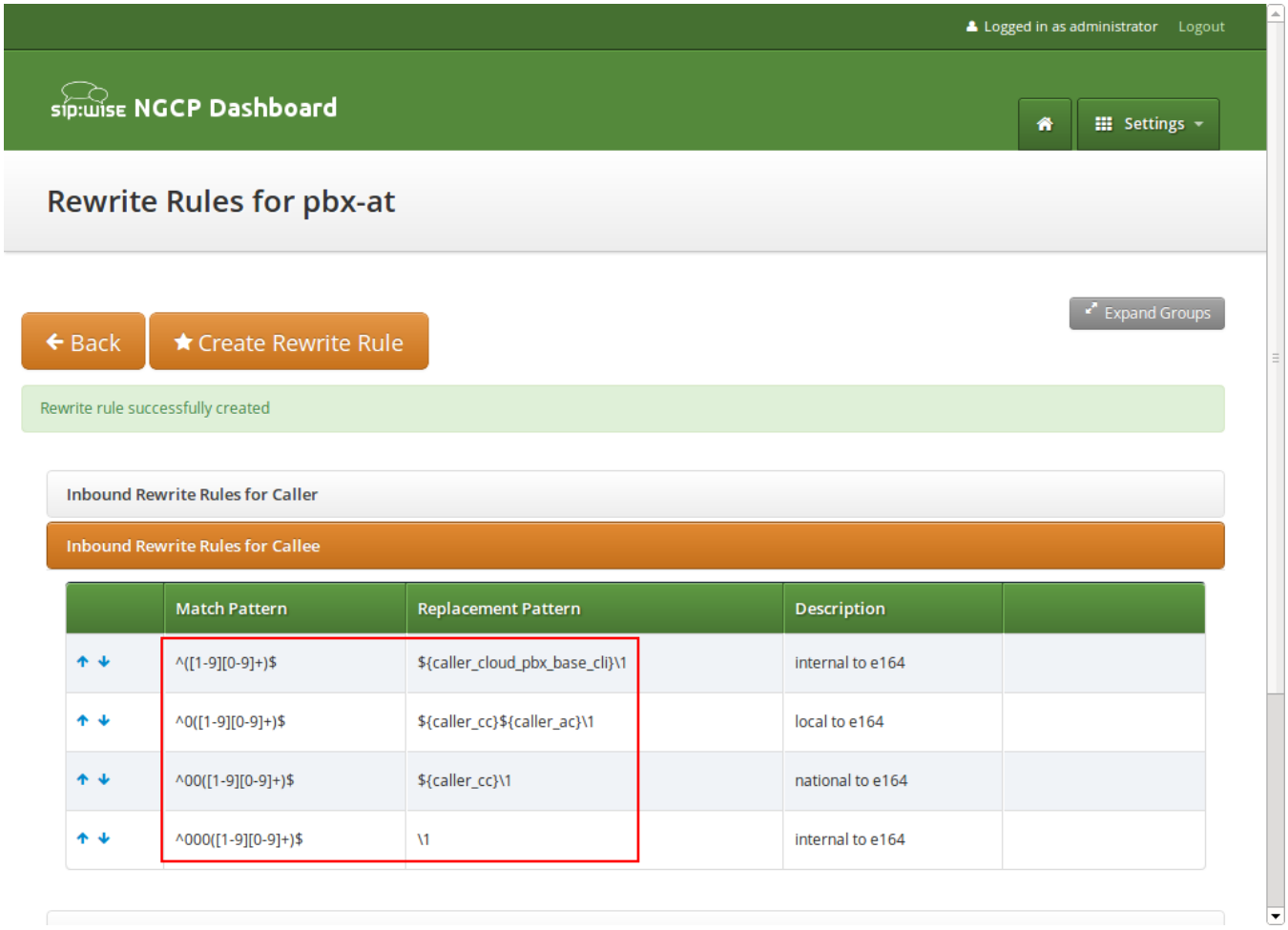


Figure 115. Inbound Rewrite Rule for Callee

Outbound Rewrite Rules for Caller

When a call goes to a PBX subscriber, it needs to be normalized in a way that it's call-back-able, which means that it needs to have the break-out code prefixed. We create a rule to show the calling number in international format including the break-out code. For PBX-internal calls, the most common setting is to show the short extension of the caller and caller name that is provisioned to a PBX subscriber (in order to do that set domain preferences *outbound_from_display* and *outbound_from_user* accordingly, so you don't have to worry about that in rewrite rules).

Adding a break-out code (use break-out code 0 and prefix country code by 00)

- **Match Pattern**: `^([1-9][0-9]+)$`
- **Replacement Pattern**: `000\1`
- **Description**: `e164 to full international`
- **Direction**: `Outbound`
- **Field**: `Caller`

Copyright © Sipwise GmbH - All rights reserved

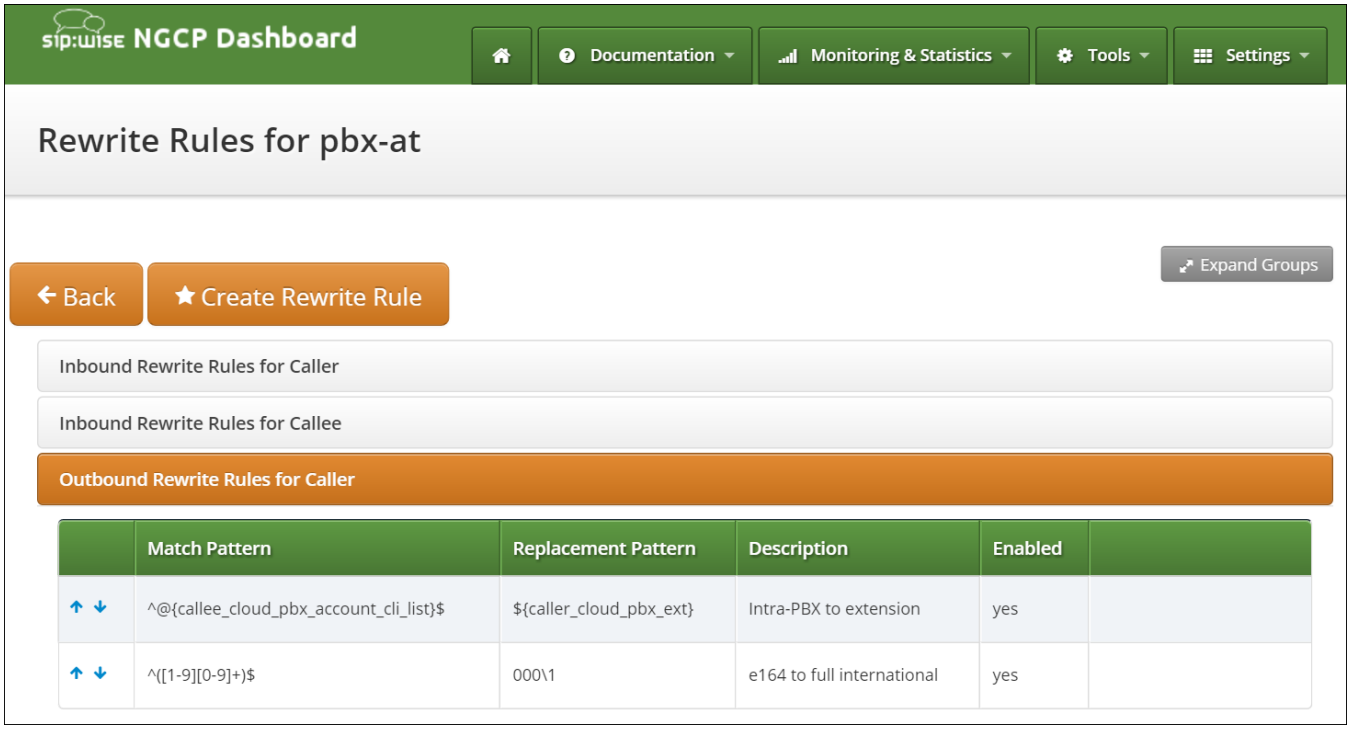


Figure 116. Outbound Rewrite Rule for Caller

Create a new *Rewrite Rule Set* for each dial plan you'd like to support. You can later assign it to customer domains and even to subscribers, if a specific subscriber of a PBX customer would like to have his own dial plan.

8.1.3. Creating Customers and Pilot Subscribers

As with a normal SIP Account, you have to create a *Customer* contract per customer, and one *Subscriber*, which the customer can use to log into the web interface and manage his PBX environment.

Creating a PBX Customer

Go to *SettingsCustomers* and click *Create Customer*. We need a *Contact* for the customer, so press *Create Contact*.

Copyright © Sipwise GmbH - All rights reserved

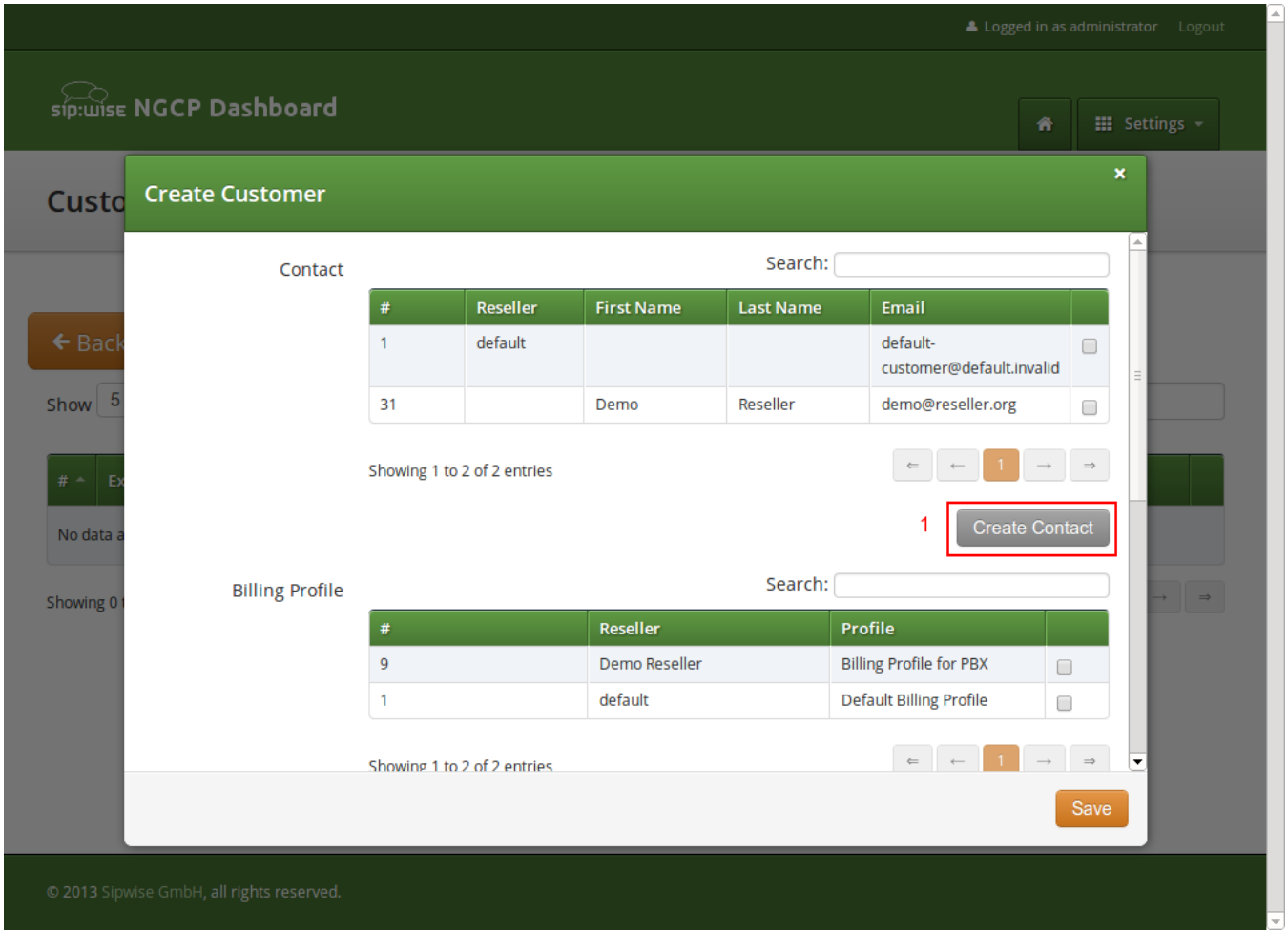


Figure 117. Create PBX Customer Part 1

Fill in the desired fields (you need to provide at least the *Email Address*) and press *Save*.

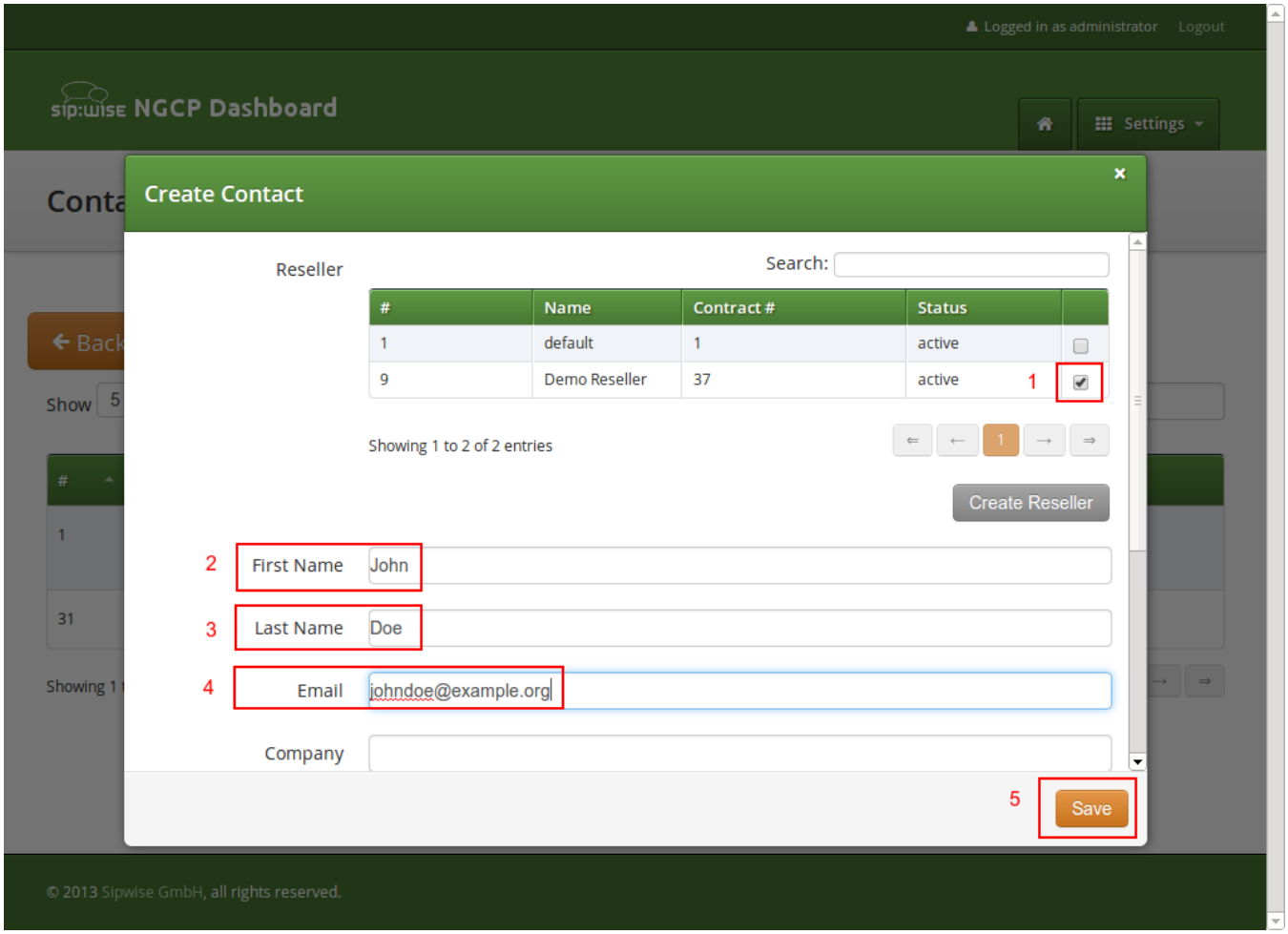


Figure 118. Create PBX Customer Contact

The new *Contact* will be automatically selected now. Also select a *Billing Profile* you want to use for this customer. If you don't have one defined yet, press *Create Billing Profile*, otherwise select the one you want to use.

Copyright © Sipwise GmbH - All rights reserved

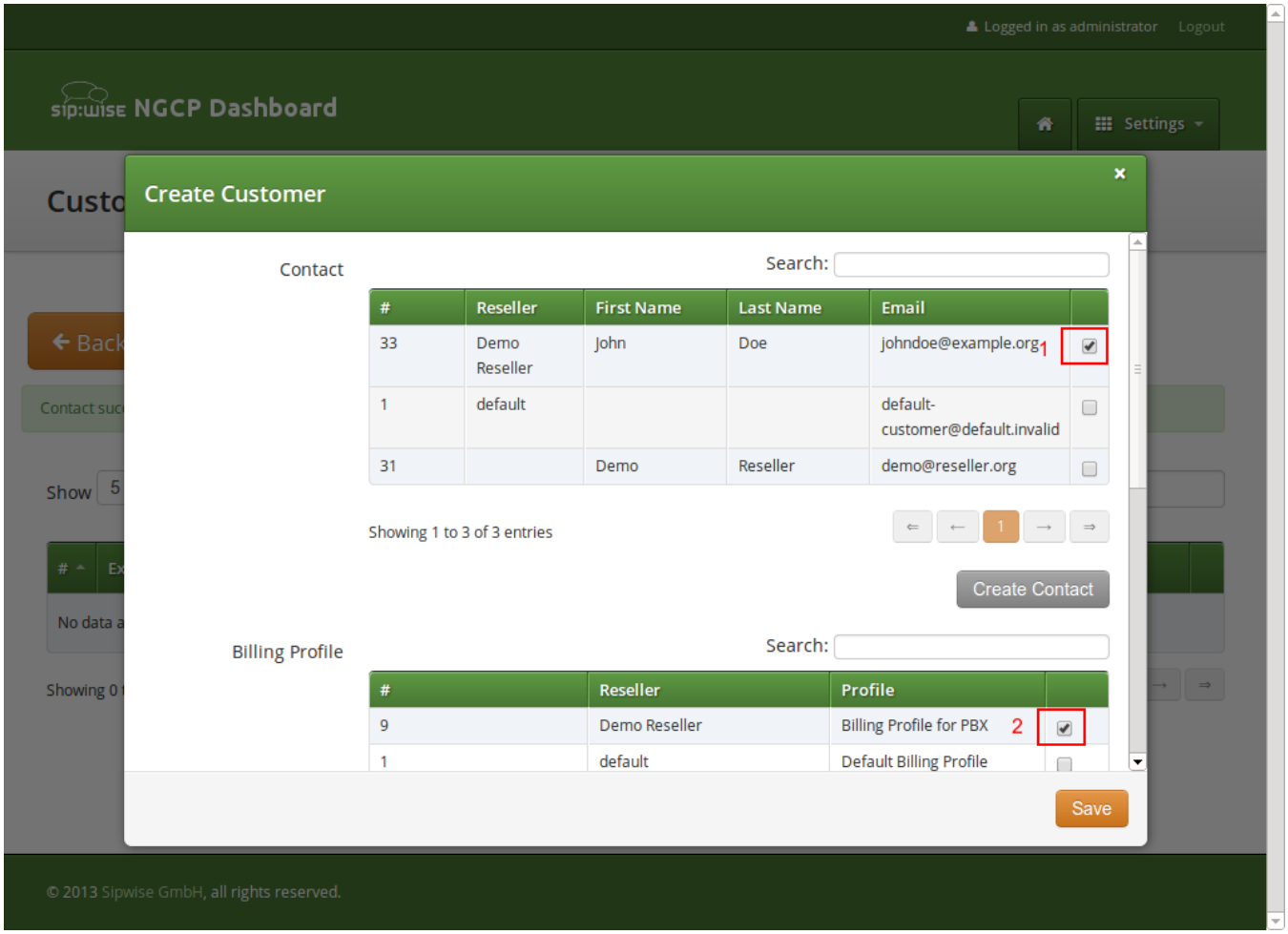


Figure 119. Create PBX Customer Part 2

Next, you need to select the *Product* for the PBX customer. Since it's going to be a PBX customer, select the product *Cloud PBX Account*.

Since PBX customers are supposed to manage their subscribers by themselves, they are able to create them via the web interface. To set an upper limit of subscribers a customer can create, define the value in the *Max Subscribers* field.

IMPORTANT

As you will see later, both PBX subscribers and PBX groups are normal subscribers, so the value defined here limits the overall amount of subscribers **and** groups. A customer can create an unlimited amount of subscribers if you leave this field empty.

Press *Save* to create the customer.

Copyright © Sipwise GmbH - All rights reserved

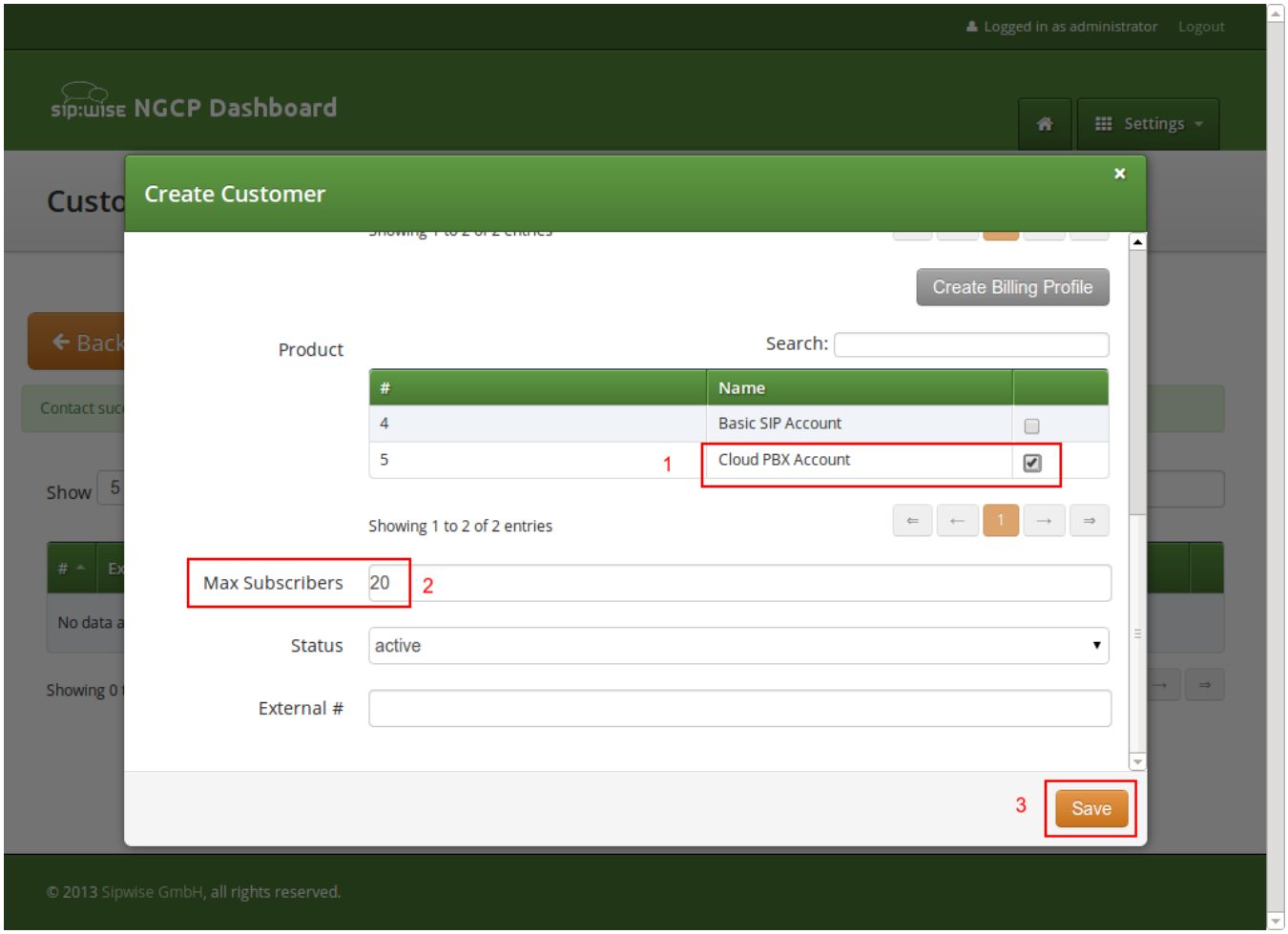


Figure 120. Create PBX Customer Part 3

Creating a PBX Pilot Subscriber

Once the customer is created, you need to create at least one *Subscriber* for the customer, so he can log into the web interface and manage the rest by himself.

Click the *Details* button on the newly created customer to enter the detailed view.

Copyright © Sipwise GmbH - All rights reserved

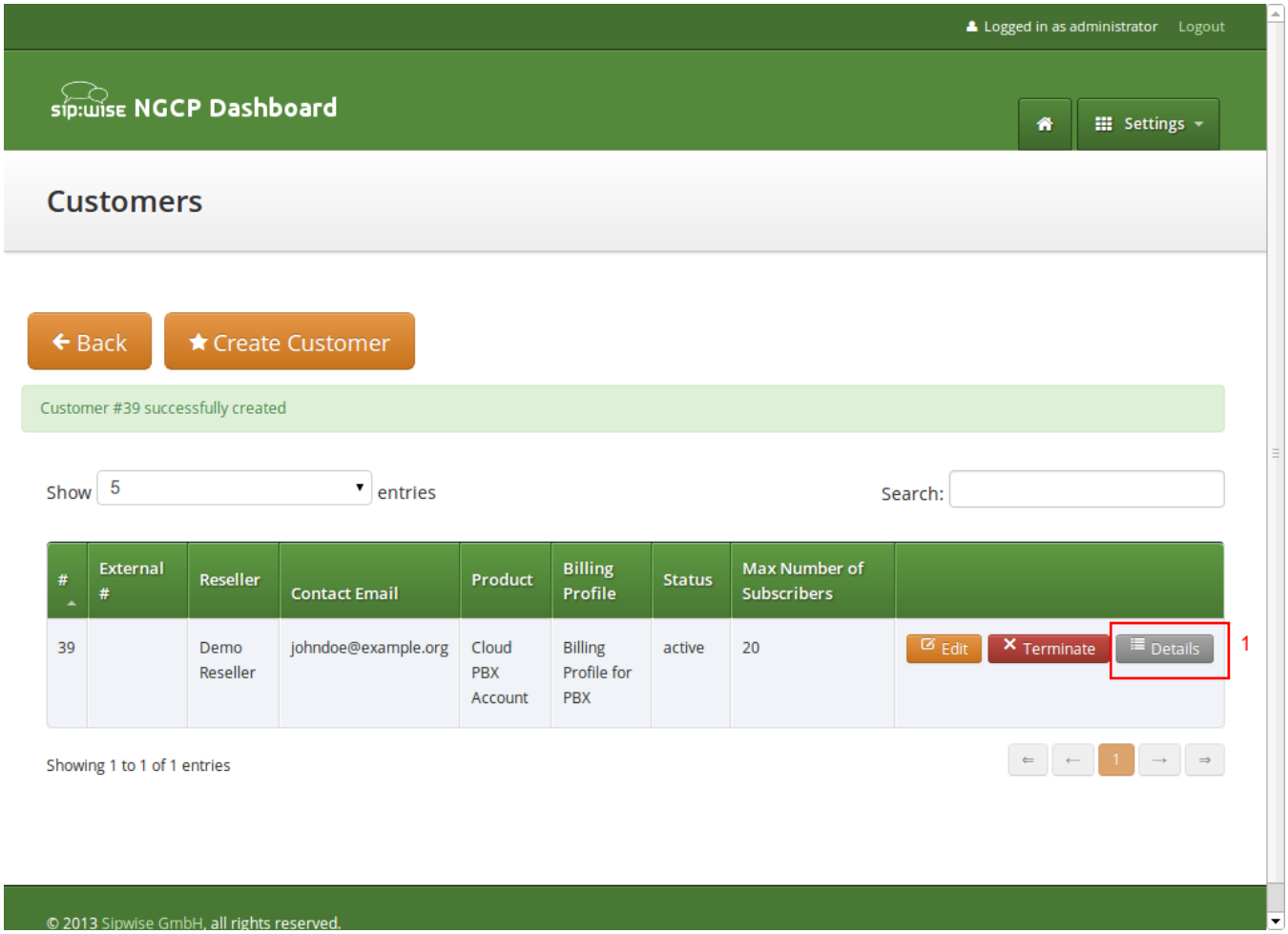


Figure 121. Go to Customer Details

To create the subscriber, open the *Subscribers* row and click *Create Subscriber*.

Copyright © Sipwise GmbH - All rights reserved

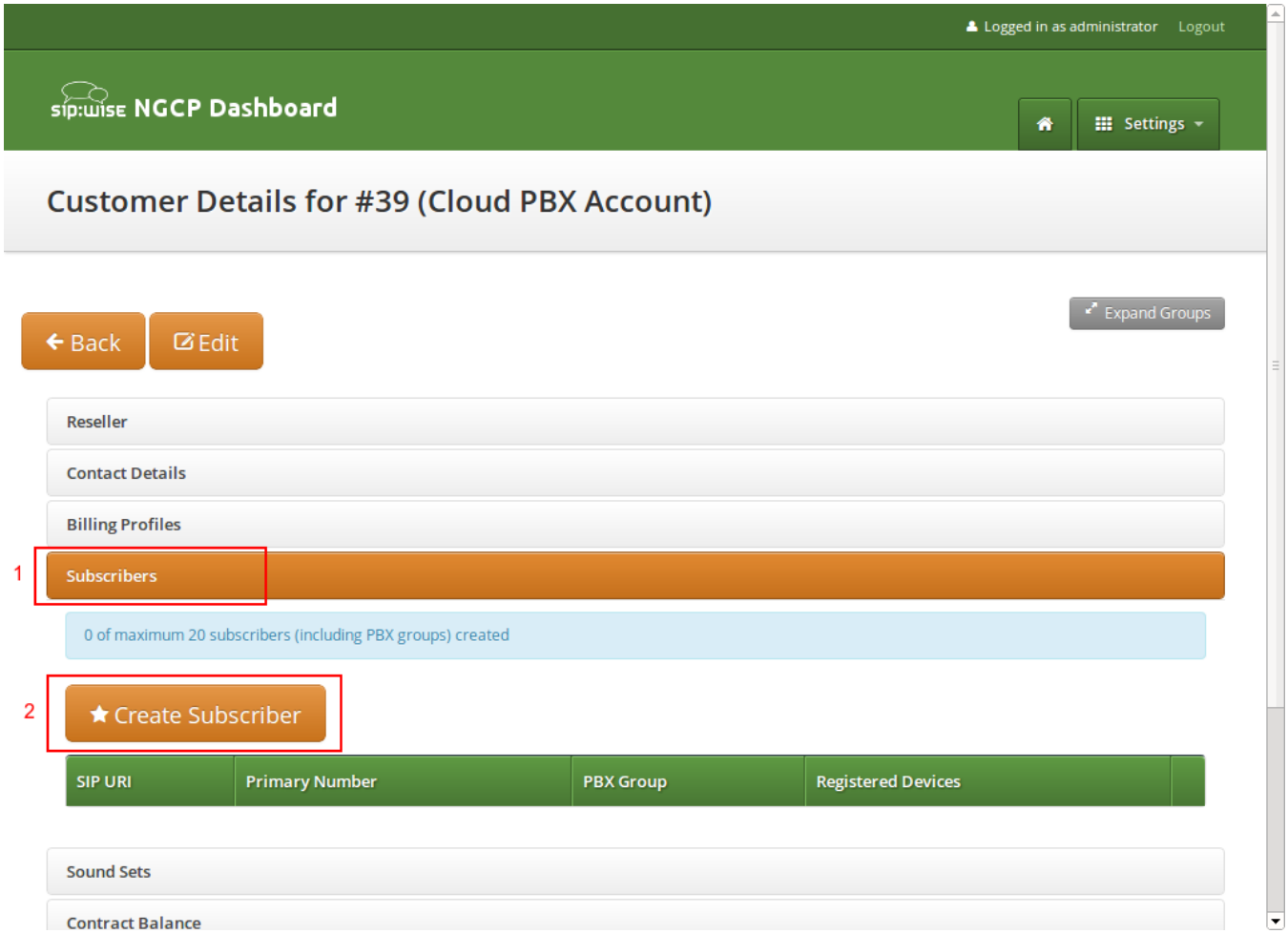


Figure 122. Go to Create Subscriber

For your pilot subscriber, you need a SIP domain, a pilot number (the main number of the customer PBX), the web credentials for the customer to log into the web interfaces, and the SIP credentials to authenticate via a SIP device.

IMPORTANT

In a PBX environment, customers can create their own subscribers. As a consequence, each PBX customer should have its own SIP domain, in order to not collide with subscribers created by other customers. This is important because two customers are highly likely to create a subscriber (or group, which is also only a subscriber) called office. If they are in the same SIP domain, they'd both have the SIP URI office@pbx.example.org, which is not allowed, and the an end customer will probably not understand why office@pbx.example.org is already taken, because he (for obvious reasons, as it belongs to a different customer) will not see this subscriber in his subscribers list.

TIP

To handle one domain per customer, you should create a wild-card entry into your DNS server like *.pbx.example.org, which points to the IP address of pbx.example.org, so you can define SIP domains like customer1.pbx.example.org or customer2.pbx.example.org without having to create a new DNS entry for each of them. For proper secure access to the web interface and to the SIP and XMPP services, you should also obtain a SSL wild-card certificate for *.pbx.example.org to avoid certification warnings on customers' web browsers and SIP/XMPP clients.

Copyright © Sipwise GmbH - All rights reserved

So to create a new domain for the customer, click *Create Domain*.

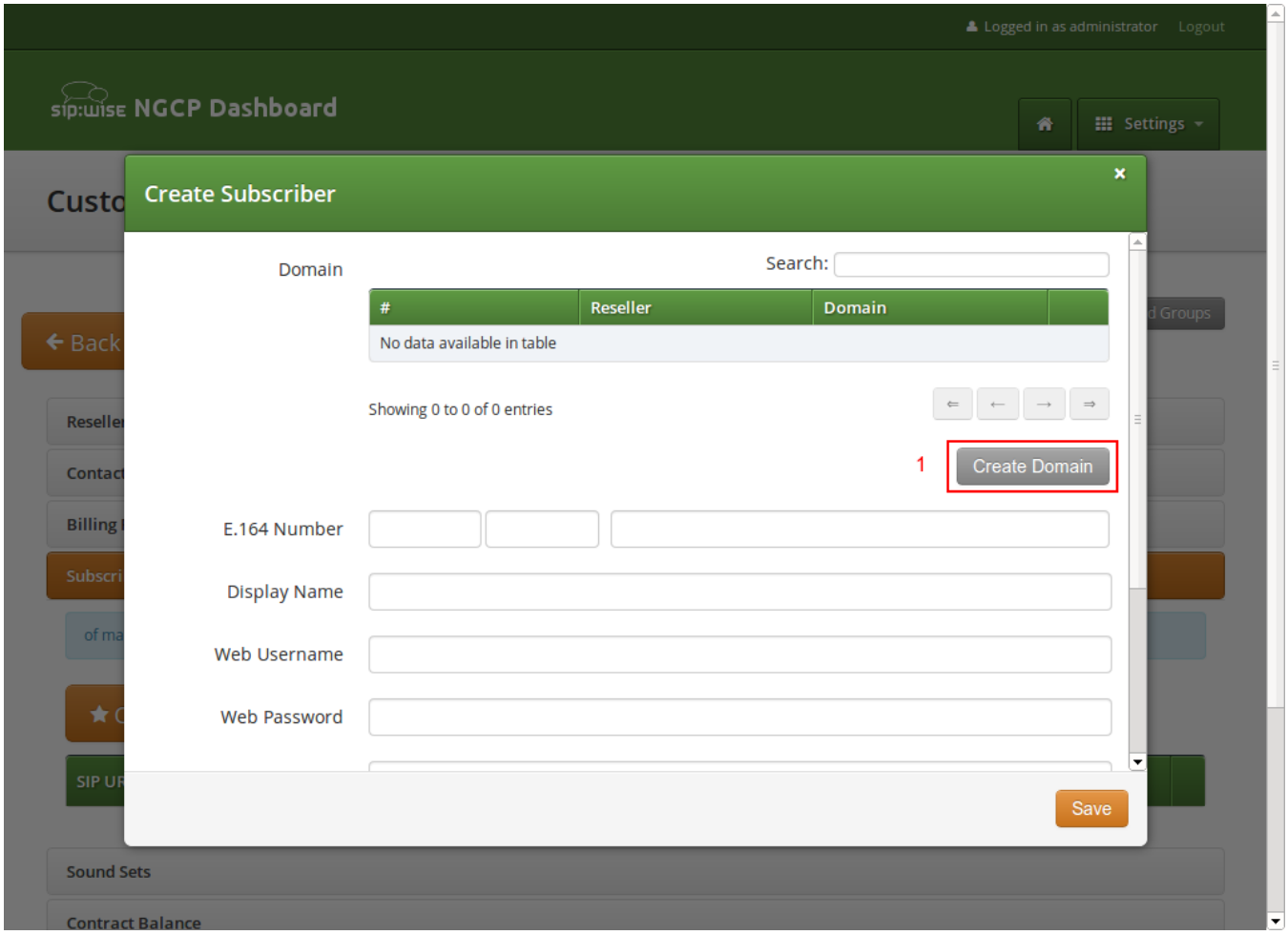


Figure 123. Go to Create Customer Domain

Specify the domain you want to create, and select the PBX *Rewrite Rule Set* which you created in Preparing PBX Rewrite Rules, then click *Save*.

Copyright © Sipwise GmbH - All rights reserved

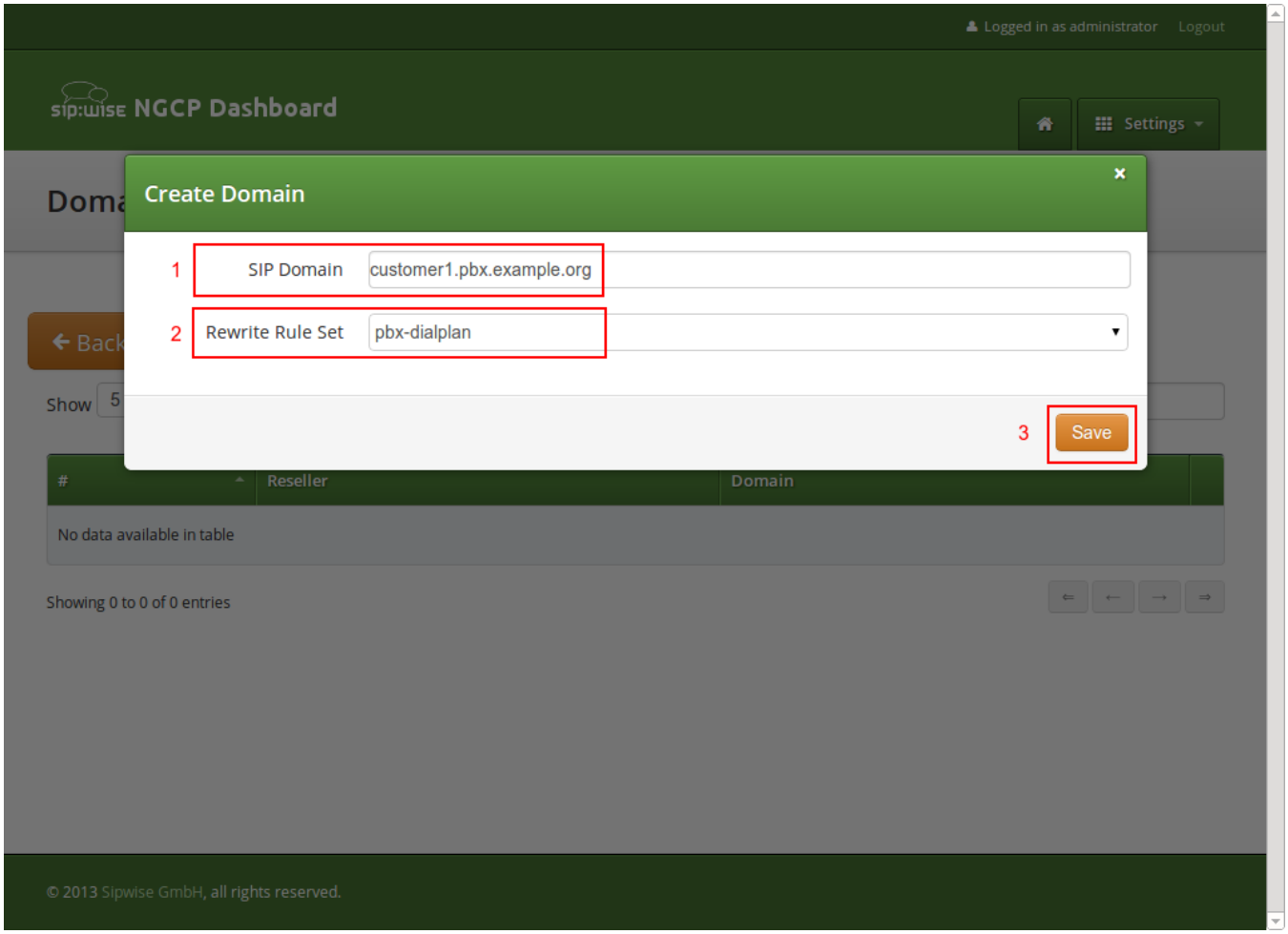


Figure 124. Create Customer Domain

Finish the subscriber creation by providing an E.164 number, which is going to be the base number for all other subscribers within this customer, the web username and password for the pilot subscriber to log into the web interface, and the sip username and password for a SIP device to connect to the PBX.

The parameters are as follows:

- **Domain**: The domain in which to create the pilot subscriber. *Each customer should get his own domain as described above to not collide with SIP usernames between customers.*
- **E.164 Number**: The primary number of the PBX. Calls to this number are routed to the pilot subscriber, and each subsequent subscriber created for this customer will use this number as its base number, suffixed by an individual extension. You can later assign alias numbers also for DID support.
- **Display Name**: This field is used on phones to identify subscribers by their real names instead of their number or extension. On outbound calls, the display name is signalled in the Display-Field of the From header, and it's used as a name in the XMPP contact lists.
- **Web Username**: The username for the subscriber to log into the customer self-care web interface. This is optional, if you don't want a subscriber to have access to the web interface.
- **Web Password**: The password for the subscriber to log into the customer self-care web interface.
- **SIP Username**: The username for the subscriber to authenticate on the SIP and XMPP service. It is automatically used for devices, which are auto-provisioned via the *Device Management*, or can be

 Copyright © Sipwise GmbH - All rights reserved

used manually by subscribers to sign into the SIP and XMPP service with any arbitrary clients.

- **SIP Password**: The password for the subscriber to authenticate on the SIP and XMPP service.

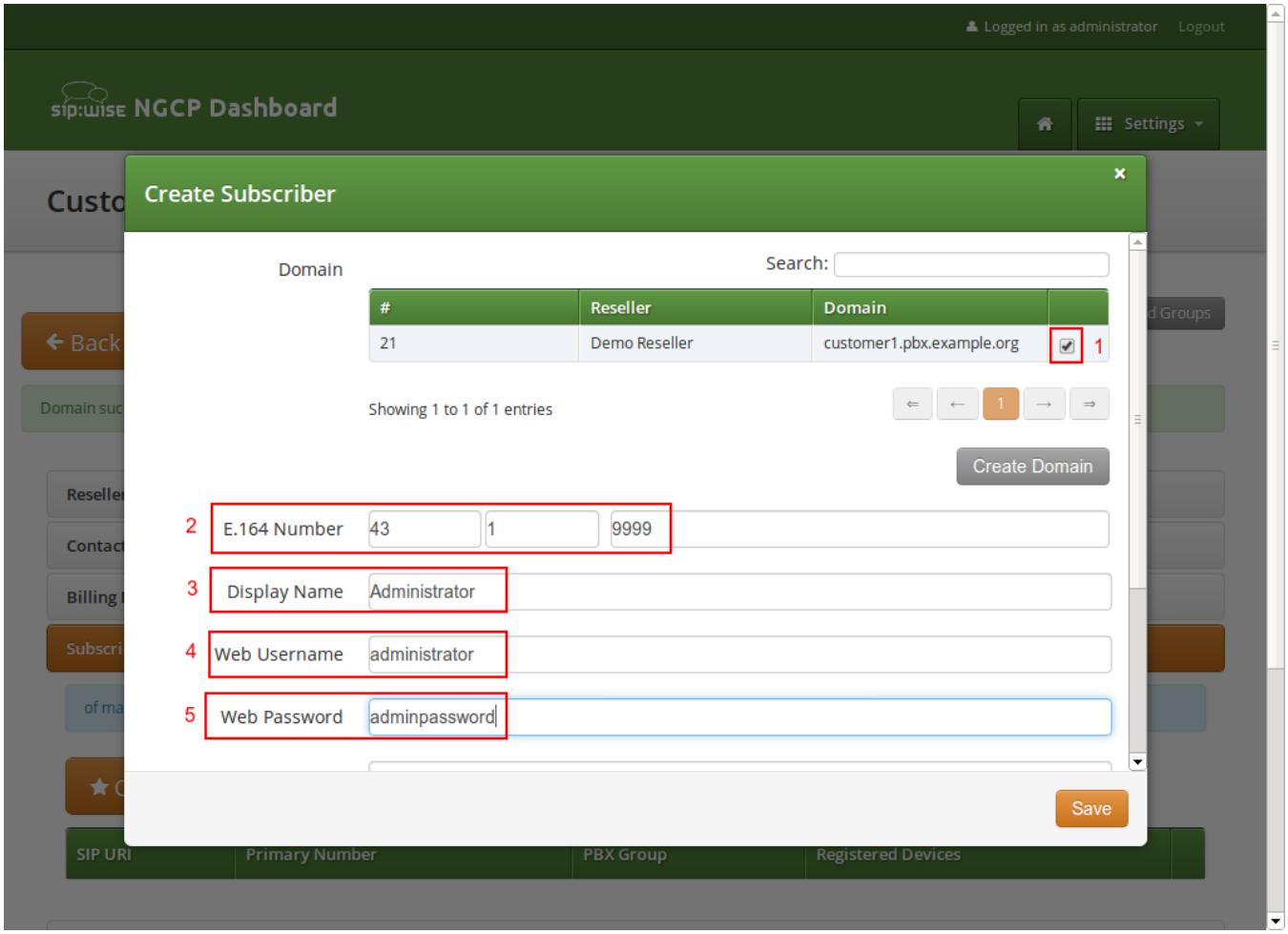


Figure 125. Create Pilot Subscriber Part 1

Copyright © Sipwise GmbH - All rights reserved

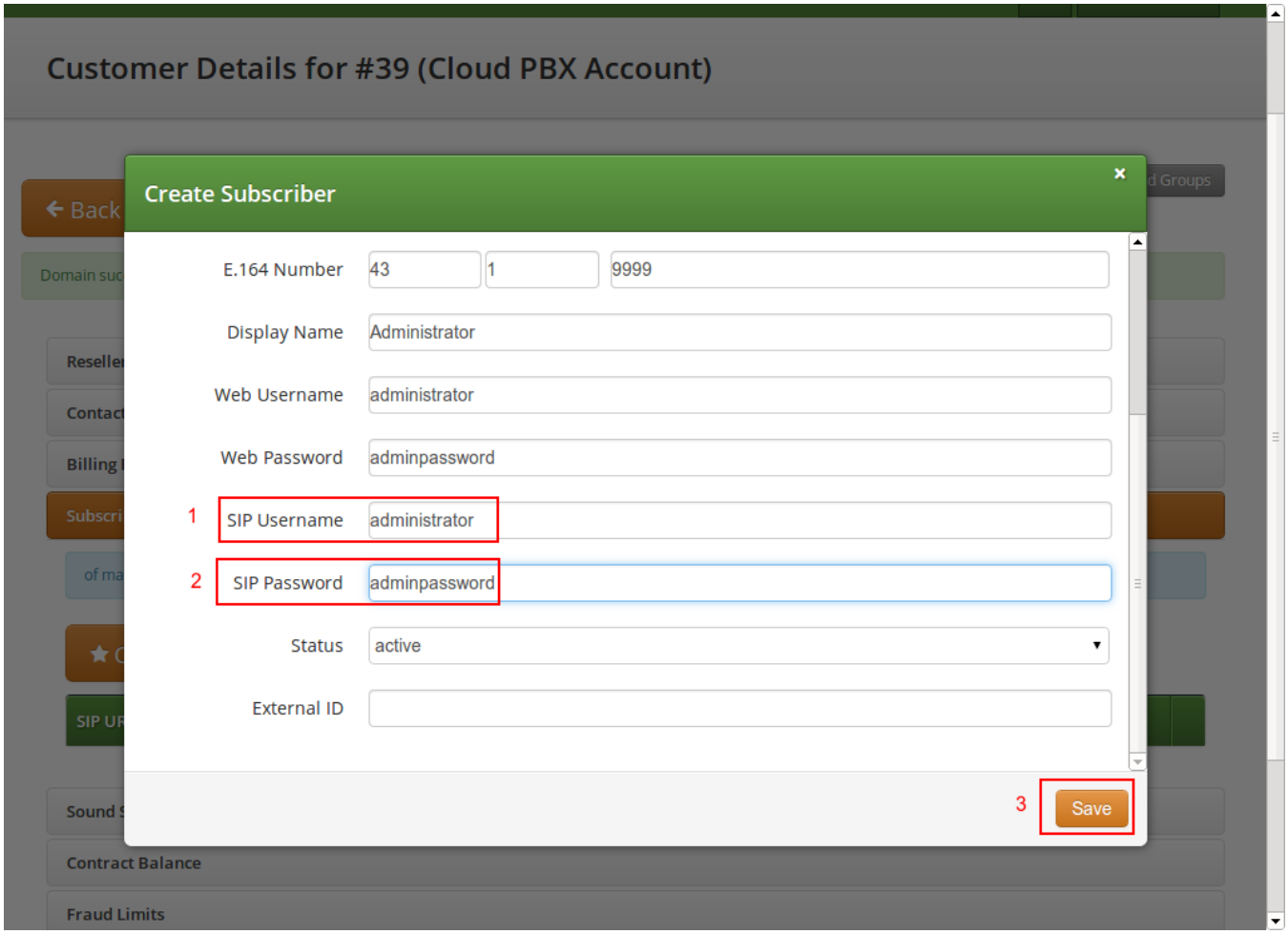


Figure 126. Create Pilot Subscriber Part 2

Once the subscriber is created, he can log into the customer self-care interface at https://<your-ip>/ and manage his PBX, like creating other users and groups, assigning Devices to subscribers and configure the Auto Attendant and more.

As an administrator, you can also do this for the customer, and we will walk through the typical steps as an administrator to configure the different features.

Go to the *Customer Details* of the PBX customer you want to configure, e.g. by navigating to *Settings Customers* and clicking the *Details* button of the customer you want to configure.

8.1.4. Creating Regular PBX Subscribers

Since we already created a pilot subscriber, more settings now appear on the *Customer Details* view. The sections we are interested in for now are the *Subscribers* and *PBX Groups* sections.

Copyright © Sipwise GmbH - All rights reserved

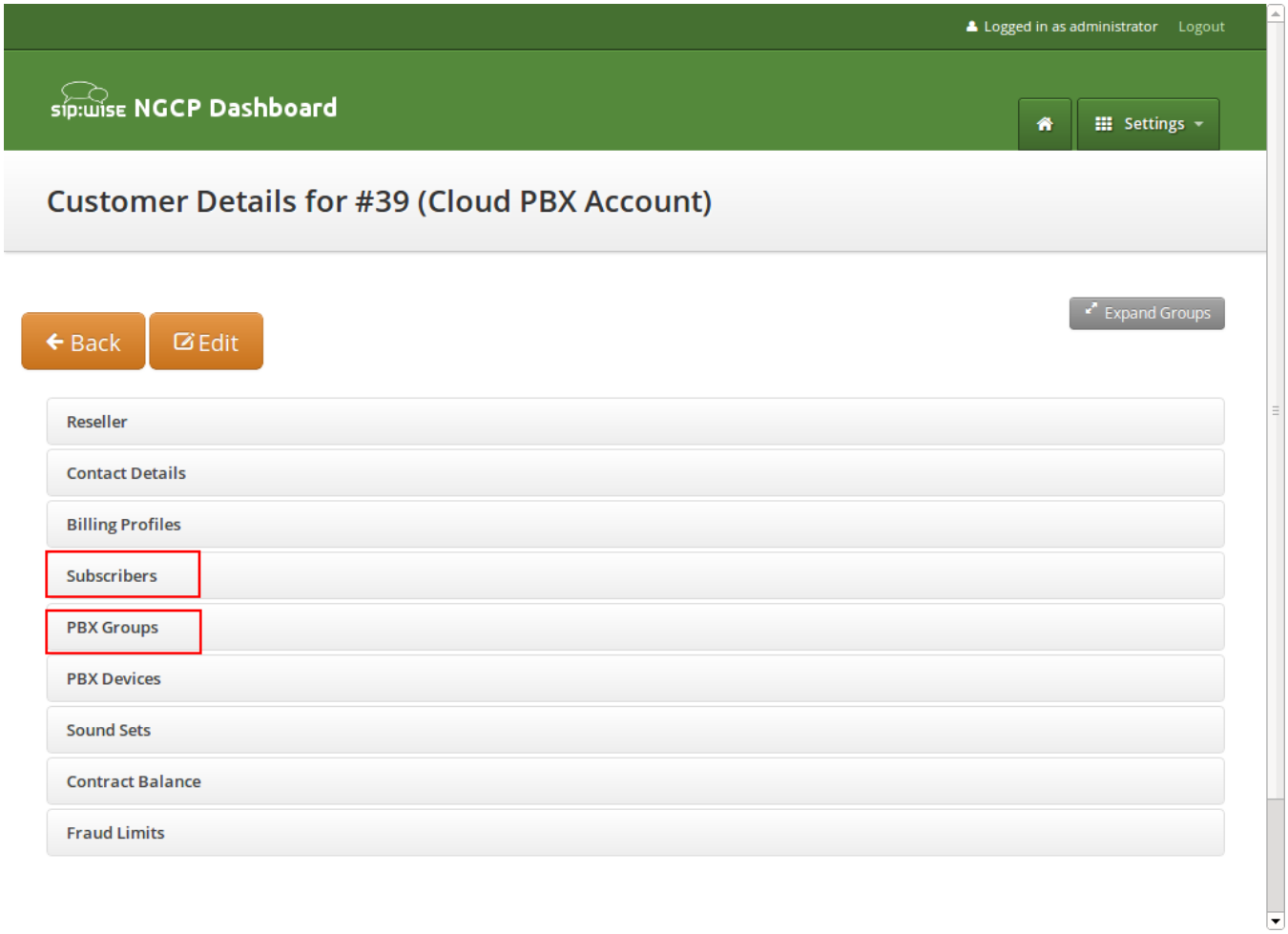


Figure 127. Subscribers and PBX Groups

To create another subscriber for the customer PBX, open the *Subscribers* row and click *Create Subscriber*.

Copyright © Sipwise GmbH - All rights reserved

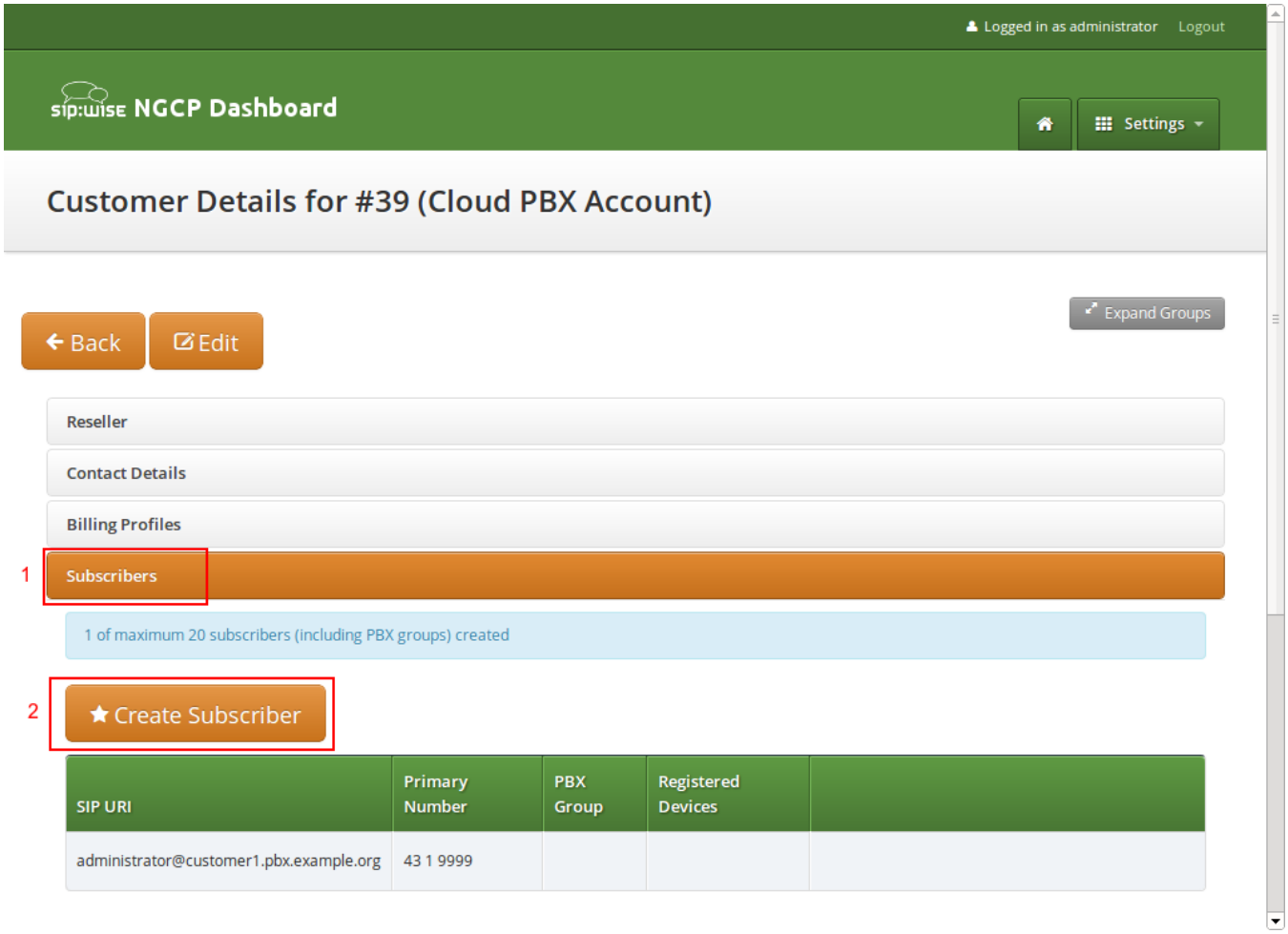


Figure 128. Create a Subscriber Extension

When creating another subscriber in the PBX after having the pilot subscriber, some fields are different now, because the *Domain* and *E.164 Number* are already pre-defined at the pilot subscriber level.

What you need to define for a new subscriber is the *Group* the subscriber is supposed to be in. We don't have a group yet, so create one by clicking *Create Group*.

A *PBX Group* has four settings:

- **Name**: The name of the group. This is used to identify a group when assigning it to subscribers on one hand, and also subscribers are pushed as server side contact lists to XMPP clients, where they are logically placed into their corresponding groups.
- **Extension**: The extension of the group, which is appended to the primary number of the pilot subscriber, so you can actually call the group from the outside. If our pilot subscriber number is 43 1 9999 and the extension is 100, you can reach the group from the outside by dialing 43 1 9999 100. Since PBX Groups are actually normal subscribers in the system, you can assign *Alias Numbers* to it for DID later, e.g. 43 1 9998.
- **Hunting Policy**: If you call a group, then all members in this group are ringing based on the policy you choose. Serial Ringing causes each of the subscribers to be tried one after another, until one of them picks up or all subscribers are tried. Parallel Ringing causes all subscribers in the group to be tried in parallel. Note that a subscriber can have a call-forward configured to some external number (e.g. his mobile phone), which will work as well.

 Copyright © Sipwise GmbH - All rights reserved

- **Serial Hunting Timeout**: This value defines for how long to ring each member of a group in case of serial hunting until the next subscriber is being tried.

We will only fill in the *Name* and *Extension* for now, as the hunting policy can be changed later if needed. Click *Save* to create the group.

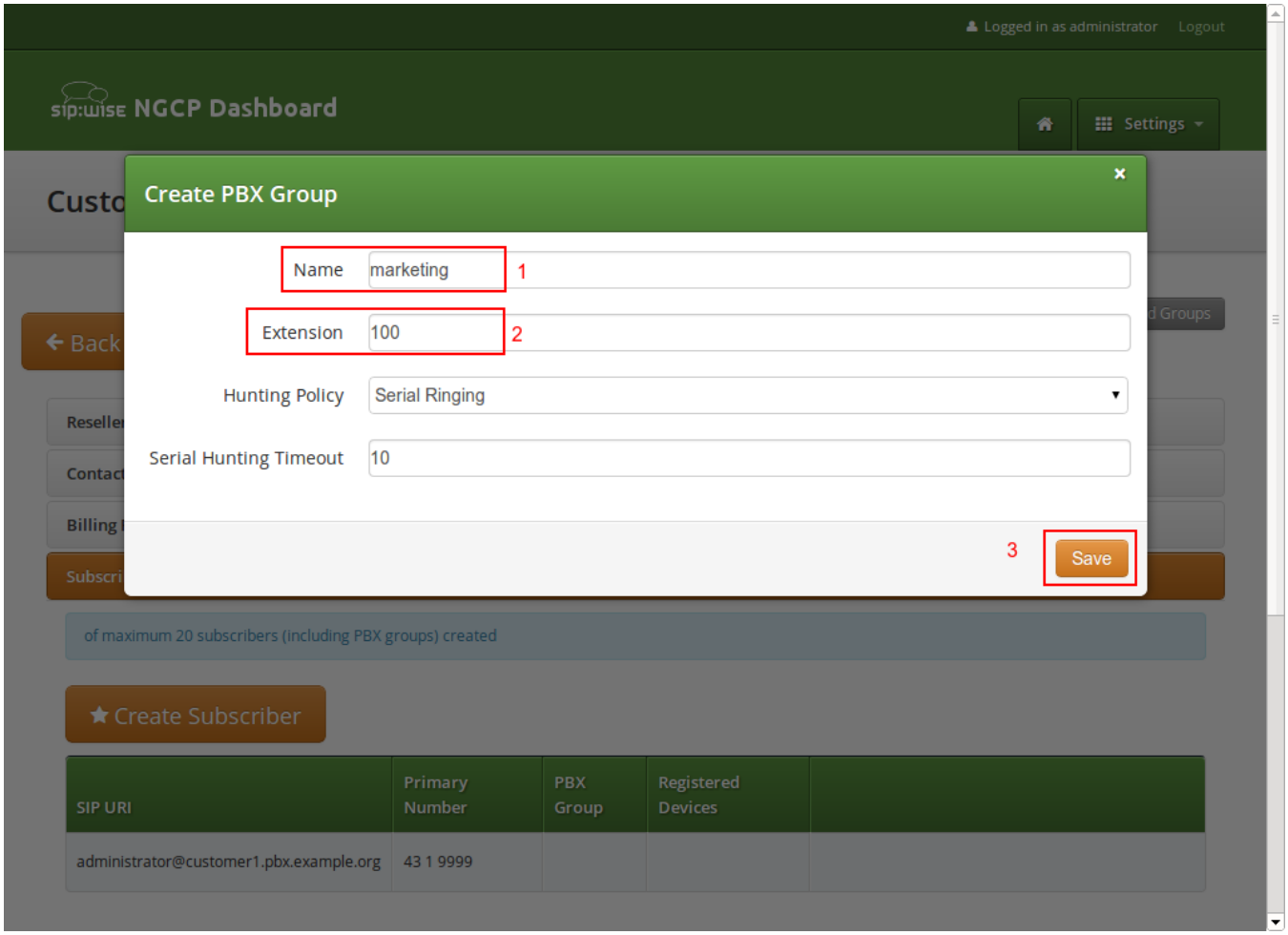


Figure 129. Create a PBX Group

Once the group is created and selected, fill out the rest of the form as needed. Instead of the *E.164 Number*, you can now only choose the *Extension*, which is appended to the primary number of the pilot subscriber and is then used as primary number for this particular subscribers. Again, if your pilot number is 43 1 9999 and you choose extension 101 here, the number of this subscriber is going to be 43 1 9999 101. Also, you can again later assign more alias numbers (e.g. 43 1 9997) to this subscriber for DID.

The rest of the fields is as usual, with *Display Name* defining the real name of the user, *Web Username* and *Web Password* allowing the subscriber to log into the customer self-care interface, and the *SIP Username* and *SIP Password* to allow signing into the SIP and XMPP services.

Copyright © Sipwise GmbH - All rights reserved

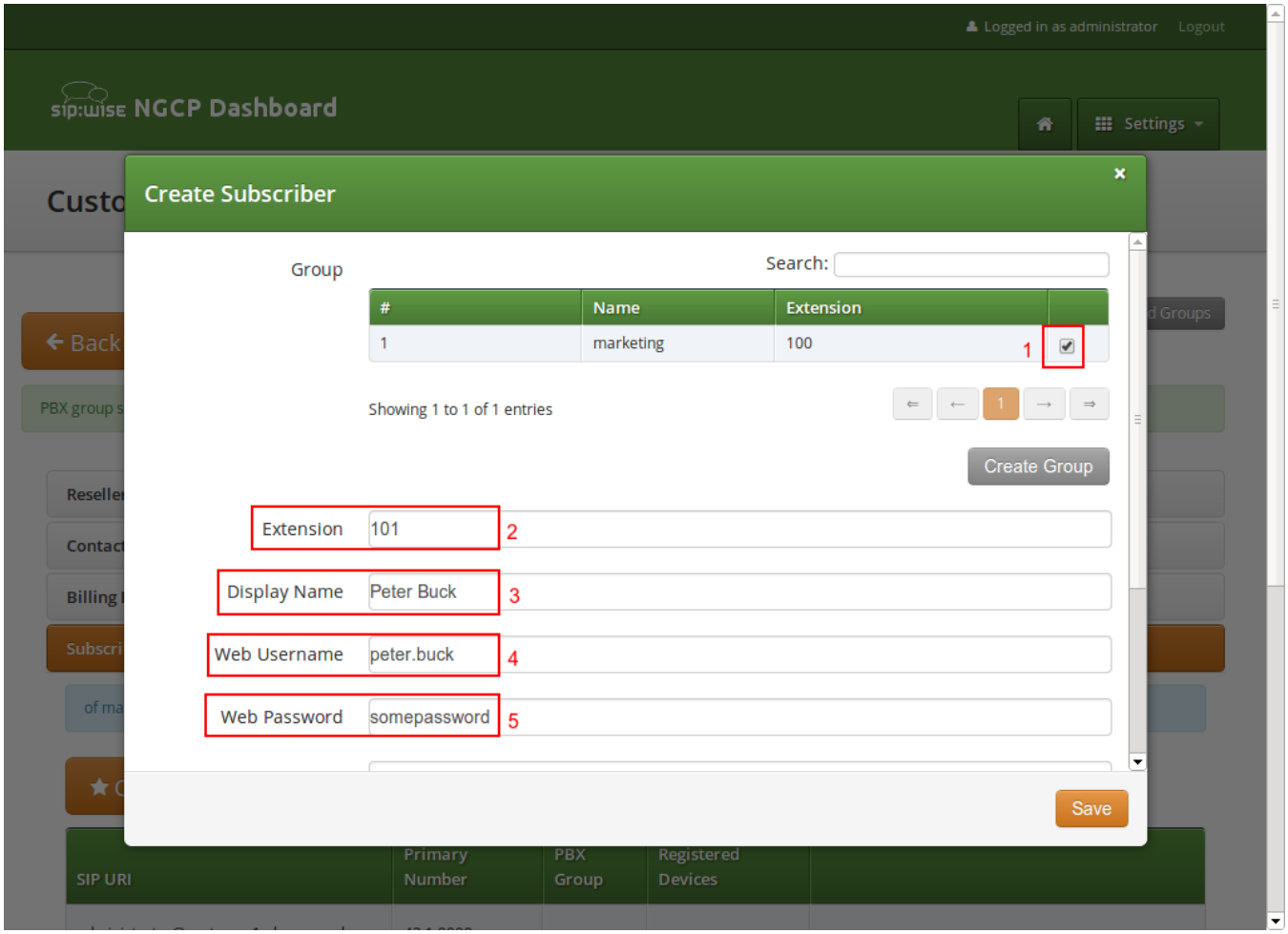


Figure 130. Finish PBX Subscriber Creation Part 1

Click *Save* to create the subscriber.

Copyright © Sipwise GmbH - All rights reserved

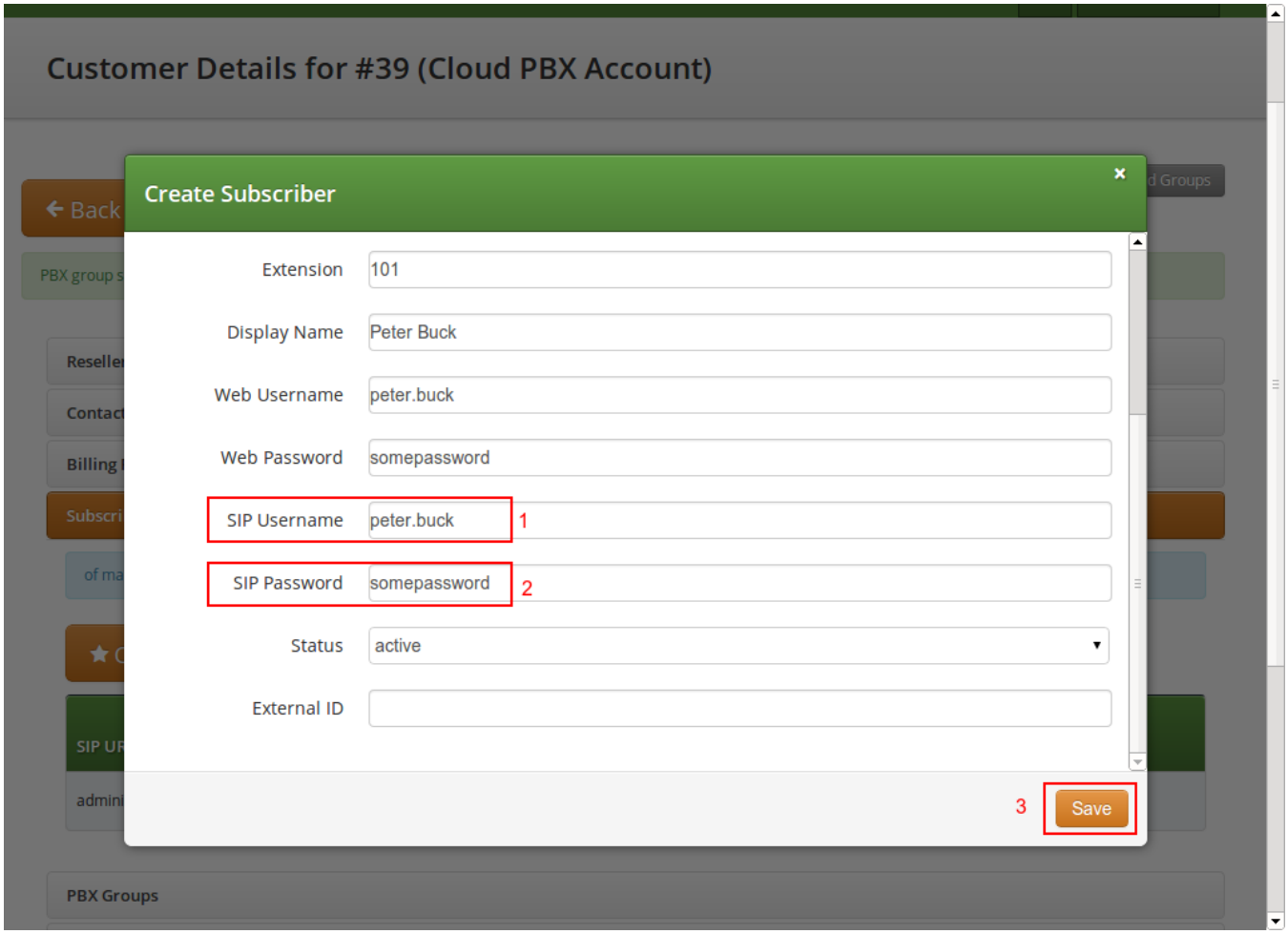


Figure 131. Finish PBX Subscriber Creation Part 2

Repeat the steps to create all the subscribers and groups as needed. An example of a small company configuration in terms of subscribers and groups might look like this:

Copyright © Sipwise GmbH - All rights reserved

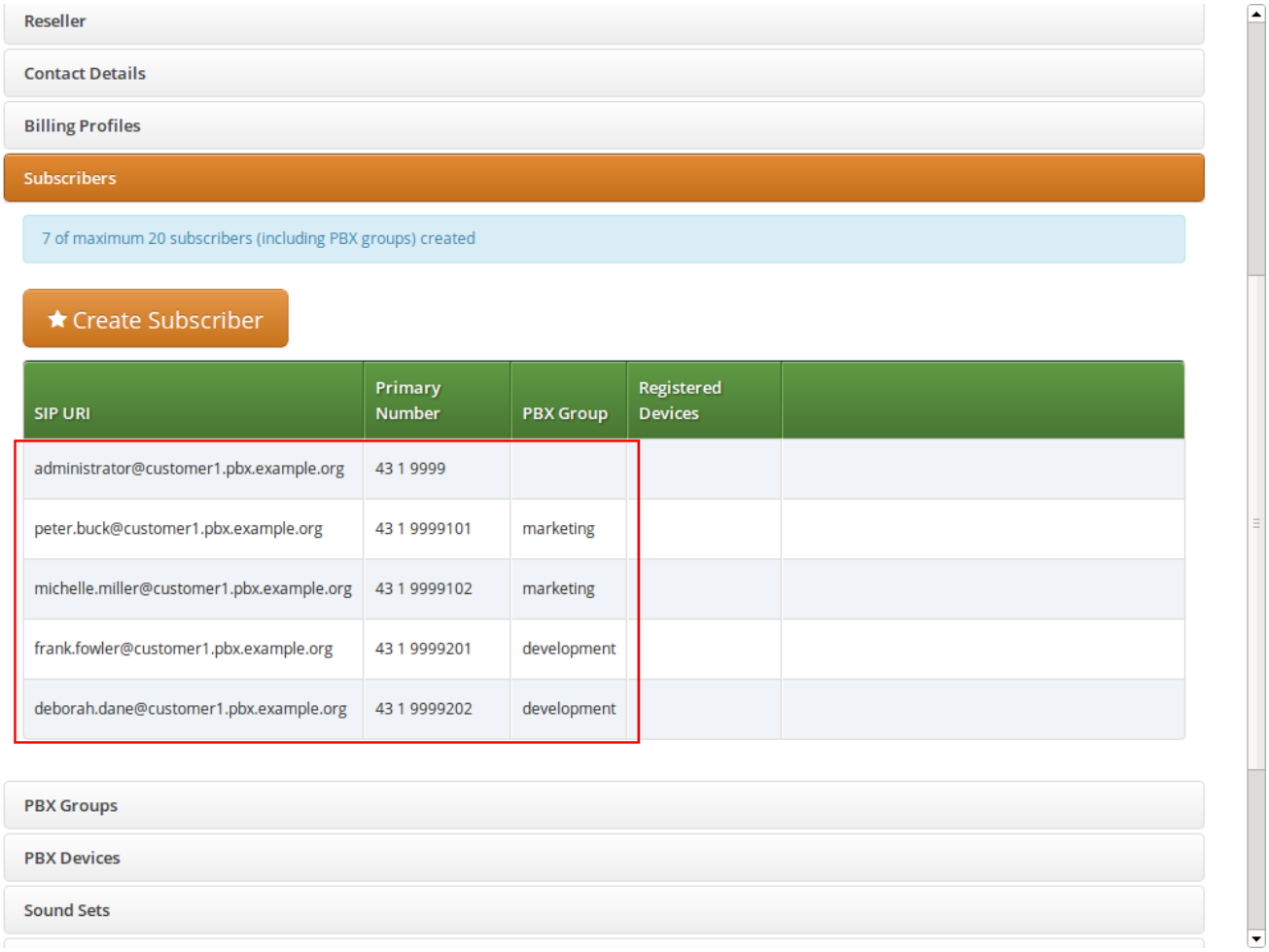


Figure 132. Example of Subscribers List

TIP

The subscribers can be reached via 3 different ways. First, you can call them by their SIP URIs (e.g. by dialing frank.fowler@customer1.pbx.example.org) from both inside and outside the PBX. Second, you can dial by the full number (e.g. 43 1 9999 201; depending on your rewrite rules, you might need to add a leading \+ or 00 or leave out the country code when dialing from the outside, and adding a 0 as break-out digit when dialing from the inside) from both inside and outside the PBX. Third, you can dial the extension (e.g. 201) from inside the PBX. If the subscriber also has an alias number assigned, you can dial that number also, according to your dial-plan in the rewrite rules.

8.1.5. Device/Aliases (how to register a phone on an alias)

Subscribers belonging to CloudPBX customers can create a special type of alias numbers called **Device/Alias**. Device/Alias differs in two important things compared to standard aliases:

- A device can register directly on a Device/Alias number. To do that configure the Device/Alias number on the authentication username of the device instead of using the sip username defined during the creation of the subscriber. The password to use is instead still the same.
- Devices registered directly on the alias number can make calls as devices registered using the subscribers' sip username. Incoming calls, instead, behave differently:

  Incoming calls directed to the SIP Uri, to the full number, to the extension or to 'standard aliases' will let ring **ONLY** devices registered using the subscriber's sip username.

 Copyright © Sipwise GmbH - All rights reserved

Incoming calls directed to Device/Aliases will let ring **ONLY** devices registered on that particular Device/Alias.

How to create a Device/Alias

To create a Device/Alias proceed as for the creation of a standard alias but flag the *Is Device ID* option shown just under the *Alias Number* definition.

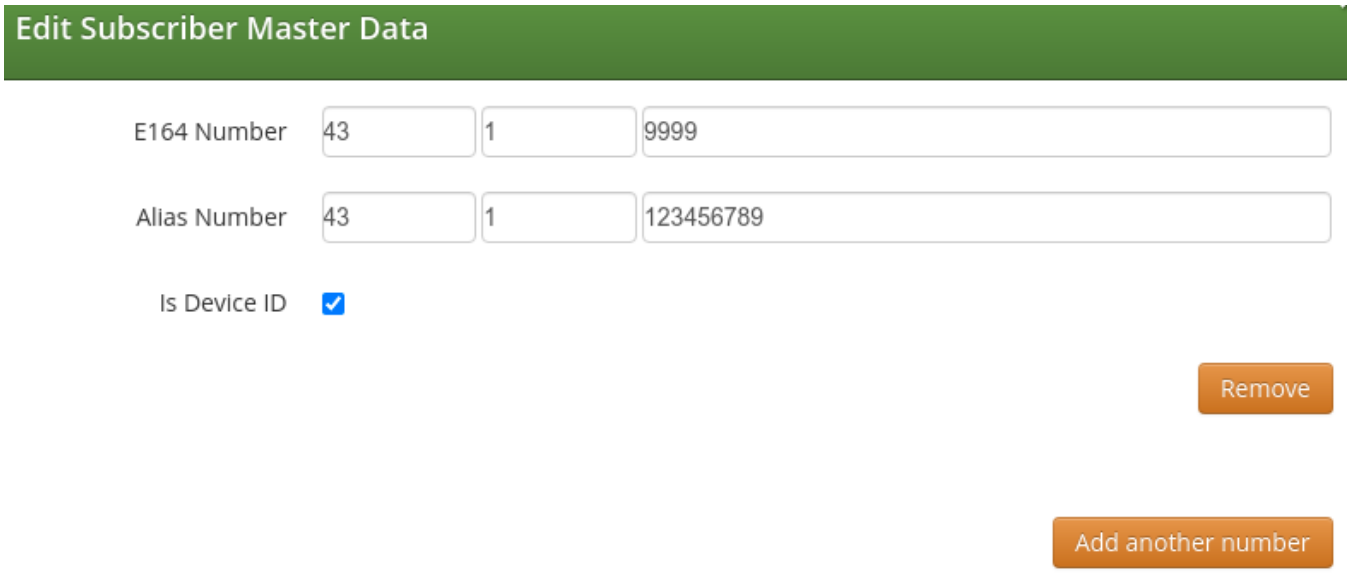


Figure 133. Device/Alias Creation

To easily distinguish which is the type of alias associated to a subscriber, Device/Alias are reported with a handset logo just beside the number in the subscriber's master data.

Primary Number	43 1 9999202
Alias Numbers	43 1 123456789 43 1 000000000
Extension	202

Figure 134. Show Device/Alias Master Data

IMPORTANT Internal calls to Device/Aliases won't work in combination with Extended matching inbound routing user preference.

lookup_all_registrations preference

The *lookup_all_registrations* subscriber's preference allows to modify the incoming call behaviour described above. In particular, if the preference is flagged the behaviour is modified as following:

- Incoming calls directed to the SIP Uri, to the full number, to the extension or to 'standard aliases' will let ring **BOTH** devices registered using the subscriber's sip username and devices registered on all the Device/Aliases.
- Incoming calls directed to Device/Aliases will continuous let ring **ONLY** devices registered on that particular Device/Alias.

Copyright © Sipwise GmbH - All rights reserved

8.1.6. Assigning Subscribers to a Device

You can register any SIP phone with the system using a SIP subscriber credentials. However, the platform supports *PBX Device Provisioning* of certain vendors and models, as described in PBX Device Provisioning.

To configure a physical device, expand the *PBX Devices* section in the *Customer Details* page and click *Create Device*.

Set up three general parameters for the new device, which are:

- **Device Profile**: The actual device profile you want to use. This has been pre-configured in the *Device Management* by the administrator or reseller, and the customer can choose from the list of profiles (which is a combination of an actual device plus its corresponding configuration).
- **MAC Address/Identifier**: The MAC address of the phone to be added. The information can usually either be found on the back of the phone, or in the phone menu itself.
- **Station Name**: Since you can (depending on the actual device) configure more lines on a phone, you can give it a station name, like Reception or the name of the owner of the device.

In addition to that information, you can configure the lines (subscribers) you want to use on which key, and the mode of operation (e.g. if it's a normal private phone line, or if you want to monitor another subscriber using BLF, or if you want it to act as shared line using SLA).

For example, a *Cisco SPA504G* has 4 keys you can use for private and shared lines as well as BLF on the phone itself, and in our example we have an *Attendant Console* attached to it as well, so you have 32 more keys for BLF.

The settings per key are as follows:

- **Subscriber**: The subscriber to use (for private/shared lines) or to monitor (for BLF).
- **Line/Key**: The key where to configure this subscriber to.
- **Line/Key Type**: The mode of operation for this key, with the following options (depending on which options are enabled in the *Device Model* configuration for this device:

  Private Line: Use the subscriber as a regular SIP phone line. This means that the phone will register the subscriber, and you can place and receive phone calls with/for this subscriber.

  Shared Line: The subscriber is also registered on the system and you can place and receive calls. If another phone has the same subscriber also configured as shared line, both phones will ring on incoming calls, and you can pick the call up on either of them. You cannot place a call with this subscriber though if the line is already in use by another subscriber. However, you can "steal" a running call by pressing the key where the shared line is configured to barge into a running call. The other party (the other phone where the shared line is configured too) will then be removed from the call (but can steal the call back the same way).

  BLF Key: The *Busy Lamp Field* monitors the call state of another subscriber and provides three different functionalities, depending on the actual state:

    Speed Dial: If the monitored subscriber is on-hook, the user can press the button and directly call the monitored subscriber.

    Call Pickup: If the monitored subscriber is ringing, the user can press the button to pick up the call on his own phone.

Copyright © Sipwise GmbH - All rights reserved

State Indication: It the monitored subscriber is on the phone, the key is indicating that the monitored subscriber is currently busy.

In our example, we will configure a private line on the first key, and the BLF for another subscriber on the second key.

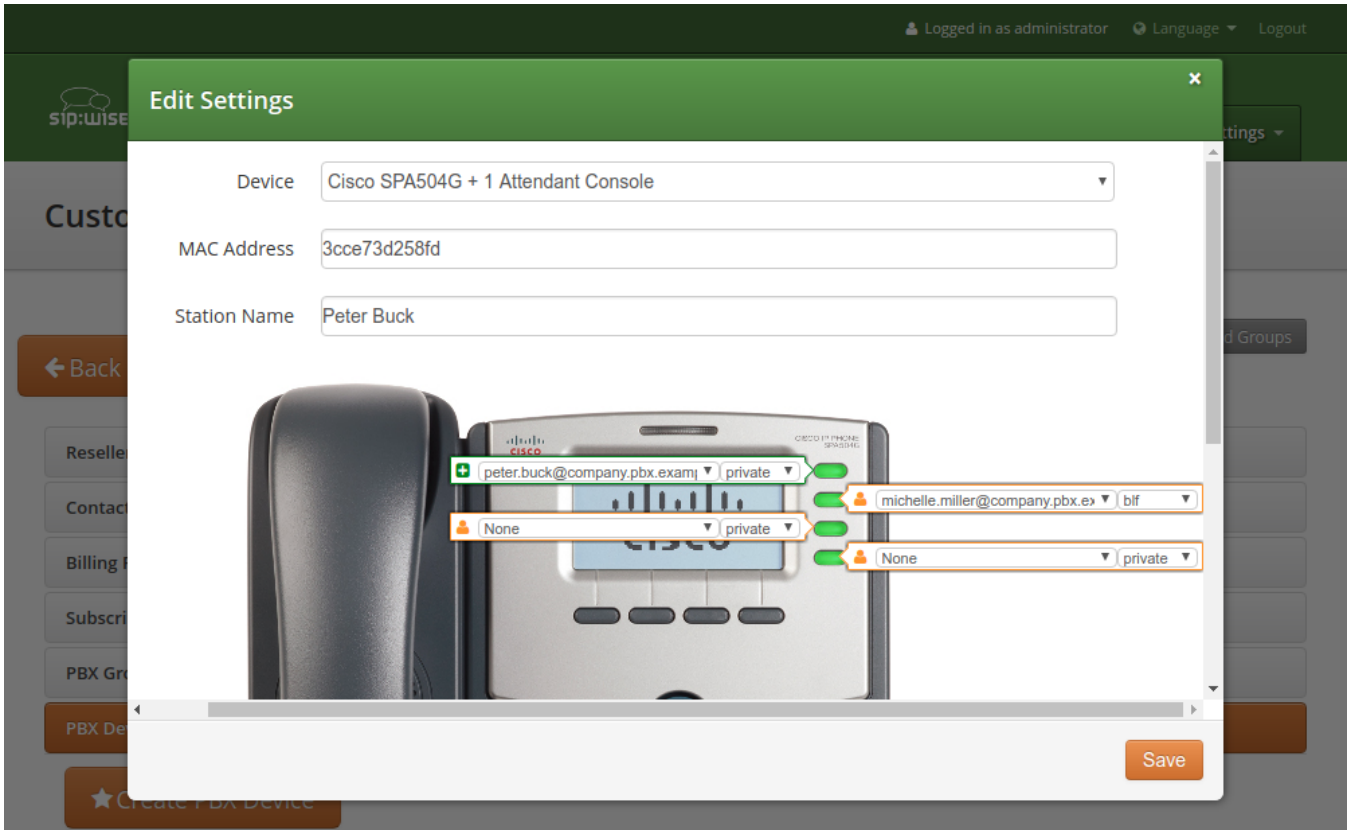


Figure 135. Configuring a PBX Device

Once the PBX device is saved, you will see it in the list of *PBX Devices*.

Initial provisioning of a PBX Device

Depending on a manufacturer and the model, there are two ways of provisioning a device:

- (legacy) putting the provisioning URL directly into the device via a web browser (this option is used e.g. for Cisco devices);
- (recommended) using the device's Zero Touch Provisioning (ZTP) feature. For Yealink it is called Redirection and Provisioning Service (RPS). Also available are Snom redirection and provisioning service (SRAPS) and Easy Deployment Server (EDS) from ALE.

Direct device provisioning

Since a stock device obtained from an arbitrary distributor doesn't know anything about your system, it can't fetch its configuration from there. For that to work, you need to push the URL of where the phone can get the configuration to the phone once.

In order to do so, click the *Sync Device* button on the device you want to configure for the very first time.

Copyright © Sipwise GmbH - All rights reserved

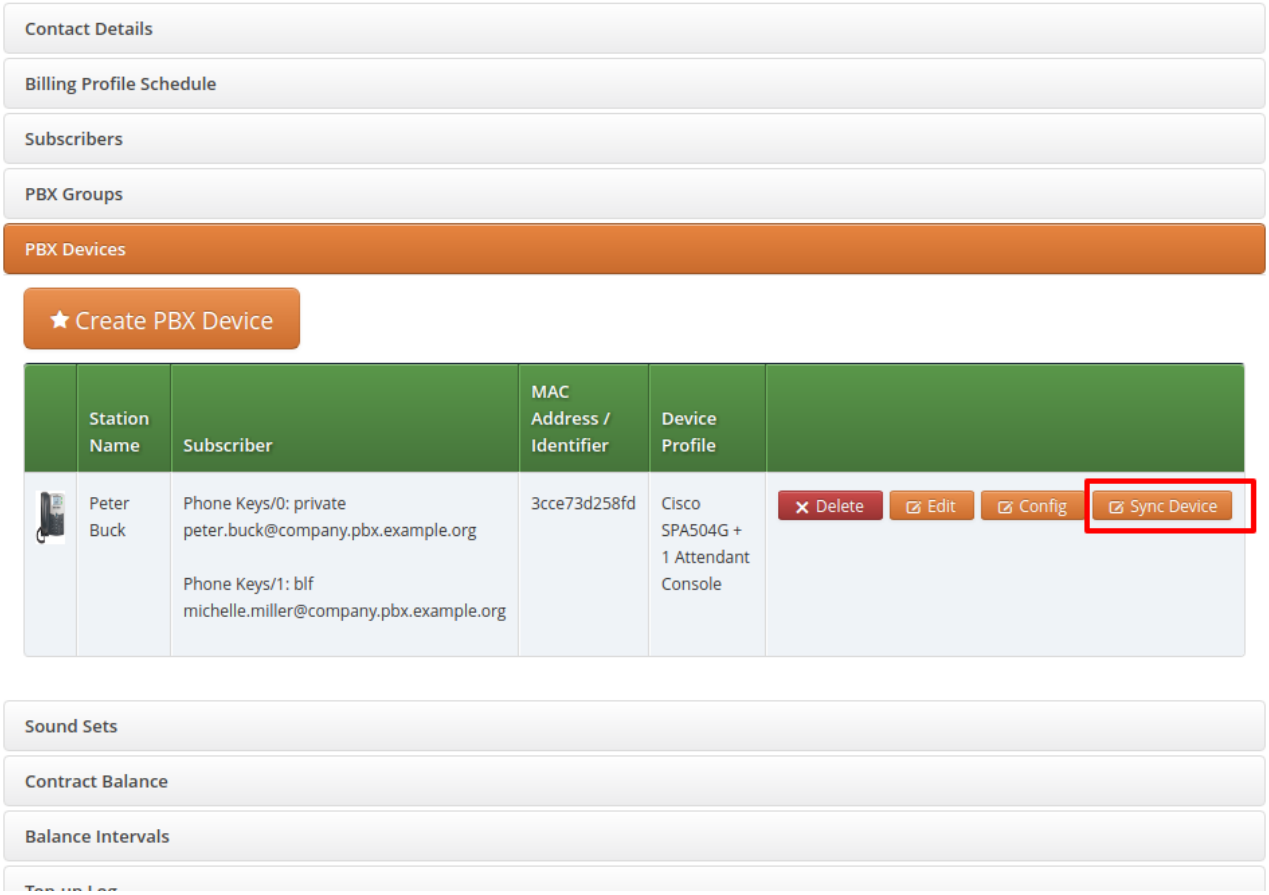


Figure 136. Go to Sync Device

IMPORTANT

As you will see in the next step, you need the actual IP address of the phone to push the provisioning URL onto it. That implies that you need access to the phone to get the IP, and that your browser is in the same network as the phone in order to be able to connect to it, in case the phone is behind NAT.

Enter the IP Address of the phone (on Cisco SPAs, press Settings 9, where Settings is the paper sheet symbol, and note down the Current IP setting), then click *Push Provisioning URL*.

Copyright © Sipwise GmbH - All rights reserved

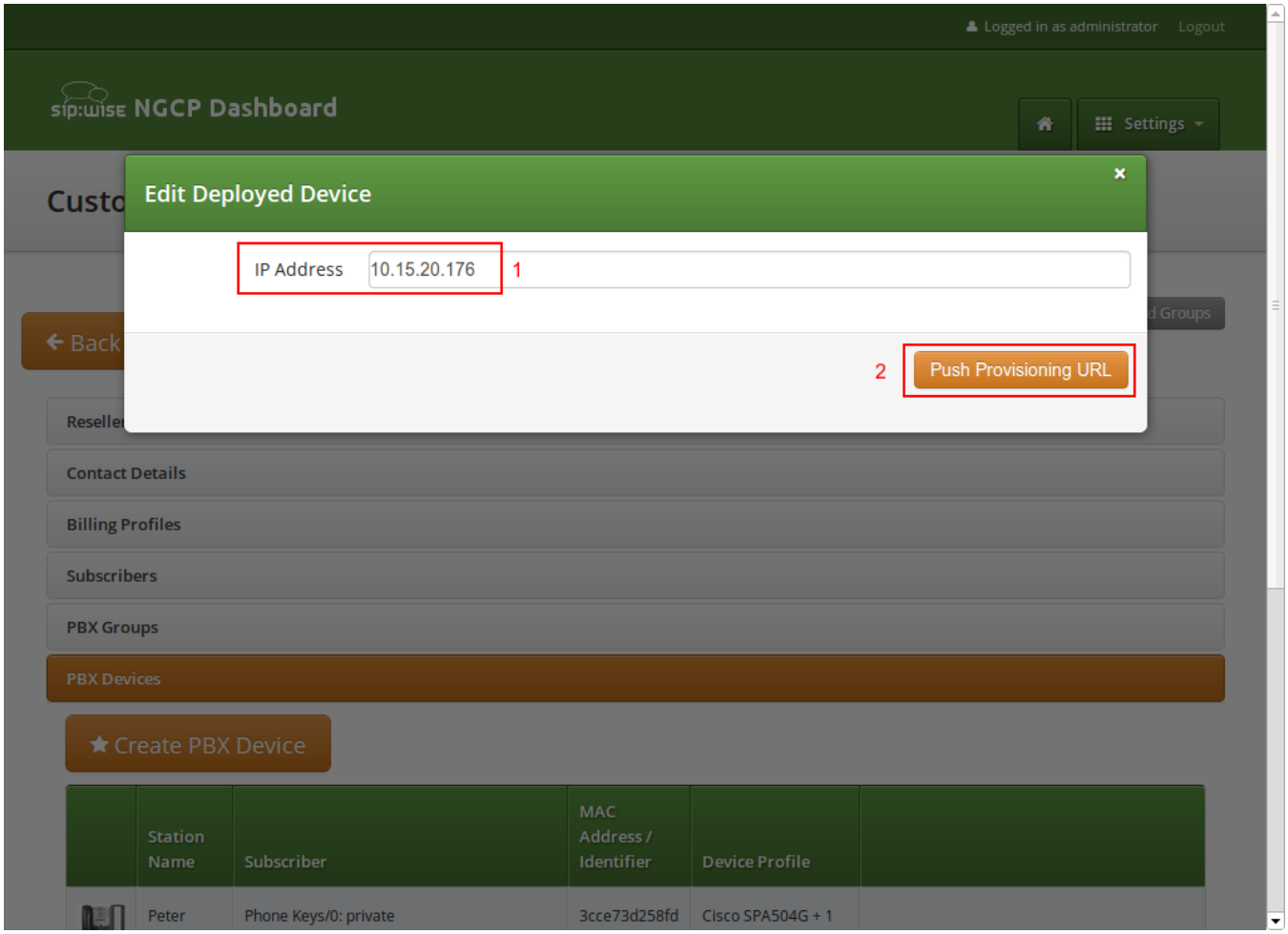


Figure 137. Sync Device

You will be redirected directly to the phone, and the Provisioning URL is automatically set. If everything goes right, you will see a confirmation page from the phone that it's going to reboot.

Copyright © Sipwise GmbH - All rights reserved

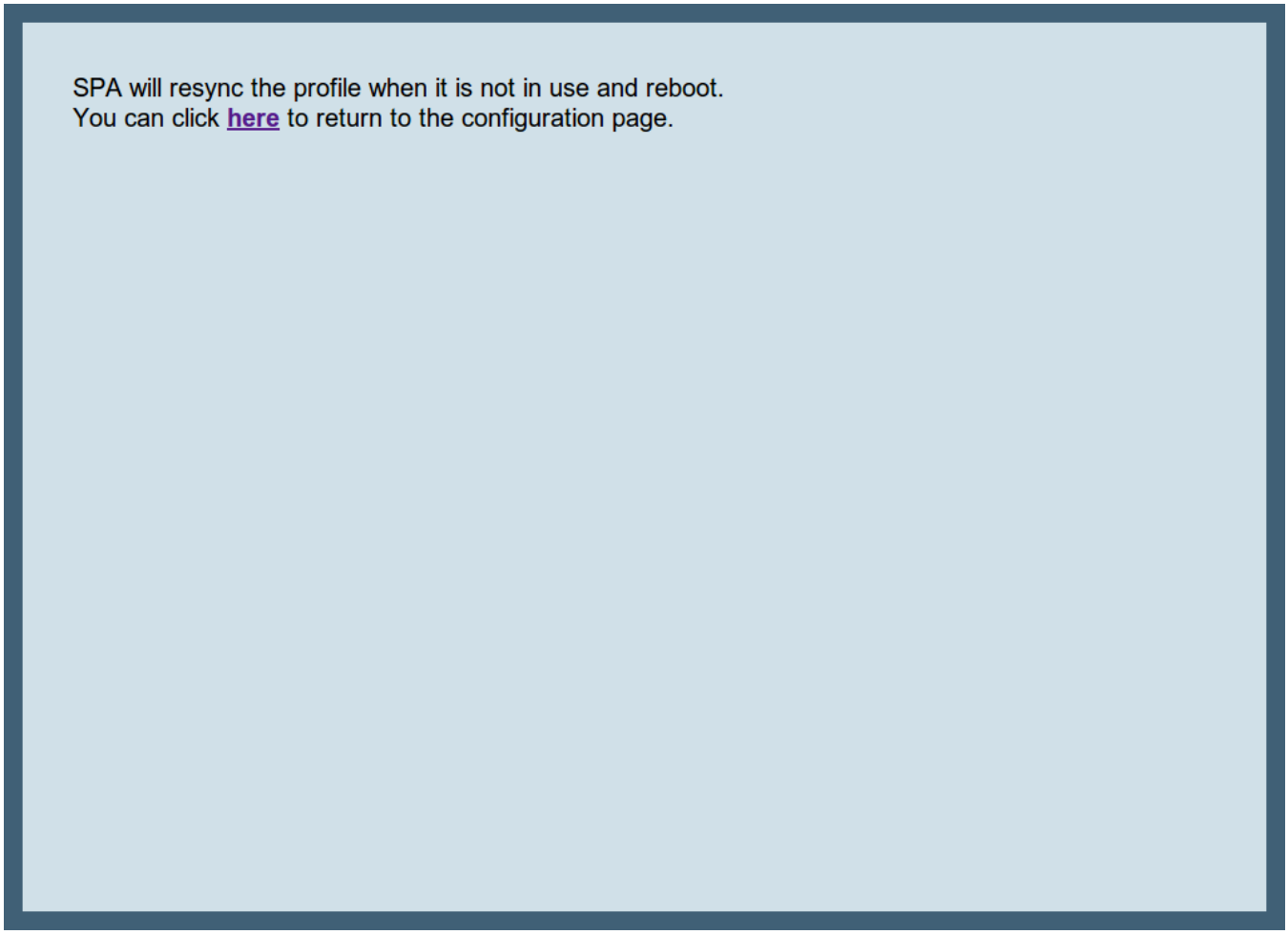


Figure 138. Device Sync Confirmation from Phone

You can close the browser window/tab and proceed to sync the next subscriber.

TIP You only have to do this step once per phone to tell it the actual provisioning URL, where it can fetch the configuration from. From there, it will regularly sync with the server automatically to check for configuration changes and apply them automatically.

Provisioning a device using ZTP/RPS/SRAPS/EDS

All ALE, Polycom, Panasonic, Snom, Grandstream and Yealink phones supported by Sipwise C5 can be provisioned using ZTP/RPS/SRAPS/EDS service without physically accessing the devices. You only need to input MAC addresses of corresponding devices and associate them with subscribers. Sipwise C5 will then immediately supply this information to the remote system of the corresponding device vendor. When a subscriber unpacks the phone and connects it to the Internet for the first time, the phone will contact the manufacturer's service and get its provisioning URL to Sipwise C5. Then, the phone downloads all required items from Sipwise C5 and automatically configures itself. Immediately after that, the subscriber can make the first call.

To prepare a PBX device for zero-touch provisioning, follow these steps:

- Go to the PBX Devices section of the corresponding customer and click *Create PBX Device*.
- Specify the device and its SIP lines parameters:

 Copyright © Sipwise GmbH - All rights reserved

Select the required device model

Input the device MAC address

Specify the name of this line for your convenience

Select a subscriber from the list for the corresponding SIP line. Some devices support multiple lines and you can provision all of them at once.

Select the line type: *private*, *shared* or *BLF*.

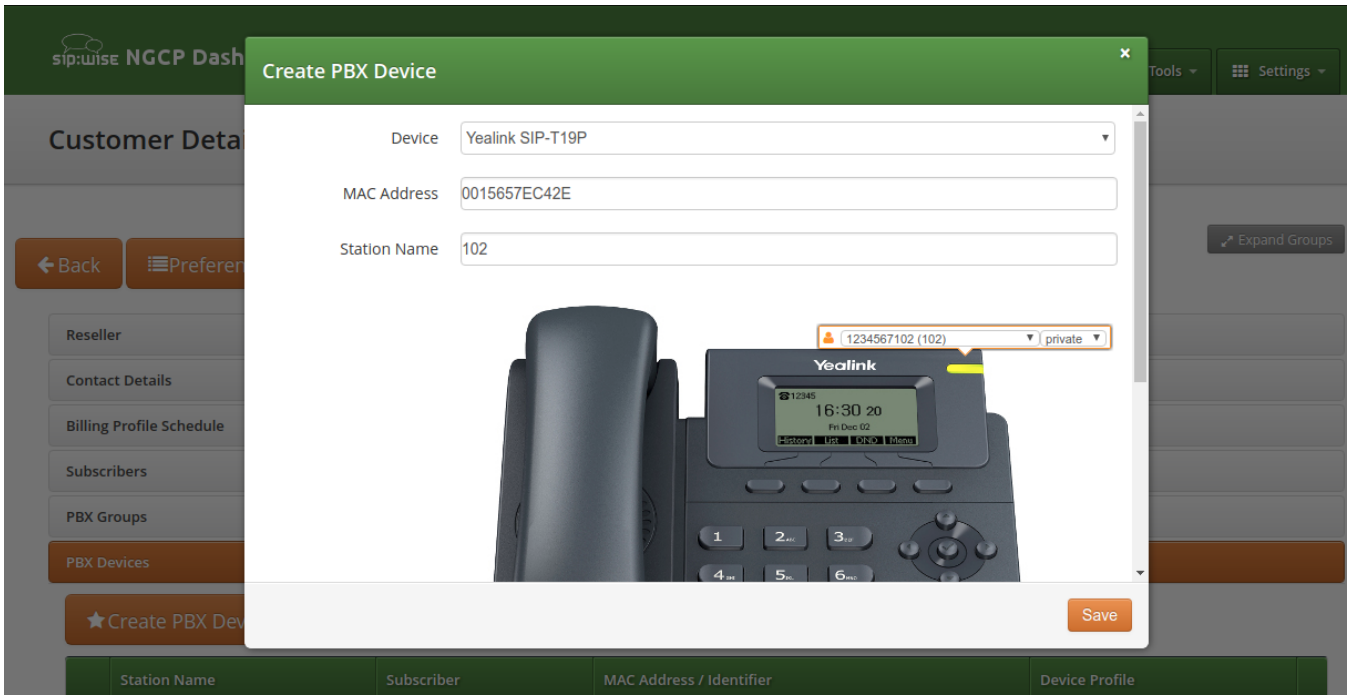


Figure 139. Create a PBX device

- Click *Save*. You will see the device in the list of customer's PBX devices.

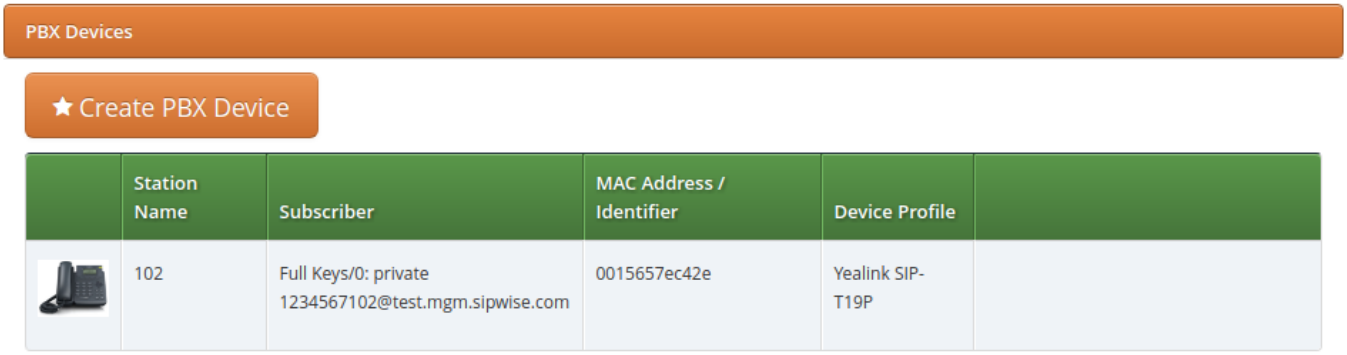


Figure 140. Created a new PBX device

TIP If you have already provisioned a specific device on another platform or for another reseller, then you might need to delete that MAC address manually from the ZTP/RPS service (it is device vendor specific).

When the PBX device provisions itself, it will become registered with your SIP proxy server. From then, it will be listed in the subscriber's *Registered Devices* page.

Copyright © Sipwise GmbH - All rights reserved



Figure 141. Registered devices

If you need to troubleshoot the provisioning process, the following logs would help you:

- /var/log/ngcp/nginx (e.g. SSL errors are collected here: autoprov_bootstrap_error.log and autoprov_error.log)
- /var/log/ngcp/panel-debug.log (general provisioning logs)

TIP In case you would like to edit a device model, firmware, configuration or profile, refer to Adjusting the PBX Devices Configuration

8.1.7. Configuring Sound Sets for the Customer PBX

In the *Customer Details* view, there is a row *Sound Sets*, where the customer can define his own sound sets for *Auto Attendant*, *Music on Hold* and the *Office Hours Announcement*.

To create a new sound set, open the *Sound Sets* row and click *Create Sound Set*.

If you do this as administrator or reseller, the Reseller and/or Customer is pre-selected, so keep it as is. If you do this as customer, you don't see any *Reseller* or *Customer* fields.

So the important settings are:

- **Name**: The name of the sound set as it will appear in the *Subscriber Preferences*, where you can assign the sound set to a subscriber.
- **Description**: A more detailed description of the sound set.
- **Default for Subscribers**: If this setting is enabled, then the sound set is automatically assigned to all already existing subscribers which do NOT have a sound set assigned yet, and also for all newly created subscribers.

Fill in the settings and click *Save*.

 Copyright © Sipwise GmbH - All rights reserved

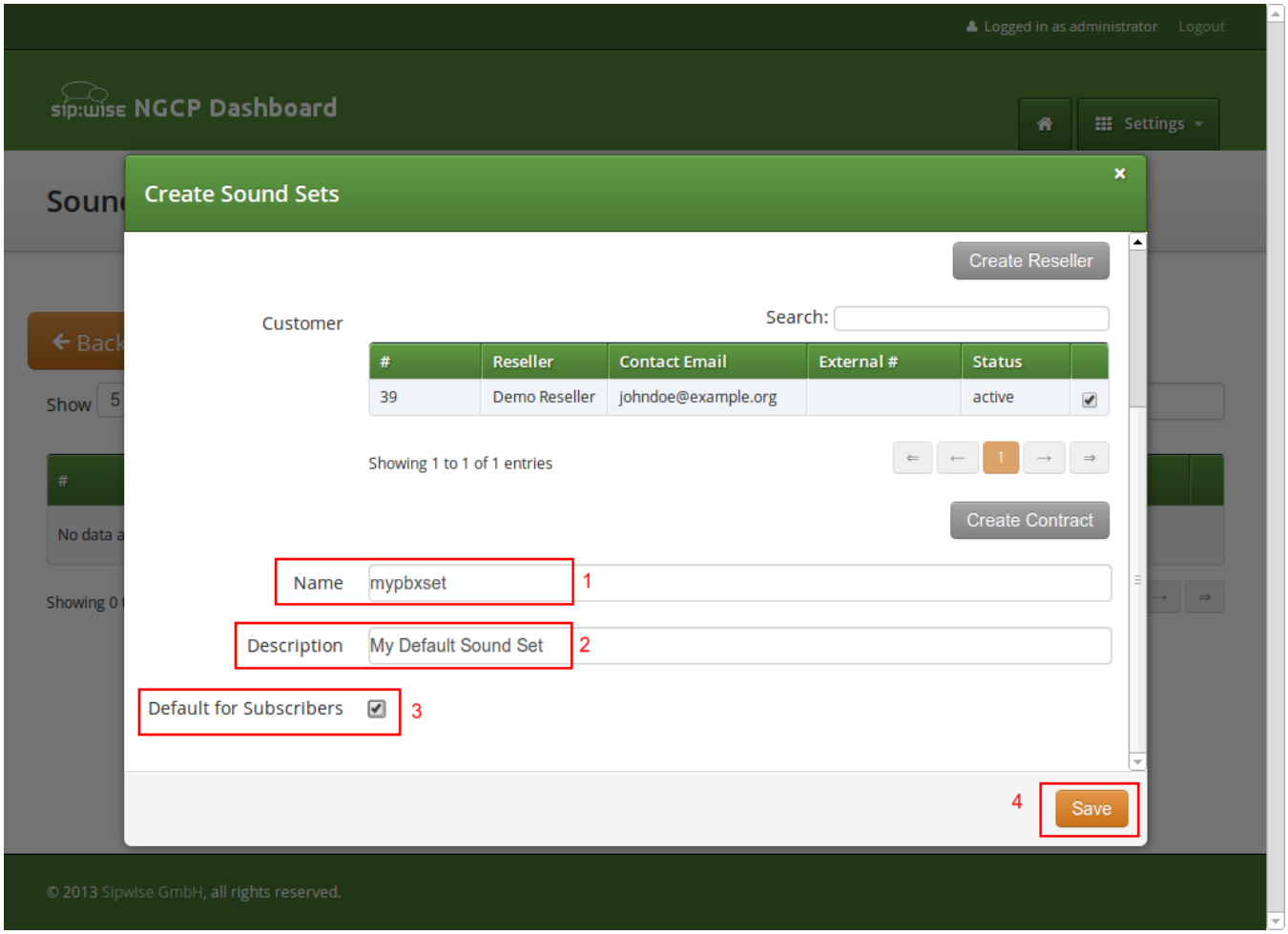


Figure 142. Create Customer Sound Set

To upload files to your Sound Set, click the *Files* button for the Sound Set.

Uploading a Music-on-Hold File

Open the *music_on_hold* row and click *Upload* on the *music_on_hold* entry. Choose a WAV file from your file system, and click the *Loopplay* setting if you want to play the file in a loop instead of only once. Click *Save* to upload the file.

Copyright © Sipwise GmbH - All rights reserved

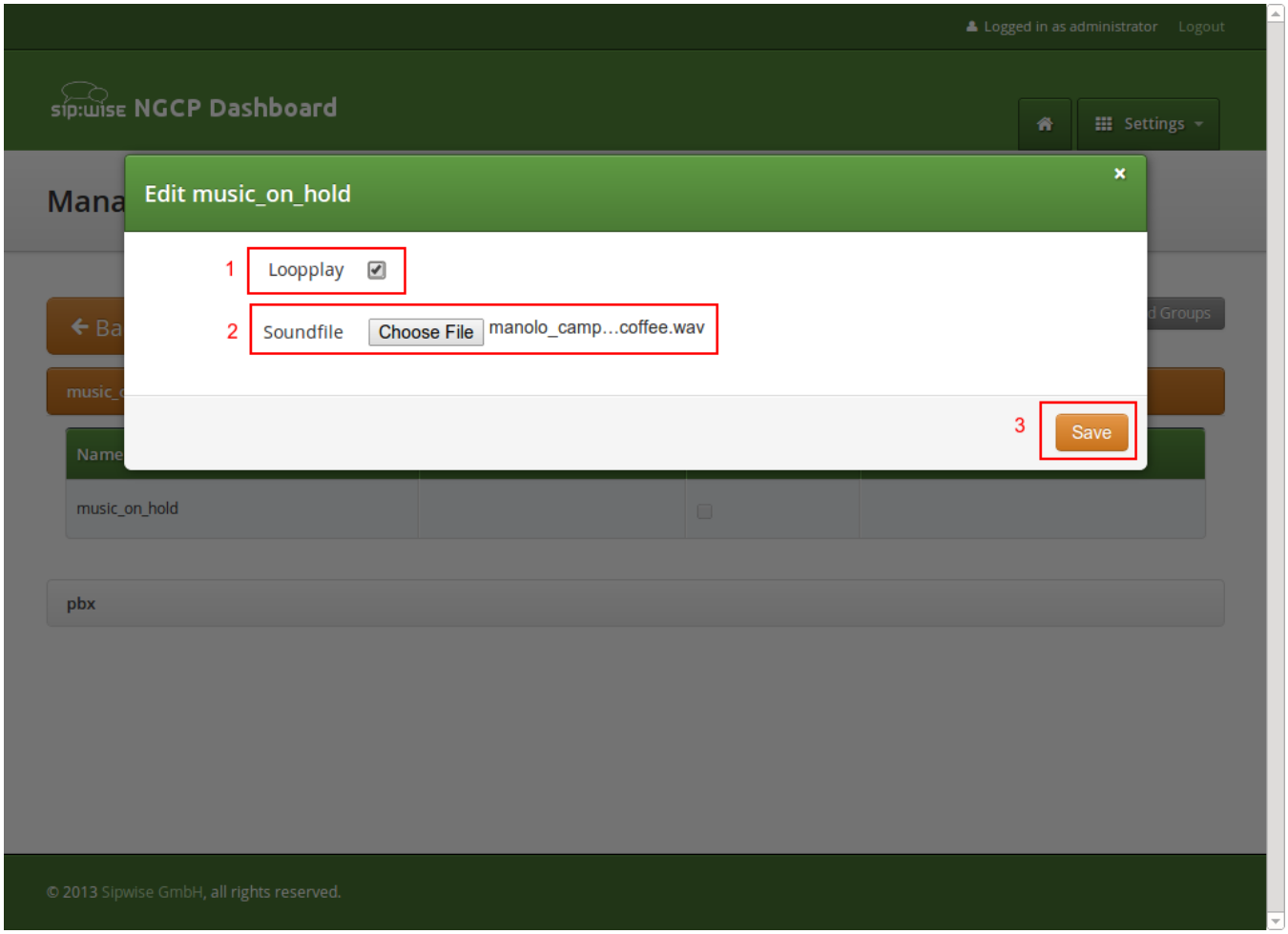


Figure 143. Upload MoH Sound File

8.1.8. Auto-Attendant Function

The *Auto-Attendant* is a built-in IVR feature that is available to Cloud PBX subscribers. It provides an automatic voice menu that enables the caller to select from a number of destinations, which could be other PBX subscribers or groups.

Another typical use case for the *Auto-Attendant* function is when the customer would like to have an "office assistant" that automatically takes incoming calls and routes them to the desired extension (i.e. to a subscriber).

The *Auto-Attendant* offers 2 ways of selecting the final call destination:

- *option selection*: selecting one of the pre-configured destinations by pressing a single digit (0-9)
- *extension dialing*: entering an arbitrary PBX extension number directly

Enabling the Auto-Attendant

The Auto-Attendant feature can be activated for any subscriber in the Customer PBX individually. There are three steps involved:

1. You have to prepare a *Sound Set* to have Auto-Attendant sound files.
2. You have to configure the destinations for the various options you provide (e.g. pressing 1 should go

Copyright © Sipwise GmbH - All rights reserved

to the marketing subscriber, 2 to development and 3 to some external number).

3. You have to set a Call Forward to the Auto-Attendant.

To do so, go to *Customer Details* and in the *Subscribers* section, click the *Preferences* button of the subscriber, where the Auto-Attendant should be set.

Preparing the Sound Set

Create a Sound Set and upload the Sound Files for it as described below. Afterwards in the *Subscriber Preferences* view, set the *Customer Sound Set* preference to the Sound Set to be used. To do so, click *Edit* on the *Customer Sound Set* preference and assign the set to be used.

Uploading Auto-Attendant Sound Files

When configuring a Call Forward to the *Auto-Attendant*, it will play the following files:

- aa_welcome: This is the welcome message (the greeting) which is played when someone calls the Auto-Attendant.
- each available pair of aa_X_for/aa_X_option: Each menu item in the Auto-Attendant consists of two parts. The for part, which plays something like *Press One for*, and the option part, which play something like *Marketing*. The Auto-Attendant only plays those menu options where both the for part and the option part is present, so if you only have 3 destinations you'd like to offer, and you want them to be on keys 1, 2 and 3, you have to upload files for aa_1_for, aa_1_option, aa_2_for, aa_2_option and aa_3_for and aa_3_option.

IMPORTANT The sound files only define the general structure of what is being played to the caller. The actual destinations behind your options are configured separately in Configuring the Auto-Attendant Slots.

An example configuration could look like this:

Copyright © Sipwise GmbH - All rights reserved

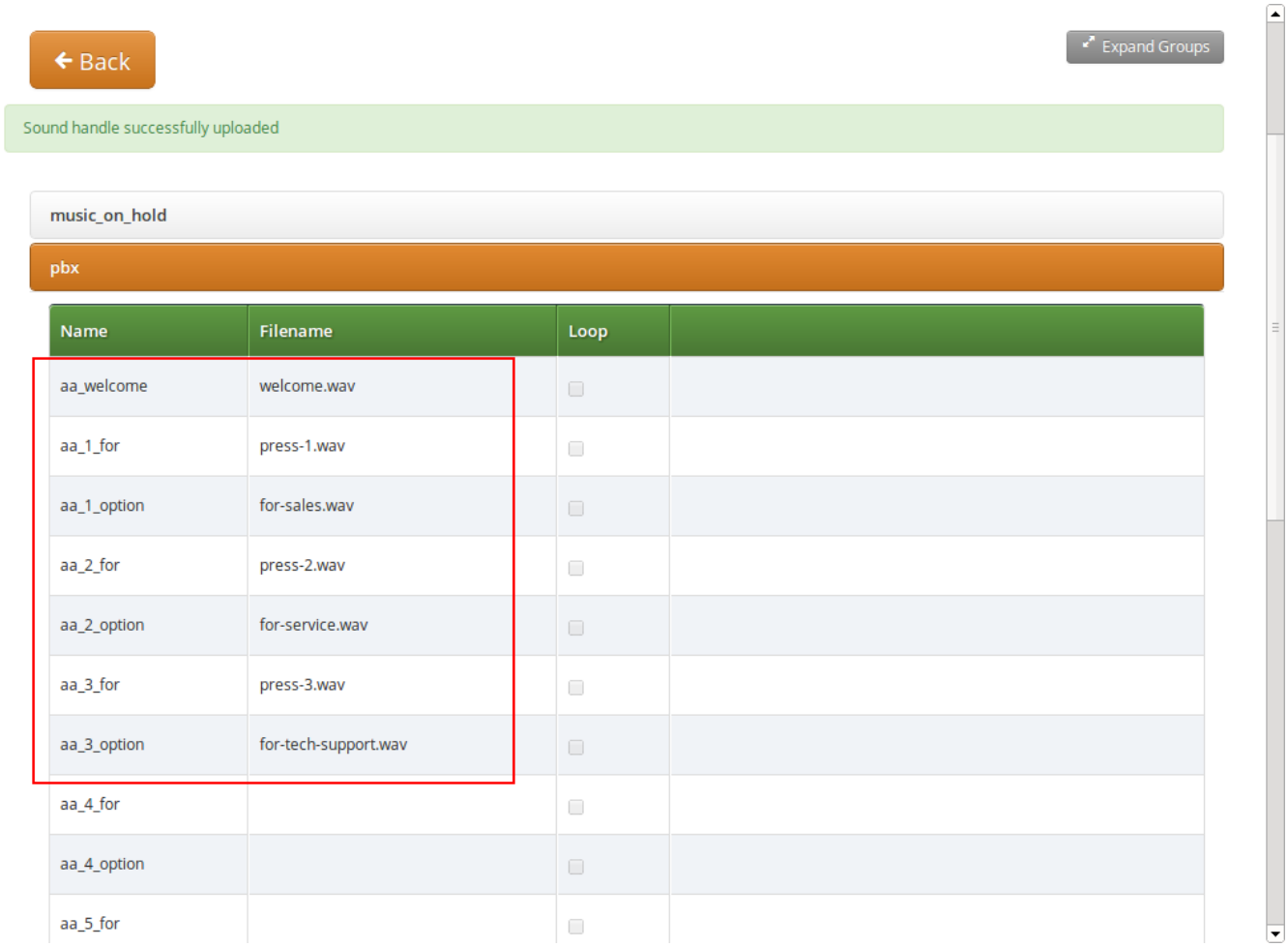


Figure 144. Upload Auto-Attendant Options Sound Files

In order to activate the **extension dialing** function within the Auto-Attendant, you have to upload the following prompt files:

- aa_star_for, aa_star_option: the announcement "Press star for connecting to an extension" (or similar message, depending on customer's needs)
- aa_enter_extension: will instruct the caller to enter the phone number of the extension he wants to connect to
- aa_invalid_extension: will be played when the phone number entered does not match any of the customer's extensions

Copyright © Sipwise GmbH - All rights reserved

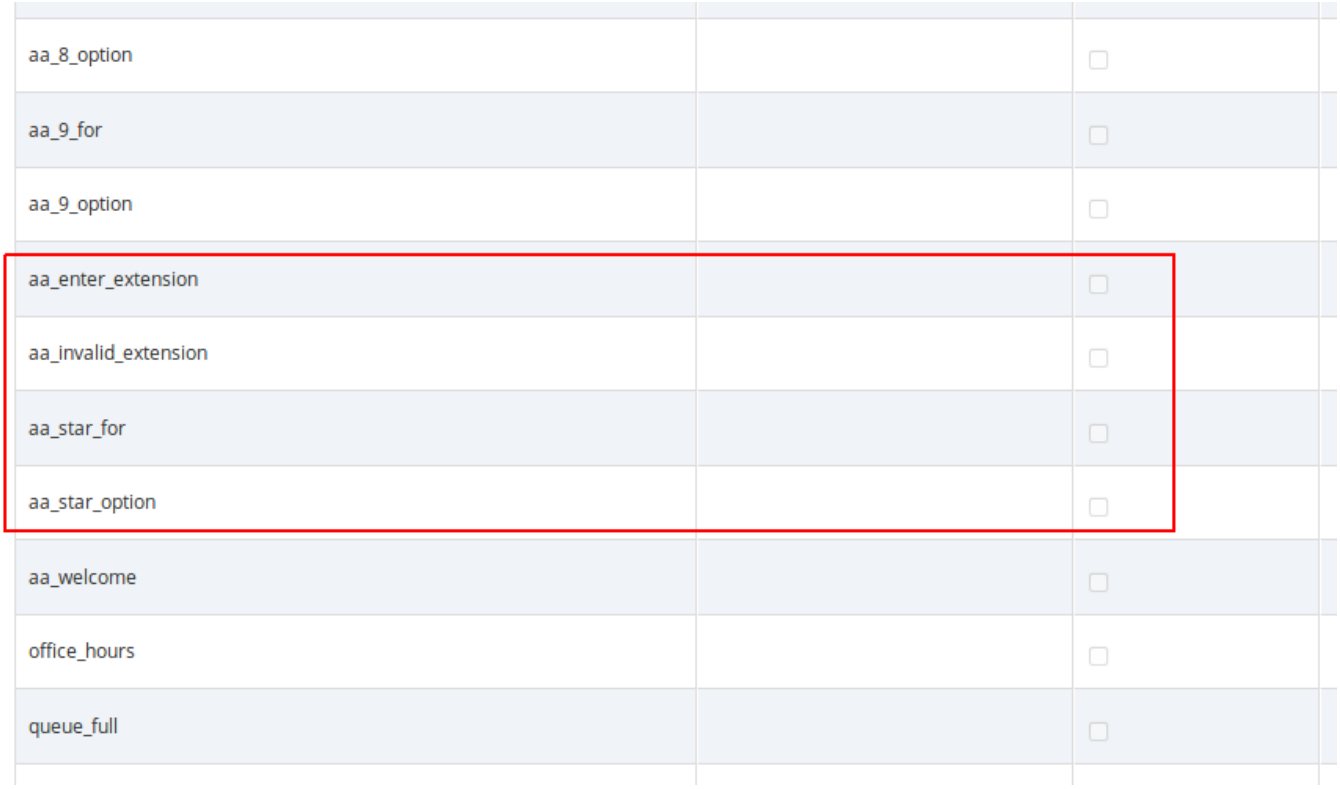


Figure 145. Upload Auto-Attendant Extension Dialing Sound Files

In order to enable the Auto-Attendant to play a prompt when the caller hasn't selected any menu item within the timeout period, you have to upload a file to the "aa_timeout" prompt in the "pbx" section of the Sound Set.

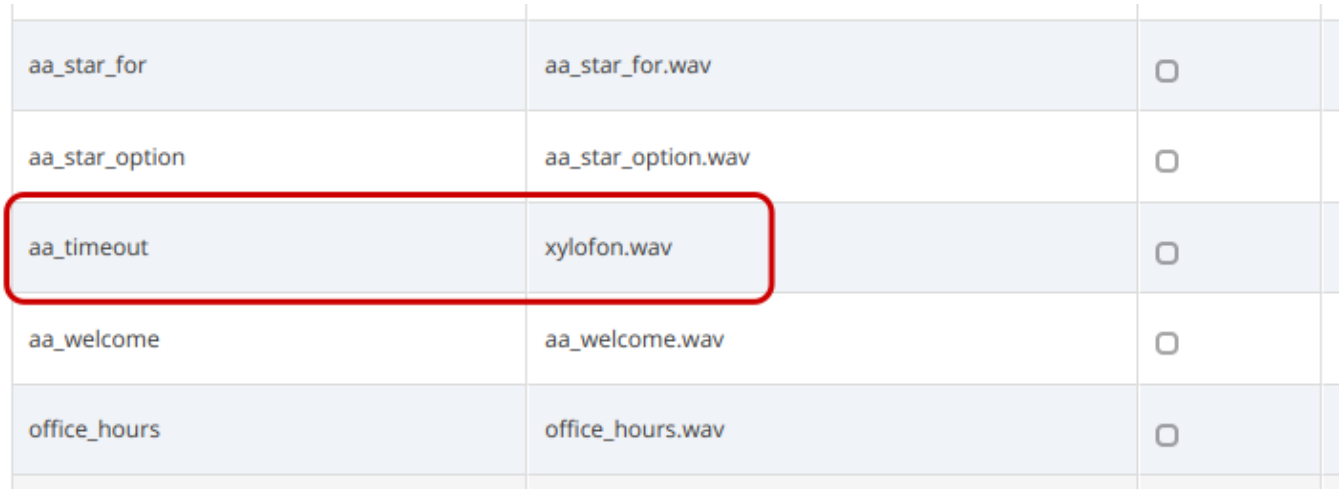


Figure 146. Upload Auto-Attendant Timeout Sound File

Auto-Attendant Flowchart with Voice Prompts

The illustration below shows the sequence of voice prompts played when Auto-Attendant feature is activated and a caller listens the IVR menu.

Copyright © Sipwise GmbH - All rights reserved

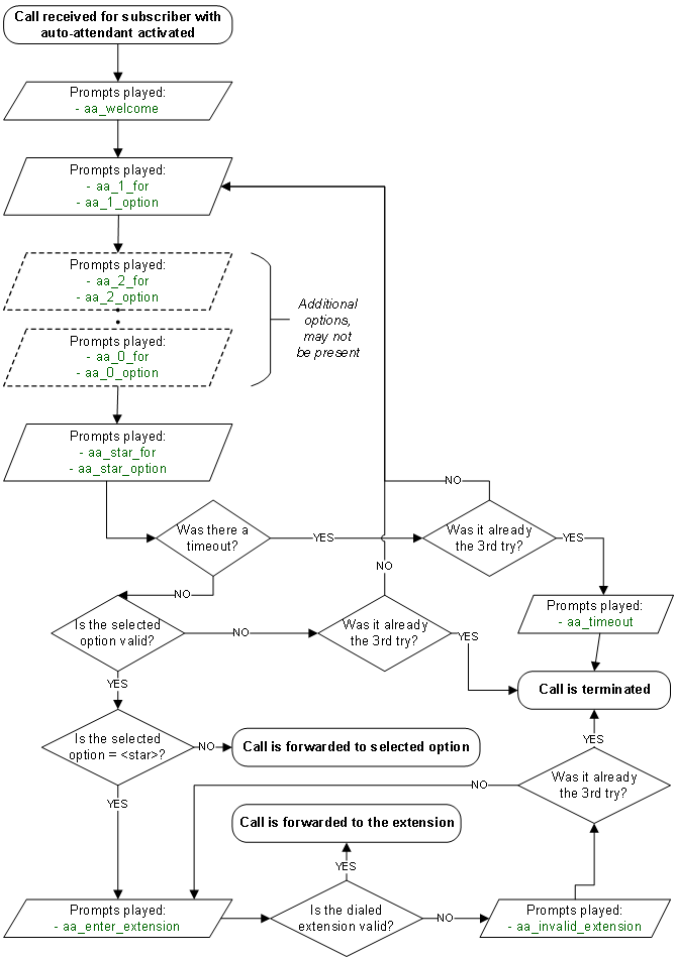


Figure 147. Flowchart of Auto-Attendant

Configuring the Auto-Attendant Slots

In the *Auto-Attendant Slots* section, click the *Edit Slots* button to configure the destination options. There are up to 10 available slots to configure, from keys *0* to *9*.

TIP Be aware that only configured slots will be prompted in the Auto-Attendant menu.

Click *Add another Slot* to add a destination option, select the Key the destination should be assigned to, and enter a Destination. The destination can be a subscriber username (e.g. marketing), a full SIP URI (e.g. sip:michelle.miller@customer1.pbx.example.org or any external SIP URI) or a number or extension (e.g. 491234567 or 101).

Repeat the step for every option you want to add, then press *Save*.

Copyright © Sipwise GmbH - All rights reserved

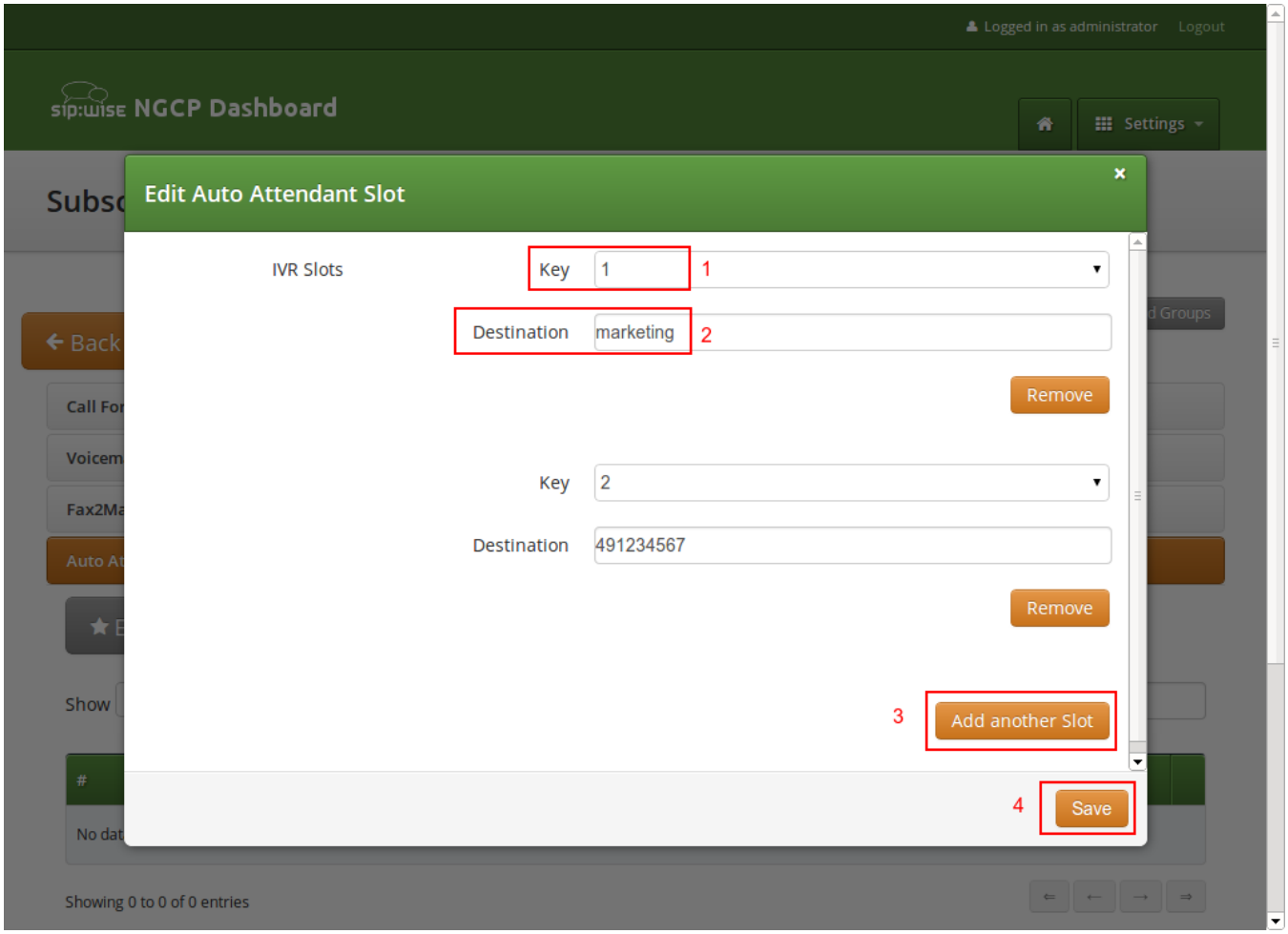


Figure 148. Define the Auto-Attendant Slots

TIP You have a possibility to define the 'default' slot for cases when a caller doesn't pick out anything. Other than that, you can additionally set a sound file for this action.

Activating the Auto-Attendant

Once the Sound Set and the Slots are configured, activate the Auto-Attendant by setting a Call Forward to Auto-Attendant.

To do so, open the *Call Forwards* section in the *Subscriber Preferences* view and press *Edit* on the Call Forward type (e.g. *Call Forward Unconditional* if you want to redirect callers unconditionally to the Auto-Attendant).

Select *Auto-Attendant* and click *Save* to activate the Auto-Attendant.

Copyright © Sipwise GmbH - All rights reserved

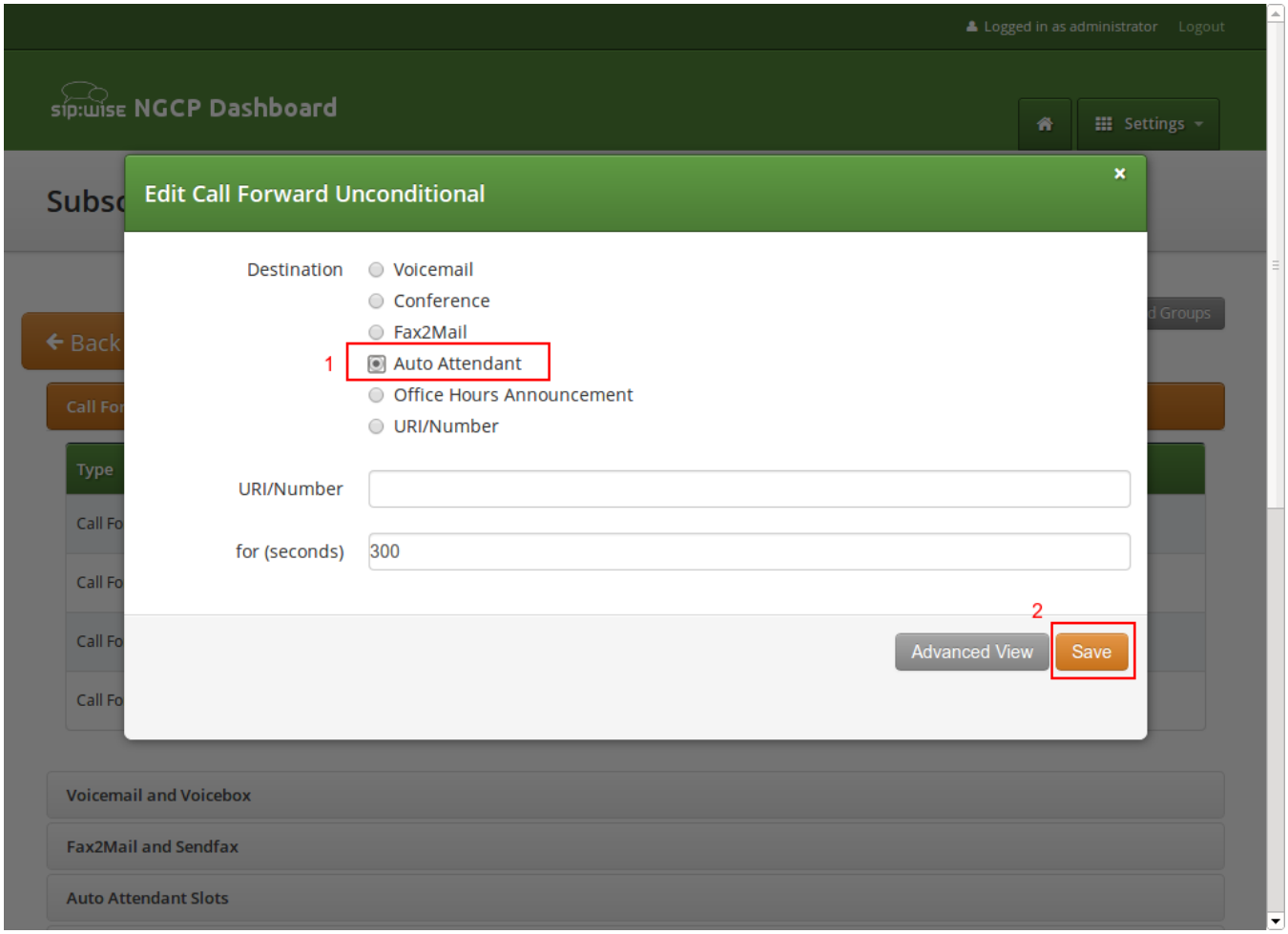


Figure 149. Set a Call Forward to Auto-Attendant

TIP As with any other Call Forward, you can define more complex forwarding rules in the *Advanced View* to only forward the call to the Auto-Attendant during specific time periods, or as a fallback if no one picks up the office number.

Handling input errors in the Auto-Attendant

If the caller enters an invalid menu item (0-9), i.e. a choice that hasn't got a valid destination or is not configured at all, the Auto-Attendant application will repeat the available choices again. The same happens when the caller enters an invalid extension number after selecting "star". After trying to select the wrong destination 3 times, the Auto-Attendant will terminate the call.

If the caller doesn't select any menu item for a pre-configured timeout, the Auto-Attendant will repeat the menu items again. When the caller doesn't select any menu item for 3 times, the Auto-Attendant will do the following:

- *terminate the call immediately* if the "aa_timeout" prompt is not defined in the Sound Set
- *play the "aa_timeout" prompt* before terminating the call, if that prompt is defined in the Sound Set, and then terminate the call

 Copyright © Sipwise GmbH - All rights reserved

8.1.9. Cloud PBX Groups with Busy Members

A *huntgroup* (HG) or a *PBX Group* is a Cloud PBX feature that distributes the calls between members of the group according to the configured hunt policy and timeout. The PBX group belongs to a customer and one Cloud PBX subscriber can be a member of one or more of the huntgroups of the customer. *Call Waiting* is a CPE (phone) feature that allows you to take another call while you're already on the phone.

Multiple incoming calls to the huntgroup may result in multiple calls delivered to the same subscriber if the Call Waiting feature is enabled on his phone, regardless whether the huntgroup members are busy at this time. Hence, busy subscribers may get a second incoming call. It may be an expected behavior (since one subscriber may have multiple devices and/or clients that all ring in parallel) or not, depending on the setup.

Therefore, Sipwise C5 Cloud PBX module offers *busy huntgroup member mode* feature to check the busy status of individual huntgroup members before routing a call to them. This will leave subscribers on active phone calls undisturbed by calls to huntgroup.

IMPORTANT

From 10.4, *busy hunt group member mode* is the new name for what used to be previously named *Ignore Busy Hunt Group Members*. Not only the name is changed, but the configuration is also moved from the main configuration file: /etc/ngcp-config/config.yml to the domain preferences or to the subscriber preferences as described below. This evolution offers more flexibility in the configuration allowing to change the behavior on subscriber basis instead of system wide. The 10.4 upgrade procedure automatically migrates the config.yml setting to all your domains preferences.

The configuration of the *busy huntgroup member mode* feature is done via the `busy_hg_member_mode` parameter available at domain or subscriber preferences:

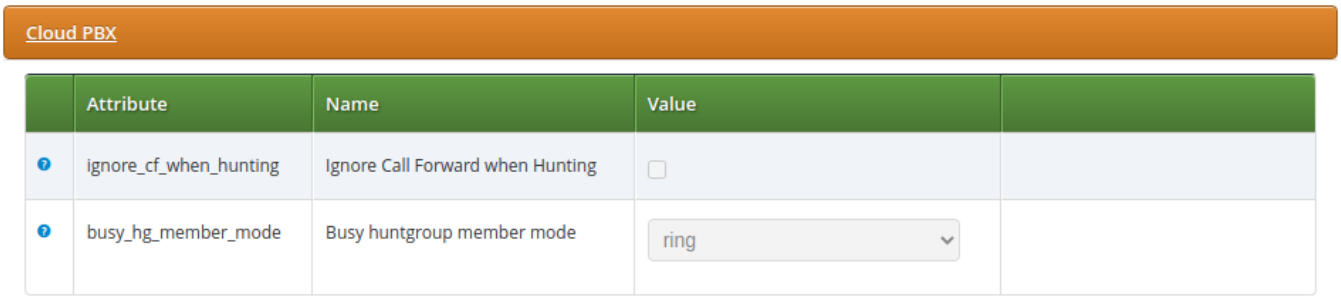


Figure 150. Busy huntgroup member mode configuration for domain

Copyright © Sipwise GmbH - All rights reserved

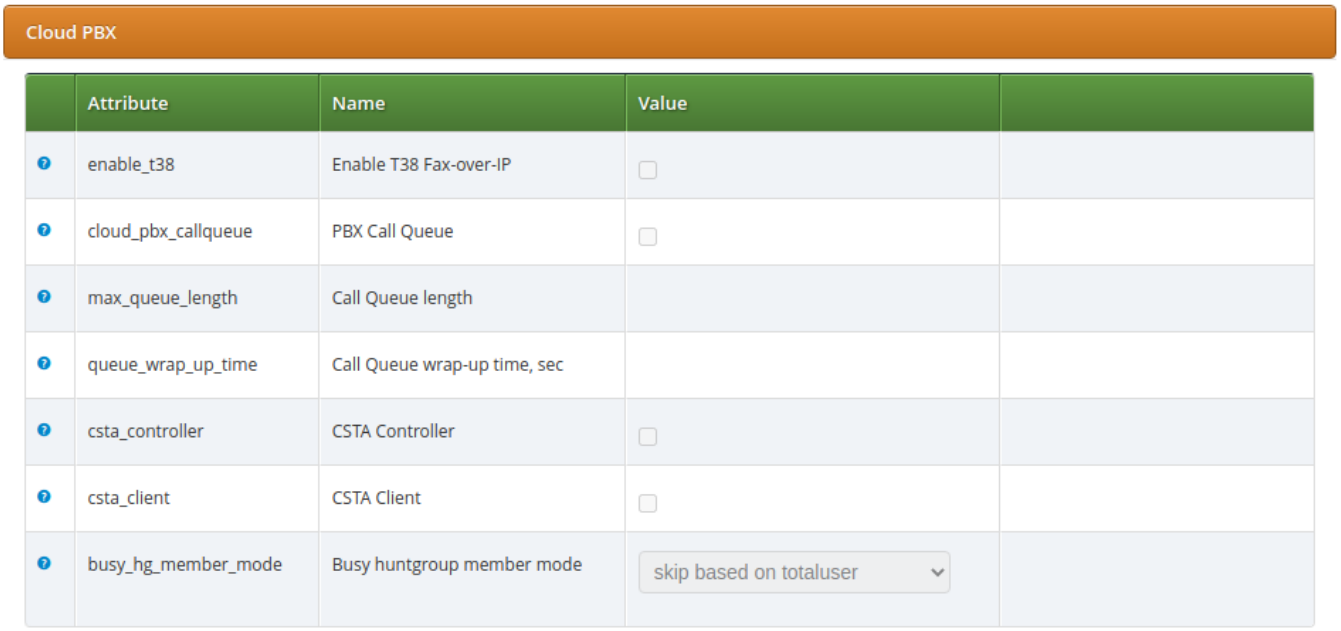


Figure 151. Busy huntgroup member mode configuration for subscriber

The `busy_hg_member_mode` preference may take the following values:

- `ring` (default value): A hunt group member is alerted of all incoming calls to his groups, regardless of his busy state.
- `skip_totaluser`: A hunt group member is not alerted of incoming calls to his groups because he is busy for the following reasons (call distribution continues with other available huntgroup members):

  he is involved in one or more incoming or outgoing calls in alerting or active phase

  an incoming call intended for him has been forwarded to another destination an this call is in alerting or active phase.
- `skip_activeuser`: A hunt group member is not alerted of incoming calls to his groups because he is involved in one or more incoming or outgoing calls in alerting or active phase but NOT busy for the calls that have been forwarded (call distribution continues with other available huntgroup members).
- `use domain default` (subscriber prefs. only): use the setting of the domain preference `busy_hg_member_mode`.

IMPORTANT

This feature does not present an extended server-side Call Waiting functionality. It concerns only the huntgroups' behavior. Hence subscriber would still be able to receive multiple calls when called directly (not via huntgroup) with Call Waiting enabled on his phone.

As an example, if the `busy_hg_member_mode` preference is set to `skip_totaluser`, a huntgroup call will not be distributed to a subscriber in the following situations:

Use Case 1

Subscriber receives an incoming call. A second call is made to the HG. The subscriber should NOT receive this call via HG extension.

 Copyright © Sipwise GmbH - All rights reserved

Use Case 2

Subscriber makes an outgoing call. A second call is made to the HG. The subscriber should NOT receive this call via HG extension.

Use Case 3

Subscriber with a call forward enable (CFU, CFB, CFNA, CFT) receives a call to his extension (not extension of HG) which is then forwarded. A second call is made to the HG. The subscriber should NOT receive the call via HG extension.

In order to prevent the forwarded calls from keeping the subscriber as "busy" for the purpose of this feature the platform administrator should set the domain or the subscriber preference `busy_hg_member_mode` to the value `skip_activeuser`:

While User Cases 1 an 2 will behave in the same way as described above, the change of behavior happens in Use Case 3:

Use Case 3

Subscriber with a call forward enabled (CFU, CFB, CFNA, CFT) receives a call to his extension (not extension of HG) which is then forwarded. A second call is made to the HG. The subscriber should receive the call as normal.

8.1.10. Configuring Call Queues

The Sipwise C5 platform offers call queueing feature for Cloud PBX subscribers. For any subscriber within the PBX Sipwise C5 system administrator or the subscriber himsef may activate the *Call Queue*. This is done individually for each subscriber on demand.

If call queue activation has been done and the subscriber receives more than 1 call at a time, then the second and all further callers will be queued until the subscriber finishes his call with the first caller and gets free.

Activating the Call Queue

The call queue configuration is available at the path: *Subscribers select one Details Preferences Cloud PBX*.

Following configuration parameters may be set for call queueing:

- cloud_pbx_callqueue : shows the status of call queueing (enabled / disabled); by default it is disabled
- max_queue_length : the length of call queue, i.e. the maximum number of callers in a queue; the default is 5
- queue_wrap_up_time : the delay in seconds between the ending of the previous call and the connection of the next queued caller with the subscriber; the default is 10

In order to change the actual setting, press the *Edit* button in the relevant row.

Copyright © Sipwise GmbH - All rights reserved

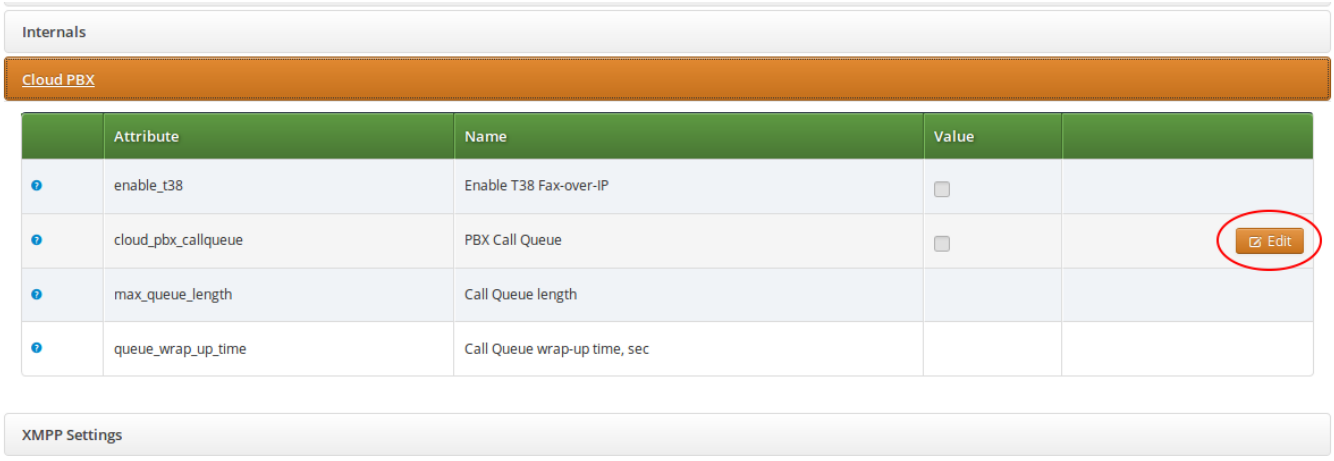


Figure 152. Call Queue Configuration

Call Queue Voice Prompts

Queued callers first hear a greeting message then information about their position in the queue and finally a waiting music / signal.

Table 36. Call Queue Voice Prompts

Prompt handle	Prompt content
`queue_greeting`	All lines are busy at the moment, you are being queued.
`queue_prefix`	You are currently number…
`queue_suffix`	… in the queue, please hold the line.
`queue_full`	All lines are busy at the moment, please try again later.
`queue_waiting_music`	<waiting music>

Call Queue Flowchart with Voice Prompts

The following illustration shows which voice prompts are played to the caller when the call gets into a queue.

Copyright © Sipwise GmbH - All rights reserved

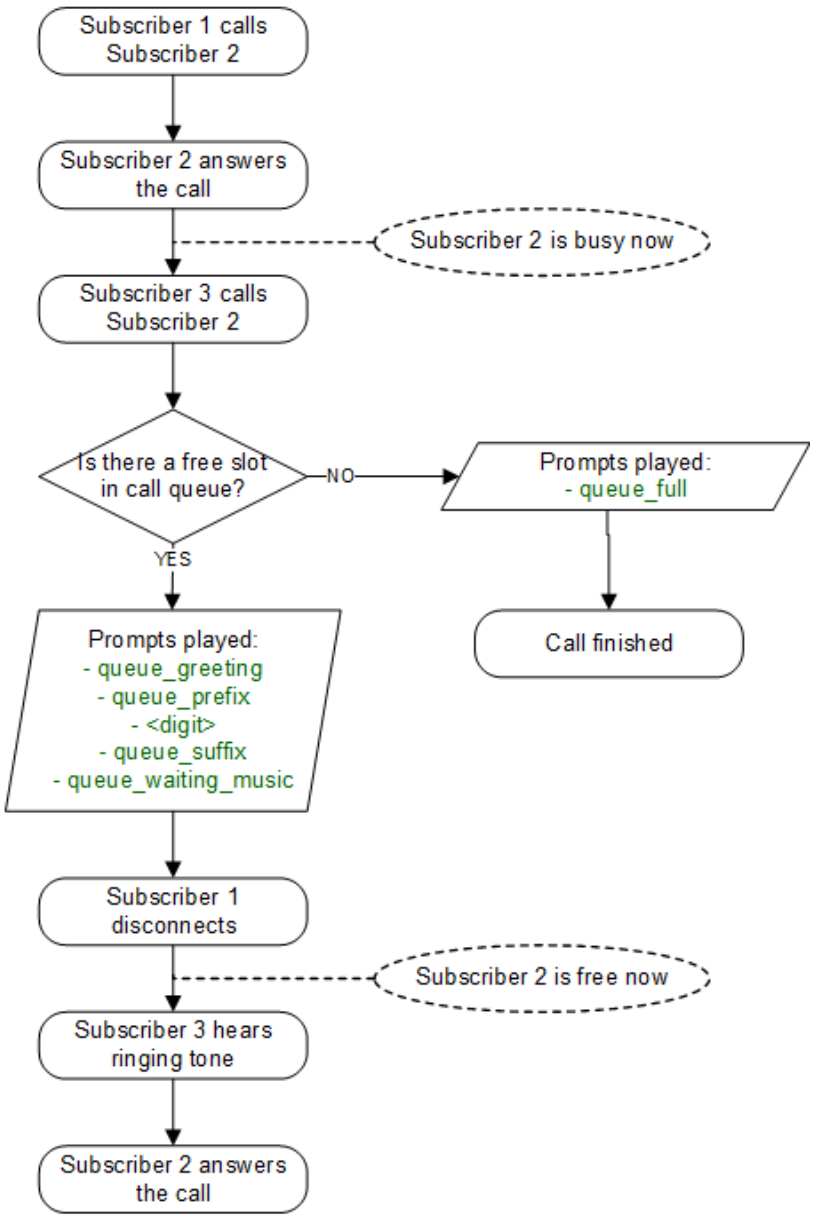


Figure 153. Flowchart of Call Queue

8.1.11. Device Auto-Provisioning Security

Server Certificate Authentication

The Cisco SPA phones can connect to the provisioning interface of the PBX via HTTP and HTTPS. When perform secure provisioning over HTTPS, the phones validate the server certificate to check if its a legitimate Cisco provisioning server. To pass this check, the provisioning interface must provide a certificate signed by Cisco for that exact purpose.

The following steps describe how to obtain such a certificate.

First, a new SSL key needs to be generated:

Copyright © Sipwise GmbH - All rights reserved

```
$ openssl genrsa -out provisioning.key 2048
Generating RSA private key, 2048 bit long modulus
...+++
.....................................................+++
e is 65537 (0x10001)
```

Next, a certificate signing request needs to be generated as follows. Provide your company details.

IMPORTANT The **Common Name (e.g. server FQDN or YOUR name)** field is crucial here. Provide an FQDN which the phones will later use via DNS to connect to the provisioning interface, for example *pbx.example.org*. Cisco does **NOT** support wild-card certificates.

IMPORTANT Leave the password empty when asked for it (press Enter without entering anything).

```
$ openssl req -new -key provisioning.key -out provisioning.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a
DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

Country Name (2 letter code) [AU]:AT
State or Province Name (full name) [Some-State]:Vienna
Locality Name (eg, city) []:Vienna
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Sipwise GmbH
Organizational Unit Name (eg, section) []:Operations
Common Name (e.g. server FQDN or YOUR name) []:pbx.example.org
Email Address []:office@sipwise.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Finally, compress the provisioning.csr file via ZIP and send it to our Cisco sales representative. If in doubt, you can try to send it directly to ciscosb-certadmin@cisco.com asking them to sign it.

IMPORTANT Only send the CSR file. **Do NOT send the key file, as this is your private key!**

IMPORTANT Ask for both the signed certificate AND a so-called *combinedca.crt* which is needed to perform client authentication via SSL. Otherwise you can not restrict access to Cisco SPAs only.

Copyright © Sipwise GmbH - All rights reserved

You will receive a signed CRT file, which Sipwise can use to configure the PBX provisioning interface.

Client Certificate Authentication

If a client connects via HTTPS, the server also checks for the client certificate in order to validate that the device requesting the configuration is indeed a legitimate Cisco phone, and not a fraudulent user with a browser trying to fetch user credentials.

Cisco Client Root Certificate can be obtained from Download Client Certificates website.

8.1.12. Device Bootstrap and Resync Workflows

The IP phones supported by the PBX need to initially be configured to fetch their configuration from the system. Since the phones have no initial information about the system and its provisioning URL, they need to be boot-strapped. Furthermore, changes for a specific device might have to be pushed to the device immediately instead of waiting for it to re-fetch the configuration automatically.

The following sections describe the work-flows how this is accomplished without having the customer directly accessing the phone.

ALE Device Bootstrap

Initial Bootstrapping

ALE provides a zero-touch provisioning mechanism in their firmwares, which causes the factory-reset phones to connect to a ALE web service at *http://eds.al.enterprise.com/* to check if a custom provisioning URL is configured for the MAC address of the phone. If an association between the MAC and a provisioning URL is found, the web service redirects the phone to the provisioning URL, where the phone connects to in order to obtain the configuration file.

Copyright © Sipwise GmbH - All rights reserved

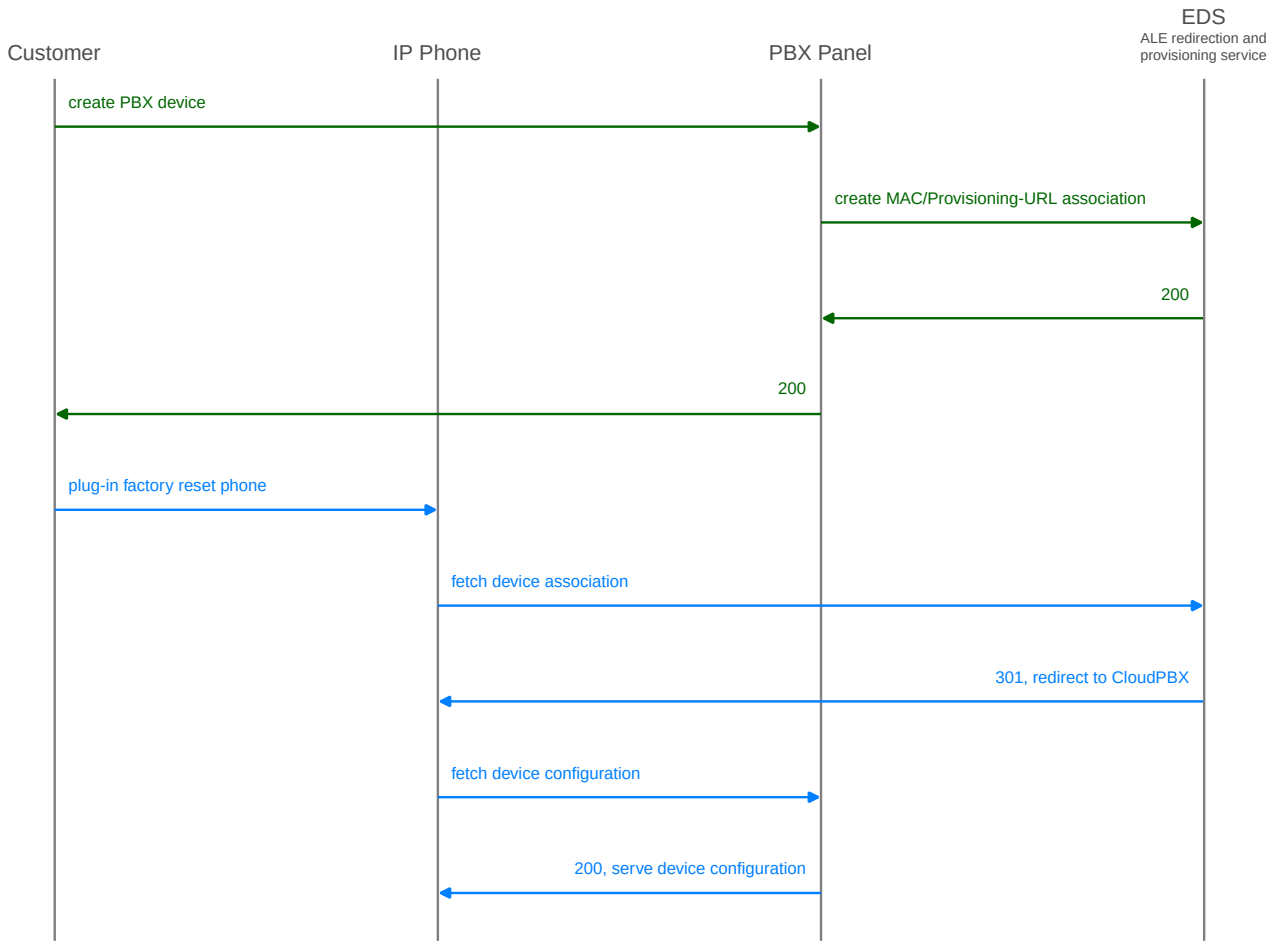


Figure 154. Initially bootstrap an ALE phone

The CloudPBX module ensures that when an end customer creates an ALE device, the MAC address is automatically provisioned on the ALE web service via an API call, so the customer's phone can use the correct provisioning URL to connect to the auto-provisioning server of the CloudPBX.

As a result, no customer interaction is required to bootstrap ALE phones, other than just creating the phone with the proper MAC on the CloudPBX web interface.

Updating ALE Firmware

For Myriad Devices:

NGCP provides version 2.13.01.000.1899 by default, but also offers versions 2.12.04.1729 and 2.11.01.1604. If another firmware version is needed, these can be downloaded at *https://www.aledevice.com/site/download* and manually uploaded to NGCP

IMPORTANT	If firmware older than 2.11.01.1604 is used, direct upgrades to the latest firmware can no longer be guaranteed. To prevent this, upgrade phones manually to version 2.11.01.1604 or newer

IMPORTANT	If firmware older than 2.13.01.000.1899 is used, position of accounts set in NGCP will no longer align with the phone display. To prevent this, upgrade phones to version 2.13.01.000.1899 or newer

Copyright © Sipwise GmbH - All rights reserved

For Halo Devices: NGCP provides version 2.12.05.000.1194 for devices H3G and H6 and version 2.10.00.0001078 for device H2P. If another firmware version is needed, these can be downloaded at *https://www.aledevice.com/site/download* and manually uploaded to NGCP

For Cloud Edition series devices: NGCP provides version 1.53.37.4117 for all Cloud Edition devices. If another firmware version is needed, these can be downloaded at *https://www.aledevice.com/site/download* and manually uploaded to NGCP

Factory Reset

For already provisioned phones, the end customer might need to perform a factory reset.

On Myriad, H3G and H6 phones, holding down the conference button for 10 seconds will give you the option to reset the device.

On H2P phone, holding down the "OK" button for 8 seconds will give you the option to reset the device

On Cloud Edition phones, following steps need to be done.

- Start up the phone
- When the boot up screen shows up, during step 1 press keys 1, 3, 7, and 9 at the same time
- Input **46*46*253** and the phone should reset

The default keypad password is *0000*. The WEB username for factory-reset phones is *admin* with password *123456*.

See more details here: ALE FAQ.

Bootstrap URL

The default ALE bootstrap URI is: http://<ngcp.fqdn>:1445/device/autoprov/bootstrap/{mac} You can find the URI above on EDS WEB interface for newly provisionied ALE devices. The variable "{mac}" will expand to the device MAC address automatically on ALE device before connecting to NGCP. You can change the default "Bootstrap URI" on page "Edit Device Model". See more details in PBX bootstrap, firmwares and security.

NOTE	it is highly recommended to use a secure bootstrap config: https://<ngcp.fqdn>:1444/device/autoprov/config/{mac}

Cisco SPA Device Bootstrap

Initial Bootstrapping

Copyright © Sipwise GmbH - All rights reserved

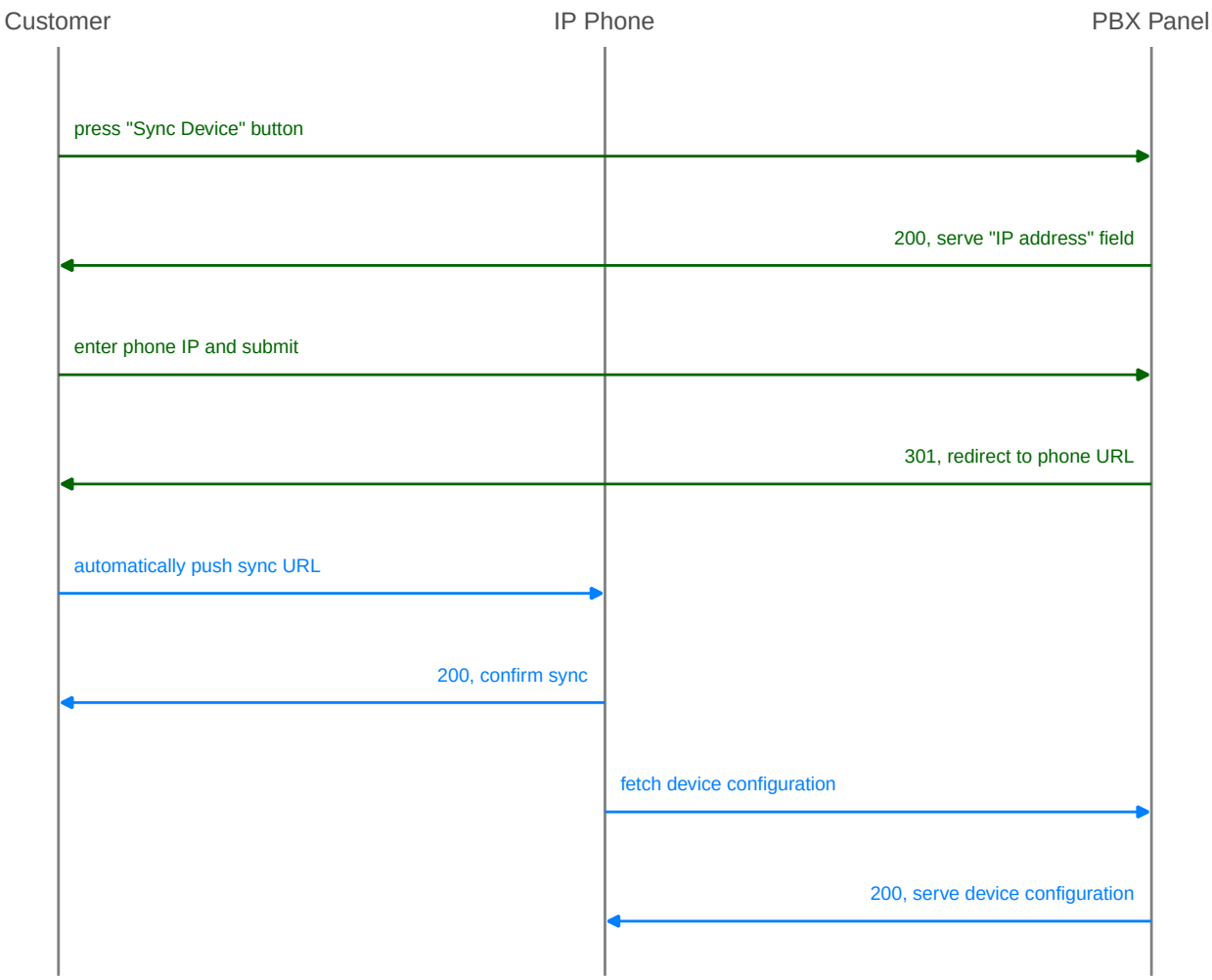


Figure 155. Initially bootstrap a PBX device

Subsequent Device Resyncs

If one of the subscribers configured on a PBX device is registered via SIP, the system can trigger a re-sync of the phone directly via SIP without having the customer enter the IP of the phone again. This is accomplished by sending a special `NOTIFY` message to the subscriber:

```
NOTIFY sip:subscriber@domain SIP/2.0
To: <sip:subscriber@domain>
From: <sip:subscriber@domain>;tag=some-random-tag
Call-ID: some-random-call-id
CSeq: 1 NOTIFY
Subscription-State: active
Event: check-sync
Content-Length: 0
```

In order to prevent unauthorized re-syncs, the IP phone challenges the request with its own SIP credentials, so the NOTIFY is sent twice, once without authentication, and the second time with the subscriber's own SIP credentials.

 Copyright © Sipwise GmbH - All rights reserved

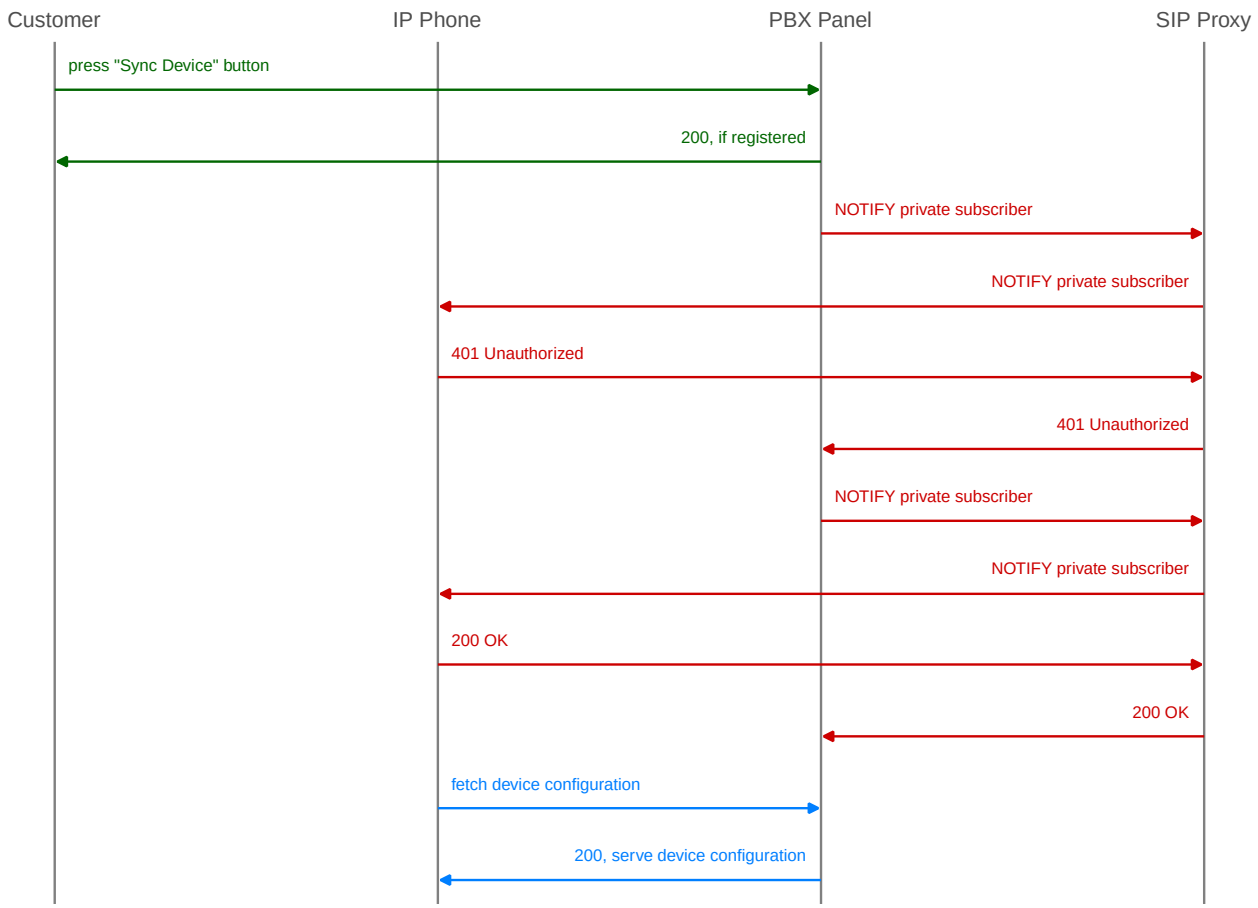


Figure 156. Resync a registered PBX device

Panasonic Device Bootstrap

Initial Bootstrapping

Panasonic provides a zero-touch provisioning mechanism in their firmwares, which causes the factory-reset phones to connect to a Panasonic web service at *https://provisioning.e-connecting.net* to check if a custom provisioning URL is configured for the MAC address of the phone. If an association between the MAC and a provisioning URL is found, the web service redirects the phone to the provisioning URL, where the phone connects to in order to obtain the configuration file.

Copyright © Sipwise GmbH - All rights reserved

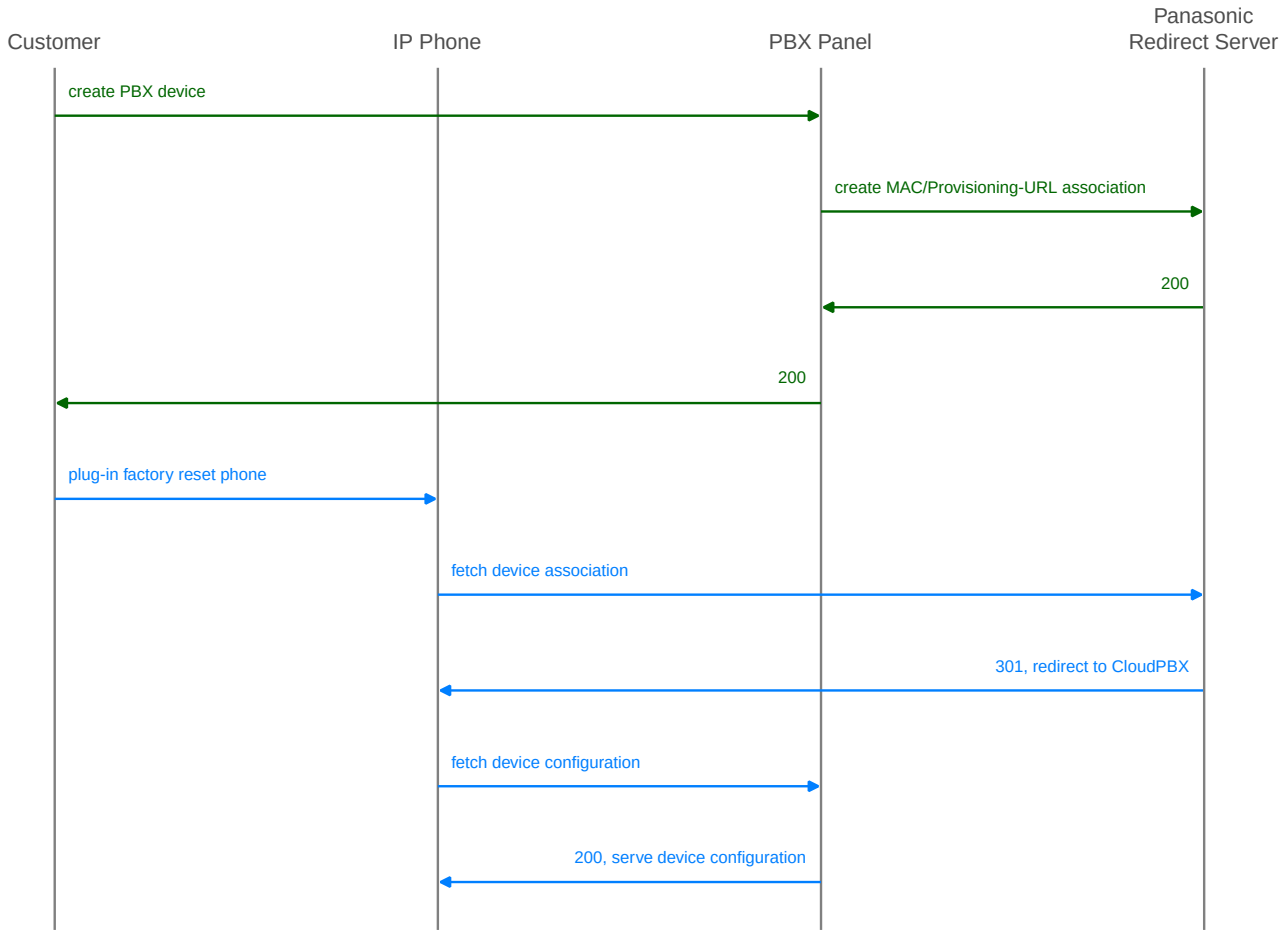


Figure 157. Initially bootstrap a Panasonic phone

The CloudPBX module ensures that when an end customer creates a Panasonic device, the MAC address is automatically provisioned on the Panasonic web service via an API call, so the customer's phone can use the correct provisioning URL to connect to the auto-provisioning server of the CloudPBX.

As a result, no customer interaction is required to bootstrap Panasonic phones, other than just creating the phone with the proper MAC on the CloudPBX web interface.

Factory Reset

For already provisioned phones, the end customer might need to perform a factory reset:

- Press *Settings* or *Setup*
- Enter *#136*
- Select *Factory Setting* and press *Enter*
- Select *Yes* and press *Enter*
- Select *Yes* and press *Enter*

The default username for factory-reset phones is *admin* with password *adminpass*.

Subsequent Device Resyncs

 Copyright © Sipwise GmbH - All rights reserved

The same procedure as with Cisco SPA phones applies, once a subscriber configured on the phone is registered.

SNOM Device Bootstrap

Initial Bootstrapping

SNOM provides a zero-touch provisioning mechanism in their firmwares, which causes the factory-reset phones to connect to a SNOM web service at *https://sraps.snom.com/* to check if a custom provisioning URL is configured for the MAC address of the phone. If an association between the MAC and a provisioning URL is found, the web service redirects the phone to the provisioning URL, where the phone connects to in order to obtain the configuration file.

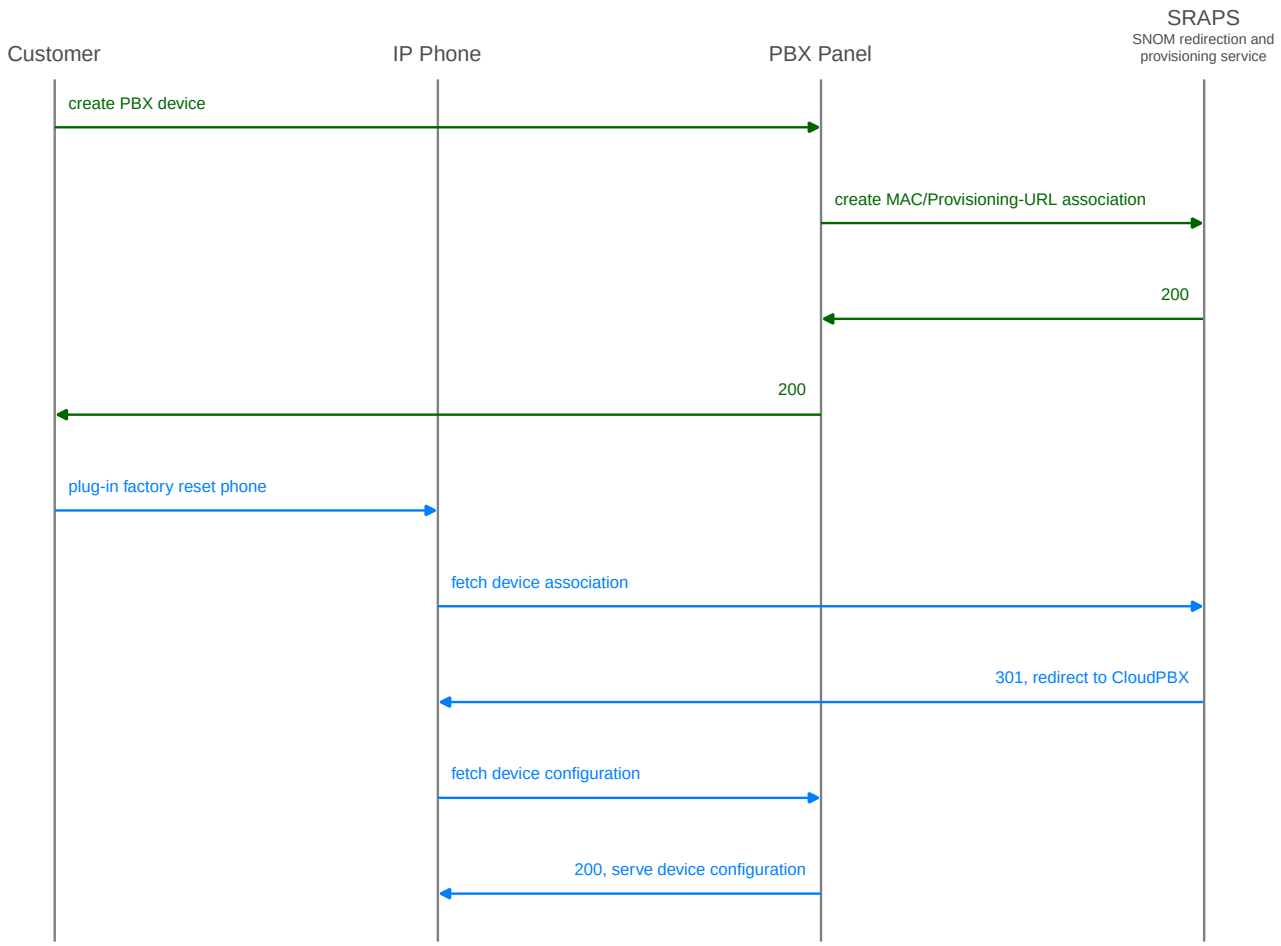


Figure 158. Initially bootstrap a SNOM phone

The CloudPBX module ensures that when an end customer creates a SNOM device, the MAC address is automatically provisioned on the SNOM web service via an API call, so the customer's phone can use the correct provisioning URL to connect to the auto-provisioning server of the CloudPBX.

As a result, no customer interaction is required to bootstrap SNOM phones, other than just creating the phone with the proper MAC on the CloudPBX web interface.

Linking SNOM Handsets with SNOM M900 Base Station

To add a handset to a base station and to assign a subscriber to the handset, open preferences of a

Copyright © Sipwise GmbH - All rights reserved

subscriber and add the IPUI of the handset into the settings field 'gpp0' (the gpp number can be changed in device config if necessary).

IMPORTANT IPUIs have to start with 0x (e.g. if IPUI is 123456789, set gpp0 equal to 0x123456789)

IMPORTANT If a new IPUI is added, a base station config reload is necessary. It happens periodically on automated bases or base station reboot.

Updating SNOM Firmware

For D-Series Phones: It is recommended to use the SNOM Update Center URL (usually *http://downloads.snom.com*) to update the firmware of phones and base stations. SNOM firmware files are signed and the phones check the signatures of the files during provisioning. This makes sure only a proper SNOM firmware can be installed on the phones via HTTP. HTTPS transport might be unreliable on newly started devices due to wrong date/time. The firmware can be downloaded from NGCP as well. In the Device Configuration XML config, there is a commented line pointing to NGCP.

<firmware>[% config.bootstrap ? firmware.booturl : firmware.baseurl %]/[% config.mac #%]/from/0/latest/</firmware>

For M-Series Base Stations: M-Series base stations require the user to define the *version* and *branch* of the firmware to update to. By default, this version is set to V510B01. If a different firmware version is needed, it has to be manually changed in the Device Configuration XML config.

<fp_fwu_sw_version>510</fp_fwu_sw_version>–

<fp_fwu_branch_version>1</fp_fwu_branch_version>

IMPORTANT Base stations with firmware below version V500B1 can't directly upgrade to the newest firmware. Please double check vendor upgrade recommendations and limitations on SNOM website.

For Handsets to use with M-Series Base Stations: Handsets require you to add the version, branch and type of device you want to update separately like this:

<pp_fwu_sw_version type="M65">510</pp_fwu_sw_version>–

<pp_fwu_branch_version type="M65">1</pp_fwu_branch_version>

IMPORTANT If a new handset is added to a base station, a reboot is required to apply the update settings

IMPORTANT Handsets require the included charger to complete the firmware upgrade.

Additionally, handsets also require you to add the IPUI of the handset you want to update. This step can be done via NGCP. Go to the subscriber settings, search for setting *gpp*, use an unused gpp preference and add the IPUI in here. It is possible to use any general purpose field from subscriber preferences (gpp1, gpp2, ...) until it matches PBX Device Configuration for XML attribute "<subscr_dect_ipui>".

 Copyright © Sipwise GmbH - All rights reserved

IMPORTANT IPUIs have to start with 0x, for example 0x123456789

Acquiring the IP of SNOM Base Stations

To get the IP address of the base stations close to you, use a SNOM Handset, go to the applications menu and type *47*. All base stations should show up in a list view with their IP Address.

Adding Vertical Service Code Wrappers into D-Series Templates

The Vertical Service Code (VSC) Interface offers a list of codes which can be masked by using specific "wrappers" tags within SNOM D-Series devices. Some examples for Reminder, Call Forward Unconditional and Speed Dial features are provided as follows:

```
<functionKeys e="2">
<fkey idx="3" context="1" lp="on" default_text="Reminder"
icon_type="kIconTypeFkeyClock" perm="">url
http://192.168.91.254/sipwise/vcs_mb.xml#sub=*[@id="vcs_55"]&amp;var:sc_
vcs_55_on=*55&amp;var:sc_vcs_55_default=0830</fkey>
<fkey idx="4" context="1" lp="on" default_text="CFU"
icon_type="kIconTypeFkeyTransfer" perm="">url
http://192.168.91.254/sipwise/vcs_mb.xml#sub=*[@id="vcs_72"]&amp;var:sc_
vcs_72_on=*72&amp;var:sc_vcs_72_off=#72</fkey>
<fkey idx="5" context="1" lp="on" default_text="SpeedDial"
icon_type="kIconTypeFkeySpeedDial" perm="">url
http://192.168.91.254/sipwise/vcs_mb.xml#sub=*[@id="vcs_50"]&amp;var:sc_
vcs_50_on=*50&amp;var:sc_vcs_50_default_slot=1</fkey>
</functionKeys>
```

[* TAGS default *]

IMPORTANT On the example above, the XML file 'vcs_mb.xml' needs to be placed on an external HTTP Server, or needs to be provisioned into the phone.

Please, contact SNOM for further assistance, or check SNOM Service Hub (Keyword: VCS Sipwise) if needed.

Factory Reset

For already provisioned phones, the end customer might need to perform a factory reset:

- Press **## (the phone starts to reboot)
- Hold # during boot
- Press *1* to choose the option *Settings reset*

The default keypad password is *0000*. The WEB username for factory-reset phones is *admin* with password *admin*.

See more details on SNOM Wiki.

Bootstrap URL

Copyright © Sipwise GmbH - All rights reserved

The default SNOM bootstrap URI is: http://<ngcp.fqdn>:1445/device/autoprov/bootstrap/{mac} You can find the URI above on SRAPS WEB interface for newly provisionied SNOM device. The variable "{mac}" will expand to device MAC address automatically on SNOM device before connecting to NGCP. You can change the default "Bootstrap URI" on page "Edit Device Model". See more details in PBX bootstrap, firmwares and security.

NOTE it is highly recommended to use a secure bootstrap config: https://<ngcp.fqdn>:1444/device/autoprov/config/{mac}

The bootstrap config should point to the proper firmware:

```
http://<ngcp.fqdn>:1445/device/autoprov/firmware/{mac}/from/{firmware_ve
rsion}/next
```

The variable '{firmware_version}' is available for SNOM firmware version 1.10.47+.

For earlier SNOM firmwares it is recommended to build a custom firmware upgrade vector using:

```
http://<ngcp.fqdn>:1445/device/autoprov/firmware/{mac}/from/0/latest
OR
http://<ngcp.fqdn>:1445/device/autoprov/firmware/{mac}/from/0/next/<tag>
```

See more details in PBX bootstrap, firmwares and security.

Yealink Device Bootstrap

Initial Bootstrapping

Yealink provides a zero-touch provisioning mechanism in their firmwares, which causes the factory-reset phones to connect to a Yealink web service at *https://rps.yealink.com* to check if a custom provisioning URL is configured for the MAC address of the phone. If an association between the MAC and a provisioning URL is found, the web service redirects the phone to the provisioning URL, where the phone connects to in order to obtain the configuration file.

If both Cisco SPA and Yealink phones are used, an issue with the Cisco-signed server certificate configured on the provisioning port (1444 by default) of the CloudPBX provisioning server arises. Yealink phones by default only connect to trusted server certificates, and the Cisco CA certificate used to sign the server certificate is not trusted by Yealink. Therefore, a two-step approach is used to disable the trusted check via a plain insecure http port (1445 by default) first, where only device-generic config options are served. No user credentials are provided in this case, because no SSL client authentication can be performed. The generic configuration disables the trusted check, and at the same time changes the provisioning URL to the secure port, where the Yealink phone is now able to connect to.

Copyright © Sipwise GmbH - All rights reserved

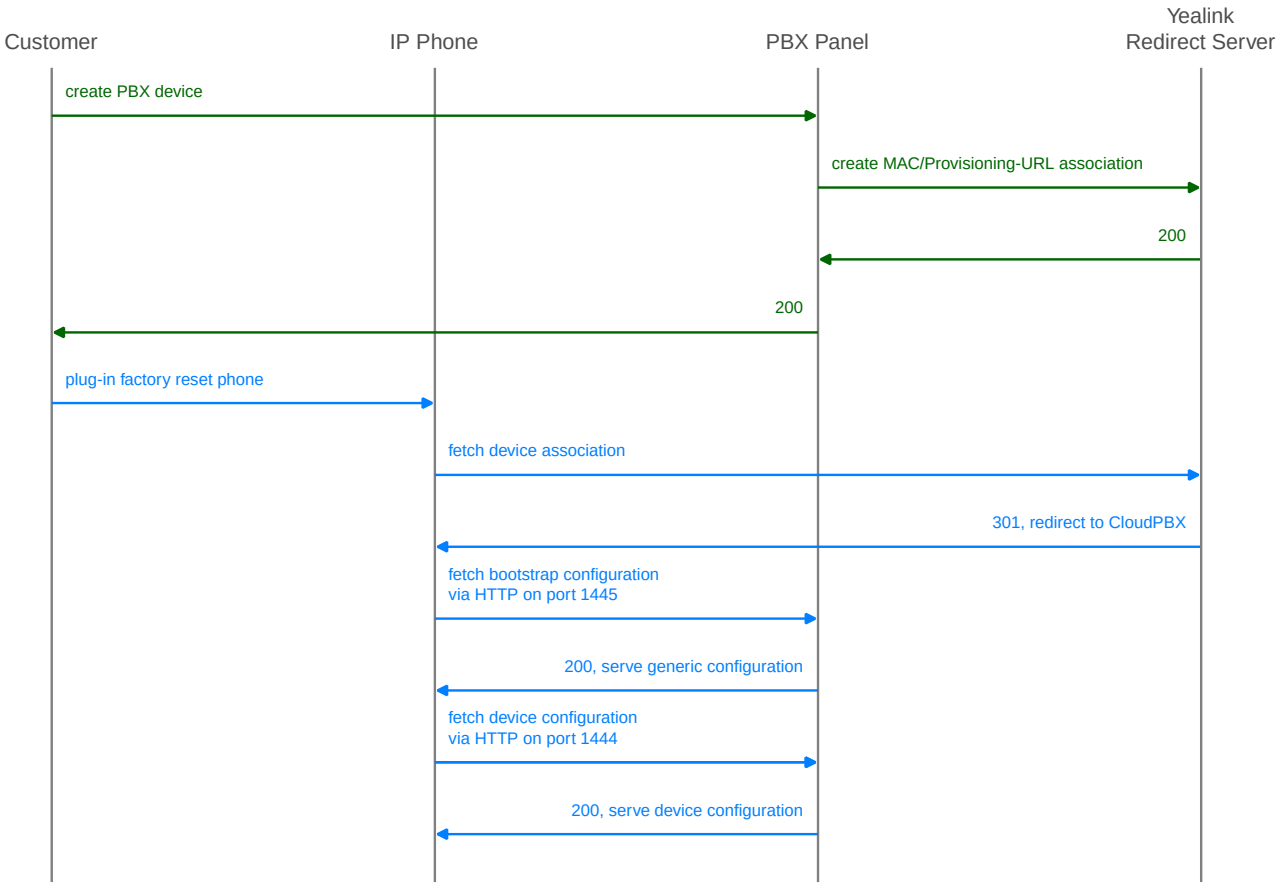


Figure 159. Initially bootstrap a Yealink phone

The CloudPBX module ensures that when an end customer creates a Yealink device, the MAC address is automatically provisioned on the Yealink web service via an API call, so the customer's phone can use the correct insecure bootstrap provisioning URL to connect to the auto-provisioning server of the CloudPBX for the generic configuration, which in turn provides the information on where to connect to for the secure, full configuration.

As a result, no customer interaction is required to bootstrap Yealink phones, other than just creating the phone with the proper MAC on the CloudPBX web interface.

Factory Enable Yealink Auto-Provisioning

Older Yealink firmwares don't automatically connect to the Yealink auto-provisioning server on initial boot, so it needs to be enabled manually by the end customer.

- Log in to *http://phone-ip/servlet?p=hidden&q=load* using *admin* and *admin* as user/password when prompted
- Change *Redirect Active* to *Enabled*
- Press *Confirm* and power-cycle phone

Subsequent Device Resyncs

The same procedure as with Cisco SPA phones applies, once a subscriber configured on the phone is registered.

Copyright © Sipwise GmbH - All rights reserved

Audiocodes Mediant Device Bootstrap and Configuration

Initial Bootstrapping

An Audiocodes device provides a zero-touch provisioning mechanism in its firmware which causes a factory-reset device to connect to the URL built into the firmware. This URL is pointing to Sipwise C5 provisioning server (in case of Sipwise C5 Carrier: web01 node) listening on TCP port 1444 for HTTPS sessions.

The prerequisites for the device provisioning are that the device has a routable IP address and can reach the IP address of Sipwise C5 provisioning interface.

The Audiocodes device should request the firmware file or CLI configuration file from Sipwise C5 platform. The firmware versions and CLI config versions are decoupled from each other; Sipwise C5 can not enforce specific version of the firmware on the device. Instead, it should be requested by the device itself. In other words, provisioning is a 'pull' and not a 'push' process.

Sipwise C5 expects the provisioning request from the Audiocodes device after SSL handshake and serves the requested file to the device if the device provides valid MAC address as the part of the URL. The MAC address is used to identify the device to Sipwise C5 platform. The firmware and CLI config files are provided at the following URLs:

- the base URL to download firmwares: https://<NGCP_IP>:1444/device/autoprov/firmware/001122334455/from/0/latest
- the base URL to download CLI config: https://<NGCP_IP>:1444/device/autoprov/config/001122334455

where 001122334455 should be replaced with the actual device's MAC address and <NGCP_IP> with IP address of Sipwise C5 provisioning interface.

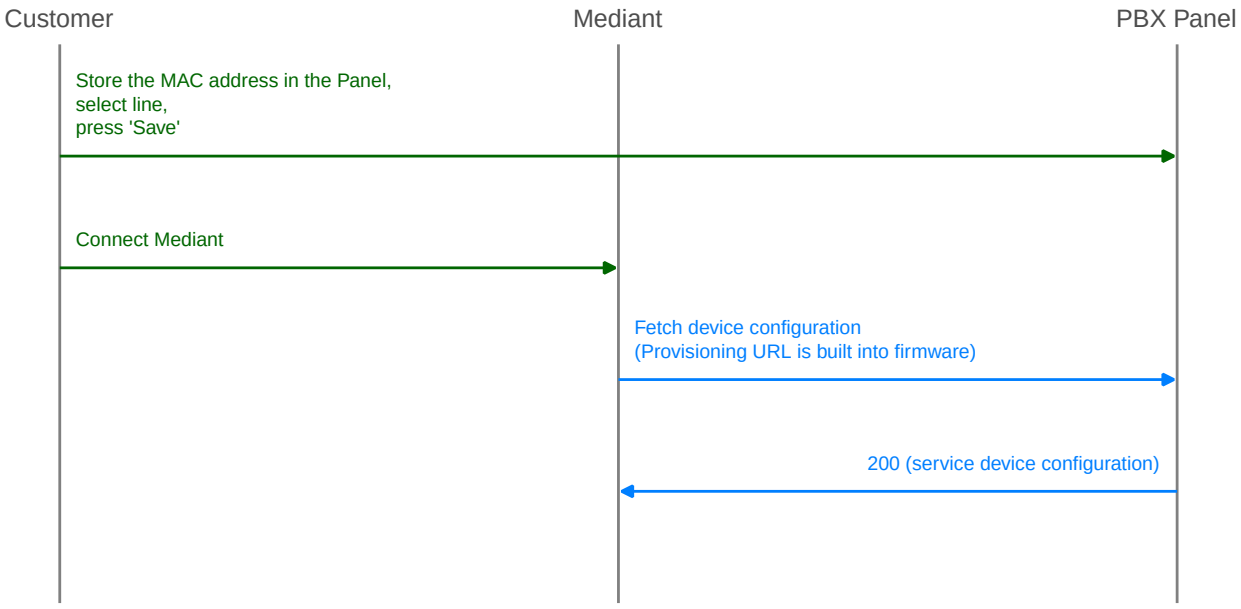


Figure 160. Initially bootstrap a Mediant gateway

Device management basics

The list of device models, firmwares and configurations are global to a reseller and are available for end

Copyright © Sipwise GmbH - All rights reserved

customer. This data is initially provided by Sipwise as bulk upload of all supported phone models. The firmwares and settings are stored in the database on the DB node pair(s). The Sipwise C5 leverages the Cloud PBX module with its template system to generate the configurations and firmware files from database on the fly. Please refer to the following chapters in Sipwise C5 handbook for the current information on how to perform device management:

- Uploading device firmwares
- Creating device configuration
- Creating device profiles

Parameterizing the Device Configuration Template

The device-specific parameters are filled in by the system individually when a physical device fetches its configuration file. Parameters from Sipwise C5 panel:

- username: Subscriber Details Master Data SIP Username
- password: Subscriber Details Master Data SIP Password
- domain: Subscriber Details Master Data Domain
- extension: Subscriber Details Master Data Extension
- area code: Subscriber Preferences Number Manipulations ac
- country code: Subscriber Preferences Number Manipulations cc

The produced **CLI config file** has the following structure:

1. SIP account credentials:

   ```
   "sip-definition account 0"
   ```

   user-name [username]

   password [password]

   host-name [domain]

   register reg

   contact-user "[country code][area code][extension]"

2. IP Groups:

   ```
   "voip-network ip-group 1" and "voip-network ip-group 2"
   ```

   sip-group-name [domain]

3. Proxy and registration settings:

   ```
   "sip-definition proxy-and-registration"
   ```

   set gw-name [domain]

Copyright © Sipwise GmbH - All rights reserved

4. Manipulations:

   manipulation-name "from trunk domain":

   ```
   "sbc manipulations message-manipulations 3"
   ```

   action-value "[% line.domain %]"

   manipulation-name "clip no screening":

   ```
   "sbc manipulations message-manipulations 8"
   ```

   action-value "\'<sip:+[country code][area code][extension]@' + param.ipg.dst.host + \'>'"

Specific CLI parameters are:

- [IPPBX_Hostname]
- [IPPBX_server_IP]

which are used at the following configuration parameters:

- Proxy settings:

  ```
  "voip-network proxy-ip 1"
  ```

  proxy-address [IPPBX_Hostname]

- Manipulations:

  ```
  "sbc manipulations message-manipulations 1"
  ```

  action-value [IPPBX_Hostname]

8.1.13. PBX bootstrap, firmwares and security

In theory, it is enough to provide the device with one config file which contains all parameters inside (including SIP credentials and a link to the new firmware). However, the connection from device to the server has to be secure due to the sensitive data inside the config file. A SSL tunnel must be used and the server must authenticate the phone using the client SSL certificate.

Due to the real world logistic speed the arrived device might have an outdated firmware which is considered as "insecure client" on the recently updated SSL server. Examples here could be an incompatible TLS version, SSL cipher suites and curves…

The only option to connect such a device to the server is to use more modern firmware and the only way to receive it is to download it from server. Some vendors provide initial/constant firmware upgrade as a part of their zero-touch provisioning platforms while it is not always possible to use it.

The NGCP platform also provides bootstrap functionality for such old devices/firmwares using

 Copyright © Sipwise GmbH - All rights reserved

'bootstrap' concept. Please note the difference:

- config: https://<ngcp.server>:1444/device/autoprov/config/<mac>
- bootstrap: http://<ngcp.server>:1445/device/autoprov/bootstrap/<mac>

The first SSL-based link provides a valid device config with all details, while the second/insecure link only provides a minimal config to update the firmware. Modern firmwares are usually SHA256 signed and can be securely re-distributed using insecure channels.

By default, the bootstrap URI is http://...:1445/device/autoprov/bootstrap/<mac> and can be changed on page "Edit Device Model" (field 'Bootstrap URI').

NOTE the bootstrap URI contains vendor specific variables, e.g. lower cased "{mac}" for SNOM and upper cased {MAC} for Panasonic. Please check the list of supported variables on the device vendor Website. Keep in mind, some variables might not be supported by very old firmware.

NGCP provides three access points by default: "config" (port 1444), "bootstrap" (port 1445) and "cisco" (port 1447). The last one is necessary due to the specific SSL setup for Cisco devices, see Server Certificate Authentication. Using "bootstrap"/"cisco" entrance is not mandatory and only necessary to bootstrap devices with very old firmware.

NGCP provides a flexible way to build a chain of bootstraps/configs to achieve the desired upgrade path. Every config/bootstrap contains the link on the next config/bootstrap. Device can (and often should) upgrade several firmwares to achieve the latest firmware version. Some vendors do not support upgrading from old, ancient firmware to the latest firmware directly. The specific upgrade path should be built.

NGCP supports various URIs to fetch any firmware using version and/or tag. For example, you can request some specific firmware version using:

```
https://<ngcp.fqdn>:1444/device/autoprov/firmware/<mac>/version/{firmwar
e_version}
https://<ngcp.fqdn>:1447/device/autoprov/firmware/<mac>/version/{firmwar
e_version}
http://<ngcp.fqdn>:1445/device/autoprov/firmware/<mac>/version/{firmware
_version}
```

All methods are available for any "config/bootstrap/cisco" entrances. Also firmware can be requested using "Firmware tag" (NGCP seraches versions first and repeats the search using the tag if nothing found by version):

```
https://<ngcp.fqdn>:1444/device/autoprov/firmware/<mac>/version/{firmwar
e_tag}
https://<ngcp.fqdn>:1447/device/autoprov/firmware/<mac>/version/{firmwar
e_tag}
http://<ngcp.fqdn>:1445/device/autoprov/firmware/<mac>/version/{firmware
_tag}
```

Copyright © Sipwise GmbH - All rights reserved

It is also possible to request the "latest" and the "next" firmware from your current possition

```
https://<ngcp.fqdn>:1444/device/autoprov/firmware/<mac>/from/<firmware_v
ersion>/latest
https://<ngcp.fqdn>:1444/device/autoprov/firmware/<mac>/from/<firmware_v
ersion>/next
https://<ngcp.fqdn>:1444/device/autoprov/firmware/<mac>/from/<firmware_v
ersion>/latest/<tag>
https://<ngcp.fqdn>:1444/device/autoprov/firmware/<mac>/from/<firmware_v
ersion>/next/<tag>
```

For example, NGCP has 7 firmwares configured for SNOM D715 with MAC 0004138aa403. For simplicity, let's call those firmware versions: 1, 2, 3, 4, 5, 6, 7. The firmwares '4' and '6' have the same tag 'mytag'. The latest firmware is firmware with id '7' and it has no tag set.

The endpoint 'latest' gives access to firmware '6' only (if 'mytag' is used):

```
> curl
http://myserver.com:1445/device/autoprov/firmware/0004138aa403/from/0/la
test
firmware 7

> curl
http://myserver.com:1445/device/autoprov/firmware/0004138aa403/from/0/la
test/mytag
firmware 6
```

The enpoint 'next' allows to address the complete chain of firmwares using tags:

Copyright © Sipwise GmbH - All rights reserved

```
> curl
http://myserver.com:1445/device/autoprov/firmware/0004138aa403/from/0/ne
xt
firmware 1  # the next firmware after version '0' is firmware '1'

> curl
http://myserver.com:1445/device/autoprov/firmware/0004138aa403/from/0/ne
xt/mytag
firmware 4  # the next firmware after version '0' with tag 'mytag' is
firmware '4'

> curl
http://myserver.com:1445/device/autoprov/firmware/0004138aa403/from/2/ne
xt/mytag
firmware 4  # the same as above, the firmware after version '2' with tag
'mytag' is firmware '4'

> curl
http://myserver.com:1445/device/autoprov/firmware/0004138aa403/from/4/ne
xt/mytag
firmware 6  # the next firmware after version '4' with tag 'mytag' is
firmware '6'
```

8.1.14. Device Provisioning and Deployment Workflows

This chapter provides information and hints for preparing and performing the deployment of certain VoIP devices at customer sites, that have a customer-facing interface which also needs customisation.

Audiocodes Mediant Device Provisioning Workflow

Audiocodes ISDN gateways and eSBCs are devices used to connect legacy (ISDN) PBX and IP-PBX to Sipwise C5 platform and maintain their operations within the Operator's network. Sipwise C5 offers a *SipConnect 1.1* compliant signaling and media interface to connect SIP trunks to the platform. In addition to this interface, Sipwise C5 provides an auto-provisioning mechanism to configure SIP endpoints like IP phones, media gateways and eSBCs.

Provisioning URL

An Audiocodes device needs to obtain the provisioning URL of Sipwise C5 in one way or the other to request its device configuration and subsequently download specific firmwares, obtain SIP credentials to connect to the network facing side, and configure the customer facing side for customer devices to connect either via ISDN or SIP. Typical ways of obtaining the provisioning URL for a SIP endpoint are:

- using DHCP option-66 (in a pre-staging environment or directly at the customer premise) where vendor-specific Redirect Servers are configured in the default configuration or firmware
- getting pre-configured per deployment from the SIP endpoint vendor
- getting pre-configured per deployment by a 3rd party distributor

The assumption is that Audiocodes devices are supplied with a firmware (and all required SSL certificates) being pre-configured and the provisioning URL pointing to an Operator URL Sipwise C5 is

Copyright © Sipwise GmbH - All rights reserved

serving, before handing the devices over to field service engineers doing the truck rolls.

Field Configuration

The Sipwise C5 provides a SipConnect 1.1 compliant interface on the network side for the Audiocodes devices. This interface defines the numbering formats of the calling and called party, the SIP header mechanisms to provide CLI restriction, the RTP codecs, etc.

On the customer facing side, however, those variables might be different from deployment to deployment:

- An IP-PBX might choose to only send its extension as calling party number, or might choose to send the full number in national format.
- It might choose to use the SIP From-header mechanisms to suppress displaying of the CLI, or use the SIP Privacy header.
- The same uncertainty exists to some extent for a legacy PBX connecting via ISDN to the Audiocodes device.

The assumption here is that a field service engineer is NOT supposed to change the Audiocodes configuration in order to make the customer interface work, as this will lead to big issues in maintaining those local changes, especially if a replacement of the device is necessary. Instead, the Audiocodes configuration must ensure that all different kinds of variants in terms of SIP headers, codecs and number formats are translated correctly to the network side and vice versa. If it turns out that there are scenarios in the field which are not handled correctly, temporary local changes might be performed to finish a truck roll, but those changes MUST be communicated to the platform operator, and the server-side configuration templates must be adapted to handle those scenarios gracefully as well.

For deployments with ISDN interfaces on the customer facing side of the Audiocodes, different *Device Profiles* with specific *Device Configurations* per *Device Model* must exist to handle certain scenarios, specifically whether the ISDN interface is operating in Point-to-Point or Point-to-Multipoint mode. Configuration options like which side is providing the clock-rate are to be defined up-front, and the PBX must be reconfigured to adhere to the configuration.

Network Configuration

On the network facing side, both the ISDN and eSBC style deployments have to be designed to obtain an IP address via DHCP. The definition of the IP address ranges is up to the Operator. It may or may not be NAT-ed, but it is advised to use a private IP range directly routed in the back-bone to avoid NAT.

On the customer facing side, networking is only relevant for the eSBC deployment. In order to make the IP-PBX configuration as stream-lined as possible, a pre-defined network should be established on the customer interface of the Audiocodes device.

TIP

The proposal is to define a network 192.168.255.0/24 with the Audiocodes device using the IP 192.168.255.2 (leaving the 192.168.255.1 to a possible gateway). The IP-PBX could obtain its IP address via DHCP from a DHCP server running on the Audiocodes device (e.g. serving IP addresses in the range of 192.168.255.100-254), or could have it configured manually (e.g. in the range of 192.168.255.3-99). Since the Audiocodes device IP on the customer side is always fixed at 192.168.255.2, the IP-PBX for each customer can be configured the same way, pointing the SIP proxy/registrar or outbound proxy always to this IP.

Copyright © Sipwise GmbH - All rights reserved

The customer facing side is outside the Sipwise demarcation line, that's why the network configuration mentioned above only serves as proposal and any feedback is highly welcome. However, it must be communicated how the customer facing network is going to be configured, because Sipwise C5 needs to incorporate this configuration into the Audiocodes configuration templates.

Audiocodes Mediant Device Deployment Workflow

Pre-Configuration on Sipwise C5 platform

1. Before connecting a customer to a SIP trunk, it must be clear which Audiocodes *Device Model* is going to be used (depending on if, which and how many ISDN ports are necessary) and which *Device Profile* for the *Device Model* is required (eSBC mode, ISDN P-to-P or P-to-MP mode). Based on that, the correct physical device must be picked.
2. Next, the customer has to be created on Sipwise C5 . This step requires the creation of the customer, and the creation of a subscriber within this customer. For the subscriber, the proper E.164 numbers or number blocks must be assigned, and the correct subscriber preferences must be set for the network interface to adhere to the SipConnect 1.1 interface. This step is automated by a script provided by Sipwise until the provisioning work-flow is fully integrated with Operator's OSS/BSS systems. *Required parameters are:*

   an external customer id to relate the customer entity on Sipwise C5 with a customer identifier in Operator's IT systems

   a billing profile name

   a subscriber username and password, the domain the subscriber is configured for

   the numbers or number blocks assigned to the subscriber, and the network provided number of the subscriber

   optional information is geographic location information and IP network information to properly map emergency calls
3. Finally, the association between the MAC address of the Audiocodes device and the SIP subscriber to be used on the SIP trunk must be established. This step is also automated by a script provided by Sipwise. *Required parameters are:*

   the subscriber id

   the Device Profile to be used

   and the MAC address of the Audiocodes device

Installation

Once the above requirements are fulfilled and the customer is created on Sipwise C5 , the Audiocodes device can be installed at the customer premise.

When the Audiocodes device boots, it requests the configuration file from Sipwise C5 by issuing a GET request via HTTPS.

For **authentication and authorization** purposes, Sipwise C5 requests an SSL client certificate from the device and will check whether it's signed by a Certificate Authority known to Sipwise C5 . Therefore, Audiocodes must provide the CA certificate used to sign the devices' client certificates to Sipwise to allow for this process. Also, Sipwise C5 will provide an SSL server certificate to the device. The device must validate this certificate in order to prevent man-in-the-middle attacks. Options here are to have:

Copyright © Sipwise GmbH - All rights reserved

- Sipwise provide a self-signed certificate to Audiocodes for Audiocodes or a 3rd party distribution partner to configure it as trusted CA in the pre-staging process
- the Operator provide a certificate signed by a CA which is already in the trust store of the Audiocodes devices.

Once the secured HTTPS connection is established, Sipwise C5 will provide a CLI style configuration file, with its content depending on the pre-configured *Device Profile* and subscriber association to the device's MAC address.

The configuration includes the firmware version of the latest available firmware configured for the *Device Model*, and a URL defining from where to obtain it. The configuration details on how the Audiocodes devices manage the scheduling of firmware updates are to be provided by Audiocodes or its partners, since this is out of scope for Sipwise. Ideally, firmware updates should only be performed if the device is idle (no calls running), and within a specific time-frame (e.g. between 1 a.m. and 5 a.m. once a certain firmware version is reached, including some random variation to prevent all devices to download a new firmware version at the same time).

Device Replacement

If a customer requires the replacement of a device, e.g. due to hardware issues or due to changing the number or type of ISDN interfaces, a new association of the new device MAC, its *Device Profile* and the subscriber must be established.

In order to make the change as seamless as possible for the customer, a new device is created for the customer with the new MAC, a proper *Device Profile*, but the same subscriber as used on the old device. Once the new device boots at the customer premise, it will obtain its configuration and will register with the same subscriber as the old device (in case it's still operational). For inbound calls to the customer, this will cause parallel ringing to take place, and it's up to the customer or the field engineer when to re-configure or re-cable the PBX to connect to one or the other device.

Once the old device is decommissioned, the old MAC association can be deleted on Sipwise C5 .

8.1.15. Adjusting the PBX Devices Configuration

Usually, everything required for PBX devices autoprovisioning is uploaded automatically as described in PBX Device Provisioning. In case you would like to introduce changes into a PBX device configuration, create a custom PBX device profile or even upload a newer firmware, this section will help you.

The *Device Management* is used by admins and resellers to define the list of device models, firmwares and configurations available for end customer usage. These settings are pre-configured for the default reseller up-front by Sipwise and have to be set up for every reseller separately, so a reseller can choose the devices he'd like to serve and potentially tweak the configuration for them. List of available pre-configured devices.

End customers choose from a list of *Device Profiles*, which are defined by a specific *Device Model*, a list of *Device Firmwares* and a *Device Configuration*. The following sections describe the setup of these components.

To do so, go to *SettingsDevice Management*.

Copyright © Sipwise GmbH - All rights reserved

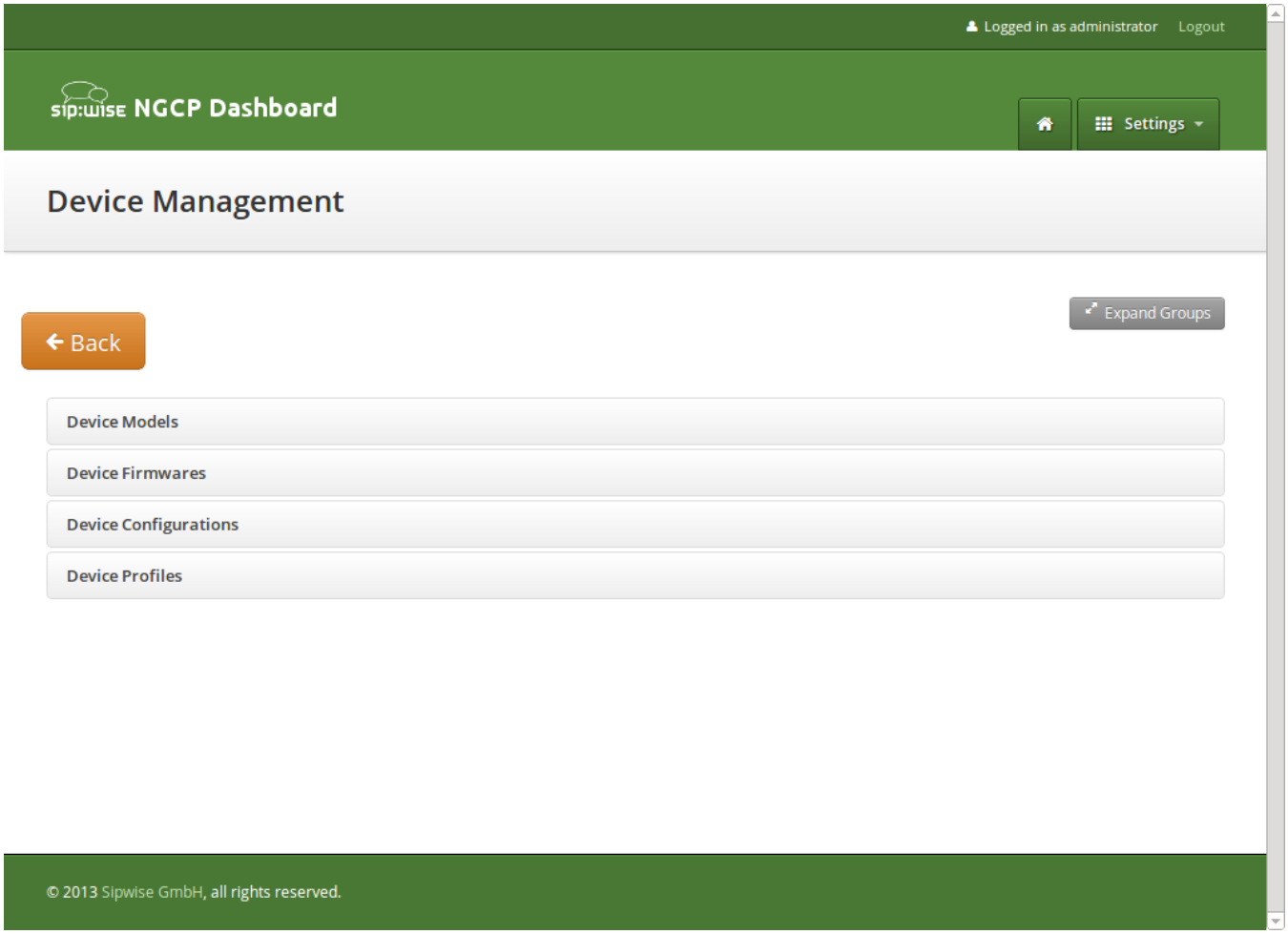


Figure 161. Device Management

Setting up Device Models

A *Device Model* defines a specific hardware device, like the vendor, model name, the number of keys and their capabilities. For example a Cisco SPA504G has 4 keys, which can be used for private lines, shared lines (SLA) and busy lamp field (BLF). If you have an additional attendant console, you get 32 more buttons, which can only do BLF.

In this example, we will create a Cisco SPA504G with an additional Attendant Console.

Expand the *Device Models* row and click *Create Device Model*.

First, you have to select the reseller this device model belongs to, and define the vendor and model name.

Copyright © Sipwise GmbH - All rights reserved

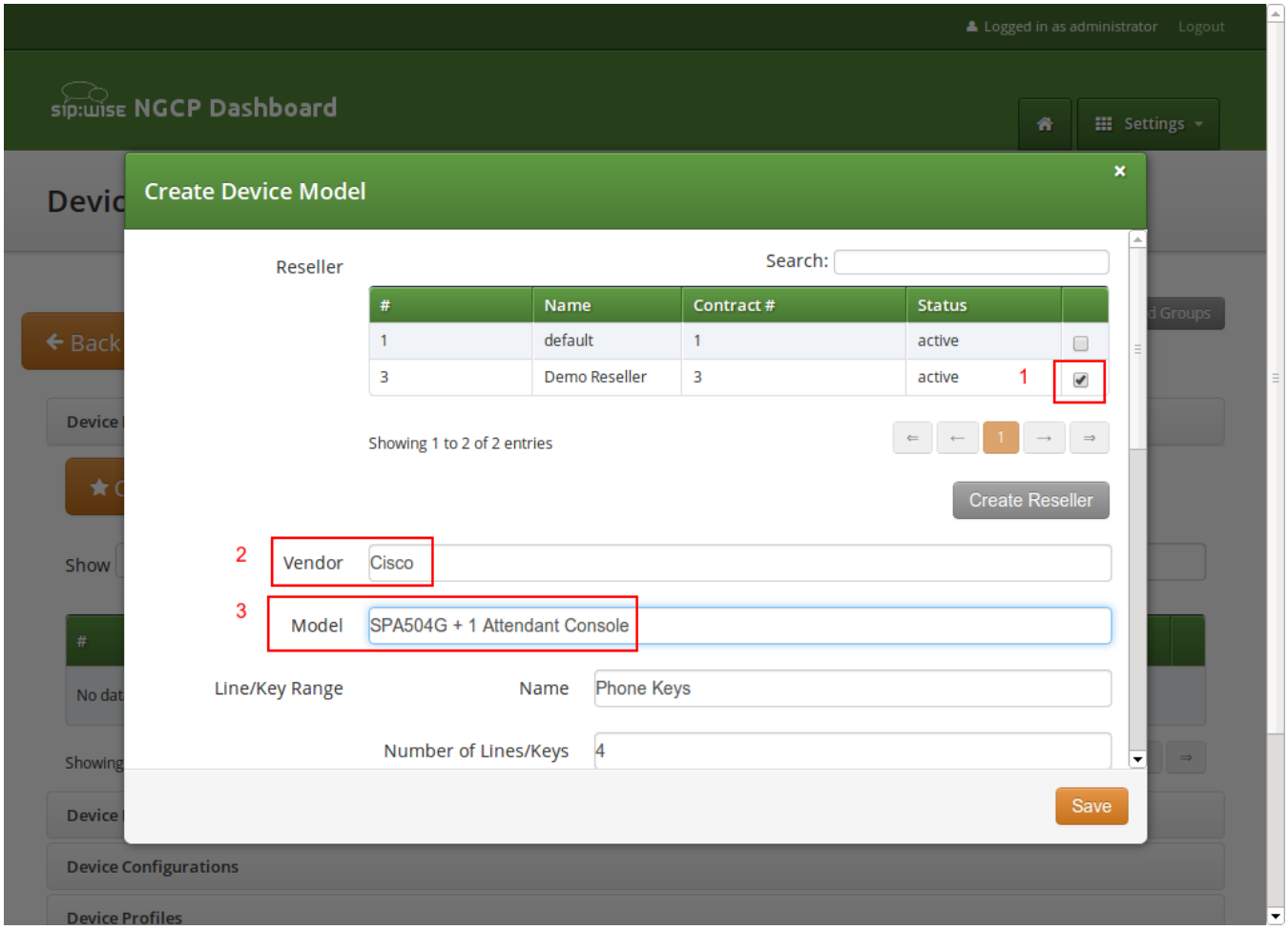


Figure 162. Create Device Model Part 1

In the *Line/Key Range* section, you can define the first set of keys, which we will label *Phone Keys*. The name is important, because it is referenced in the configuration file template, which is described in the following sections. The SPA504G internal phone keys support private lines (where the customer can assign a normal subscriber, which is used to place and receive standard phone calls), shared lines (where the customer can assign a subscriber which is shared across multiple people) and busy lamp field (where the customer can assign other subscribers to be monitored when they get a call, and which also acts as speed dial button to the subscriber assigned for BLF), so we enable all 3 of them.

Copyright © Sipwise GmbH - All rights reserved

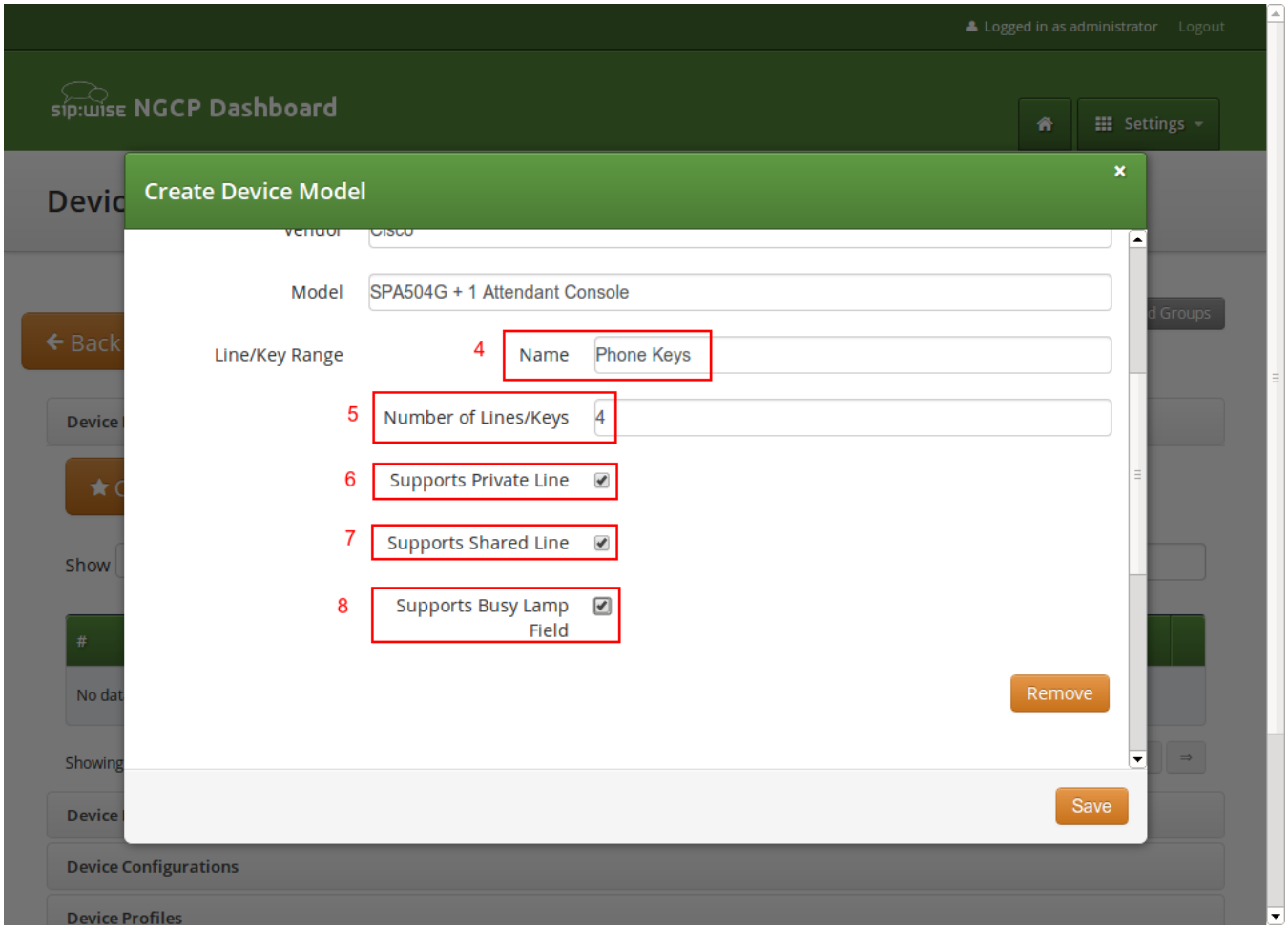


Figure 163. Create Device Model Part 2

In order to also configure the attendant console, press the *Add another Line/Key Range* button to specify the attendant console keys.

Again provide a name for this range, which will be *Attendant Console 1* to match our configuration defined later. There are 32 buttons on the attendant console, so set the number accordingly. Those 32 buttons only support BLF, so make sure to **uncheck** the private and shared line options, and only check the `Supports Busy Lamp Field` option.

Copyright © Sipwise GmbH - All rights reserved

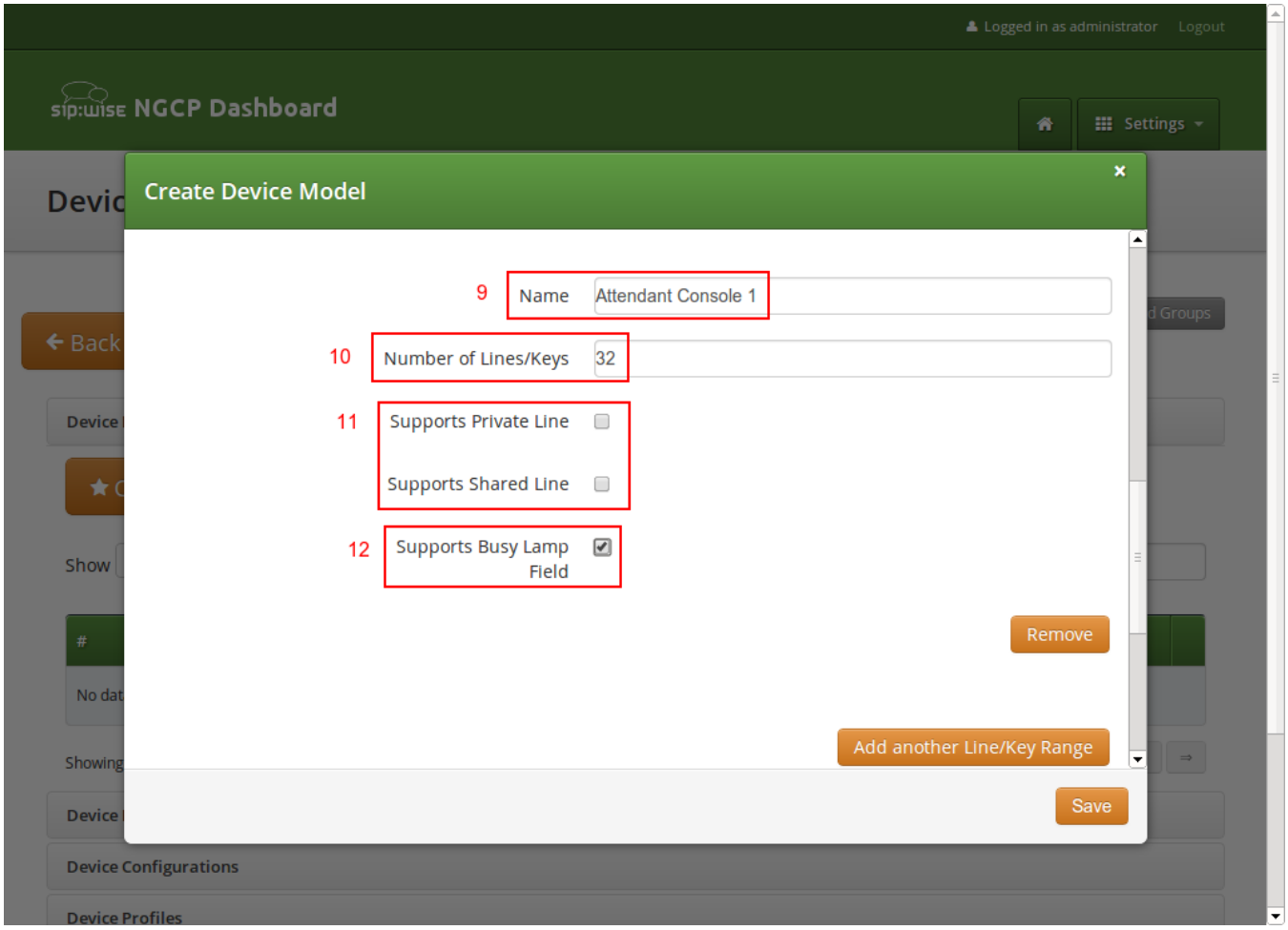


Figure 164. Create Device Model Part 3

The last two settings to configure are the *Front Image* and *MAC Address Image* fields. Upload a picture of the phone here in the first field, which is shown to the customer for him to recognize easily how the phone looks like. The MAC image is used to tell the customer where he can read the MAC address from. This could be a picture of the back of the phone with the label where the MAC is printed, or an instruction image how to get the MAC from the phone menu.

The rest of the fields are left at their default values, which are set to work with Cisco SPAs. Their meaning is as follows:

- *Bootstrap Sync URI*: If a stock phone is plugged in for the first time, it needs to be provisioned somehow to let it know where to fetch its configuration file from. Since the stock phone doesn't know about your server, you have to define an HTTP URI here, where the customer is connected with his web browser to set the according field.
- *Bootstrap Sync HTTP Method*: This setting defines whether an HTTP GET or POST is sent to the Sync URI.
- *Bootstrap Sync Params*: This setting defines the parameters appended to the Sync URI in case of a GET, or posted in the request body in case of POST, when the customer presses the *Sync* button later on.

Finally press *Save* to create the new device model.

Copyright © Sipwise GmbH - All rights reserved

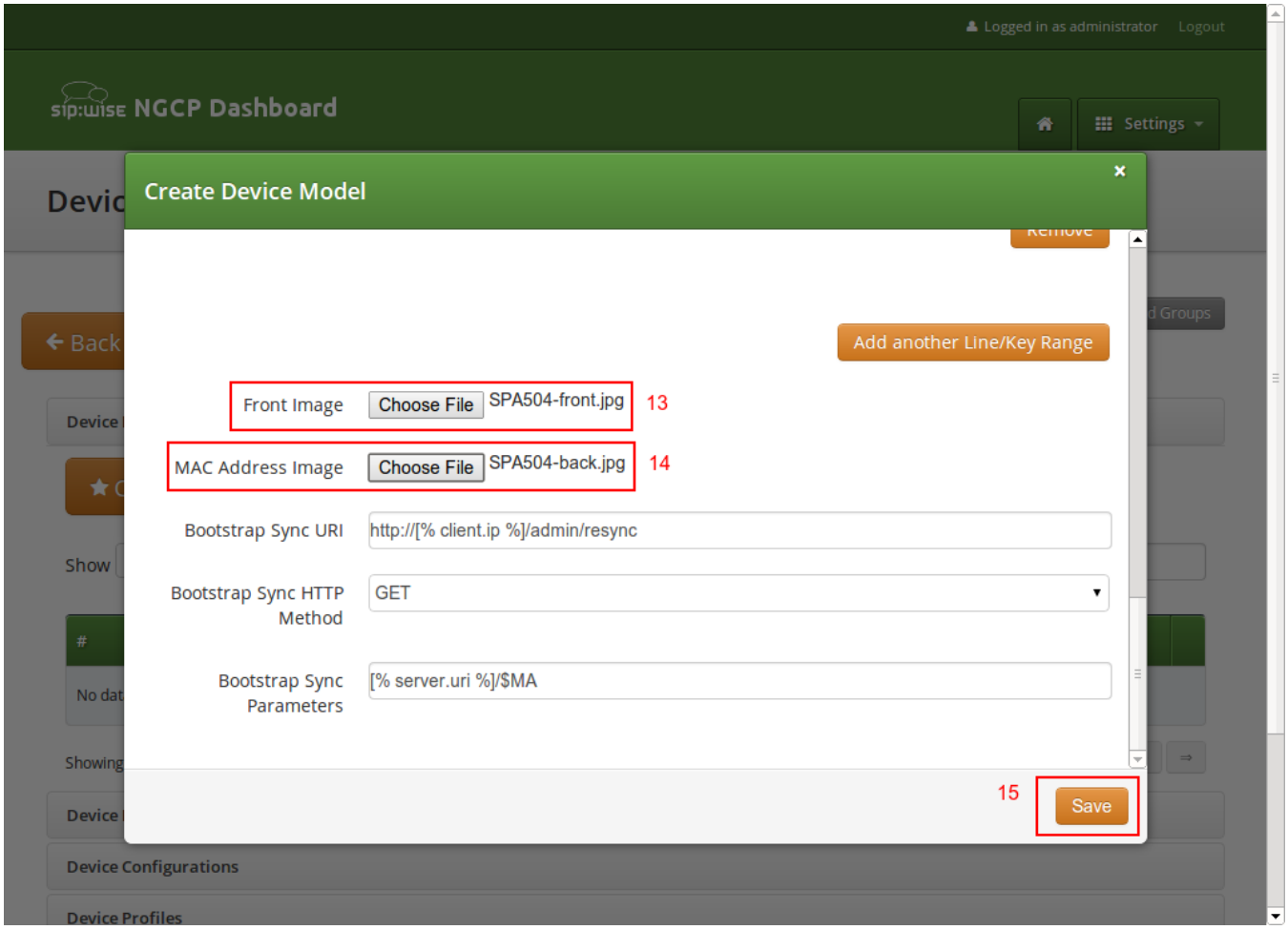


Figure 165. Create Device Model Part 4

Uploading Device Firmwares

A device model can optionally have one or more device firmware(s). Some devices like the Cisco SPA series don't support direct firmware updates from an arbitrary to the latest one, but need to go over specific firmware steps. In the device configuration discussed next, you can return the *next* supported firmware version, if the phone passes the current version in the firmware URL.

Since a stock phone purchased from any shop can have an arbitrary firmware version, we need to upload all firmwares needed to get from any old one to the latest one. In case of the Cisco SPA3x/SPA5x series, that would be the following versions, if the phone starts off with version 7.4.x:

- spa50x-30x-7-5-1a.bin
- spa50x-30x-7-5-2b.bin
- spa50x-30x-7-5-5.bin

So to get an SPA504G with a firmware version 7.4.x to the latest version 7.5.5, we need to upload each firmware file as follows.

Open the *Device Firmware* row in the *Device Management* section and press *Upload Device Firmware*.

Select the device model we're going to upload the firmware for, then specify the firmware version and choose the firmware file, then press *Save*.

Copyright © Sipwise GmbH - All rights reserved

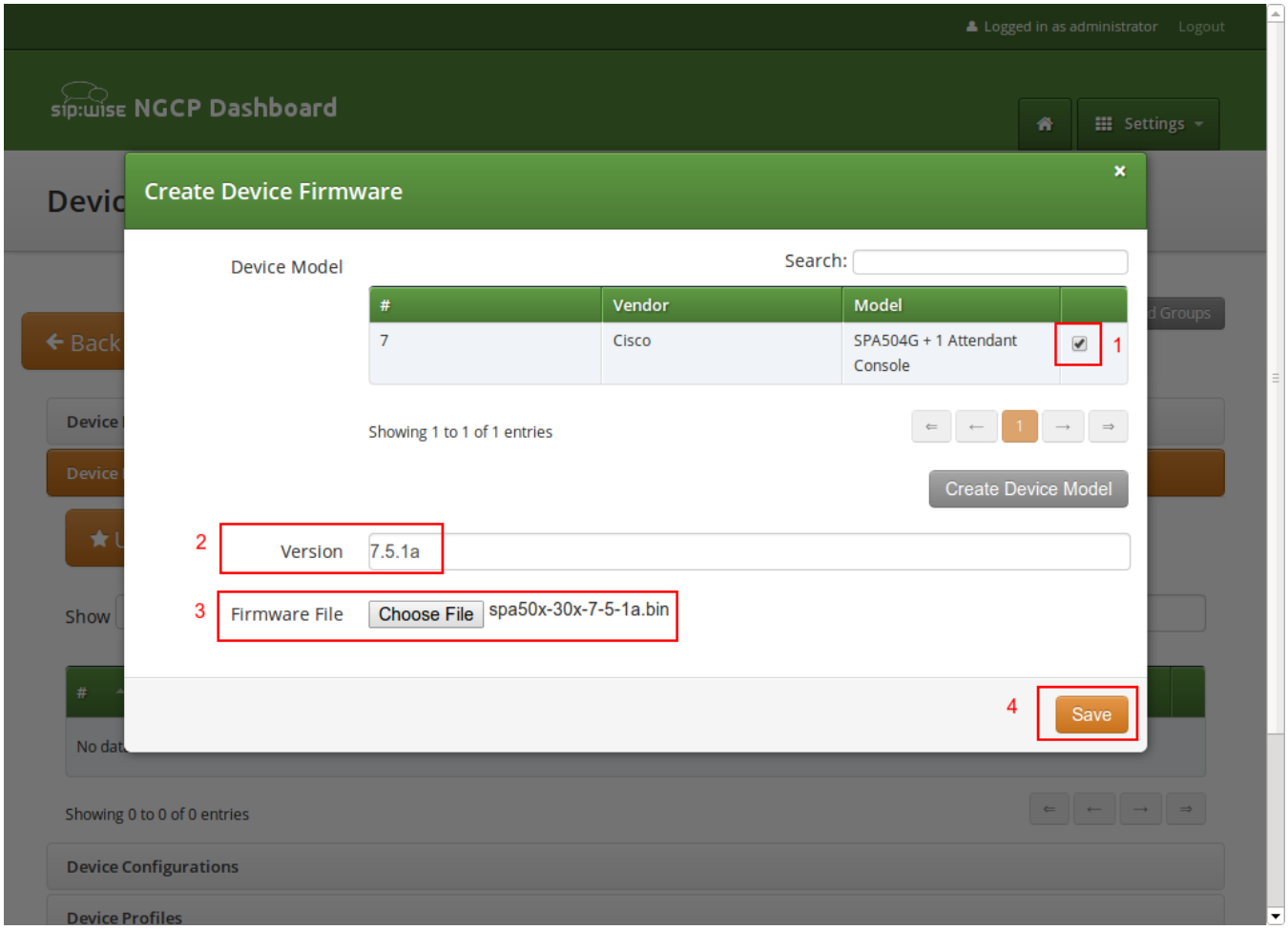


Figure 166. Upload Device Firmware

Repeat this step for every firmware in the list above (and any new firmware you want to support when it's available).

Creating Device Configurations

Each customer device needs a configuration file, which defines the URL to perform firmware updates, and most importantly, which defines the subscribers and features configured on each of the lines and keys. Since these settings are different for each physical phone at all the customers, the Cloud PBX module provides a template system to specify the configurations. That way, template variables can be used in the generic configuration, which are filled in by the system individually when a physical device fetches its configuration file.

To upload a configuration template, open the *Device Configuration* row and press *Create Device Configuration*.

Select the device model and specify a version number for this configuration (it is only for your reference to keep track of different versions). For Cisco SPA phones, keep the *Content Type* field to `text/xml`, since the configuration content will be served to the phone as XML file.

For devices other than the Cisco SPA, you might set `text/plain` if the configuration file is plain text, or `application/octet-stream` if the configuration is compiled into some binary form.

Finally paste the configuration template into the *Content* area and press *Save*.

Copyright © Sipwise GmbH - All rights reserved

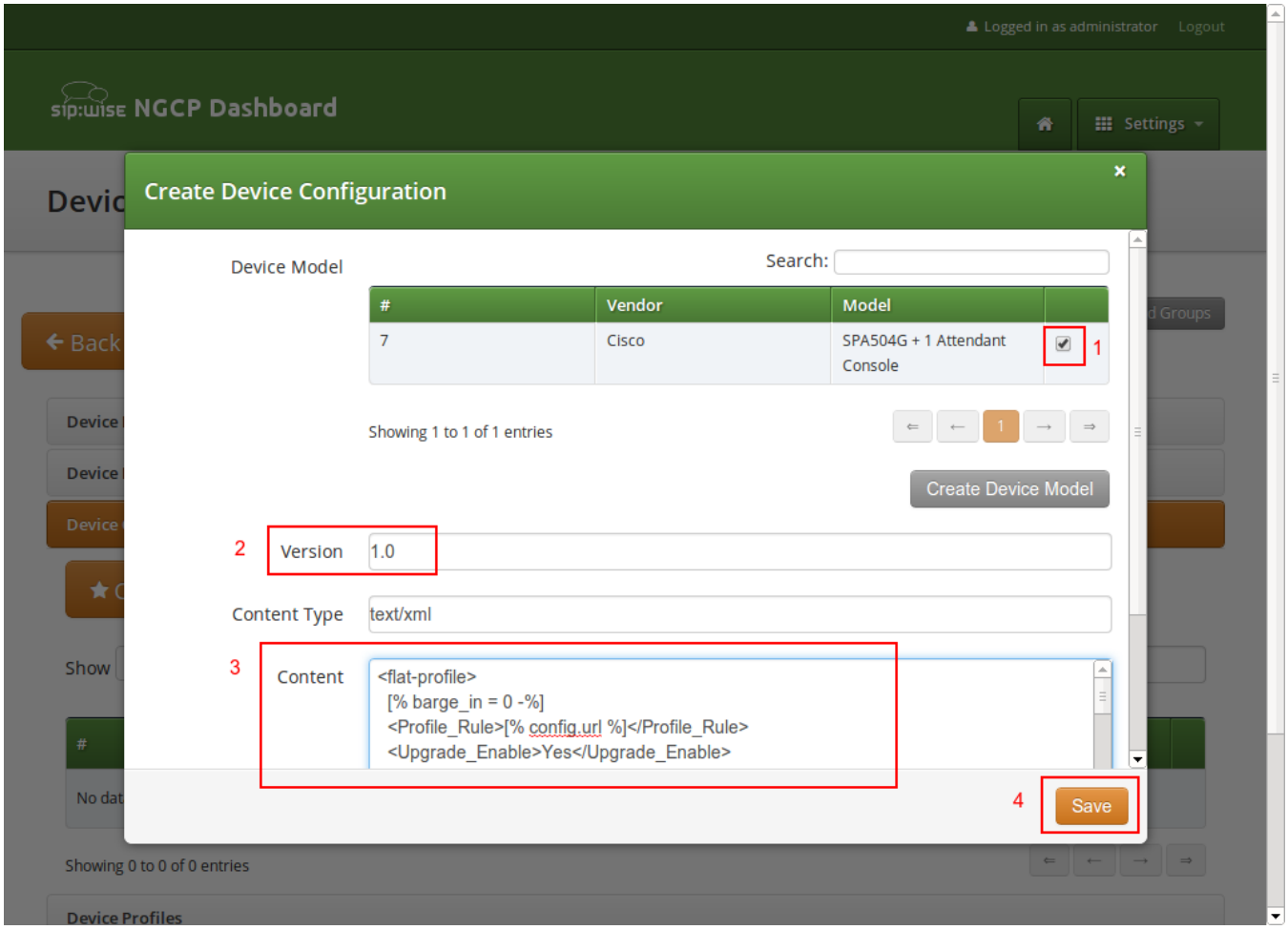


Figure 167. Upload Device Configuration

The templates for certified device models are provided by Sipwise, but you can also write your own. The following variables can be used in the template:

- `config.url`: The URL to the config file, including the device identifier (e.g. *http://sip.example.org:1444/device/autoprov/config/001122334455*).
- `config.baseurl`: The URL to the config file, without the device identifier (e.g. *http://sip.example.org:1444/device/autoprov/config/*).
- `config.mac`: The device MAC address.
- `config.filename`: The requested config filename (e.g. `001122334455-firmware.xml` in case the URL is *http://sip.example.org:1444/device/autoprov/config/001122334455-firmware.xml*).
- `firmware.maxversion`: The latest firmware version available on the system for the specific device.
- `firmware.baseurl`: The base URL to download firmwares (e.g. *http://sip.example.org:1444/device/autoprov/firmware*). To fetch the next newer firmware for a Cisco SPA, you can use the template line `[% firmware.baseurl %]/$MA/from/$SWVER/next`.
- `phone.model`: The device model. It is useful for the cases of a shared/common template for several devices within the same vendor.
- `phone.vendor`: The device vendor name. It is available for the general PBX template's flexibility.
- `phone.stationname`: The name of the station (physical device) the customer specifies for this

Copyright © Sipwise GmbH - All rights reserved

phone. Can be used to show on the display of the phone.

- `phone.lineranges`: An array of lines/keys as specified for the device model. Each entry in the array has the following keys:

  `name`: The name of the line/key range as specified in the *Device Model* section (e.g. *Phone Keys*).

  `num_lines`: The number of lines/keys in the line range (e.g. 4 in our *Phone Keys* example, or 32 in our *Attendant Console 1* example).

  `lines`: An array of lines (e.g. subscriber definitions) for this line range. Each entry in the array has the following keys:

  `keynum`: The index of the key in the line range, starting from 0 (e.g. `keynum` will be 3 for the 4th key of our *Phone Keys* range).

  `rangenum`: The index of the line range, starting from 0. The order of line ranges is as you have specified them (e.g. *Phone Keys* was specified first, so it gets `rangenum` 0, *Auto Attendant 1* gets rangenum 1).

  `type`: The type of the line/key, one of `private`, `shared` or `blf`.

  `username`: The SIP username of the line.

  `domain`: The SIP domain of the line.

  `password`: The SIP password of the line.

  `displayname`: The SIP Display Name of the line.

In the configuration template, you can adjust embedded variable references for the existing variables. Please also check the bootstrap section: PBX bootstrap, firmwares and security. If you need other specific variables, please request their development from Sipwise.

TIP In order to change the provisioning base IP and port (default 1444), you have to access `/etc/ngcp-config/config.yml` and change the value `host` and `port` under the `autoprov.server` section.

Creating Device Profiles

When the customer configures his own device, he doesn't select a *Device Model* directly, but a *Device Profile*. A device profile specifies which model is going to be used with which configuration version. This allows the operator to create new configuration files and assign them to a profile, while still keeping older configuration files for reference or roll-back scenarios. It also makes it possible to test new firmwares by creating a test device model with the new firmware and a specific configuration, without impacting any existing customer devices.

To create a *Device Profile* for our phone, open the *Device Profile* row in the *Device Management* section and press *Create Device Profile*.

Select the device configuration (which implicitly identifies a device model) and specify a *Profile Name*. This name is what the customer sees when he is selecting a device he wants to provision, so pick a descriptive name which unambiguously identifies a device. Press *Save* to create the profile.

Copyright © Sipwise GmbH - All rights reserved

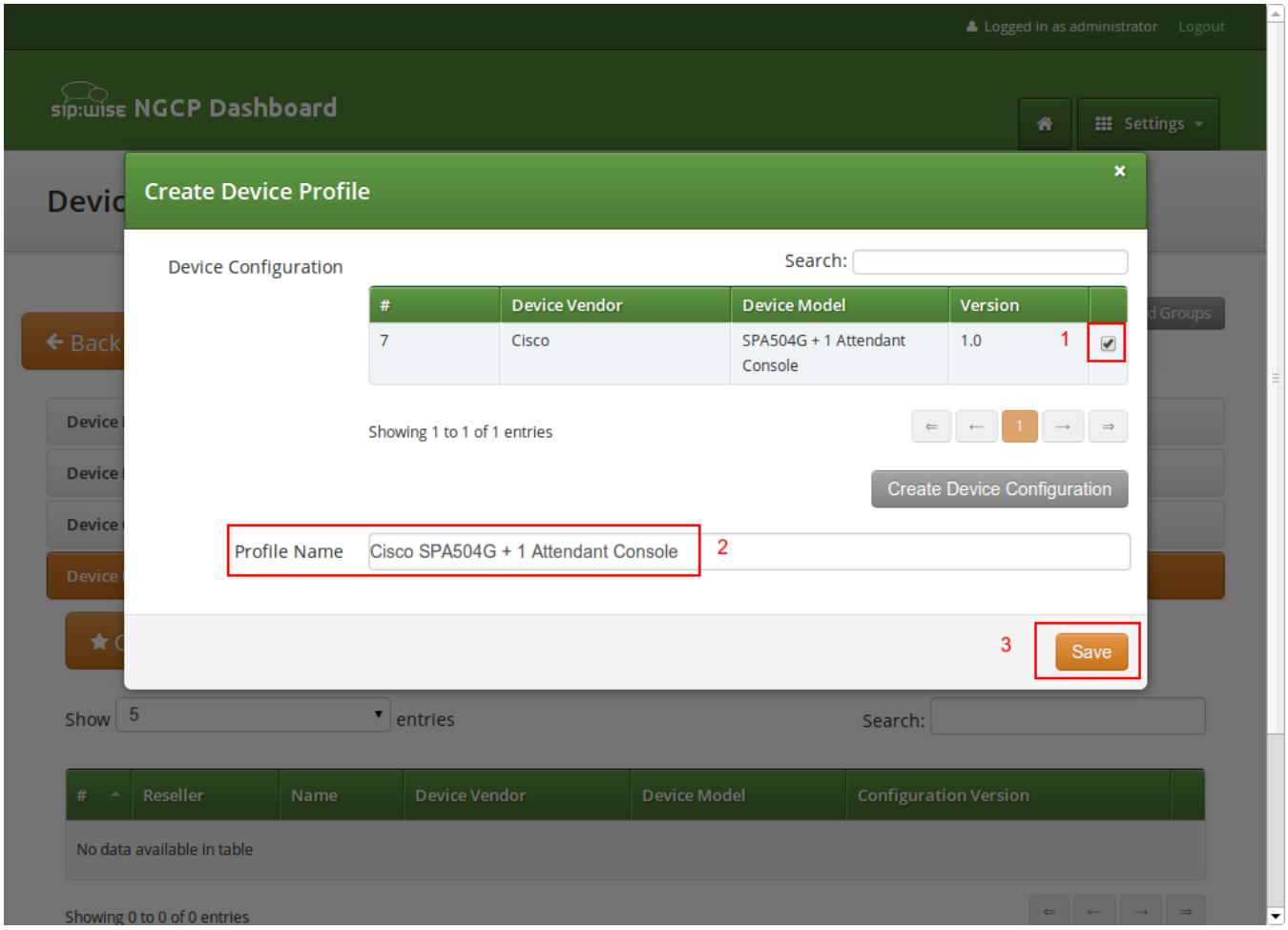


Figure 168. Create Device Profile

Repeat the steps as needed for every device you want to make available to customers.

8.1.16. List of available pre-configured devices

Vendor	Model	Available from release	Deprecated from release
ALE	8008	mr7.0.1.1	-
ALE	8018	mr7.0.1.1	-
ALE	8028	mr7.0.1.1	-
ALE	8058	mr7.0.1.1	-
ALE	8068	mr7.0.1.1	-
ALE	8078	mr7.0.1.1	-
ALE	AOM10	mr7.0.1.1	-
ALE	AOM40	mr7.0.1.1	-
ALE	AOMEL	mr7.0.1.1	-
ALE	M3	mr10.2.1.1	-
ALE	M5	mr10.2.1.1	-

Copyright © Sipwise GmbH - All rights reserved

Vendor	Model	Available from release	Deprecated from release
ALE	M7	mr10.2.1.1	-
ALE	H3P	mr10.2.1.1	-
ALE	H6	mr10.2.1.1	-
Audiocodes	Mediant800	mr4.1.1.1	mr10.5.1.1
Cisco	ATA112	mr3.4.1.1	mr10.5.1.1
Cisco	ATA122	mr3.4.1.1	mr10.5.1.1
Cisco	SPA232D	mr3.4.1.1	mr10.5.1.1
Cisco	SPA301	mr3.4.1.1	mr10.5.1.1
Cisco	SPA303	mr3.4.1.1	mr10.5.1.1
Cisco	SPA501G	mr3.4.1.1	mr10.5.1.1
Cisco	SPA502G	mr3.4.1.1	mr10.5.1.1
Cisco	SPA512G	mr3.4.1.1	mr10.5.1.1
Cisco	SPA504G	mr3.4.1.1	mr10.5.1.1
Cisco	SPA504G + SPA500S	mr3.7.1.4	mr10.5.1.1
Cisco	SPA504G + two SPA500S	mr3.7.1.4	mr10.5.1.1
Cisco	SPA514G	mr3.4.1.1	mr10.5.1.1
Cisco	SPA508G	mr3.4.1.1	mr10.5.1.1
Cisco	SPA509G	mr3.4.1.1	mr10.5.1.1
Cisco	SPA525G	mr3.4.1.1	mr10.5.1.1
Grandstream	HT814	mr5.1.1.1	-
Grandstream	GXW-4008	mr5.1.1.1	-
Grandstream	GXW-4216	mr5.1.1.1	-
Innovaphone	IP2X2X	mr3.8.3.3	mr10.5.1.1
Innovaphone	IP230-X	mr3.8.3.3	mr10.5.1.1
Innovaphone	IP232	mr3.8.3.3	mr10.5.1.1
Innovaphone	IP222	mr3.8.3.3	mr10.5.1.1
Innovaphone	IP240	mr3.8.3.3	mr10.5.1.1
Innovaphone	IP22	mr3.8.3.3	mr10.5.1.1
Innovaphone	IP111	mr3.8.3.3	mr10.5.1.1
Panasonic	KX-UT113	mr3.7.1.1	mr10.5.1.1
Panasonic	KX-UT123	mr3.7.1.1	mr10.5.1.1
Panasonic	KX-UT133	mr3.7.1.1	mr10.5.1.1
Panasonic	KX-UT136	mr3.7.1.1	mr10.5.1.1

Copyright © Sipwise GmbH - All rights reserved

Vendor	Model	Available from release	Deprecated from release
Panasonic	KX-UT248	mr3.7.1.1	mr10.5.1.1
Panasonic	KX-TGP600	mr5.1.1.1	mr10.5.1.1
Panasonic	KX-HDV330	mr5.1.1.1	mr10.5.1.1
Panasonic	KX-HDV230	mr5.1.1.1	mr10.5.1.1
Panasonic	KX-HDV130	mr5.1.1.1	mr10.5.1.1
Polycom	VVX300	mr5.4.1.1	mr10.5.1.1
Polycom	VVX400	mr5.4.1.1	mr10.5.1.1
Polycom	VVX500	mr5.4.1.1	mr10.5.1.1
SNOM	D715	mr9.1.1.1	-
SNOM	D717	mr9.1.1.1	-
SNOM	D735	mr9.1.1.1	-
SNOM	D785	mr9.1.1.1	-
SNOM	M900	mr9.1.1.1	-
SNOM	M65	mr9.1.1.1	-
SNOM	M70	mr9.1.1.1	-
SNOM	M80	mr9.1.1.1	-
SNOM	M85	mr9.1.1.1	-
SNOM	M90	mr9.1.1.1	-
SNOM	A190 (no uaCSTA)	mr9.1.1.1	-
Yealink	CP860	mr5.2.1.1	-
Yealink	SIP-T19P	mr3.7.1.1	mr10.5.1.1
Yealink	SIP-T20P	mr3.7.1.1	mr10.5.1.1
Yealink	SIP-T21P	mr3.7.1.1	mr10.5.1.1
Yealink	SIP-T22P	mr3.7.1.1	mr10.5.1.1
Yealink	SIP-T23P	mr3.7.1.1	-
Yealink	SIP-T23G	mr3.7.1.1	-
Yealink	SIP-T26P	mr3.7.1.1	mr10.5.1.1
Yealink	SIP-T28P	mr3.7.1.1	mr10.5.1.1
Yealink	SIP-T28P + EXP39	mr3.8.1.1	mr10.5.1.1
Yealink	SIP-T28P + two EXP39	mr3.8.1.1	mr10.5.1.1
Yealink	SIP-T32G	mr3.7.1.1	mr10.5.1.1
Yealink	SIP-T38G	mr3.7.1.1	mr10.5.1.1
Yealink	SIP-T41P	mr3.7.1.1	mr10.5.1.1
Yealink	SIP-T42G	mr3.7.1.1	mr10.5.1.1

Copyright © Sipwise GmbH - All rights reserved

Vendor	Model	Available from release	Deprecated from release
Yealink	SIP-T46G	mr3.7.1.1	mr10.5.1.1
Yealink	SIP-T48G	mr3.7.1.1	mr10.5.1.1
Yealink	SIP-T53	mr8.5.1.1	-
Yealink	SIP-T54W	mr8.5.1.1	-
Yealink	SIP-T57W	mr8.5.1.1	-
Yealink	W52P	mr3.7.1.6	-

NOTE Deprecated phones are phones for which the device manufacturer has dropped hardware and software support entirely. These are still available for autoprovisioning, but will no longer be supported and potentially removed in a future version of NGCP

Alcatel-Lucent Enterprise (ALE) Devices

Model	IPv6	TLS	SRTP	Auto provisioning	Private Line	Shared Line	Busy Lamp	Speed Dial	Extension Boards
8008	Y	Y	Y	redirect	1-2	0	1-6	N	N
8018	Y	Y	Y	redirect	1-2	0	1-6	N	N
8028	Y	Y	Y	redirect	1-2	0	1-6	N	N
8058	Y	Y	Y	redirect	1-4	1	1-10	N	Y
8068	Y	Y	Y	redirect	1-4	1	1-10	N	Y
8078	Y	Y	Y	redirect	1-4	1	1-10	N	Y
AOM10	N	N	N	device	1	0	1	Y	Y
AOM40	N	N	N	device	1	0	1	Y	Y
AOMEL	N	N	N	device	1	0	1	Y	Y
M3	Y	Y	Y	redirect	1-8	0	1-20	N	N
M5	Y	Y	Y	redirect	1-8	0	1-28	N	N
M7	Y	Y	Y	redirect	1-8	0	1-28	N	N
H2P	Y	Y	Y	redirect	1-2	0	0	Y	N
H3G	Y	Y	Y	redirect	1-3	0	1-8	Y	N
H6	Y	Y	Y	redirect	1-4	0	1-12	Y	N

Audiocodes Devices

Model	IPv6	TLS	SRTP	Auto provisioning	Private Line	Shared Line	Busy Lamp	Speed Dial
Mediant800	Y	Y	Y	dhcp	1	0	0	N

Copyright © Sipwise GmbH - All rights reserved

Cisco Devices

IP Phones

Model	IPv6	TLS	SRTP	Auto provisioning	Private Line	Shared Line	Busy Lamp	Extension Boards
SPA301	N	Y	Y	http	1	1	0	N
SPA303	N	Y	Y	http	1-3	1-3	1-2	N
SPA501G	N	Y	Y	http	1-8	1-8	1-7	N
SPA502G	N	Y	Y	http	1	1	0	N
SPA512G	N	N	Y	http	1	1	0	N
SPA504G	N	Y	Y	http	1-4	1-4	1-3	2
SPA514G	N	N	Y	http	1-4	1-4	1-3	N
SPA508G	N	Y	Y	http	1-8	1-8	1-7	N
SPA509G	N	Y	Y	http	1-12	1-12	1-11	N
SPA525G	N	Y	N	http	1-5	1-5	1-4	N

Analog Adapters

Model	IPv6	TLS	SRTP	Auto provisioning	Private Line	Shared Line	Busy Lamp
SPA232D	N	Y	Y	http	1-6	0	0
ATA112	Y	Y	Y	http	1-2	0	0
ATA122	Y	Y	Y	http	1-2	0	0

Extension Boards

Model	Ports	Buttons	Busy Lamp	Supported phones
SPA500S	2	32	1-32	SPA500

Grandstream Devices

Analog Adapters

Model	IPv6	TLS	SRTP	Auto provisioning	Private Line	Shared Line	Busy Lamp
HT814	N	Y	Y	redirect	4	N	N
GXW-4008	N	Y	Y	redirect	8	N	N
GXW-4216	N	Y	Y	redirect	16	N	N

Innovaphone Devices

IP Phones

Copyright © Sipwise GmbH - All rights reserved

Model	IPv6	TLS	SRTP	Auto provisioning	Private Line	Shared Line	Busy Lamp	Extension Boards
IP232	N	Y	Y	dhcp	1	0	1-16	2
IP222	N	Y	Y	dhcp	1	0	1-16	2
IP240	N	N	N	dhcp	1	0	1-15	2
IP111	N	Y	Y	dhcp	1	0	1-16	0

Analog Adapters

Model	IPv6	TLS	SRTP	Auto provisioning	Private Line	Shared Line	Busy Lamp
IP22	N	Y	Y	dhcp	1	0	0

Extension Boards

Model	Ports	Buttons	Busy Lamp	Supported phones
IP2X2X	2	64	1-32	IP2x2
IP230-X	2	30	1-30	IP230

Panasonic Devices

IP Phones

Model	IPv6	TLS	SRTP	Auto provisioning	Private Line	Shared Line	Busy Lamp	Extension Boards
KX-UT113	N	N	N	redirect	1-2	1-2	0	N
KX-UT123	N	N	N	redirect	1-2	1-2	0	N
KX-UT133	N	N	N	redirect	1-4	1-4	1-23	N
KX-UT136	N	N	N	redirect	1-4	1-4	1-23	N
KX-UT248	N	N	Y	redirect	1-6	1-6	1-23	N
KX-TGP600	Y	Y	Y	redirect	1-8	N	N	N
KX-HDV330	Y	Y	Y	redirect	1-12	Y	Y	N
KX-HDV230	Y	Y	Y	redirect	1-6	Y	Y	N
KX-HDV130	Y	Y	Y	redirect	1-2	Y	Y	N

Polycom Devices

IP Phones

Model	IPv6	TLS	SRTP	Auto provisioning	Private Line	Shared Line	Busy Lamp	Extension Boards
VVX300	N	N	Y	redirect	1-6	1-6	Y	N
VVX400	N	N	Y	redirect	1-12	1-12	Y	N

Copyright © Sipwise GmbH - All rights reserved

Model	IPv6	TLS	SRTP	Auto provisioning	Private Line	Shared Line	Busy Lamp	Extension Boards
VVX500	N	N	Y	redirect	1-12	1-12	Y	N

SNOM Devices

IP Phones

Model	IPv6	TLS	SRTP	Auto provisioning	Private Line	Shared Line	Busy Lamp	Extension Boards
D715	Y	Y	Y	redirect	1-4	N	Y	Y
D717	Y	Y	Y	redirect	1-6	N	Y	Y
D735	Y	Y	Y	redirect	1-12	N	Y	Y
D785	Y	Y	Y	redirect	1-12	N	Y	Y

Base Stations

Model	IPv6	TLS	SRTP	Auto provisioning	Private Line	Shared Line	Busy Lamp	Extension Boards
M900	Y	Y	Y	redirect	1-4	N	N	N

Handsets and Headsets

Model	IPv6	TLS	SRTP	Auto provisioning	Private Line	Shared Line	Busy Lamp	Extension Boards
M65	Y	Y	Y	redirect	1	N	N	N
M70	Y	Y	Y	redirect	1	N	N	N
M80	Y	Y	Y	redirect	1	N	N	N
M85	Y	Y	Y	redirect	1	N	N	N
M90	Y	Y	Y	redirect	1	N	N	N
A190	Y	Y	Y	redirect	1	N	N	N

Yealink Devices

IP Phones

Model	IPv6	TLS	SRTP	Auto provisioning	Private Line	Shared Line	Busy Lamp	Extension Boards
CP860	Y	Y	Y	redirect	1	N	N	N
SIP-T19P	Y	Y	Y	redirect	1	1	0	N
SIP-T20P	Y	Y	Y	redirect	1	1	0	N
SIP-T21P	Y	Y	Y	redirect	1-2	1-2	1	N
SIP-T22P	Y	Y	Y	redirect	1-3	1-3	1-2	N

Copyright © Sipwise GmbH - All rights reserved

Model	IPv6	TLS	SRTP	Auto provisioning	Private Line	Shared Line	Busy Lamp	Extension Boards
SIP-T23P	Y	Y	Y	redirect	1-3	1-3	1-2	N
SIP-T23G	Y	Y	Y	redirect	1-3	1-3	1-2	N
SIP-T26P	Y	Y	Y	redirect	1-3	1-3	1-12	N
SIP-T28P	Y	Y	Y	redirect	1-6	1-6	1-15	2
SIP-T32G	Y	Y	Y	redirect	1-3	1-3	1-2	N
SIP-T38G	Y	Y	Y	redirect	1-6	1-6	1-15	N
SIP-T41P	Y	Y	Y	redirect	1-3	1-3	1-14	N
SIP-T42G	Y	Y	Y	redirect	1-3	1-3	1-14	N
SIP-T46G	Y	Y	Y	redirect	1-6	1-6	1-26	N
SIP-T48G	Y	Y	Y	redirect	1-6	1-6	1-28	N
W52P	N	Y	Y	redirect	1-5	1-5	0	N

8.1.17. Phone features

Cisco phones

SPA301

1) Soft keys

Not available.

2) Hard keys

- vm
- hold/unhold

3) Line keys

Not available.

4) VSC

- directed pickup
- park/unpark

SPA303

1) Soft keys

Idle:

redial	lcr	dir	dnd >
< cfwd	unpark		

Copyright © Sipwise GmbH - All rights reserved

Idle with missed calls:

lcr			miss

Call:

hold	endCall	conf	xfer >
< bxfer	park		

Call on hold:

resume	endCall	newCall	redial >
< dir	cfwd	dnd	

Ringing:

answer	ignore		

2) Hard keys

- vm
- hold/unhold

3) Line keys

- BLF monitoring
- directed pickup

4) VSC

- directed pickup

SPA501G

1) Soft keys

Idle:

redial	lcr	dir	dnd >
< cfwd	unpark		

Idle with missed calls:

lcr			miss

Call:

hold/resume	endCall	conf	xfer >
< bxfer	park		

Copyright © Sipwise GmbH - All rights reserved

Call on hold:

resume	endCall	newCall	redial >
< dir	cfwd	dnd	

Ringing:

answer	reject		

2) Hard keys

- vm
- hold/unhold

3) Line keys

- BLF monitoring
- directed pickup

4) VSC

- directed pickup

SPA502G

1) Soft keys

Idle:

redial	lcr	dir	dnd >
< cfwd	unpark		

Idle with missed calls:

lcr			miss

Call:

hold/resume	endCall	conf	xfer >
< bxfer	park		

Call on hold:

resume	endCall	newCall	redial >
< dir	cfwd	dnd	

Ringing:

answer	reject		

Copyright © Sipwise GmbH - All rights reserved

2) Hard keys

- vm
- hold/unhold

3) Line keys

Not available.

4) VSC

- directed pickup

SPA504G

1) Soft keys

Idle:

redial	lcr	dir	dnd >
< cfwd	unpark		

Idle with missed calls:

lcr			miss

Call:

hold/resume	endCall	conf	xfer >
< bxfer	park		

Call on hold:

resume	endCall	newCall	redial >
< dir	cfwd	dnd	

Ringing:

answer	reject		

2) Hard keys

- vm
- hold/unhold

3) Line keys

- BLF monitoring
- directed pickup

Copyright © Sipwise GmbH - All rights reserved

4) VSC

- directed pickup

SPA512G

1) Soft keys

Idle:

redial	lcr	dir	dnd >
< cfwd	unpark		

Idle with missed calls:

lcr			miss

Call:

hold/resume	endCall	conf	xfer >
< bxfer	park		

Call on hold:

resume	endCall	newCall	redial >
< dir	cfwd	dnd	

Ringing:

answer	reject		

2) Hard keys

- vm
- hold/unhold

3) Line keys

Not available.

4) VSC

- directed pickup

SPA514G

1) Soft keys

Idle:

Copyright © Sipwise GmbH - All rights reserved

redial	lcr	dir	dnd >
< cfwd	unpark		

Idle with missed calls:

lcr			miss

Call:

hold/resume	endCall	conf	xfer >
< bxfer	park		

Call on hold:

resume	endCall	newCall	redial >
< dir	cfwd	dnd	

Ringing:

answer	reject		

2) Hard keys

- vm
- hold/unhold

3) Line keys

- BLF monitoring
- directed pickup

4) VSC

- directed pickup

SPA509G

1) Soft keys

Idle:

redial	lcr	dir	dnd >
< cfwd	unpark		

Idle with missed calls:

lcr			miss

Call:

Copyright © Sipwise GmbH - All rights reserved

hold/resume	endCall	conf	xfer >
< bxfer	park		

Call on hold:

resume	endCall	newCall	redial >
< dir	cfwd	dnd	

Ringing:

answer	reject		

2) Hard keys

- vm
- hold/unhold

3) Line keys

- BLF monitoring
- directed pickup

4) VSC

- directed pickup

SPA508G

1) Soft keys

Idle:

redial	lcr	dir	dnd >
< cfwd	unpark		

Idle with missed calls:

lcr			miss

Call:

hold/resume	endCall	conf	xfer >
< bxfer	park		

Call on hold:

resume	endCall	newCall	redial >
< dir	cfwd	dnd	

Copyright © Sipwise GmbH - All rights reserved

Ringing:

answer	reject		

2) Hard keys

- vm
- hold/unhold

3) Line keys

- BLF monitoring
- directed pickup

4) VSC

- directed pickup

SPA525G

1) Soft keys

Idle:

Redial	call Rtn	Directory	DND >
< Forward	Unpark		

Idle with missed calls:

Call Rtn			Miss

Call:

Hold	End Call	Conf	Transfer >
BlindXfer	Park		

Call on hold:

Resume	EndCall	EewCall	Redial >
< Directory	Forward	DND	

Ringing:

Answer	Ignore		

2) Hard keys

- vm
- hold/unhold

Copyright © Sipwise GmbH - All rights reserved

3) Line keys

- BLF monitoring
- directed pickup

4) VSC

- directed pickup

Yealink phones

T19P

1) Soft keys

Idle:

History		DND	Menu

Idle with missed calls:

Exit			View

Call:

Tran	Hold	Conf	Cancel

Call on hold:

Tran	Resume	NewCall	Cancel

Ringing:

Answer	FWD	Silence	Reject

2) Hard keys

- vm
- redial
- transfer

3) Line keys

Not available.

4) VSC

- transfer park
- directed pick up
- park/unpark

Copyright © Sipwise GmbH - All rights reserved

T20P

1) Soft keys

Idle:

History		DND	Menu

Idle with missed calls:

Exit			View

Call:

Tran	Hold	Conf	Cancel

Call on hold:

Tran	Resume	NewCall	Cancel

Ringing:

Answer	FWD	Silence	Reject

2) Hard keys

- vm
- redial
- transfer

3) Line keys

- BLF monitoring
- directed pickup

4) VSC

- transfer park
- park/unpark

T21P

1) Soft keys

Idle:

History		DND	Menu

Idle with missed calls:

Copyright © Sipwise GmbH - All rights reserved

Exit			View

Call:

Tran	Hold	Conf	Cancel

Call on hold:

Tran	Resume	NewCall	Cancel

Ringing:

Answer	FWD	Silence	Reject

2) Hard keys

- vm
- redial
- transfer

3) Line keys

- BLF monitoring
- directed pickup

4) VSC

- transfer park
- park/unpark

T22P

1) Soft keys

Idle:

History		DND	Menu

Idle with missed calls:

Exit			View

Call:

Tran	Hold	Conf	Cancel

Call on hold:

Tran	Resume	NewCall	Cancel

Copyright © Sipwise GmbH - All rights reserved

Ringing:

Answer	FWD	Silence	Reject

2) Hard keys

- vm
- redial
- transfer

3) Line keys

- BLF monitoring
- directed pickup

4) VSC

- park/unpark
- transfer park

T23P

1) Soft keys

Idle:

History		DND	Menu

Idle with missed calls:

Exit			View

Call:

Tran	Hold	Conf	Cancel

Call on hold:

Tran	Resume	NewCall	Cancel

Ringing:

Answer	FWD	Silence	Reject

2) Hard keys

- vm
- redial
- transfer

Copyright © Sipwise GmbH - All rights reserved

3) Line keys

- BLF monitoring
- directed pickup

4) VSC

- park/unpark
- transfer park

T23G

1) Soft keys

Idle:

History	Dir	DND	Menu

Idle with missed calls:

Exit			View

Call:

Tran	Hold	Conf	EndCall

Call on hold:

Tran	Resume	NewCall	EndCall

Ringing:

Answer	FWD		Reject

2) Hard keys

- vm
- redial
- transfer

3) Line keys

- BLF monitoring
- directed pickup

4) VSC

- unpark
- transfer park

Copyright © Sipwise GmbH - All rights reserved

T26P

1) Soft keys

Idle:

History		DND	Menu

Idle with missed calls:

Exit			View

Call:

Tran	Hold	Conf	Cancel

Call on hold:

Tran	Resume	NewCall	Cancel

Ringing:

Answer	FWD	Silence	Reject

2) Hard keys

- vm
- redial
- transfer

3) Line keys

- BLF monitoring
- directed pickup

4) VSC

- unpark
- transfer park

T28P

1) Soft keys

Idle:

History		DND	Menu

Idle with missed calls:

Copyright © Sipwise GmbH - All rights reserved

Exit			View

Call:

Tran	Hold	Conf	Cancel

Call on hold:

Tran	Resume	NewCall	Cancel

Ringing:

Answer	FWD	Silence	Reject

2) Hard keys

- vm
- redial
- transfer

3) Line keys

- BLF monitoring
- directed pickup

4) VSC

- park/unpark
- transfer park

T32G

1) Soft keys

Idle:

History		DND	Menu

Idle with missed calls:

Exit			View

Call:

Tran	Hold	Conf	Cancel

Call on hold:

Tran	Resume	NewCall	Cancel

Copyright © Sipwise GmbH - All rights reserved

Ringing:

Answer	FWD	Silence	Reject

2) Hard keys

- vm
- redial
- transfer

3) Line keys

- BLF monitoring
- directed pickup

4) VSC

- unpark
- transfer park

T38G

1) Soft keys

Idle:

History		DND	Menu

Idle with missed calls:

Exit			View

Call:

Tran	Hold	Conf	Cancel

Call on hold:

Tran	Resume	NewCall	Cancel

Ringing:

Answer	FWD	Silence	Reject

2) Hard keys

- vm
- redial
- transfer

Copyright © Sipwise GmbH - All rights reserved

3) Line keys

- BLF monitoring
- directed pickup

4) VSC

- unpark
- transfer park

T41P

1) Soft keys

Idle:

History		DND	Menu

Idle with missed calls:

Exit			View

Call:

Tran	Hold	Conf	Cancel

Call on hold:

Tran	Resume	NewCall	Cancel

Ringing:

Answer	FWD	Silence	Reject

2) Hard keys

- vm
- redial
- transfer

3) Line keys

- BLF monitoring
- directed pickup

4) VSC

- park/unpark
- transfer park

Copyright © Sipwise GmbH - All rights reserved

T42G

1) Soft keys

Idle:

History		DND	Menu

Idle with missed calls:

Exit			View

Call:

Tran	Hold	Conf	Cancel

Call on hold:

Tran	Resume	NewCall	Cancel

Ringing:

Answer	FWD	Silence	Reject

2) Hard keys

- vm
- redial
- transfer

3) Line keys

- BLF monitoring
- directed pickup

4) VSC

- park/unpark
- transfer park

T46G

1) Soft keys

Idle:

History		DND	Menu

Idle with missed calls:

Copyright © Sipwise GmbH - All rights reserved

Exit			View

Call:

Tran	Hold	Conf	Cancel

Call on hold:

Tran	Resume	NewCall	Cancel

Ringing:

Answer	FWD	Silence	Reject

2) Hard keys

- vm
- redial
- transfer

3) Line keys

- BLF monitoring
- directed pickup

4) VSC

- park/unpark
- transfer park

T48G

1) Soft keys

Idle:

History		DND	Menu

Idle with missed calls:

Exit			View

Call:

Tran	Hold	Conf	Cancel

Call on hold:

Tran	Resume	NewCall	Cancel

Copyright © Sipwise GmbH - All rights reserved

Ringing:

Answer	FWD	Silence	Reject

2) Hard keys

- vm
- redial
- transfer

3) Line keys

- BLF monitoring
- directed pickup

4) VSC

- park/unpark
- transfer park

W52P

1) Soft keys

Idle:

History	Line

Idle with missed calls:

Exit	View

Call:

Ext. Call	Options

Call on hold:

Resume	Line

Ringing:

Accept	

2) Hard keys

- vm
- redirect

3) VSC

Copyright © Sipwise GmbH - All rights reserved

- park/unpark
- transfer park

Panasonic phones

KX-UT113

1) Soft keys

Idle:

Settings	Call Log	Phone book	

Call:

Blind		Phone book	

Call on hold:

	Call Log	Phone book	

Ringing:

Answer		Reject	

2) Hard keys

- vm
- forward/dnd
- hold/unhold
- redial
- recall
- transfer
- conf

3) Line keys

Not available.

4) VSC

- park/unpark
- transfer park

KX-UT123

1) Soft keys

Idle:

Copyright © Sipwise GmbH - All rights reserved

Settings	Call Log	Phone book	

Call:

Blind		Phone book	

Call on hold:

	Call Log	Phone book	

Ringing:

Answer		Reject	

2) Hard keys

- vm
- forward/dnd
- hold/unhold
- redial
- recall
- transfer
- conf

3) Line keys

Not available.

4) VSC

- park/unpark
- transfer park

KX-UT133

1) Soft keys

Idle:

Settings	Call Log	Phone book	

Call:

Blind		Phone book	

Call on hold:

	Call Log	Phone book	

Copyright © Sipwise GmbH - All rights reserved

Ringing:

Answer		Reject	

2) Hard keys

- vm
- forward/dnd
- hold/unhold
- redial
- recall
- transfer
- conf

3) Line keys

- BLF monitoring
- directed pickup

4) VSC

- unpark
- transfer park

KX-UT136

1) Soft keys

Idle:

Settings	Call Log	Phone book	

Call:

Blind		Phone book	

Call on hold:

	Call Log	Phone book	

Ringing:

Answer		Reject	

2) Hard keys

- vm
- forward/dnd

Copyright © Sipwise GmbH - All rights reserved

- hold/unhold
- redial
- recall
- transfer
- conf

3) Line keys

- BLF monitoring
- directed pickup

4) VSC

- park/unpark
- transfer park

KX-UT248

1) Soft keys

Idle:

Settings	Call Log	Phone book	

Call:

Blind		Phone book	

Call on hold:

	Call Log	Phone book	

Ringing:

Answer		Reject	

2) Hard keys

- vm
- forward/dnd
- hold/unhold
- redial
- recall
- transfer
- conf

3) Line keys

Copyright © Sipwise GmbH - All rights reserved

- BLF monitoring
- directed pickup

4) VSC

- park/unpark
- transfer park

Innovaphone

IP222

1) Soft keys

Idle:

Setup	All Calls	Home	Calls	My favorites	Phonebook

Call:

Hold	Transfer	Park	Cancel

Call on hold:

Resume	Transfer	Park	Cancel

Ringing:

Answer	Transfer	Silence	Reject

2) Hard keys

- hold
- redial

3) Line keys

- BLF monitoring

4) VSC

- unpark
- transfer park

IP232

1) Soft keys

Idle:

Setup	All Calls	Home	Calls	My favorites	Phonebook

Copyright © Sipwise GmbH - All rights reserved

Call:

Hold	Transfer	Park	Cancel

Call on hold:

Resume	Transfer	Park	Cancel

Ringing:

Answer	Transfer	Silence	Reject

2) Hard keys

- hold
- redial

3) Line keys

- BLF monitoring

4) VSC

- unpark
- transfer park

IP111

1) Soft keys

Idle:

Setup	All Calls	Home	Calls	My favorites	Phonebook

Call:

Hold	Transfer	Park	Cancel

Call on hold:

Resume	Transfer	Park	Cancel

Ringing:

Answer	Transfer	Silence	Reject

2) Hard keys

- hold
- redial

Copyright © Sipwise GmbH - All rights reserved

3) Line keys

- BLF monitoring

4) VSC

- unpark
- transfer park

IP240

1) Soft keys

Not available.

2) Hard keys

- hold
- redial
- conference
- dnd
- forward

3) Line keys

- BLF monitoring

4) VSC

- transfer park
- unpark

8.1.18. Shared line appearance

In PBX environment, shared line appearance is supported for PBX subscribers. In comparison to the private line, subscriber registering for the shared line will, immediately after the successful registration, subscribe for Call-Info event. This subscribe is challenged for authentication and if the credentials are provided, subscriber is notified that the subscription is active. In the respective NOTIFY message, this is reflected in Subscription-State header set to active. NOTIFY also contains information about the status of the shared line in Call-Info header. If the appearance is not used, Call-Info header will describe the state as idle. In the NOTIFY message, this is reflected as “appearance-index=*;appearance-state=idle”.

If there is incoming call to the subscriber, the appearance index is created after the call is accepted and state will be set to active. Call-Info header will contain “appearance-index=1; appearance-state=active”. After call is finished and appearance is not used elsewhere, appearance index is removed and state is set to idle. In the case of outgoing call, subscription to line-seize event is required to be able to dial. Before dialing can be started, SUBSCRIBE to line-seize is sent. Consequently, subscriber receives NOTIFY for line-seize with active subscription state. Call-info subscription is updated accordingly, appearance is created and its state is set to seized. As soon as the call starts ringing, Call-Info status is updated to progressing and line-seize subscription is set to terminated with “reason=noresource” in Subscription-State header. When the call is accepted, Call-Info status is changed to active and set

 Copyright © Sipwise GmbH - All rights reserved

again to idle when call is finished. Also, the appearance index is removed.

8.2. Sipwise sip:phone App (SIP client)

Sipwise provides a commercial Unified Communication Client for full end-to-end integration of voice, video, chat and presence features. The application is called sip:phone and is a mobile app for iOS and Android.

The clients are fully brandable to the customer's corporate identity. They are not part of the standard delivery and need to be licensed separately. This handbook discusses the mobile client in details.

8.2.1. Zero Config Launcher

Part of the mobile apps is a mechanism to sign up to the service via a 3rd party website, which is initiated on the login screen and rendered within the app. During the sign-up process, the 3rd party service is supposed to create a new account and subscriber in Sipwise C5 (e.g. automatically via the API) and provide the end user with the access credentials.

The mobile apps come with a zero config mechanism to simplify the end-customer log in using these credentials (especially ruling out the need to manually enter them). It makes it possible to deliver the access credentials via a side channel (e.g. Email, SMS) packed into a URL. The user only clicks the URL, and it automatically launches the app with the correct credentials. The following picture shows the overall workflow.

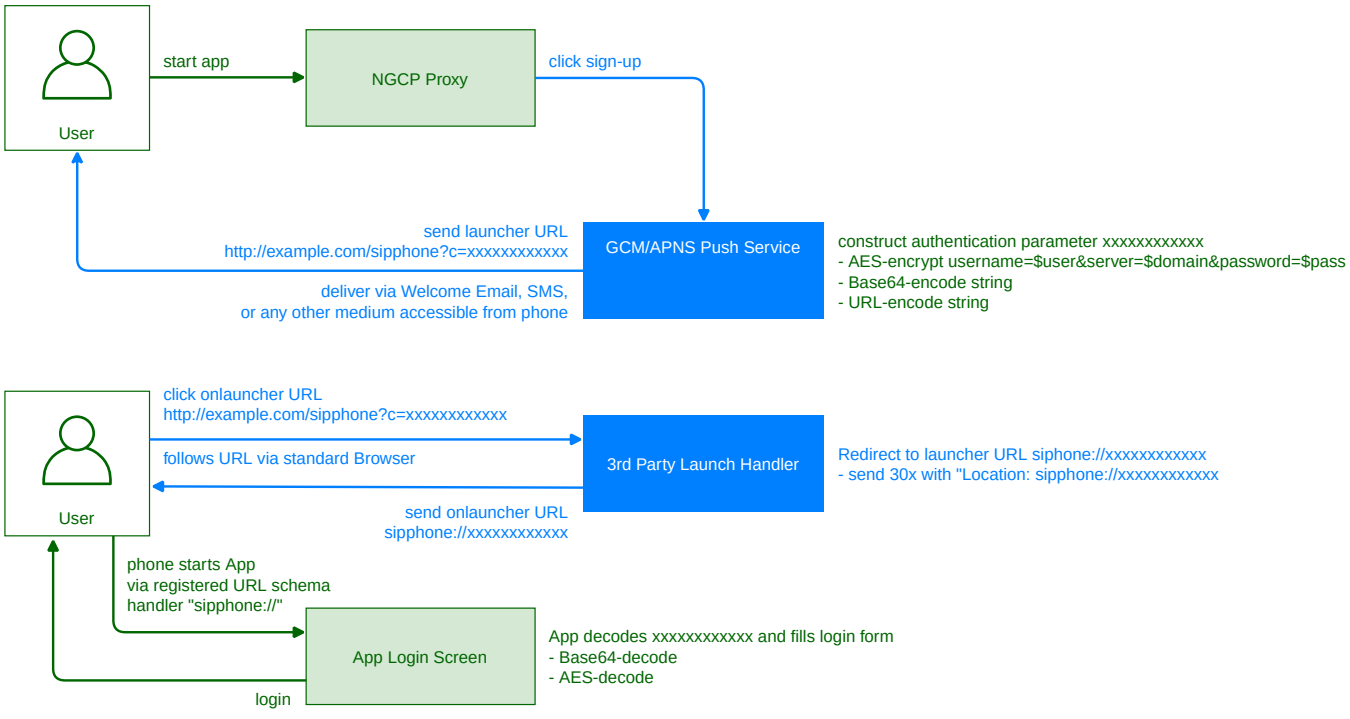


Figure 169. Provisioning Push Workflow

There are two components provided by a 3rd party system. One is the *3rd Party Sign-Up Form*, and the other is the *3rd Party Launch Handler*. The purpose of these components is to allow an end customer to open a link with the access credentials via the sip:phone app.

Copyright © Sipwise GmbH - All rights reserved

3rd Party Sign-Up Form

The 3rd Party Sign-Up Form is a website the app shows to the end user when he taps the sign-up link on the app *Login Screen*. There, the end customer usually provides his contact details like name, address, phone number and email address, etc. After validation, the website creates an account and a subscriber in Sipwise C5 via the API.

After successfully creating the account and the subscriber, this site needs to construct a specially crafted URL, which is sent back to the end customer via a side channel. Ideally, this channel would be an SMS if you want to verify the end customer's mobile number, or an email if you want to check the email address.

The sip:phone app registers a URL schema handler for URLs starting with sipphone://. If you start such a link, the app performs a Base64 decoding of the string right after the sipphone:// prefix and then decrypts the resulting binary string via AES using the keys defined during the branding step. The resulting string is supposed to be

username=$user&server=$domain&password=$password&fsurl=$fsurl&fsttl=$fsttl&country=$country.

Therefore, the *3rd Party Sign-Up Form* needs to construct this string using the credentials defined while creating the subscriber via Sipwise C5 API, then encrypt it via AES, and finally perform a Base64 encoding of the result.

The parameters of the string are as follows:

- username: The SIP username for the login (e.g. *testuser*)
- password: The SIP password for the login (e.g. *testpass*)
- server: The server string containing either the IP address or a domain resolving via DNS SRV or A records to the load-balancer IP of the deployment (e.g. *sip.example.org*)
- fsurl: The filesharing URL for the apps to upload data (e.g. *https://sip.example.org:1446/rtc/fileshare/uploads*)
- fsttl: The number of seconds a shared file is valid by default (e.g. *1209600* for 14 days)
- country: The ISO country code for the app to normalize phone numbers (e.g. *de* for Germany)

An example Perl code performs encoding of such a string. The AES key and initialization vector ($key and $iv) are the standard values of the sip:phone app and should work until you specified other values during the branding process.

Copyright © Sipwise GmbH - All rights reserved

```
#!/usr/bin/perl -w
use strict;
use Crypt::Rijndael;
use MIME::Base64;
use URI::Escape;

my $key = 'iBmTdavJ8joPW3HO';
my $iv = 'tww21lQe6cmywrp3';

my $plain = do { local $/; <> };
# pkcs#5 padding to 16 bytes blocksize
my $pad = 16 - (length $plain) % 16;
$plain .= pack('C', $pad) x $pad;

my $cipher = Crypt::Rijndael->new(
        $key,
        Crypt::Rijndael::MODE_CBC()
);
$cipher->set_iv($iv);
my $encrypted = $cipher->encrypt($plain);
# store b64-encoded string and print to STDOUT
my $b64 = encode_base64($encrypted, '');
print $b64, "\n";
# print to STDOUT using URL escaping also
print uri_escape($b64), "\n";
```

This snippet takes a string from STDIN, encrypts it via AES, encodes it via Base64 and sends the result to STDOUT. It also writes the second line with the same string, but this time, the URL is escaped. To test it, you would run it as follows on a shell, granted it's stored at /path/to/encrypt.pl.

```
echo -n
'username=testuser&server=example.org&password=testpass&fsurl=https://ex
ample.org:1446/rtc/fileshare/uploads&fsttl=3600&country=at' \
  | /path/to/encrypt.pl
```

This command would result in the output strings like CI8VN8toaE40w8E4OH2rAuFj3Qev9QdLI/Wv/VaBCVK2yNkBZjxE9eafXkkrQfmYdeu01PquS5P40zhUq8Mfjg== and like CI8VN8toaE40w8E4OH2rAuFj3Qev9QdLI%2FWv%2FVaBCVK2yNkBZjxE9eafXkkrQfmYdeu01PquS5P40zhUq8Mfjg%3D%3D. The sip:phone can use the former string to automatically fill in the login form of the Login Screen if started via a Link like sipphone://CI8VN8toaE40w8E4OH2rAuFj3Qev9QdLI/Wv/VaBCVK2yNkBZjxE9eafXkkrQfmYdeu01PquS5P40zhUq8Mfjg==.

Here is the same code in PHP.

Copyright © Sipwise GmbH - All rights reserved

```
#!/usr/bin/php
<?php
$key = "iBmTdavJ8joPW3HO";
$iv = "tww21lQe6cmywrp3";

$clear = fgets(STDIN);
$cipher = fnEncrypt($clear, $key, $iv);

echo $cipher, "\n";
echo urlencode($cipher), "\n";

function fnEncrypt($clear, $key, $iv) {
        $pad = 16 - strlen($clear) % 16;
        $clear .= str_repeat(pack('C', $pad), $pad);
        return rtrim(base64_encode(mcrypt_encrypt(
                MCRYPT_RIJNDAEL_128, $key, $clear,
                MCRYPT_MODE_CBC, $iv)), "\0");
}
?>
```

Similar to the Perl code, you can call it like this:

```
echo -n
'username=testuser&server=example.org&password=testpass&fsurl=https://ex
ample.org:1446/rtc/fileshare/uploads&fsttl=3600&country=at' \
  | /path/to/encrypt.php
```

However, a URL with the sipphone:// schema is not displayed as a link in an SMS or an Email client and thus can not be clicked by the end customer, so you need to make a detour via a regular http:// URL. To do so, you need a *3rd Party Launch Handler* to trick the phone to open such a link.

Therefore, that the *3rd Party Sign-Up Form* needs to return a link containing a URL pointing to the *3rd Party Launch Handler* and pass the URL escaped string gathered above to the client via an SMS or an Email. Since it is the regular http:// link, it is clickable on the phone and can be launched from virtually any client (SMS, Email, etc.), which correctly renders an HTML link.

A possible SMS sent to the end customer (via the phone number entered in the sign-up from) could, therefore, look as follows (trying to stay below 140 chars).

```
http://example.org/p?c=CI8VN8toaE40w8E4OH2rAuFj3Qev9QdLI
%2FWv%2FVaBCVK2yNkBZjxE9eafXkkrQfmYdeu01PquS5P40zhUq8Mfjg%3D%3D to
launch sipphone
```

An HTML Email could look like this:

Copyright © Sipwise GmbH - All rights reserved

```
Welcome to Example.org,
<a
href="http://www.example.org/sipphone?c=CI8VN8toaE40w8E4OH2rAuFj3Qev9QdL
I
%2FWv%2FVaBCVK2yNkBZjxE9eafXkkrQfmYdeu01PquS5P40zhUq8Mfjg%3D%3D">
click here
</a> to log in.
```

That way, you can do both: verify the contact details of the end customer, and send the end customer the login credentials in a secure manner.

3rd Party Launch Handler

The URL http://www.example.org/sipphone mentioned above can be any simple script, and its sole purpose is to send back a 301 Moved Permanently or 302 Moved Temporarily with a Location: sipphone://xxxxxxxxxxxx header to tell the phone to open this link via the sip:phone app. The *xxxxxxxxxxxx* is the plain (non-URL-escaped) string generated by the above script.

An example CGI script performing this task follows.

```
#!/usr/bin/perl -w
use strict;
use CGI;

my $q = CGI->new;
my $c = $q->param('c');
print CGI::redirect("sipphone://$c");
```

The script takes the URL parameter c from the URL http://www.example.org/sipphone?c=CI8VN8toaE40w8E4OH2rAuFj3Qev9QdLI%2FWv%2FVaBCVK2yNkBZjxE9eafXkkrQfmYdeu01PquS5P40zhUq8Mfjg%3D%3D crafted above and puts its content into a Location header using the sipphone:// schema, and finally sends a 301 Moved Permanently back to the phone.

The phone follows the redirect by opening the URL using the sip:phone app, which in turn decrypts the content and fills in the login form.

NOTE Future versions of Sipwise C5 will be shipped with this launch handler integrated into the system. Up until and including the version mr10.5.7, this script needs to be installed on any webserver manually.

8.2.2. Mobile Push Notification

The *mobile push* functionality provides the remote start of a mobile application on incoming calls via the Google FCM or the Apple APNS notification services. It enables you to offer your subscribers a modern and convenient service on mobile devices.

Copyright © Sipwise GmbH - All rights reserved

CAUTION

Although suspending an application on a phone and waking it up via the mobile push notification service extends battery life, the whole mobile push notification concept is the best effort framework provided by Apple and Google for iOS and Android respectively, and therefore does not guarantee 100% reliability.

Architecture

If the *mobile push* functionality is enabled and there are no devices registered for a subscriber, the call-flow looks as follows.

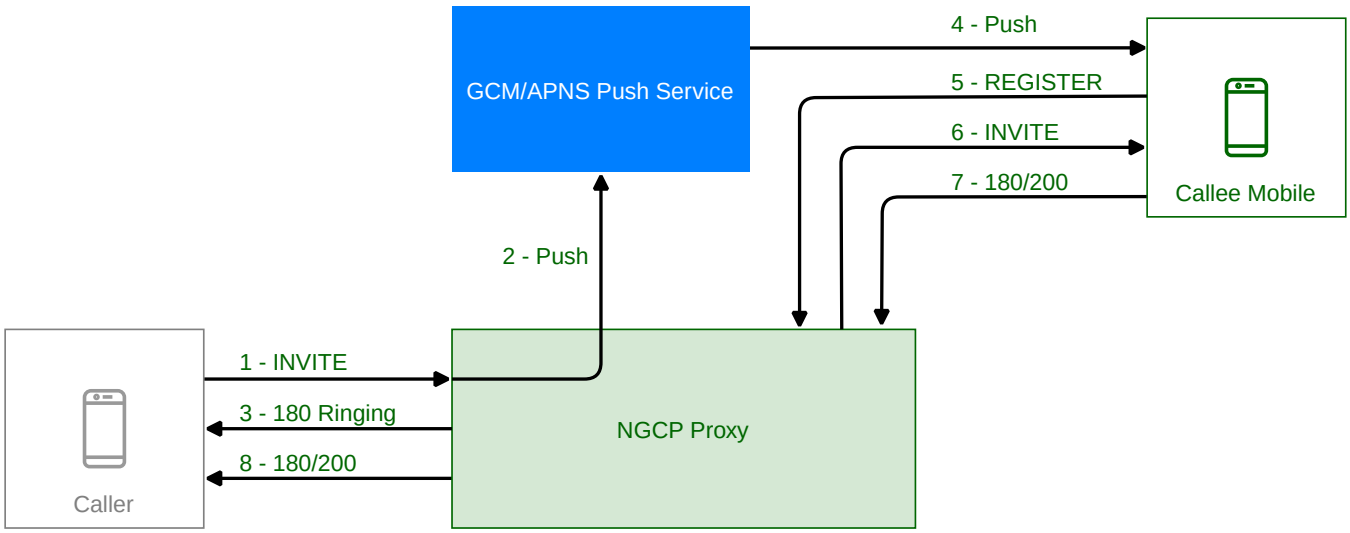


Figure 170. Mobile Push Workflow

1. The caller sends INVITE to the Proxy
2. The callee is offline, the Proxy sends the PUSH request to the GCM/APNS service
3. If configured, Proxy sends a 18x message to the caller to trigger remote ringing
4. The GCM/APNS service delivers the PUSH request to the callee
5. The callee accepts the PUSH request and confirms the mobile application start (unattended on Android), then the mobile application registers at the Proxy
6. The newly received registration triggers in the Proxy a sending of an INVITE to the callee's mobile device
7. The callee accepts the call
8. The response is sent back to the caller. Hence, the call setup is completed

In case of timeout (no registration notification within certain period of time), Proxy rejects the call request with an error.

Sipwise C5 allows to trigger mobile push notifications even if the callee has a device already registered, thus allowing to have a parallel forking to deskphone and mobile phone. In this case ff the *mobile push* functionality is enabled the call flow looks as follow.

Copyright © Sipwise GmbH - All rights reserved

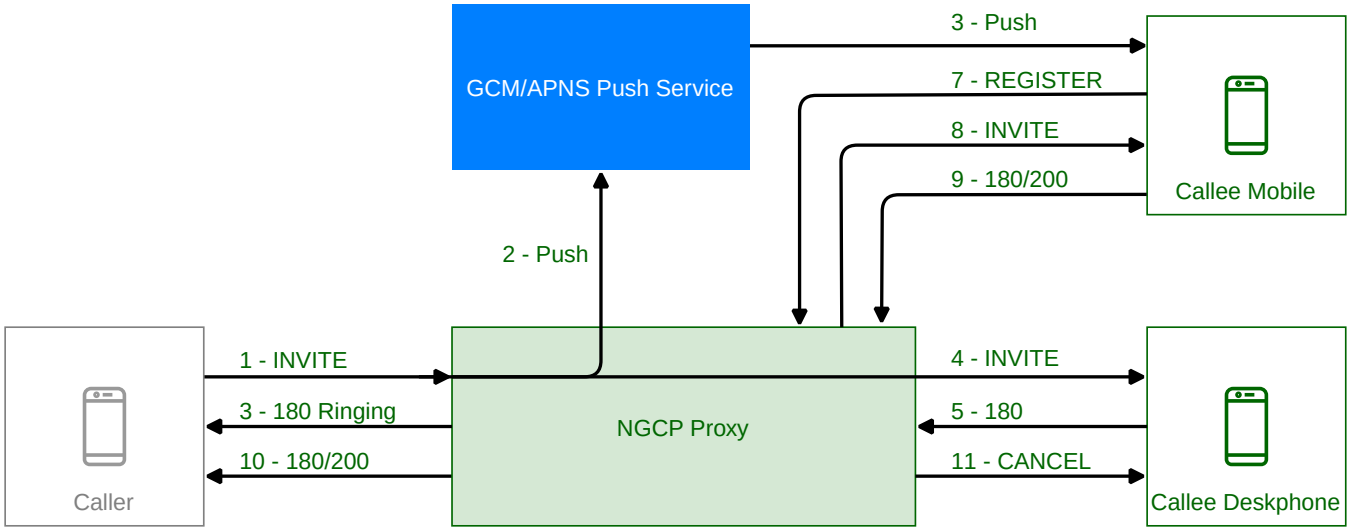


Figure 171. Parallel Mobile Push Workflow

1. The caller sends INVITE to the Proxy
2. The callee is online, the Proxy anyway sends the PUSH request to the GCM/APNS service
3. If configured, Proxy sends a 18x message to the caller to trigger remote ringing
4. At the same time Proxy sends the INVITE to the callee's registered device (deskphone)
5. The GCM/APNS service delivers the PUSH request to the callee
6. The callee's mobile device accepts the PUSH request and confirms the mobile application start (unattended on Android), then the mobile application registers at the Proxy
7. The newly received registration triggers in the Proxy a sending of an INVITE to the callee's mobile device
8. The callee accepts the call using one of its devices
9. The response is sent back to the caller. Hence, the call setup is completed
10. Proxy sends a CANCEL message to the other ringing device

The Configuration Checklist

Follow this checklist to make sure you've completed all the steps. If you miss anything, the service may not work as expected.

Name	Description	Link
Obtain a trusted SSL certificate from a CA	Required for either application	Obtain the Trusted SSL Certificate
Create an Apple developer account and enable the push notification service	For iOS mobile application	Create an Apple Account and Enable the Push Notification Service
Obtain the Apple certificate for the app	For iOS mobile application	Obtain an Apple SSL Certificate and a Private Key
Obtain the API key for the app from Google	For Android mobile application	Obtain the API Key for the App from Google

Copyright © Sipwise GmbH - All rights reserved

Name	Description	Link
Provide the required information to developers	It is required to make beta builds and publish the apps	Provide the Required Information to Developers
Adjust the configuration	Adjust the config.yml file and apply the changes (usually performed by Sipwise)	Adjust Sipwise C5 Configuration (Usually Performed by Sipwise)
Recheck your DNS Zone configuration	Check that the DNS Zone is correctly configured	Recheck Your DNS Zone Configuration
Add DNS SRV records	Create specific DNS SRV records for SIP and XMPP services	Add SRV Records to DNS
Check NTP configuration	Ensure that all your servers show exact time	Check NTP Configuration
Enable Apple/Google Mobile Push in the Admin Panel	It can be enabled for a domain or separate subscribers	Enable Apple/Google Mobile Push
Configure a mobile application	Check that subscribers can easily install and use your application	Perform Tests

Obtain the Trusted SSL Certificate

A *trusted* SSL certificate is required, and we suggest obtaining it before starting the configuration.

The mobile application uses respective iOS/Android libraries to establish a secure TLS connection with certain Sipwise C5 services, such as SIP/XMPP/pushd(https). A *signed* SSL certificate is required to guarantee the security of this connection.

Any Certificate Authority (CA) such as Verisign and others can provide you with the required trusted SSL certificate (a certificate and the key files) which you will use in the configuration below.

Create an Apple Account and Enable the Push Notification Service

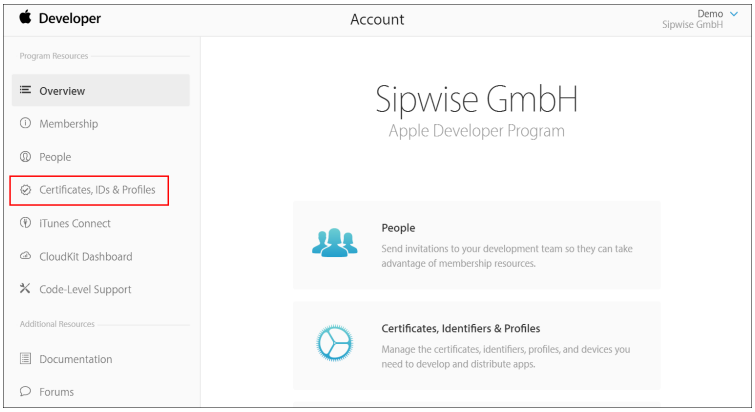
Below is a brief instruction on how to create an Apple account and enable the Push Notification Service in it. You may need to perform additional steps depending on your project.

NOTE You may only create an Apple account (step 1 below) and enroll into the Apple Developer Program (step 2 below) and Sipwise developers will do the rest. Still, you can perform all the steps by yourself.

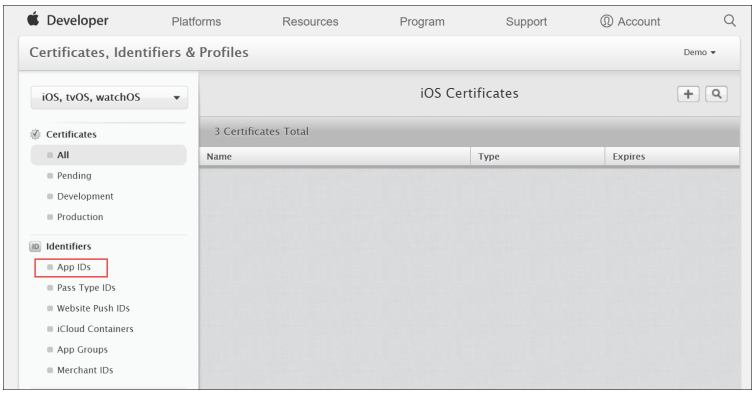
1. Create an Apple developer account to get the Apple ID for your company. For this, go to https://developer.apple.com/account
2. Enroll in the Apple Developer Program. It is required to configure push notifications as you will need a push notification certificate for your App ID, which requires the Apple Developer Program membership. Go to https://developer.apple.com/programs for more details.
3. Register an App ID:

   Sign into https://developer.apple.com/account.

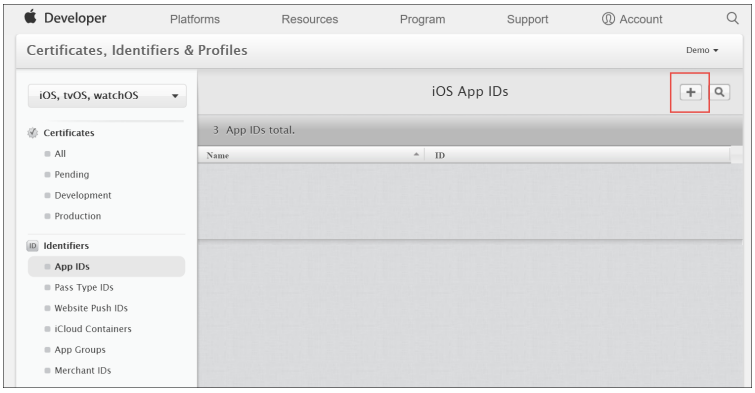
Copyright © Sipwise GmbH - All rights reserved



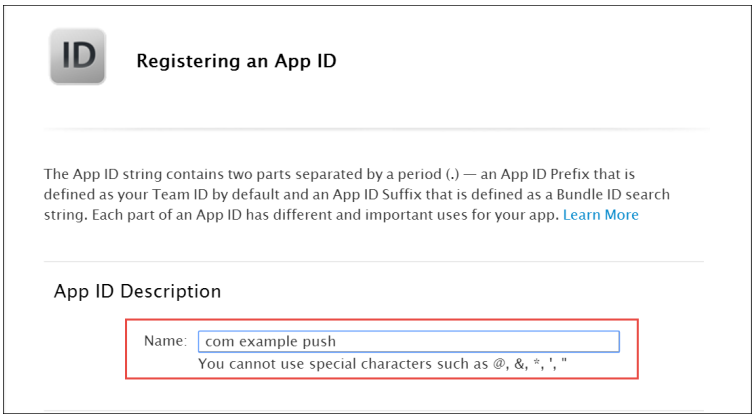
Click *Certificates, IDs & Profiles*.



Under *Identifiers*, select *App IDs*.

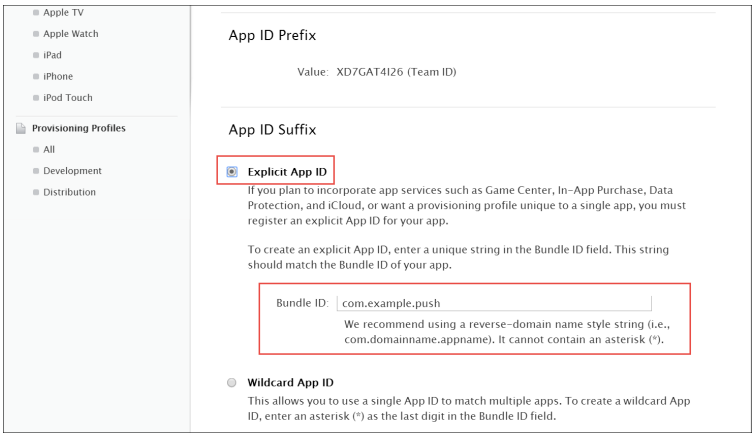


Click the *Add* button (+) in the upper-right corner.

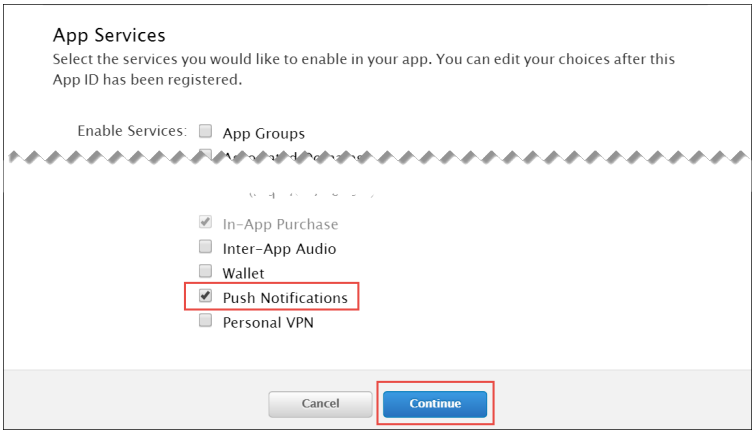


Copyright © Sipwise GmbH - All rights reserved

Enter a name for the App ID in the *App ID Description* block. This helps you identify the App ID later.

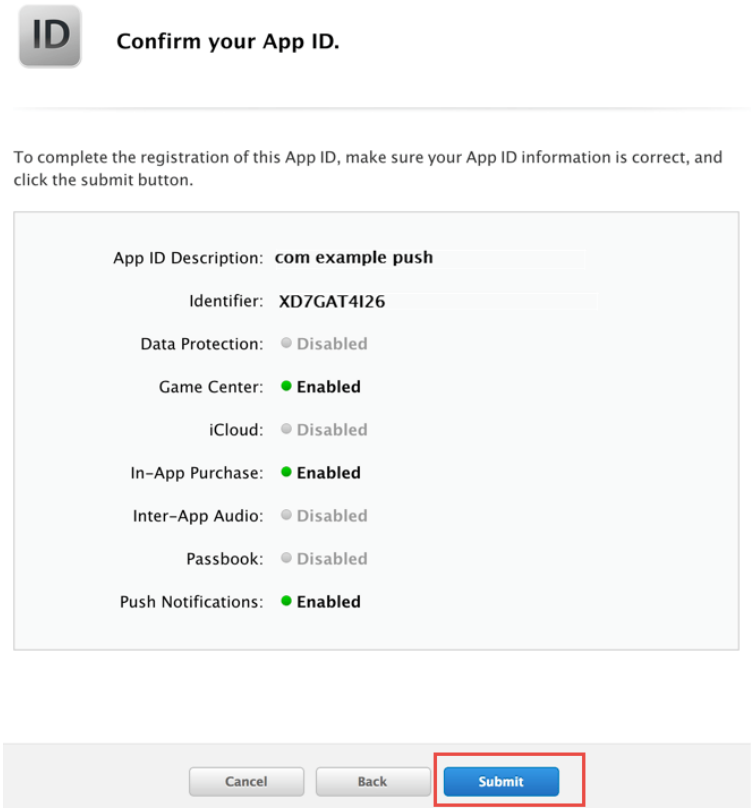


Select *Explicit App ID* and enter the app's bundle ID in the *Bundle ID* field. Note that an explicit App ID exactly matches the bundle ID of an app you are building — for example, com.example.push. An explicit App ID can *not* contain an asterisk (*).



In the App Services section enable Push Notifications. Click *Continue* to submit the form

Copyright © Sipwise GmbH - All rights reserved

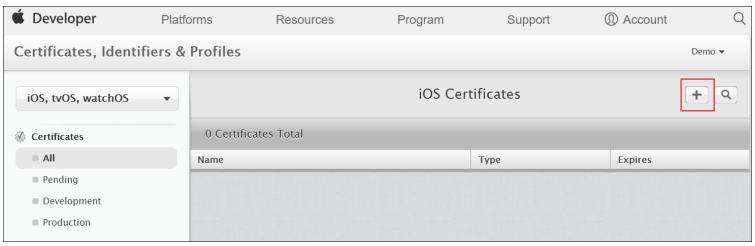


Click *Submit* to create the App ID.

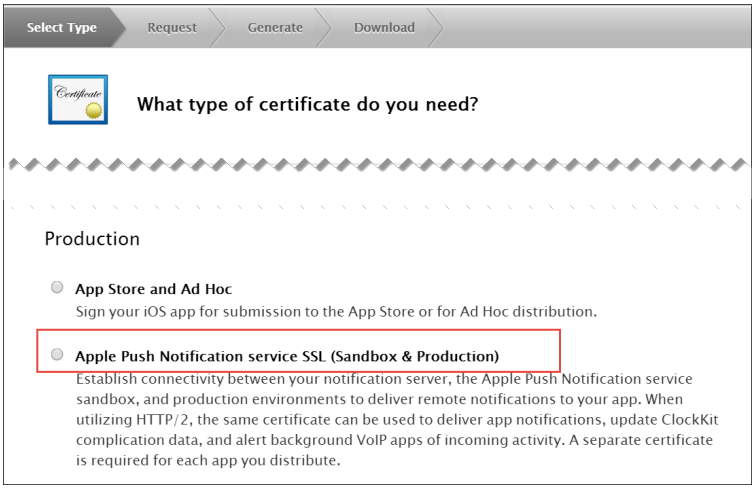
Obtain an Apple SSL Certificate and a Private Key

1. Create a CSR (Certificate Signing Request):

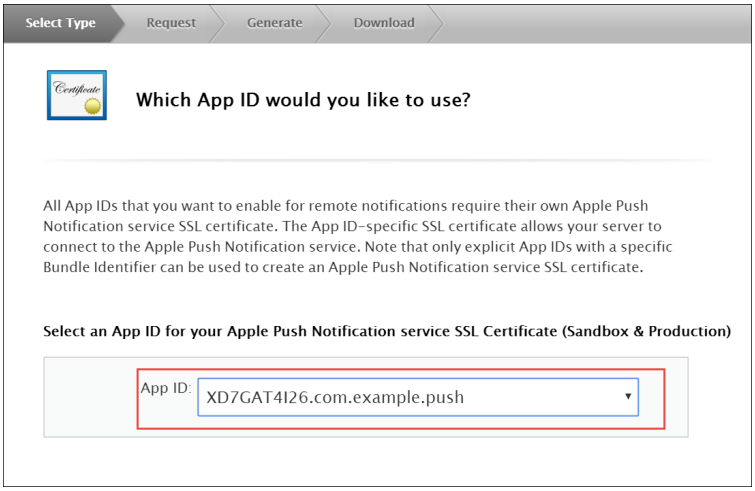
   Sign into https://developer.apple.com/account/ios/certificate.



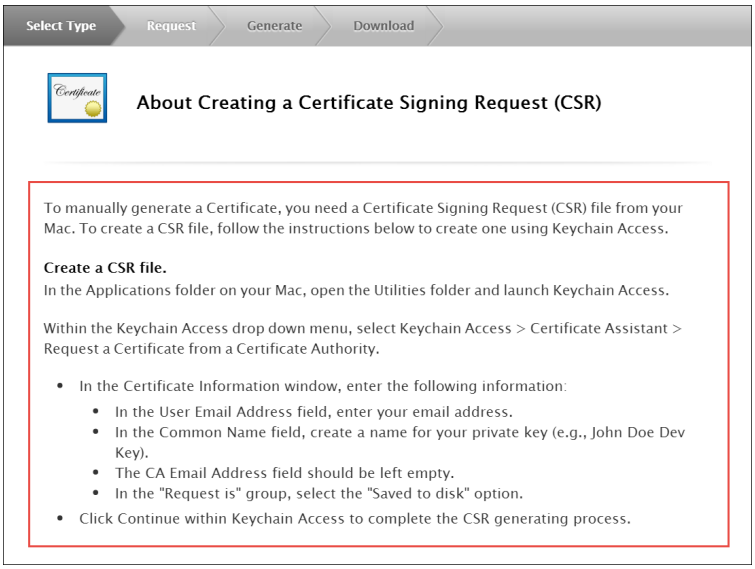
   Click the *Add* button (+) in the upper-right corner.



Select *Apple Push Notification service SSL (Sandbox & Production)* as the certificate type and click *Continue*.



Select your App ID and click *Continue*.



Read the information about creating a CSR.

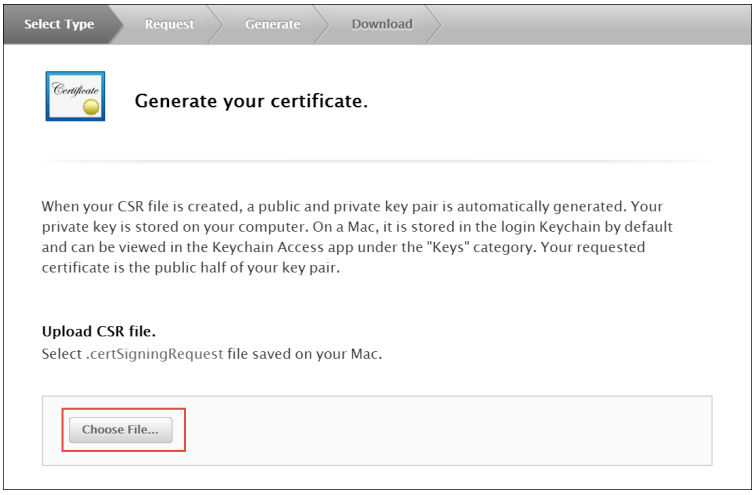
Follow the instructions to create a CSR using Keychain Access in MAC.

Copyright © Sipwise GmbH - All rights reserved

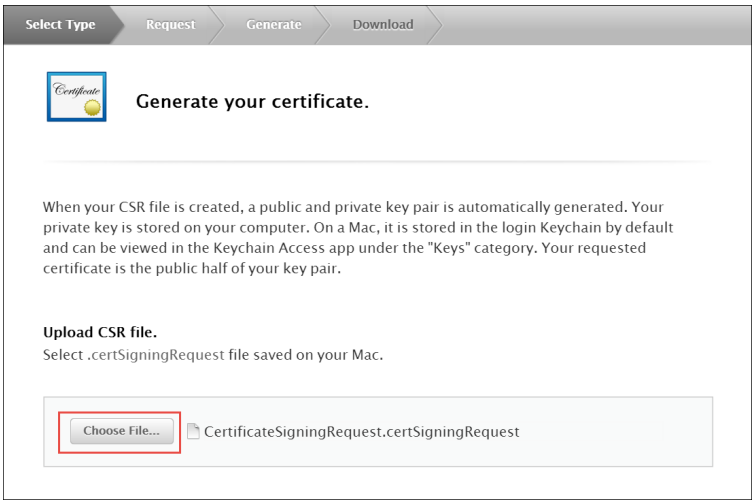
NOTE If you do not have access to a Mac, you can still create a CSR in Linux or Windows using OpenSSL, for example.

2. Get the Certificate and Private Key

   When you have the CSR file return to the browser and click *Continue*.

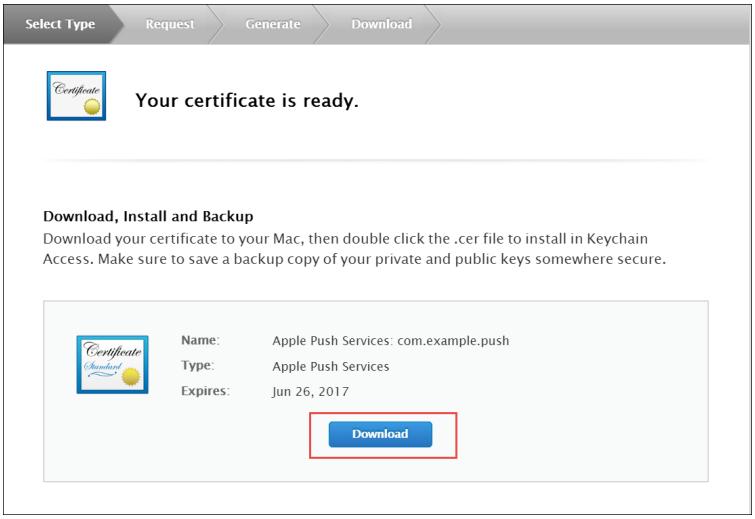


   Click *Choose File...* in your browser.



   Select the CSR file you just created and saved and click *Continue*.

Copyright © Sipwise GmbH - All rights reserved



Click *Download* to download the certificate (give it the **aps.cer** name).

Open the downloaded certificate file (it should automatically be opened in Keychain Access, otherwise open it manually in Keychain Access).

Find the certificate you just opened/imported in Keychain Access.

Expand the certificate to show the Private Key.

Select only the Private Key portion of the certificate, right-click on it and select *Export "Common Name"...* from the menu.

Choose a location (e.g. Desktop) and filename to export the .p12 file to and click *Save*.

Optionally pick a password for the .p12 file to protect its private key contents and click *OK*. (You will then need to enter your log-in password to permit the export).

3. Generate a PEM file from the p12 file:

   Open up your terminal and run the following commands to create a PEM file from the p12 file (If you input a password for the p12 file, you will need to enter it here):

```
cd ~/Desktop
openssl x509 -in aps.cer -inform der -out PushChatCert.pem
openssl pkcs12 -in PushChatCert.p12 -out PushCertificate.pem -nodes
-clcerts
openssl pkcs12 -nocerts -out PushChatKey.pem -in PushChatKey.p12
```

Obtain the API Key for the App from Google

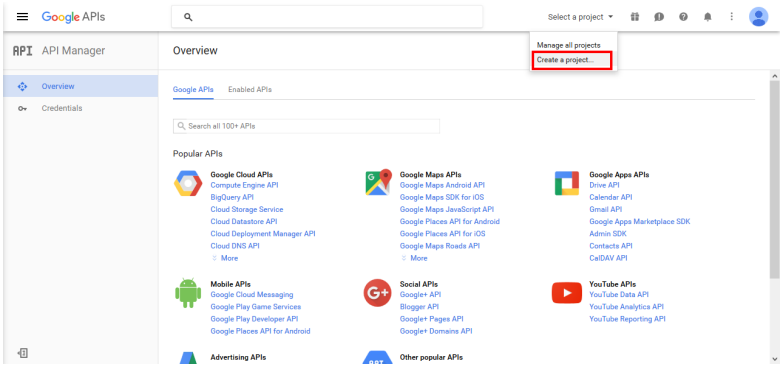
You can use Google Firebase Cloud Messaging (FCM) to send push notifications to your subscribers with Android-based mobile devices. Google Cloud Messaging is a free service that acts as an intermediary between Sipwise C5 and devices of your subscribers. Google's Cloud Connection Server (CCS), a part of GCP, manages the persistent connections with mobile devices to deliver your push notifications.

While communicating with CCS, Sipwise C5 identifies itself using an API key. To get it, follow the steps below.

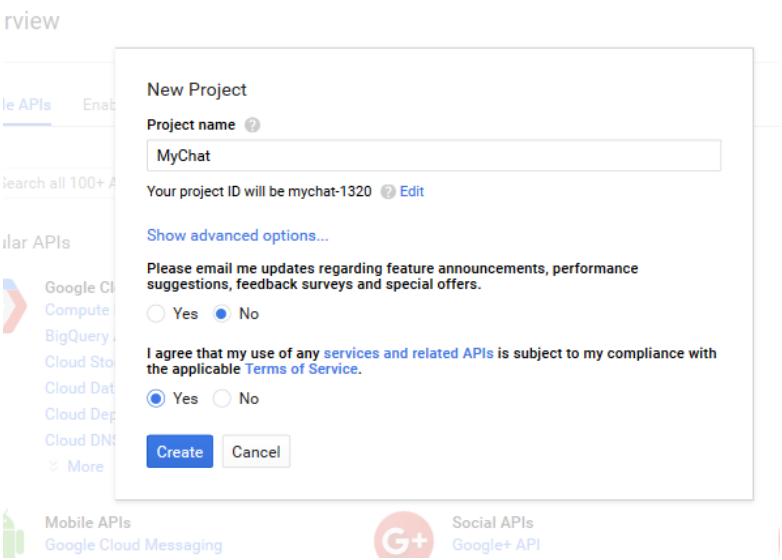
1. Create a new project in the Google APIs Console page. For this go to

Copyright © Sipwise GmbH - All rights reserved

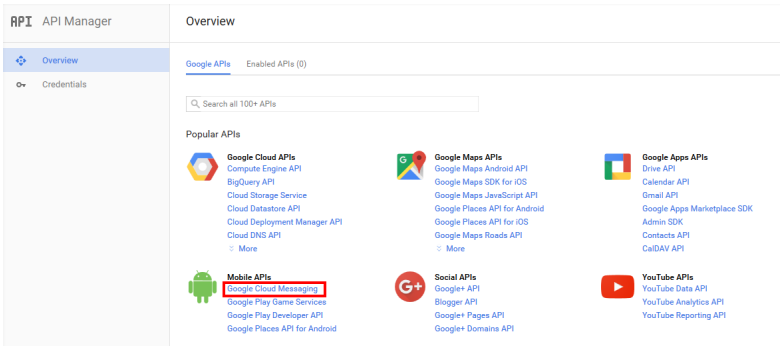
code.google.com/apis/console.



2. Click *Create a Project..*

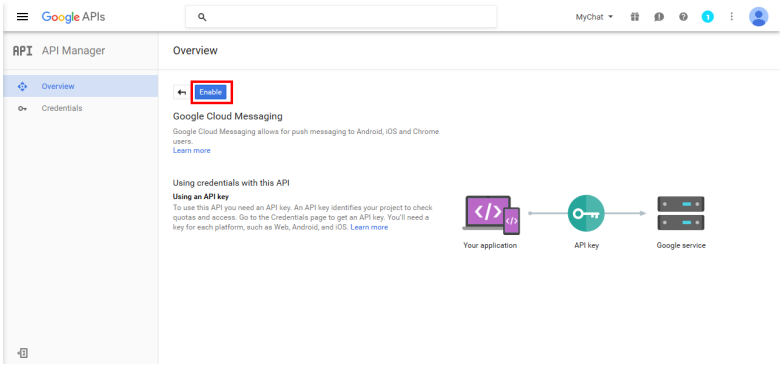


3. Input the project name, agree with the *Terms of Service* and click *Create*.

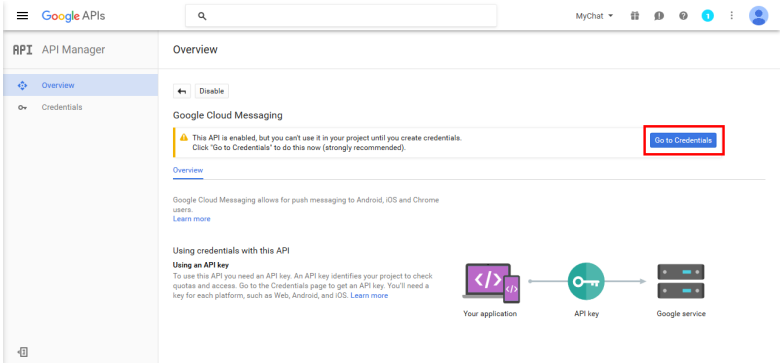


4. Click *Google Cloud Messaging* on the Overview page.

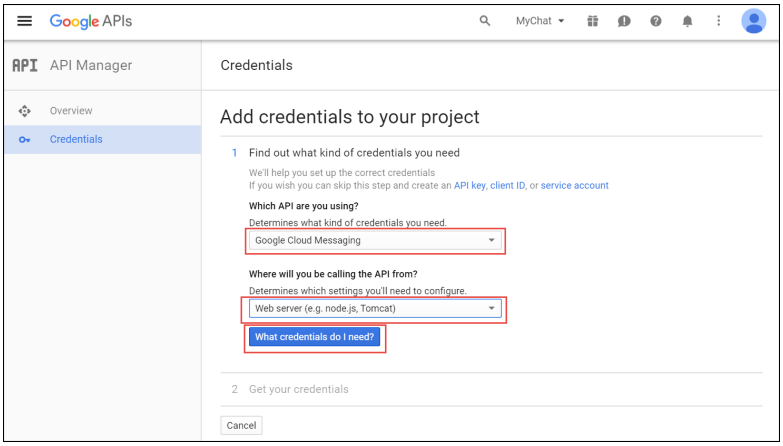
Copyright © Sipwise GmbH - All rights reserved



5. Click *Enable* for the Google Cloud Messaging.

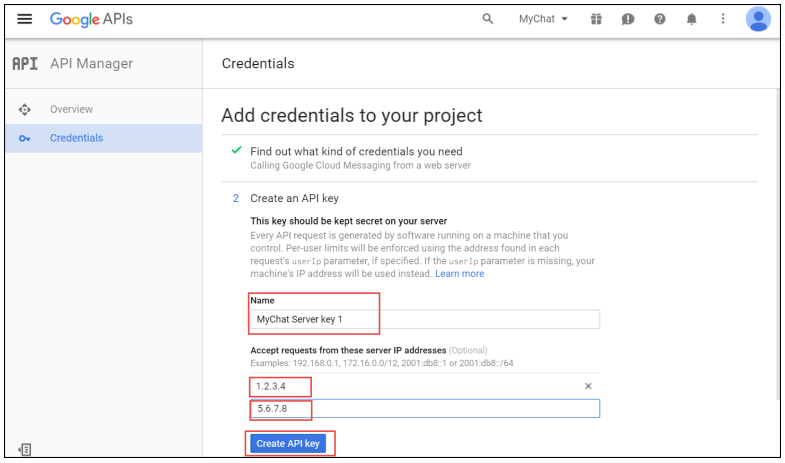


6. Click *Go to Credentials*.



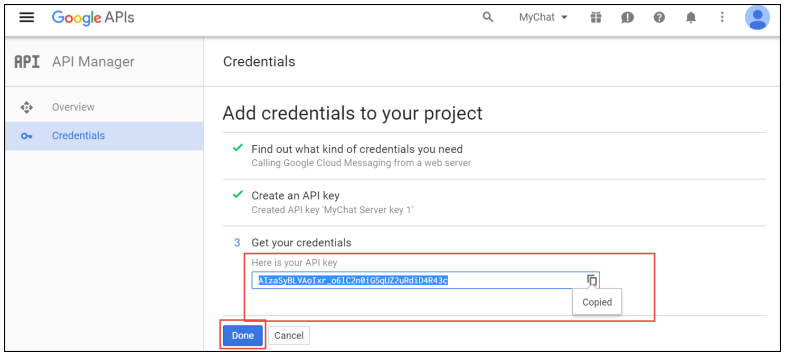
7. Select Google Cloud Messaging and Web Server from the corresponding lists and click *What credentials do I need*?

Copyright © Sipwise GmbH - All rights reserved

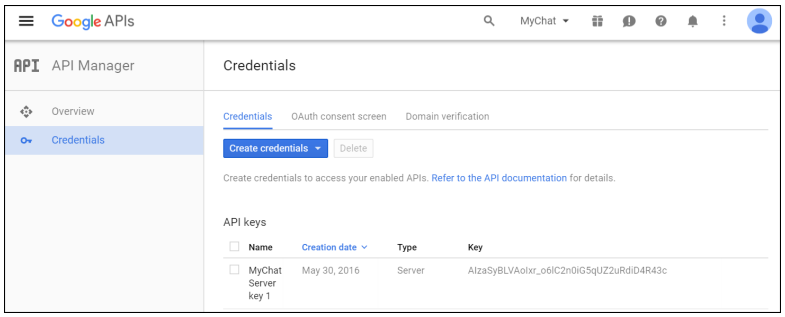


8. Adjust the API Key name and input the IP addresses of *all* your load balancers under *Accept requests from these server IP addresses*. Click *Create API key*.

NOTE You may skip adding the IP addresses, otherwise list *ALL* your load balancers.



9. Copy your API key and click *Done*. Save the API key for future use.



Provide the Required Information to Developers

Please, provide Sipwise developers with the following files and information so that they can make beta builds and submit the application to the App Store:

- Access to your Apple developer account
- The trusted SSL certificate and its private key
- The Apple SSL certificate and its private key

For the Android application, provide the following:

Copyright © Sipwise GmbH - All rights reserved

- Access to your Google developer account
- Google application API key

Adjust Sipwise C5 Configuration (Usually Performed by Sipwise)

1. Upload the Apple SSL certificate (**PushChatCert.pem**) and the private key (**PushChatKey.pem**) to */etc/ngcp-config/shared-files/ssl/*
2. Upload the trusted SSL certificate (**CAsigned.crt**) and the private key (**CAsigned.key**) to */etc/ngcp-config/shared-files/ssl/*
3. Specify the corresponding paths and names in the pushd section of the config.yml file:

   apns: section (For iOS mobile application)

   certificate: \'/etc/ngcp-config/shared-files/ssl/PushChatCert.pem'

   enable: yes

   key: \'/etc/ngcp-config/shared-files/ssl/PushChatKey.pem'

   enable: yes

   android: section (for Android mobile application)

   enable: yes

   key: \'google_server_api_key_here'

   ssl: yes

   sslcertfile: /etc/ngcp-config/shared-files/ssl/CAsigned.crt

   sslcertkeyfile: /etc/ngcp-config/shared-files/ssl/CAsigned.key

   You can find an example of */etc/ngcp-config/config.yml* configuration in the config.yml overview section.

4. Apply your changes:

```
ngcpcfg apply 'enabled the backup feature.'
ngcpcfg push all
```

Recheck Your DNS Zone Configuration

Check that your **NS** and **A** DNS records are correctly configured.

Let's consider the following example: * the load-balancers have the lb01a.example.com and the lb01b.example.com names * the shared name is lb01.example.com and the shared IP address is 1.1.1.1 * the service name is voipservice.example.com

The following DNS records must be present:

Server Name	Record type	IP Address
lb01a.example.com	A	1.2.3.4
lb01b.example.com	A	5.6.7.8

Copyright © Sipwise GmbH - All rights reserved

lb01.example.com	A	1.1.1.1
voipservice.example.com	A	1.1.1.1

Add SRV Records to DNS

Add at least one record for each service: **xmpp-server**, **xmpp-client**, **sips**.

A regular SRV record has the following form:

```
_service._proto.name. TTL class SRV priority weight port target
```

- service: the symbolic name of the service (xmpp-server, xmpp-client, sips).
- proto: the transport protocol of the desired service (TCP).
- name: the domain name (ending in a dot).
- TTL: standard DNS time to live field.
- class: the standard DNS class field (this is always IN).
- priority: the priority of the target host (lower value means more preferred).
- weight: a relative weight for records with the same priority (the higher the value, the more requests will be sent).
- port: the TCP or UDP port of the service.
- target: the canonical hostname of the machine providing the service (ending in a dot).

Here are examples of the SRV records:

```
_xmpp-server._tcp.voipservice.example.com. 18000 IN SRV 10 50 5269
voipservice.example.com.
_xmpp-client._tcp.voipservice.example.com. 18000 IN SRV 10 50 5222
voipservice.example.com.
_sips._tcp.voipservice.example.com. 18000 IN SRV 10 100 5061
voipservice.example.com.
```

You can always check whether the required SRV records are configured by executing the following commands:

```
dig SRV _xmpp-client._tcp.voipservice.example.net
dig SRV _xmpp-server._tcp.voipservice.example.net
dig SRV _sips._tcp.voipservice.example.net
```

Check NTP Configuration

We strongly suggest that the clocks of all the nodes within the platform are synchronized. To ensure this, check that the NTP service is correctly configured on all your Sipwise C5 servers and works reliably. Execute the following command for quick test of time synchronization:

Copyright © Sipwise GmbH - All rights reserved

```
timedatectl
```

If the current node synchronizes with an NTP server, this server will be marked by 'NTP service: active'.

Execute the following command to get the NTP service status:

```
timedatectl timesync-status
```

When using the optional 'ntpd' implementation instead of the default 'systemd-timesyncd', execute the following command to get the time synchronization status:

```
ntpq -p
```

If the current node synchronizes with an NTP server, this server will be marked by the star ('*') symbol.

Enable Apple/Google Mobile Push

It can be enabled for a domain or separate subscribers in the web administration panel or using API.

To enable the service:

1. Go to *Domain Preferences* or *Subscriber Preferences* of the domain/subscriber you want to enable Apple/Google Mobile Push for.
2. Go to the *Internals* group and enable the **mobile_push_enable** parameter selecting one of the following options:

   Never send push: the mobile push feature is disabled.

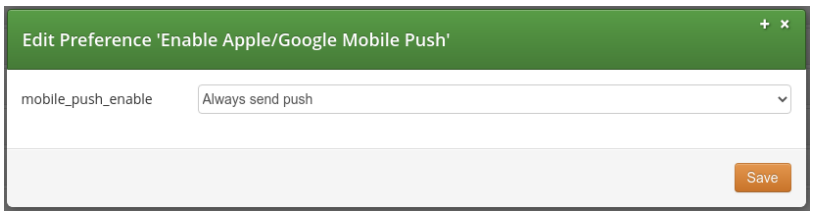
   Always send push: the mobile push will be triggered every time the subscriber receives a call in parallel with the already registered devices, if there are any.

   Send push only if no device registered: the mobile push will be triggered only if the subscriber doesn't have any registered device. In case there is one or more devices registered only those will receive the incoming call.

   Always send push, skip registered devices: the mobile push will be triggered immediately disregarding the already registered devices. That means, already registered devices do not receive the incoming call.

   Registered devices, then send push: the Sipwise C5 will call the already registered devices as first. In case of failure (devices reject the call) it will then trigger the mobile push.

   Send push, then registered devices: the Sipwise C5 will trigger the mobile push as first. In case of failure (push service return an error or the mobile devices reject the call) it will then send the call to the already registered devices.



 Copyright © Sipwise GmbH - All rights reserved

Perform Tests

Perform tests when the application is available:

1. Download and install the application.
2. Open the application and input your registration username in the username@domain.name format and password.
3. Review the quality of application branding.
4. Make test calls.
5. Test the presence functionality.
6. Test the chat and group chat.
7. Test messaging.
8. Test the sharing functionality (e.g. pictures, video and voice messages and maps).
9. Check the application phone book integration with the phone's one

Make sure that the subscribers can start using your services in the easiest possible way.

8.3. Lawful Interception

8.3.1. Introduction

The Sipwise C5, as a communications platform carrying voice, fax and messaging data has to provide means for lawful interception of the content of communication by third party entities. Those Law Enforcement Agencies (LEAs) have to be able to connect to Sipwise C5 platform in a standardized way—ETSI, 3GPP and other organisations define the interface (and data exchange) between telecommunication operators and LEAs.

High level overview of lawful interception is shown in the following figure:

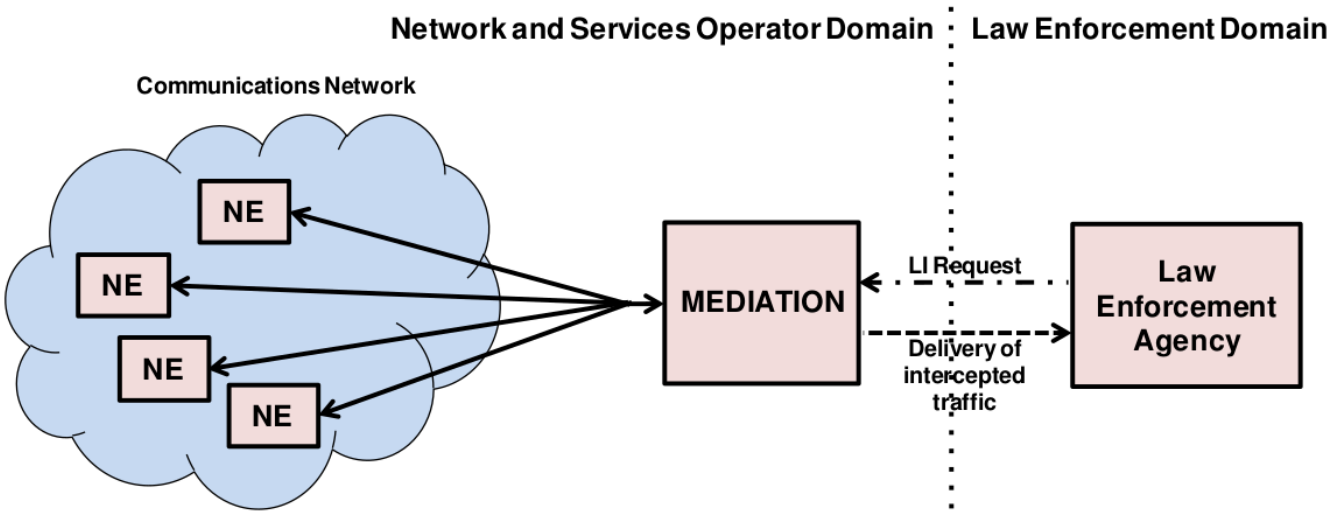


Figure 172. LI: High Level Overview

Main interfaces of lawful interception according to ETSI standard:

Copyright © Sipwise GmbH - All rights reserved

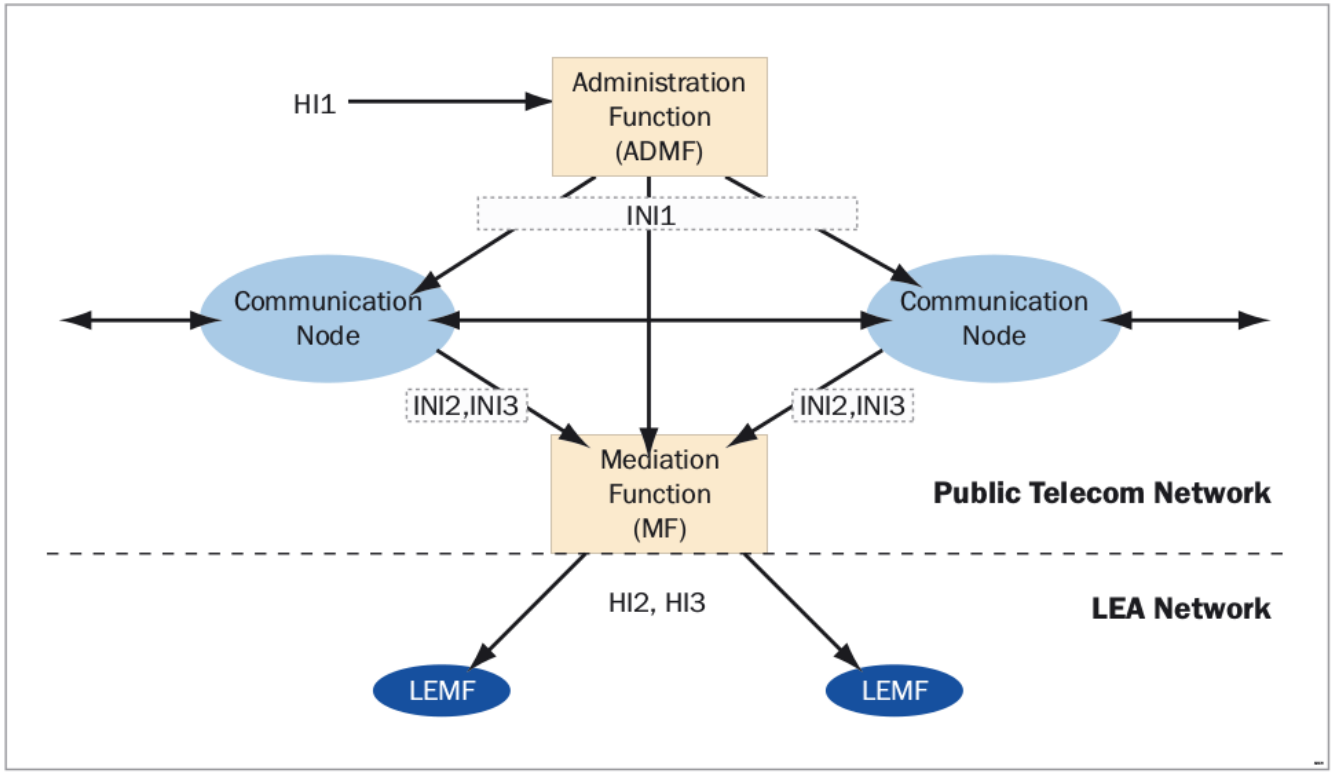


Figure 173. LI: ETSI Interfaces

Terms and Abbreviations

Content of Communication (CC)

Information exchanged between two or more users of a telecommunications service, excluding Intercept Related Information.

NOTE This includes information which may, as part of some telecommunications service, be stored by one user for subsequent retrieval by another.

CC Internal Interception Function (CC-IIF)

The CC-IIF shall cause the CC, specified by the CCTF, via the CCCI to be duplicated and passed to the MF.

Content of Communication Control Interface (CCCI)

Carries controls information from the CCTF to the CC-IIF.

CC Trigger Function (CCTF)

The purpose of the CCTF is to determine the location of the CC-IIF device associated to the target CC traffic, and to control the CC-IIF via the CCCI interface.

Content of Communication Trigger Interface (CCTI)

Carries trigger information from the IRI-IIF to the CCTF.

Handover Interface (HI)

Physical and logical interface across which the interception measures are requested from an operator, and the results of interception are delivered from an operator to an LEMF.

Copyright © Sipwise GmbH - All rights reserved

Intercept Related Information (IRI)

Collection of information or data associated with telecommunication services involving the target identity, specifically call or service associated information or data (e.g. call identifier, unsuccessful call attempts) and location information.

Intercept Related Information Internal Interception Function (IRI-IIF)

The purpose of the IRI-IIF is to generate IRI information associated with sessions, calls, connections and any other information involving interception targets identified by Law Enforcement Agency (LEA) sessions.

Internal Network Interface (INI)

Network's internal interface between the Internal Intercepting Function and a mediation function.

Law Enforcement Agency (LEA)

Organization authorized, by a lawful authorization based on a national law, to request interception measures and to receive the results of telecommunications interceptions.

Law Enforcement Monitoring Facility (LEMF)

Law enforcement facility designated as the transmission destination for the results of interception relating to a particular interception subject.

Lawful Interception Administration Function (AF)

The AF ensures that an intercept request from a LEA for IRI or CC or both is provisioned for collection from the network, and subsequent delivery to the LEMF.

Lawful Interception Mediation Function (MF)

Mechanism which passes information between an access provider or network operator or service provider and a handover interface.

1. Firstly it receives information related to active intercepts from the IRI-IIF(s) and CC-IIF(s) within the service provider network.
2. Secondly correlates and formats that IRI and CC information in real time for delivery to the LEMF over the HI2 and HI3 handover Interfaces.

X1, X2 and X3 Interfaces

The 3GPP standard for Lawful Interception defines the handover interfaces with different names compared to the ETSI standard. The X*n* interface corresponds to the INI*n* interface and is functionally identical to the INI*n* interface.

8.3.2. Architecture and Configuration of LI Service

Sipwise C5 platform implements the functions defined by LI requirements in a way that it relies on a third party provider for the Lawful Interception Mediation Function (MF).

Regarding other LI functions that are defined by ETSI / 3GPP standards there are 2 possible implementations:

1. Sipwise C5 behaves as the Administration Function (AF) but the actual call data capturing is carried out by other SIP endpoints. In this case Sipwise C5 forwards the calls to be intercepted to its **SIP peers dedicated for LI service**. Within the scope of SIP peer based solution there are still 2 modes of operation:

Copyright © Sipwise GmbH - All rights reserved

Call loopback to NGCP: the LI peer receives the call, extracts IRI and CC data and then routes the call back to NGCP. Sipwise C5 handles the looped back call as if that was initiated from Sipwise C5 and sets up the second call leg to the destination.

Call forwarded by peer directly to destination: in this case Sipwise C5 will handle the call to LI peer as an ordinary second call leg to the destination.

2. Sipwise **Sipwise C5 itself provides** the required LI functions: AF and call data capturing; IRI and CC of intercepted calls are forwarded to the third party MF from NGCP. Sipwise C5 can be configured in two modes:

   Non-Distributed: The LI roles are hosted on the PROXY nodes. The REST API endpoint to LEA will be the usual MGMT nodes.

   Distributed: The LI roles are hosted on geographically distributed LB+RTP nodes, for example one pair of LB+RTP nodes per country, each of which has different law enforcement authorities. The LB+RTP nodes will provide an *ngcp-panel* instance, which due to privacy laws is specially configured to store intercepts locally (the node will operate on its own set of data), and the external party (law enforcement, LI integration, etc.) interacts with the pertinent LB+RTP nodes via the REST API (or SOAP for older installations).

This handbook will discuss the second setup in detail in the following sections.

The below figure illustrates the logical connection of LI functions on Sipwise C5.

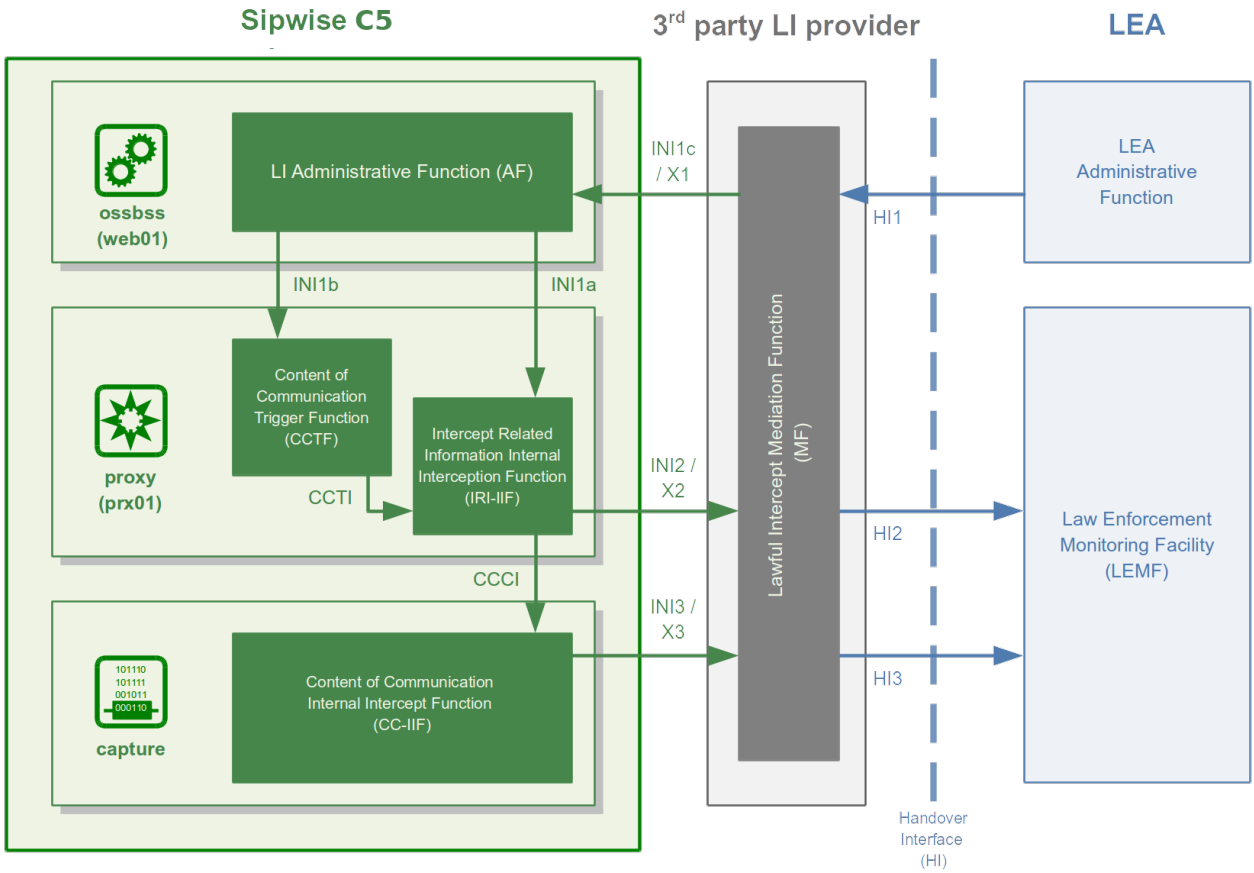


Figure 174. LI with 3rd Party Provider

Copyright © Sipwise GmbH - All rights reserved

Architecture Based on NGCP Voisniff Module

The implementation of LI services with captagent is no longer available and configurable on Sipwise C5, Sipwise requires deploying a revised solution with its ngcp-voisniff software module. This newer implementation also relies on a 3rd party LI provider representing the LI Mediation Function (MF), where Sipwise currently cooperates with Group2000, Pine and Utimaco.

Sipwise C5 components providing LI functions:

- **ngcp-panel:** this module is responsible for managing REST API for the whole NGCP in general

  runs on: the active node with web role (sp on PRO; web01 on CARRIER) on a Sipwise C5 platform

  LI functions: AF; INI1 / X1 interface towards the MF

- **kamailio-proxy:** this module serves as a generic call control function on the NGCP

  runs on: the active node typically with proxy role (sp on PRO; prx01 on CARRIER) on a Sipwise C5 platform

  LI functions: CCTF and IRI-IIF; INI2 / X2 interface towards the MF

- **ngcp-voisniff:** this module is a generic element for capturing SIP and RTP traffic on the NGCP

  runs on: the active node typically with lb role (sp on PRO; lb01 on CARRIER) on a Sipwise C5 platform

  LI functions: CC-IIF; INI3 / X3 interface towards the MF

NOTE The ngcp-voisniff module is installed by default but not activated on the Sipwise C5. It provides the possibility to get call statistics through the Admin web interface, and is not readily configured for LI services. Please contact Sipwise if you need to activate LI services on the platform.

Authentication and Confidentiality

It is required that the communication between the telecommunication operator's network element (that is: Sipwise C5) and the MF be authenticated and confidential, since the intercepted session related data and content of communication must not be disclosed to any 3rd party. For this purpose NGCP's LI service applies authentication and LI session data encryption based on public key cryptography mechanism (TLS).

Both Sipwise C5 and the MF must authenticate themselves by certificates, for this reason Sipwise C5 operator must ensure that valid certificates are deployed on the system. There is a need to contact the 3rd party LI provider, so that he can provide the necessary client certificates that Sipwise C5 will use to setup secured connection to the MF on X2 and X3 interfaces.

Similarly, the MF provider must contact Sipwise C5 operator to offer him valid client certificates that the MF element will use to establish secured connection to the Sipwise C5 on X1 interface.

Configuration of LI Service

To enable LI services on Sipwise C5 the platform administrator has to enable lawful interception through the main configuration file (config.yml).

For a distributed setup, the *cluster_sets.type* variable has to be set to 'distributed' (see cluster_sets for more information), and the *lb* nodes need to be assigned the *li* role. For a non-distributed setup, the

Copyright © Sipwise GmbH - All rights reserved

proxy nodes need to be assigned the *li* role.

From the user and program point of view, the *li* role will only be visible in a node if the *intercept.enable* setting is set to 'yes'. When the cluster is set up in a distributed mode (that is *cluster_sets.type* is set to 'distributed'), the nodes will also have the *li_dist* virtual role visible (which is synthesized at run-time and cannot be specified in the node configuration), so that these can check for a single condition instead of multiple.

Here below is a sample configuration, which shows parameters of intercept and voisniff sections.

On a CARRIER it would look like:

```
intercept:
  enable: yes
  local: no
  peer:
    acc: no
    inbound_prefix: LI_
    outbound_prefix: intercept_
  type: voisniff

voisniff:
  admin_panel: no
  daemon:
    custom_bpf: ''
    filter:
      exclude:
      - active: '0'
        case_insensitive: '1'
        pattern: '\ncseq: *\d+ +(register|notify|options)'
      include: []
      sip_ports:
      - 5060
      - 5062
    interfaces:
      extra: []
      types:
      - sip_int
      - sip_ext
      - rtp_ext
    li_x1x2x3:
      call_id:
        del_patterns:
        - _pbx\-1(?:_[0-9]{1,10})?$
        - _b2b\-1(?:_[0-9]{1,10})?$
        - _xfer\-1(?:_[0-9]{1,10})?$
      captagent:
        cin_max: '3000'
        cin_min: '0'
        x2:
          threads: 20
      client_certificate: /etc/ngcp-config/shared-
```


Copyright © Sipwise GmbH - All rights reserved

```
files/ssl/li/x23_client/x23_client_cert.pem
      enable: yes
      fix_checksums: no
      fragmented: no
      interface:
        excludes: []
      local_name: sipwise
      private_key: /etc/ngcp-config/shared-
files/ssl/li/x23_client/x23_client_cert_priv_key.pem
      split_intercept: no
      x1:
        port: '18090'
      x23:
        protocol: sipwise
    mysql_dump:
      enable: no
      max_query_len: 67108864
      num_threads: '4'
    rtp_filter: yes
    start: yes
    threads_per_interface: '10'
  partitions:
    increment: '700000'
    keep: '10'
```

On a PRO it would look like:

```
intercept:
  enable: no
  local: no
  peer:
    acc: no
    inbound_prefix: LI_
    outbound_prefix: intercept_
  type: none

voisniff:
  admin_panel: yes
  daemon:
    custom_bpf: ''
    filter:
      exclude:
      - active: '0'
        case_insensitive: '1'
        pattern: '\ncseq: *\d+ +(register|notify|options)'
      include: []
      sip_ports:
      - 5060
      - 5062
    interfaces:
      extra: []
```

Copyright © Sipwise GmbH - All rights reserved

```
      types:
      - sip_int
      - sip_ext
      - rtp_ext
    li_x1x2x3:
      call_id:
        del_patterns:
        - _pbx\-1(?:_[0-9]{1,10})?$
        - _b2b\-1(?:_[0-9]{1,10})?$
        - _xfer\-1(?:_[0-9]{1,10})?$
      captagent:
        cin_max: '3000'
        cin_min: '0'
        x2:
          threads: 20
      client_certificate: ''
      enable: no
      fix_checksums: no
      fragmented: no
      interface:
        excludes: []
      local_name: sipwise
      split_intercept: no
      x1:
        port: '18090'
      x23:
        protocol: sipwise
    mysql_dump:
      enable: yes
      max_query_len: 67108864
      num_threads: '4'
    rtp_filter: yes
    start: yes
    threads_per_interface: '2'
  partitions:
    increment: '700000'
    keep: '10'
```

Configuration Parameters

intercept.enable

Set it to `yes` if you want to activate LI service. Default: `no`

intercept.local

On CARRIER systems, if set to `yes`, intercept data will be stored on the local system instead of the central database node.

intercept.peer.acc

Calls to be intercepted may be forwarded to LI peers. The LI peer may forward the call to the original destination, without looping the call back to NGCP. Set this parameter to `yes` if you want to enable billing for such calls. Default: `no`

Copyright © Sipwise GmbH - All rights reserved

intercept.peer.inbound_prefix

Calls to be intercepted may be forwarded to LI peers. This parameter specifies the prefix that is prepended to SIP usernames when the call is looped back to NGCP, in order to avoid sending the call again to any LI peer. Used by Sipwise C5 internally. Default: `LI_`

intercept.peer.outbound_prefix

Calls to be intercepted may be forwarded to LI peers. This parameter specifies the prefix that is prepended to SIP usernames when the call is routed to an LI peer. It will be stripped off by rewrite rules of the peer, before sending the call effectively to the peer. Used by Sipwise C5 internally. Default: `intercept_`

intercept.type

The LI service provider module; allowed values are:

- `none`: LI service is not activated
- `peer`: LI service is activated and call data capturing is performed by SIP peers
- `voisniff`: LI service is activated and call data capturing is performed by the voisniff module

Default: `none`

voisniff.admin_panel

voisniff.daemon.mysql_dump.*

voisniff.partitions.*

These parameters are not used in LI configuration, but only for call statistics which can be retrieved through the Admin web interface.

voisniff.daemon.custom_bpf

Allows the operator to set a custom packet filter to be used when capturing packets no the network interfaces, overriding the default packet filter generated by the system based on other configuration settings (port ranges, etc). It's not normally necessary to set this. Default: empty

voisniff.daemon.filter.exclude

Additional filter to determine packets that need to be excluded from capturing. This configuration parameter is a list of items, each of them has 3 components:

- active: Determines whether the filter is active or not. Allowed values are: `0` (false/inactive; this is the default) or `1` (true/active).
- case_insensitive: Determines whether the `pattern` is case-insensitive (`1`; this is the default) or not (`0`).
- pattern: A regular expression providing the matching pattern for packets that have to be filtered.

voisniff.daemon.filter.include

Additional filter to determine packets that need to be included in capturing. The parameter has the same syntax as voisniff.daemon.filter.exclude.

voisniff.daemon.filter.sip_ports

A list of ports that should be considered to carry SIP traffic. Intercepted packets that do not involve one of these ports will not be attempted to be parsed as SIP packets. This filter can be disabled by having this list empty. Default: `5060` and `5062`

Copyright © Sipwise GmbH - All rights reserved

voisniff.daemon.interfaces.extra

This is a list of additional network interfaces (typically VLAN IDs) where ngcp-voisniff should listen for and capture packets. These interfaces are in addition to the list of interfaces generated by the system based on the interface types (see below).

TIP VLAN interfaces have to be listed when they are used for intercepted calls. On the other hand virtual interfaces for additional IP addresses (e.g. eth0:1) do not have to be listed separately, because the base interface (e.g. eth0) will be used to capture packets.

voisniff.daemon.interfaces.types

A list of network interface types that should be activated for interception. All interfaces that match the given types will be activated. Default: `sip_int`, `sip_ext`, and `rtp_ext`

voisniff.daemon.li_x1x2x3.call_id.del_patterns

List of NGCP-internal Call-ID suffix patterns that should be ignored when determining the original SIP Call-ID of an intercepted call.

CAUTION Please do not change these patterns unless instructed to do so by a Sipwise engineer! Changing the patterns may result in falsely recognised Call-IDs and eventually missed SIP messages during an intercepted call.

voisniff.daemon.li_x1x2x3.captagent.cin_min

voisniff.daemon.li_x1x2x3.captagent.cin_max

When using the captagent-compatible protocol, this specifies the range of intercept ID numbers (CIN) to be generated. Default: `0` through `3000`

voisniff.daemon.li_x1x2x3.captagent.x2.threads

When using the captagent-compatible protocol, this specifies the number of threads to be used for sending outgoing X2 (SIP) captures. Interception may stall if this number if set too low. Default: `20`

voisniff.daemon.li_x1x2x3.client_certificate

The client certificate that Sipwise C5 uses to connect over TLS to a 3rd party LI provider. Relevant only when using the `sipwise` outbound protocol.

voisniff.daemon.li_x1x2x3.enable

Set it to `yes` to enable LI services via X1, X2 and X3 interfaces. Default: `no`

voisniff.daemon.li_x1x2x3.fix_checksums

When enabled (= `yes`), Sipwise C5 will calculate UDP header checksum for packets sent out on X2 and X3 interfaces. This is necessary when the checksum calculation is normally left to the network interface hardware and therefore the UDP header checksum is inherently incorrect on application level. Also the UDP checksum must be calculated by *ngcp-voisniff* on re-assembled packets, so enable this option if there are fragmented packets in intercepted call traffic. Default: disabled (= `no`)

voisniff.daemon.li_x1x2x3.fragmented

When disabled (= `no`), *ngcp-voisniff* defragments all packets and sends out only reassembled packets via X2 and X3 interfaces. If the option is enabled (= `yes`), *ngcp-voisniff* will instead send out the original fragments via X2 and X3. Default: `no`

Copyright © Sipwise GmbH - All rights reserved

voisniff.daemon.li_x1x2x3.instant_intercept

When disabled (= `no`), creating a new interception object does not affect already running calls. In other words, if a call that is already running matches the parameters by a newly created interception object, that call will not start to be intercepted, only new calls established afterwards will. Enabling this option changes this behaviour so that already running calls will also start to be intercepted at the moment when a new interception object is created. Doing so creates additional processing overhead within *ngcp-voisniff*. Default: `no`

voisniff.daemon.li_x1x2x3.interface.excludes

This is a list of interfaces that must be excluded from the interception procedures. The list contains regular expressions that describe the to-be-excluded interfaces, for example: `- ^lo$` to exclude the loopback interface. Default: empty list

voisniff.daemon.li_x1x2x3.local_name

This parameter maps to the header.source field of the X2 protocol. It's an arbitrary string and can be used to identify the sending Sipwise C5 system. Default: `sipwise`

NOTE	This is currently not used.

voisniff.daemon.li_x1x2x3.private_key

The private key that Sipwise C5 uses to connect over TLS to a 3rd party LI provider. Only necessary if the client certificate file does not include the private key.

voisniff.daemon.li_x1x2x3.split_intercept

When enabled (= `yes`), *ngcp-voisniff* uses the Split-LI algorithm to detect which calls have to be intercepted. This algorithm should be enabled only on CARRIER systems with a central core and local satellites architecture and *ngcp-voisniff* running on LB nodes. Default: `no`

voisniff.daemon.li_x1x2x3.x1.port

The port number on which ngcp-voisniff listens for incoming X1 messages. Default: `18090`

CAUTION	You should leave the parameter set to the default value, unless there is a good reason to change it.

voisniff.daemon.li_x1x2x3.x23.protocol

Specified the outbound protocol to speak when delivering X2 (SIP) or X3 (RTP) data. This can be either the `sipwise` protocol using TLS connections, or the `captagent` compatible protocol using HTTP and UDP. Default: `sipwise`

voisniff.daemon.mysql_dump.enable

Master switch for call statistics collection. Default: `yes`

voisniff.daemon.mysql_dump.max_query_len

Determines how much data should be gathered into a single statement for insertion into the database. This should not normally be changed. Default: `67108864`

voisniff.daemon.mysql_dump.num_threads

The number of threads dedicated to inserting data into the database. Default: `4`

Copyright © Sipwise GmbH - All rights reserved

voisniff.daemon.rtp_filter

Determined whether to intercept RTP packets or not. Enabling the filter (set to `yes`) suppresses interception of RTP packets. Disabling it (`no`) enabled interception of RTP packets. Default: `yes`

voisniff.daemon.start

Determines whether voisniff service must be started on the platform. Set it to `yes` if you'd like to activate voisniff that is needed for LI service too. Default: `no` (on CARRIER), `yes` (on PRO)

voisniff.daemon.threads_per_interface

This is a performance tuning option and controls how many threads per enabled sniffing interface should be launched. Example: if it's set to 10 and 3 interfaces are enabled for sniffing, a total of 30 threads will be launched. Default: 2

CAUTION	Do not set it to a high number, or leave it at its default value, unless there is a performance problem with voisniff service. Please keep in mind that a high number of threads might also decrease the overall system performance of NGCP!

8.3.3. X1, X2 and X3 Interface Specification

Short description of X*n* interfaces:

- The **X1** interface is used by an LI provider to create, modify, delete and list interceptions on Sipwise C5 . It is designed as RESTful HTTP interface using JSON (with JSON-HAL in responses from the NGCP) as content type to provision interceptions.
- The **X2** interface is a TLV based interface with JSON payload with a simple request/response mechanism over a secure TLS connection, used to pass intercepted signaling data towards an LI provider.
- The **X3** interface is also a TLV based interface with a binary payload encapsulating the intercepted RTP data.

X1 Interface

The resource used to work with interceptions is always https://ngcp-ip:1443/api/interceptions/

Authentication

Authentication and authorization on Sipwise C5 API is performed via HTTP Basic Auth or SSL Client certificates.

- **HTTP Basic Auth**: With *cURL* use `--user username:password` option to specify your access credentials.

  ```
  curl -i -X GET --user myuser:mypassword
  https://example.org:1443/api/interceptions/
  ```

  Additionally use the `--insecure` option if you are testing against a self-signed server certificate.

- **SSL Client Authentication**: You can generate and download client certificates for administrators

Copyright © Sipwise GmbH - All rights reserved

and resellers via Sipwise C5 Panel in the Administrators view.

For the actual client authentication, you will need two files which you can download from the panel after creating the client certificates:

1. The client certificate generated via Sipwise C5 Panel. This is usually labelled NGCP-API-client-certificate-xxxxx.pem.
2. The CA certificate used to sign the server certificate, in case it as been self-signed or the CA is not recognized by the client host environment.

With *cURL* use `--cert /path/to/NGCPAPIclientcertificatexxxxx.pem` to specify the client certificate, and `--cacert /path/to/cacert.pem` to specify the CA certificate in case of a self-signed server certificate.

```
curl -i -X GET --cert /path/to/NGCPAPIclientcertificatexxxxx.pem \
--cacert /path/to/cacert.pem
https://example.org:1443/api/interceptions/
```

Additionally use the `--insecure` option if you are testing against a self-signed server certificate.

API Description

Collection Actions

Allowed methods for the collection as in METHOD /api/interceptions/

- OPTIONS
- POST
- GET
- HEAD

Item Actions

Allowed methods for a collection item as in METHOD /api/interceptions/id

- PATCH
- OPTIONS
- DELETE
- PUT
- GET
- HEAD

Properties

- liid (Number): The LI ID for this interception.
- number (String): The number to intercept.
- x2_host (String): The IP address of the X2 interface.
- x2_password (null, String): The password for authenticating on the X2 interface.

Copyright © Sipwise GmbH - All rights reserved

- x2_port (Number): The port of the X2 interface.
- x2_user (null, String): The username for authenticating on the X2 interface.
- x3_host (null, String): The IP address of the X3 interface.
- x3_port (null, Number): The port of the X3 interface.
- x3_required (null, Boolean): Whether to also intercept call content via X3 interface (`false` by default).

Query Parameters

- liid: Filter for interceptions of a specific interception ID
- number: Filter for interceptions of a specific number (in E.164 format)
- order_by: Order collection by a specific attribute. Possible values are: id, reseller_id, liid, number, cc_required, delivery_host, delivery_port, delivery_user, delivery_pass, modify_timestamp, create_timestamp, deleted, uuid, sip_username, sip_domain, cc_delivery_host, cc_delivery_port
- order_by_direction: Direction which the collection should be ordered by. Possible values are: asc (default), desc

API Examples

Get a specific interception

- Request:

```
curl -i --insecure --user administrator:administrator -X GET
https://localhost:1443/api/interceptions/528
```

- Response:

Copyright © Sipwise GmbH - All rights reserved

```
HTTP/1.1 200 OK
Server: nginx
Date: Tue, 01 Dec 2015 09:43:41 GMT
ContentType: application/hal+json;
profile="http://purl.org/sipwise/ngcpapi/";
  charset=utf8
ContentLength: 634
Connection: keepalive
Link: </api/interceptions/>; rel=collection
Link: <http://purl.org/sipwise/ngcpapi/>; rel=profile
Link: </api/interceptions/528>; rel="item self"
SetCookie:
ngcp_panel_session=35b56d921c36c1fc6edb8fcd0a86dd9af61ec62a; path=/;
  expires=Tue, 01Dec 2015 10:43:41 GMT; HttpOnly
StrictTransportSecurity: maxage=15768000
  {
    "_links" : {
      "collection" : {
        "href" : "/api/interceptions/"
      },
      "curies" : {
        "href" : "http://purl.org/sipwise/ngcpapi/#rel{rel}",
        "name" : "ngcp",
        "templated" : true
      },
      "profile" : {
        "href" : "http://purl.org/sipwise/ngcpapi/"
      },
      "self" : {
        "href" : "/api/interceptions/528"
      }
    },
    "id" : 528,
    "liid" : 918273,
    "number" : "0014155550132",
    "x2_host" : "192.168.42.42",
    "x2_password" : null,
    "x2_port" : 3002,
    "x2_user" : null,
    "x3_host" : "192.168.42.42",
    "x3_port" : 3003,
    "x3_required" : true
  }
```

Get all interceptions for a number

- Request:

Copyright © Sipwise GmbH - All rights reserved

```
curl -i --insecure --user administrator:administrator -X GET \
https://localhost:1443/api/interceptions/?number=0014155550132
```

- Response:

```
HTTP/1.1 200 OK
Server: nginx
Date: Tue, 01 Dec 2015 09:47:36 GMT
ContentType: application/hal+json;
profile="http://purl.org/sipwise/ngcpapi/";
  charset=utf8
ContentLength: 1283
Connection: keepalive
SetCookie:
ngcp_panel_session=238550c5737058db619b183d925b5f9a61261cfe; path=/;
  expires=Tue, 01 Dec 2015 10:47:36 GMT; HttpOnly
StrictTransportSecurity: maxage=15768000
{
   "_embedded" : {
      "ngcp:interceptions" : {
         "_links" : {
            "collection" : {
               "href" : "/api/interceptions/"
            },
            "curies" : {
               "href" : "http://purl.org/sipwise/ngcpapi/#rel{rel}",
               "name" : "ngcp",
               "templated" : true
            },
            "profile" : {
               "href" : "http://purl.org/sipwise/ngcpapi/"
            },
            "self" : {
               "href" : "/api/interceptions/520"
            }
         },
         "id" : 520,
         "liid" : 1,
         "number" : "0014155550132",
         "x2_host" : "192.168.42.42",
         "x2_password" : null,
         "x2_port" : 3002,
         "x2_user" : null,
         "x3_host" : "192.168.42.42",
         "x3_port" : 3003,
         "x3_required" : true
      }
   },
   "_links" : {
      "curies" : {
```


Copyright © Sipwise GmbH - All rights reserved

```
         "href" : "http://purl.org/sipwise/ngcpapi/#rel{rel}",
         "name" : "ngcp",
         "templated" : true
      },
      "ngcp:interceptions" : {
         "href" : "/api/interceptions/520"
      },
      "profile" : {
         "href" : "http://purl.org/sipwise/ngcpapi/"
      },
      "self" : {
         "href" : "/api/interceptions/?page=1&rows=10"
      }
   },
   "total_count" : 1
}
```

Get all interceptions for all numbers

- Request:

```
curl -i --insecure --user administrator:administrator -X GET \
https://localhost:1443/api/interceptions/
```

- Response:

```
HTTP/1.1 200 OK
Server: nginx
Date: Tue, 01 Dec 2015 09:43:18 GMT
ContentType: application/hal+json;
profile="http://purl.org/sipwise/ngcpapi/";
  charset=utf8
ContentLength: 2364
Connection: keepalive
SetCookie:
ngcp_panel_session=68398eea5bdd3885ad0517e1f6d367ccc80111fa; path=/;
  expires=Tue, 01 Dec 2015 10:43:18 GMT; HttpOnly
StrictTransportSecurity: maxage=15768000
{
   "_embedded" : {
      "ngcp:interceptions" : [
         {
            "_links" : {
               "collection" : {
                  "href" : "/api/interceptions/"
               },
               "curies" : {
                  "href" : "http://purl.org/sipwise/ngcpapi/#rel-
{rel}",
```

Copyright © Sipwise GmbH - All rights reserved

```
                  "name" : "ngcp",
                  "templated" : true
               },
               "profile" : {
                  "href" : "http://purl.org/sipwise/ngcpapi/"
               },
               "self" : {
                  "href" : "/api/interceptions/520"
               }
            },
            "id" : 520,
            "liid" : 1,
            "number" : "0014155550132",
            "x2_host" : "192.168.42.42",
            "x2_password" : null,
            "x2_port" : 3002,
            "x2_user" : null,
            "x3_host" : "192.168.42.42",
            "x3_port" : 3003,
            "x3_required" : true
         },
         {
            "_links" : {
               "collection" : {
                  "href" : "/api/interceptions/"
               },
               "curies" : {
                  "href" : "http://purl.org/sipwise/ngcpapi/#rel-
{rel}",
                  "name" : "ngcp",
                  "templated" : true
               },
               "profile" : {
                  "href" : "http://purl.org/sipwise/ngcpapi/"
               },
               "self" : {
                  "href" : "/api/interceptions/528"
               }
            },
            "id" : 528,
            "liid" : 918273,
            "number" : "0014155550132",
            "x2_host" : "192.168.42.42",
            "x2_password" : null,
            "x2_port" : 3002,            "x2_user" : null,
            "x3_host" : "192.168.42.42",
            "x3_port" : 3003,
            "x3_required" : true
         }
      ]
   },
   "_links" : {
```


 Copyright © Sipwise GmbH - All rights reserved

```
      "curies" : {
         "href" : "http://purl.org/sipwise/ngcpapi/#rel{rel}",
         "name" : "ngcp",
         "templated" : true
      },
      "ngcp:interceptions" : [
         {
            "href" : "/api/interceptions/520"
         },
         {
            "href" : "/api/interceptions/528"
         }
      ],
      "profile" : {
         "href" : "http://purl.org/sipwise/ngcpapi/"
      },
      "self" : {
         "href" : "/api/interceptions/?page=1&rows=10"
      }
   },
   "total_count" : 2
}
```


Get interception for specific LIID

- Request:

```
curl -i --insecure --user administrator:administrator -X GET \
https://localhost:1443/api/interceptions/?liid=9876
```

- Response:


```
HTTP/1.1 200 OK
Server: nginx
Date: Tue, 01 Dec 2015 09:50:41 GMT
ContentType: application/hal+json;
profile="http://purl.org/sipwise/ngcpapi/";
  charset=utf8
ContentLength: 1283
Connection: keepalive
SetCookie:
ngcp_panel_session=23960dde6bb90f0c5c84575890194c53cce120ce; path=/;
  expires=Tue, 01 Dec 2015 10:50:40 GMT; HttpOnly
StrictTransportSecurity: maxage=15768000
{
   "_embedded" : {
      "ngcp:interceptions" : {
         "_links" : {
            "collection" : {
```


Copyright © Sipwise GmbH - All rights reserved

```
                "href" : "/api/interceptions/"
             },
             "curies" : {
                "href" : "http://purl.org/sipwise/ngcpapi/#rel{rel}",
                "name" : "ngcp",
                "templated" : true
             },
             "profile" : {
                "href" : "http://purl.org/sipwise/ngcpapi/"
             },
             "self" : {
                "href" : "/api/interceptions/520"
             }
          },
          "id" : 520,
          "liid" : 1,
          "number" : "0014155550132",
          "x2_host" : "192.168.42.42",
          "x2_password" : null,
          "x2_port" : 3002,
          "x2_user" : null,
          "x3_host" : "192.168.42.42",
          "x3_port" : 3003,
          "x3_required" : true
       }
    },
    "_links" : {
       "curies" : {
          "href" : "http://purl.org/sipwise/ngcpapi/#rel{rel}",
          "name" : "ngcp",
          "templated" : true
       },
       "ngcp:interceptions" : {
          "href" : "/api/interceptions/520"
       },
       "profile" : {
          "href" : "http://purl.org/sipwise/ngcpapi/"
       },
       "self" : {
          "href" : "/api/interceptions/?page=1&rows=10"
       }
    },
    "total_count" : 1
}
```


Create interception for a specific number

- Request:

 Copyright © Sipwise GmbH - All rights reserved

```
curl -i --insecure --user administrator:administrator -X POST \
-H "ContentType: application/json" --data \
'{"liid":123, "number":"31032222203", "x2_host":"127.0.0.1",
"x2_port":12345,
  "x3_required":true, "x3_host":"127.0.0.2", "x3_port":23456}' \
https://localhost:1443/api/interceptions/
```

- Response:

```
HTTP/1.1 201 Created
TransferEncoding: chunked
Connection: close
Location: /api/interceptions/528
SetCookie:
ngcp_panel_session=e7817079d121fae4d86448b10e1fa21d0201c526; path=/;
  expires=Tue, 01 Dec 2015 10:43:18 GMT; HttpOnly
StrictTransportSecurity: maxage=15768000
```

The path to the newly created interception is found in the *Location* header of the response.

Update specific interception

- Request:

```
curl -i --insecure --user administrator:administrator -X PUT \
-H "ContentType: application/json" -H 'Prefer: return=representation'
--data \
'{"liid":918273, "number":"0014155550132", "x2_host":"192.168.42.42",
"x2_port":5000,
  "x3_required":false}' \
https://localhost:1443/api/interceptions/123
```

- Response:

Copyright © Sipwise GmbH - All rights reserved

```
HTTP/1.1 200 OK
ContentType: application/hal+json;
profile="http://purl.org/sipwise/ngcpapi/";
  charset=utf8
ContentLength: 621
Link: </api/interceptions/>; rel=collection
Link: <http://purl.org/sipwise/ngcpapi/>; rel=profile
Link: </api/interceptions/530>; rel=self
PreferenceApplied: return=representation
SetCookie:
ngcp_panel_session=0b56e4a197b0e9f6e22a998e85473a0184770740; path=/;
  expires=Tue, 01 Dec 2015 10:56:17 GMT; HttpOnly
{
   "_links" : {
      "collection" : {
         "href" : "/api/interceptions/"
      },
      "curies" : {
         "href" : "http://purl.org/sipwise/ngcpapi/#rel{rel}",
         "name" : "ngcp",
         "templated" : true
      },
      "profile" : {
         "href" : "http://purl.org/sipwise/ngcpapi/"
      },
      "self" : {
         "href" : "/api/interceptions/530"
      }
   },
   "id" : 530,
   "liid" : 918273,
   "number" : "0014155550132",
   "x2_host" : "192.168.42.42",
   "x2_password" : null,
   "x2_port" : 5000,
   "x2_user" : null,
   "x3_host" : null,
   "x3_port" : null,
   "x3_required" : false
}
```

The *Prefer: return=representation* header forces the API to return the content, otherwise status *201* with no content is returned.

Update only certain items for a specific interception

- Request:

Copyright © Sipwise GmbH - All rights reserved

```
curl -i --insecure --user administrator:administrator -X PATCH \
-H "ContentType: application/jsonpatch+json" -H 'Prefer:
return=representation' \
--data '[{"op":"replace", "path":"/x2_host",
"value":"192.168.42.42"},{"op":"replace",
  "path":"/x2_port", "value":4000}]' \
https://localhost:1443/api/interceptions/530
```

- Response:

Copyright © Sipwise GmbH - All rights reserved

```
HTTP/1.1 200 OK
Server: nginx
Date: Tue, 01 Dec 2015 10:06:06 GMT
ContentType: application/hal+json;
profile="http://purl.org/sipwise/ngcpapi/";
  charset=utf8
ContentLength: 620
Connection: close
Link: </api/interceptions/>; rel=collection
Link: <http://purl.org/sipwise/ngcpapi/>; rel=profile
Link: </api/interceptions/530>; rel=self
PreferenceApplied: return=representation
SetCookie:
ngcp_panel_session=0693129d63d543a85f96d464ff9a8f807cfc4d18; path=/;
  expires=Tue, 01 Dec 2015 11:06:06 GMT; HttpOnly
StrictTransportSecurity: maxage=15768000
{
   "_links" : {
      "collection" : {
         "href" : "/api/interceptions/"
      },
      "curies" : {
         "href" : "http://purl.org/sipwise/ngcpapi/#rel{rel}",
         "name" : "ngcp",
         "templated" : true
      },
      "profile" : {
         "href" : "http://purl.org/sipwise/ngcpapi/"
      },
      "self" : {
         "href" : "/api/interceptions/530"
      }
   },
   "id" : 530,
   "liid" : 918273,
   "number" : "0014155550132",
   "x2_host" : "192.168.42.42",
   "x2_password" : null,
   "x2_port" : 4000,
   "x2_user" : null,
   "x3_host" : null,
   "x3_port" : null,
   "x3_required" : false
}
```

Delete specific interception

- Request:

Copyright © Sipwise GmbH - All rights reserved

```
curl -i --insecure --user administrator:administrator -X DELETE \
https://localhost:1443/api/interceptions/123
```

- Response:

```
HTTP/1.1 204 No Content
Server: nginx
Date: Tue, 01 Dec 2015 10:08:49 GMT
Connection: keepalive
SetCookie:
ngcp_panel_session=570c66b66732629766f86b8ed9bd0d64902ae73e; path=/;
  expires=Tue, 01 Dec 2015 11:08:49 GMT; HttpOnly
XCatalyst: 5.90042
StrictTransportSecurity: maxage=15768000
```

X2 Interface

The communication via the X2 interface consists of request-response pairs.

Request

The request is formatted as: X2/<bodylength>/<body>

Body part has the following items:

Table 37. X2 Message Body Items

Element	Type	Length	Description
/x2/header/source	String	arbitrary length	identifier of Sipwise node which captured the data
/x2/header/destination	String	arbitrary length	identifier of LI mediation system
/x2/header/type	String	arbitrary length	always "sip" (but later potentially "xmpp" and others too)
/x2/header/version	PosInteger	arbitrary length	always "1"
/x2/header/timestamp	String	27 chars	format: YYYY-MM-DDThh:mm:ss.ffffffZ; timestamp in UTC when the X2 package is sent to mediation
/x2/body/dialogid	PosInteger	arbitrary length	globally increasing counter for each new communication dialog (e.g. call)
/x2/body/messageid	PosInteger	arbitrary length	increasing counter for each new x2 message within a dialog, starting from 0
/x2/body/timestamp	String	27 chars	format: YYYY-MM-DDThh:mm:ss.ffffffZ; timestamp in UTC when the package has been captured on the wire

Copyright © Sipwise GmbH - All rights reserved

Element	Type	Length	Description
/x2/body/interceptions			one or more elements containing the following information, one element per intercepted target:
/x2/body/interceptions/liid	PosInteger	arbitrary length	interception id ("liid") as set via X1 interface
/x2/body/interceptions/direction	String	arbitrary length	either "totarget" or "fromtarget" from the soft-switch perspective (if target is the called party, it is "totarget", if target is the calling party, it is "fromtarget").
/x2/body/data	Base64 encoded	arbitrary	content of full IP frame and up on the OSI layer; packets fragmented on the wire are provided in fully assembled format

Example of full message:

```
X2/418/
{
  "header": {
    "source": "prx01a.example.com",
    "destination": "x2destination.example.com",
    "type": "sip",
    "version": 1,
    "timestamp": "20150311T09:18:04.729803Z"
  },
  "body": {
    "dialogid": 4,
    "messageid": 0,
    "timestamp": "20150311T09:18:04.729123Z",
    "interceptions": [
      { "liid": 174, "direction": "fromtarget" },
      { "liid": 175, "direction": "totarget" }
    ],
    "data": "<base64 encoded ip,udp/tcp,sip frame>"
  }
}
```

Response

- Success: X2-ACK/0/
- Error: X2-ERR/<length>/<error string>

Keep-Alive Mechanism

A regular keep-alive mechanism with a default value of 10s is used on the connection if it is re-used across multiple messages.

- Request: X2/0/
- Response: X2-ACK/0/

Copyright © Sipwise GmbH - All rights reserved

X3 Interface

On the X3 interface TLV based packets are sent via secured (TLS) connection on a pre-established stream. X3 messages do not need to be acknowledged, except for keep-alive messages.

X3 Message Structure

Table 38. X3 Message Structure

Field	Length
Header	arbitrary
CCCID	4 bytes
MessageId	4 bytes
Timestamp	8 bytes
Payload	arbitrary

Header Details

Table 39. X3: Header Details

Field	Length	Content
type	2 bytes	always "X3"
delimiter	1 byte	always "/"
length	arbitrary	ASCII string
delimiter	1 byte	always "/"

CCCID Details

dialogid (32 bit in network byte order, reset to 0 after 2^{32}-1)

The dialogid is referencing the /x2/body/dialogid field in order to correlate an X3 packet to an X2 call.

MessageId Details

messageid (32 bit in network byte order, reset to 0 after 2^{32}-1)

The messageid is a counter within a dialog sequencing the X3 packets sent from the NGCP. This counter is not correlated in any way with X2, rather than starting at 0 with the first RTP packet captured within a dialog.

Timestamp Details

- seconds (32 bit in network byte order)
- fraction (32 bit in network byte order)

The timestamp represents the Unix epoch starting from 1970-01-01.

Payload Details

Copyright © Sipwise GmbH - All rights reserved

Table 40. X3: Payload Details

Field	Length
original ip header	20 bytes for v4, 40 bytes for v6
original udp header	8 bytes
original rtp header	variable, 12-72 bytes
original rtp payload	arbitrary

Keep-Alive Mechanism

A regular keep-alive mechanism with a default value of 10s is used on the connection if it is re-used across multiple messages.

- Request: X3/0/
- Response: X3-ACK/0/

8.4. 3rd Party Call Control

8.4.1. Introduction

The Sipwise C5 offers the possibility to perform call control through 3rd party applications. This functionality, called ***Party Call Control*** and referred to as **"PCC"** throughout this handbook, is available since mr5.1.1 release.

Incoming calls to local subscribers may be signalled to a 3rd party CAC (Call Admission Control) server. Before accepting (that is: sending the SIP *INVITE* request to the called subscriber) or rejecting the call, Sipwise C5 will wait for an explicit reply from the CAC / PCC server, or a timeout.

Short Messages received by Sipwise C5 for a local subscriber may also be signalled to the PCC server. After an explicit reply with "accepted" status from the PCC server, Sipwise C5 will forward the SM to the final recipient.

IMPORTANT Sipwise C5 does not support delivering SMs to the local subscribers directly. Local subscribers can define a *Call Forward for SMS* instead, thus allowing themselves to receive SMs on their mobile phones.

3rd party call control may be implemented in many ways, such as by server-side or client-side applications (e.g. smartphone app).

NOTE Please note that Sipwise C5 implements a proprietary protocol for PCC deployments and adapting the protocol to customer needs requires software development from Sipwise.

8.4.2. Details of Call Processing with PCC

Copyright © Sipwise GmbH - All rights reserved

Overview

The following figure presents the schema of incoming call processing when PCC is involved:

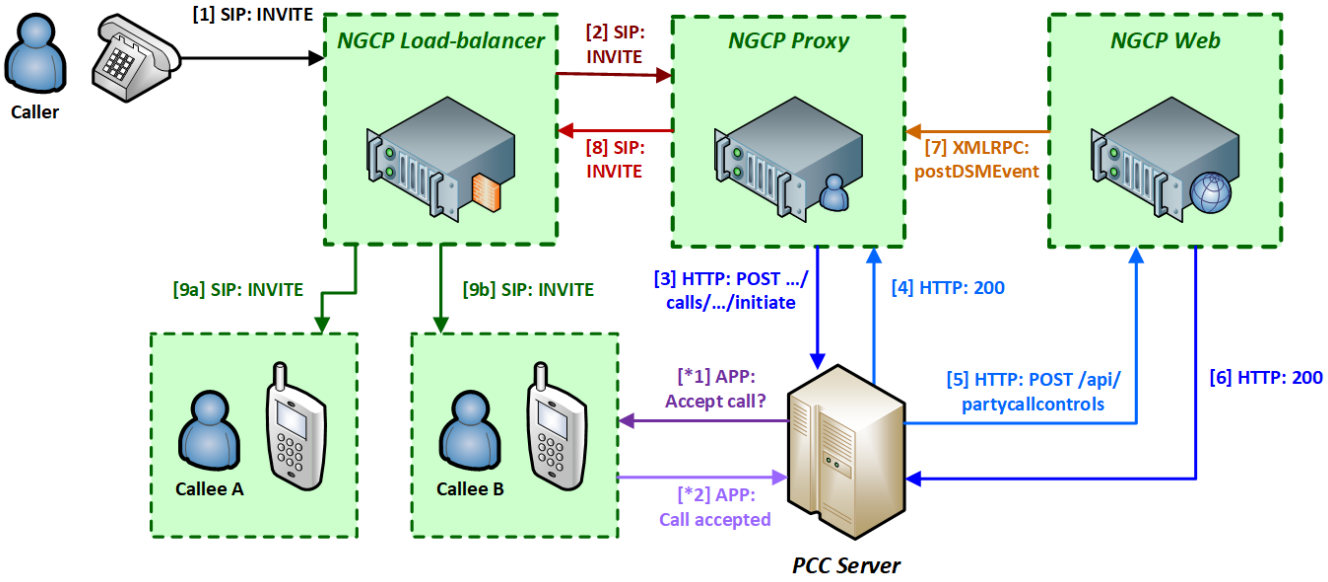


Figure 175. Overview of Party Call Control

The messages / interactions of PCC call processing are:

1. Sipwise C5 Load-Balancer receives a SIP *INVITE* message from the caller.
2. The LB forwards the *INVITE* to the PROXY component as usual with every incoming call.
3. The PROXY (*kamailio-proxy* module) checks whether the called subscriber has the PCC feature activated. If this is the case, it will send an HTTP *POST* or *GET* request (configurable) to the PCC server with the most important details of the call (such as calling and called party numbers, call-ID, a token for internal identification of the session).
4. The PCC server replies with *200 OK* HTTP status in order to indicate that it understood the request and will provide the final status (such as *ACCEPTED* or *REJECTED*) of the call later.

   Optional:

   *1) The PCC server requests the subscriber's confirmation to accept the call for instance via a smartphone app.

   *2) The subscriber indicates accepting the call to the PCC server.
5. The PCC server send an HTTP *POST* request to the WEB component of NGCP, using Sipwise C5 REST API, to signal accepting the call.
6. The WEB will reply with *200 OK* HTTP status.
7. The WEB sends an internal XMLRPC request to PROXY indicating that the incoming call can be accepted.
8. The PROXY sends the SIP *INVITE* message to the LB, i.e. it continues the call setup as usual.
9. The LB sends the *INVITE* to the subscriber.

There are more software modules within NGCP's components and those are shown separately on the diagrams in following sections of the handbook. For instance the PROXY component has the *kamailio-*

Copyright © Sipwise GmbH - All rights reserved

proxy and *sems-b2b* modules.

Successful Call Initiation at PCC Server

A subscriber with PCC activated will not receive the SIP *INVITE* request directly, but only after a series of intermediate CAC (Call Admission Control) steps, involving Sipwise C5 Proxy and the PCC server. First of those steps is the call initiation at the PCC server:

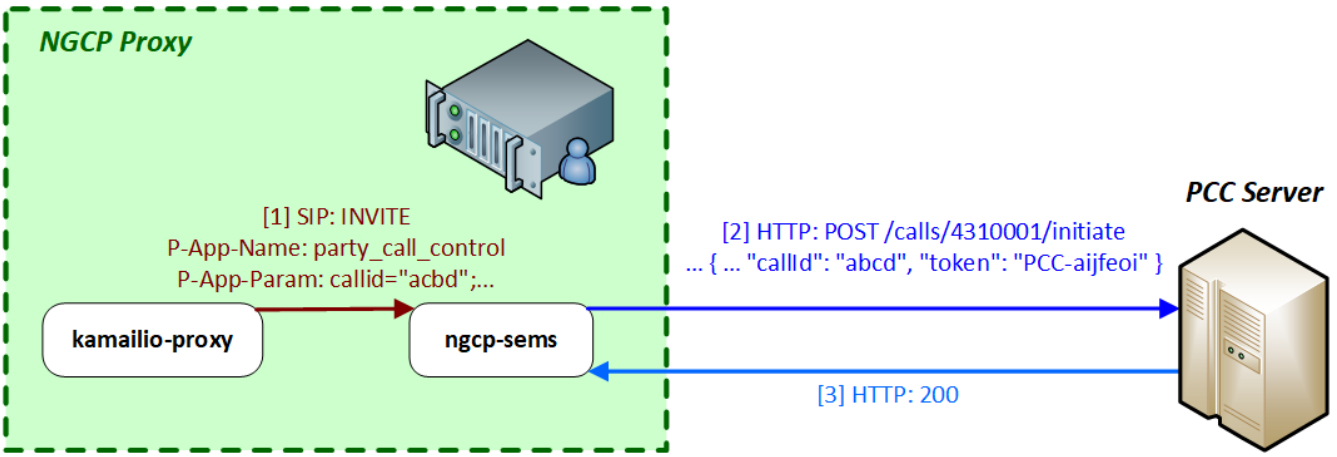


Figure 176. Successful Call Initiation with PCC

1. When *kamailio-proxy* receives the *INVITE* request from Sipwise C5 LB, it will forward the message to *sems-b2b* module with 2 private SIP headers:

   ```
   P-App-Name: party_call_control
   P-App-Param:
   callid="acbd";caller="4369912345";callee="4310001";caller_clir="0";
   ```

2. These headers will activate the PCC function in *sems-b2b* and it will send an HTTP *POST* request to the PCC server, instead of creating the second call leg directly towards Sipwise C5 LB. An example of such a request (not all details included):

   ```
   POST /calls/4310001/initiate HTTP/1.1
   Content-Type: application/json

   {
     "actualMsisdn": 4369912345,
     "callingMsisdn": 4310001,
     "actualClir": 0,
     "callId": "abcd",
     "token": "PCC-aijfeoi"
   }
   ```

   where:

   actualMsisdn: calling party number

   callingMsisdn: called party number

Copyright © Sipwise GmbH - All rights reserved

actualClir: non-0 if CLIR is active

callid: the SIP Call-ID

token: a generated token that identifies the session between Sipwise C5 and the PCC server

The target URL has the format: /calls/<called_party_num>/**initiate**

3. The PCC server replies with HTTP *200 OK* if it understood the request and can proceed with working on that.

Call Initiation at PCC Server with Error

The *sems-b2b* module on Sipwise C5 Proxy will wait for a response from PCC server, once it has sent the "initiate" request to it. If the PCC server responds with an HTTP error status, such as any *4xx*, then *sems-b2b* reports the error condition of PCC server with a SIP *487 Request Terminated* reply to *kamailio-proxy*.

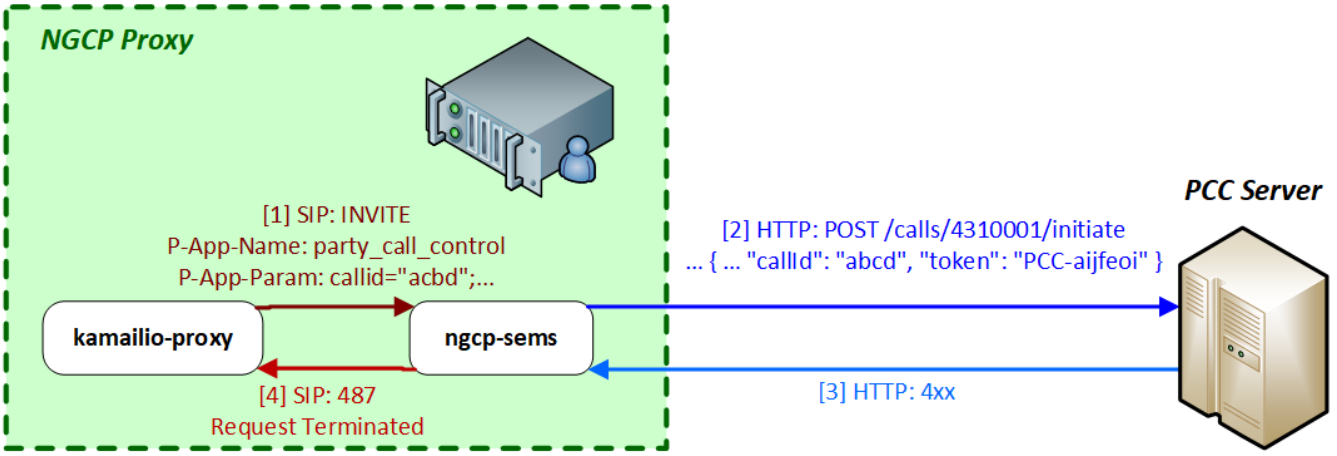


Figure 177. Call Initiation Error with PCC

Call Initiation at PCC Server with Timeout

The *sems-b2b* module on Sipwise C5 Proxy will wait for a response from PCC server, once it has sent the "initiate" request to it. If the PCC server does not respond with HTTP *200 OK* within 30 seconds (configurable) then *sems-b2b* considers the PCC is not available. In such a case *sems-b2b* sends a SIP *408 Timeout* reply to *kamailio-proxy*.

Copyright © Sipwise GmbH - All rights reserved

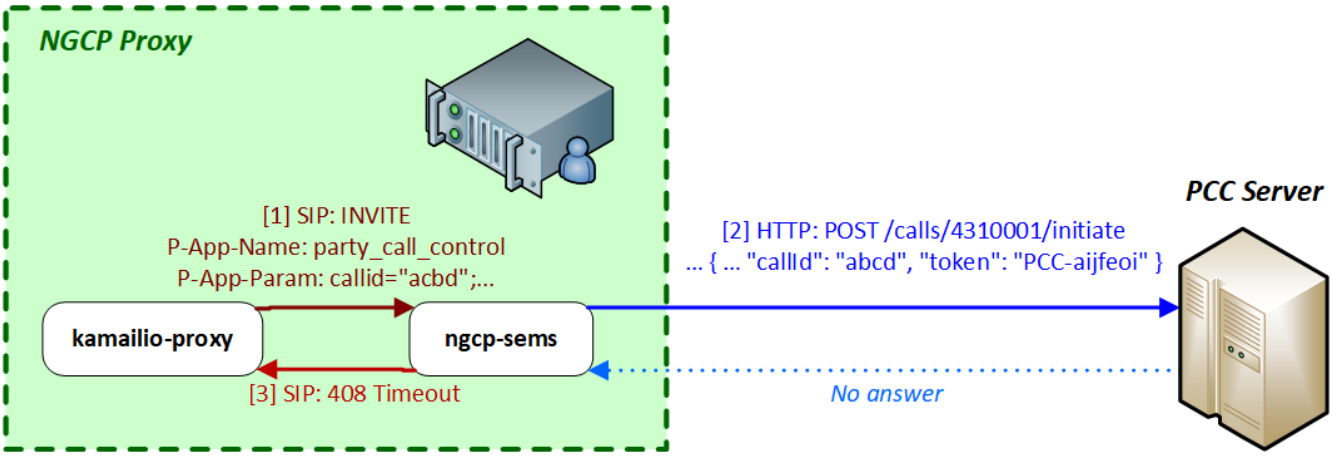


Figure 178. Call Initiation Timeout with PCC

Call Accepted by PCC Server

If the PCC server (eventually this may also be the called subscriber) accepts the call, the PCC server will send an HTTP *POST* request to the REST API interface of Sipwise C5 (Web/Management component). This request must contain a status field with the content ACCEPT (configurable) so that Sipwise C5 continues the call setup towards called party. Example:

```
POST /api/partycallcontrols HTTP/1.1
Content-Type: application/json

{
  "type": "pcc",
  "caller": 4369912345,
  "callee": 4310001,
  "status": "ACCEPT",
  "callId": "abcd",
  "token": "PCC-aijfeoi"
}
```

The target URL of the request: /api/**partycallcontrols**. The **type** parameter must have a value of **pcc**.

You can see the flow of messages in the diagram below:

Copyright © Sipwise GmbH - All rights reserved

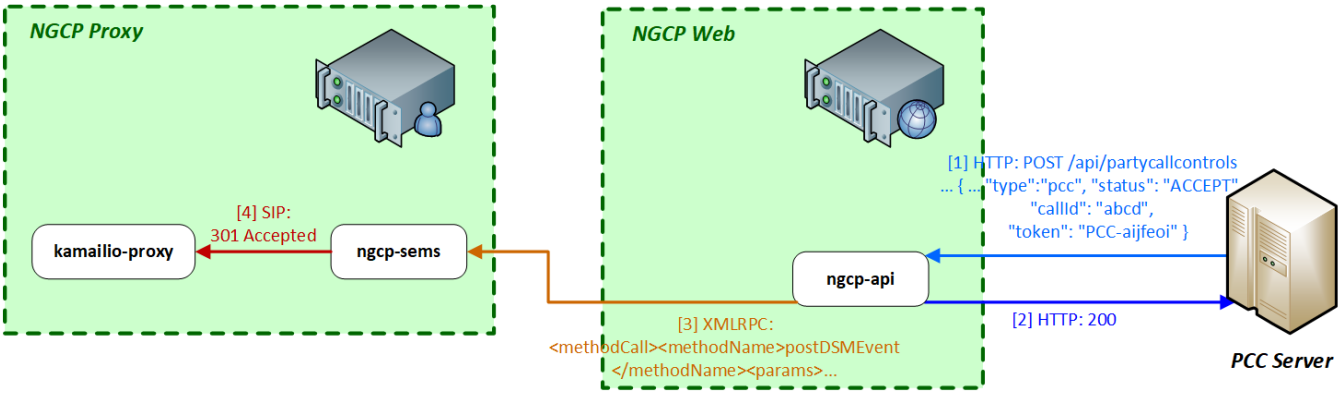


Figure 179. Call Accepted by PCC

1. The PCC server sends an HTTP *POST* request to NGCP's REST API.
2. Sipwise C5 Web will reply with *200 OK* HTTP status once the request is validated.
3. The *ngcp-panel* module generates an XMLRPC call to the *sems-b2b* module on the PROXY. An example is shown here:

Copyright © Sipwise GmbH - All rights reserved

```
<?xml version="1.0"?>
<methodCall>
  <methodName>postDSMEvent</methodName>
  <params>
    <param>
      <value><string>PCC-aijfeoi</string></value>
    </param>
    <param>
      <value><array><data>
        <value><array><data>
          <value><string>cmd</string></value>
          <value><string>handleCall</string></value>
        </data></array></value>
        <value><array><data>
          <value><string>callid</string></value>
          <value><string>abcd</string></value>
        </data></array></value>
        <value><array><data>
          <value><string>caller</string></value>
          <value><string>4369912345</string></value>
        </data></array></value>
        <value><array><data>
          <value><string>callee</string></value>
          <value><string>4310001</string></value>
        </data></array></value>
        <value><array><data>
          <value><string>status</string></value>
          <value><string>ACCEPT</string></value>
        </data></array></value>
      </data></array></value>
    </param>
  </params>
</methodCall>
```

At this point *sems-b2b* examines the following:

whether the token (listed as first param parameter of postDSMEvent) matches any of the saved session tokens

whether the callid parameter's value matches the session's SIP Call-ID

whether the status parameter's value is ACCEPT (configurable)

and if all those conditions are valid it will indicate to *kamailio-proxy* module that the call can be accepted (i.e. call setup towards the callee may continue).

4. *sems-b2b* module sends *301 Accepted* SIP response to *kamailio-proxy* and the latter can forward the SIP *INVITE* message to Sipwise C5 LB. If the status parameter's value is not ACCEPT (configurable), *sems-b2b* will reply *487 Request Terminated* to *kamailio-proxy*.

Copyright © Sipwise GmbH - All rights reserved

Indicating Call Termination at PCC Server

In the same manner as call initiation happens, call termination is also reported by Sipwise C5 towards the PCC server.

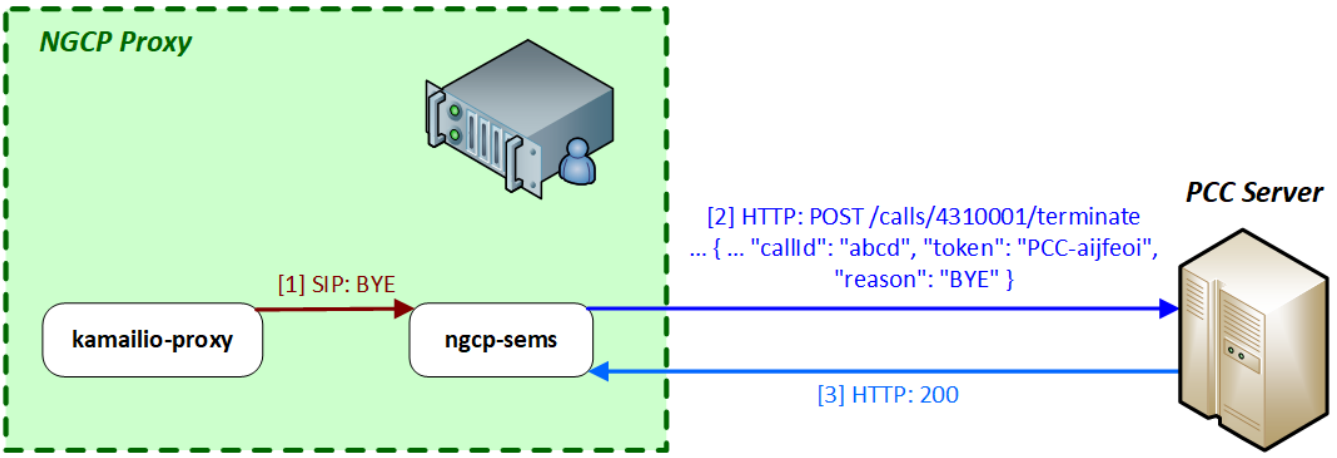


Figure 180. Call Termination with PCC

The target URL of the HTTP *POST* request for the call termination case looks like: /calls/<called_party_num>/**terminate**

The body of the request must contain the following element: "reason": "BYE", where the reason can be one of BYE, CANCEL, NOANSWER and REJECT. An example of a call termination request:

```
POST /calls/4310001/terminate HTTP/1.1
Content-Type: application/json

{
  "actualMsisdn": 4369912345,
  "callingMsisdn": 4310001,
  "actualClir": 0,
  "callId": "abcd",
  "token": "PCC-aijfeoi",
  "reason": "BYE"
}
```

Sipwise C5 will not take the response of PCC server into consideration, because the call has already been terminated at SIP protocol level.

8.4.3. Voicemail Notification

Using the PCC Framework

The PCC call control framework may also be used for voicemail notifications. The Sipwise C5 involves its elements: *asterisk* (Voicemail server) and *ngcp-vmnotify* in the process of the notification.

Copyright © Sipwise GmbH - All rights reserved

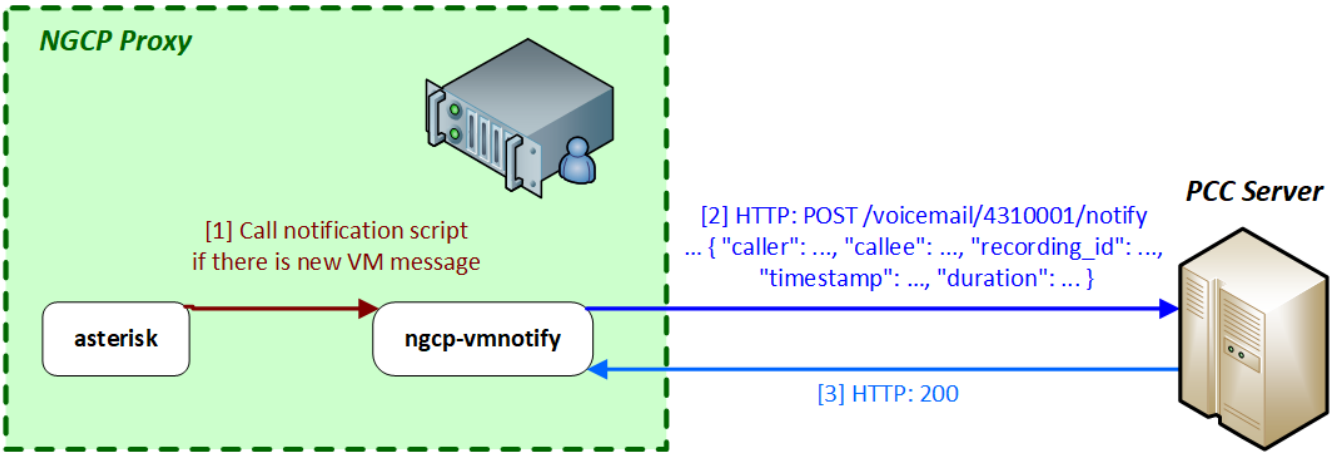


Figure 181. Voicemail Notification with PCC

1. The *asterisk* voicemail server triggers the *ngcp-vmnotify* script when a caller leaves a voicemail message in the callee's voicebox.
2. *ngcp-vmnotify* sends an HTTP *POST* request to the PCC server, as given in the example below:

```
POST /voicemail/4310001/notify HTTP/1.1
Content-Type: application/json

{
  "caller": 4369912345,
  "callee": 4310001,
  "recording_id":  45235 ,
  "timestamp": "2017-06-13T14:21:17T+01:00",
  "duration": 17
}
```

   The target URL is: /**voicemail**/<called_party_num>/**notify**

3. The PCC server replies with *200 OK* if it properly processed the request.

Using SMS

The Sipwise C5 also supports voicemail notifications in form of short messages, using the built-in SMS modules. In such a case the *ngcp-vmnotify* module will send an HTTP *POST* request to the REST API (Sipwise C5 Web), that will contain the short message and finally be stored in the central database. Afterwards the short message will be sent to the recipient by Sipwise C5 Proxy.

Copyright © Sipwise GmbH - All rights reserved

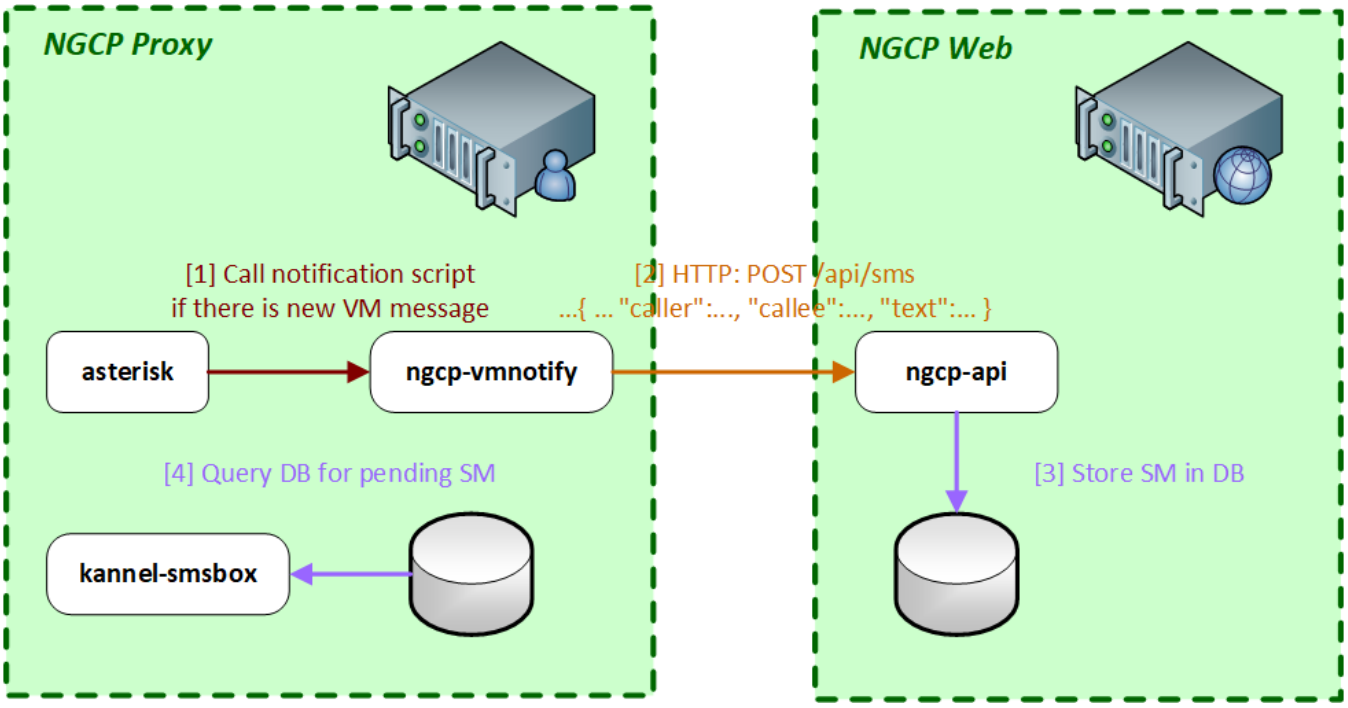


Figure 182. Voicemail Notification with SMS

1. The *asterisk* voicemail server triggers the *ngcp-vmnotify* script when a caller leaves a voicemail message in the callee's voicebox.
2. *ngcp-vmnotify* sends an API request to *ngcp-api* module, as given in the example below:

```
POST /api/sms/?skip_checks=true&skip_journal=false HTTP/1.1
Content-Type: application/json

{
  "subscriber_id": 90
  "caller": 4369912345,
  "callee" : 4310001,
  "text":  "user1 4310001 17 Tue 13 Jun 2017 14:21:17 +01:00"
}
```

   The target URL is: /api/**sms**

3. The *ngcp-api* stores the message in the database.
4. The *kannel-smsbox* module of Sipwise C5 Proxy will query the database for messages waiting for delivery and send the SM to its recipient through Sipwise C5 LB.

8.4.4. Incoming Short Message Acceptance

Indicating Incoming SM to PCC Server

The PCC server may also serve as a control point for incoming short messages. The Sipwise C5 may indicate an incoming SM to the PCC server, which in turn must explicitly accept the message, so that the message will be forwarded to the recipient.

Copyright © Sipwise GmbH - All rights reserved

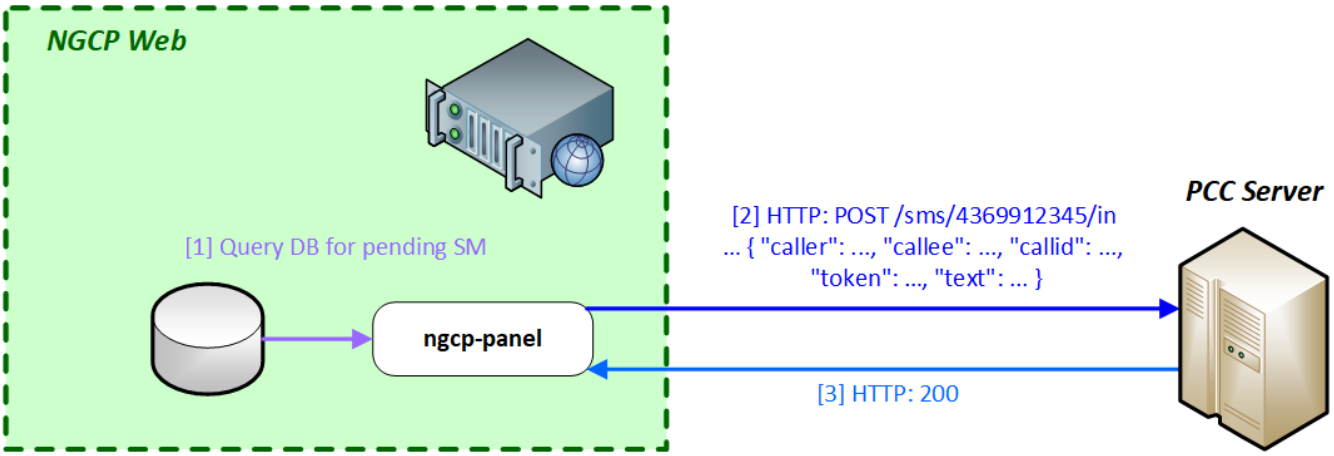


Figure 183. Short Message Notification with PCC

1. The *ngcp-panel* module on Sipwise C5 Web component will query the central database for pending incoming SMs.
2. The *ngcp-panel* will send an HTTP *POST* request to the PCC server if there is a message waiting for a subscriber. An example of such request is shown here:

   ```
   POST /sms/4310001/in HTTP/1.1
   Content-Type: application/json

   {
     "caller": 4369912345,
     "callee": 4310001,
     "token": "PCC-aijfeoi",
     "callId": "abcd",
     "text": "This is the SM text"
   }
   ```

   The target URL in this case is: /**sms**/<called_party_num>/**in**
3. The PCC server replies with *200 OK* HTTP status if it properly understood the request.

Incoming SM Accepted by PCC Server

As in the case of an incoming call, the PCC server will send an HTTP *POST* request to the REST API of NGCP, in order to signal the acceptance of the SM.

Copyright © Sipwise GmbH - All rights reserved

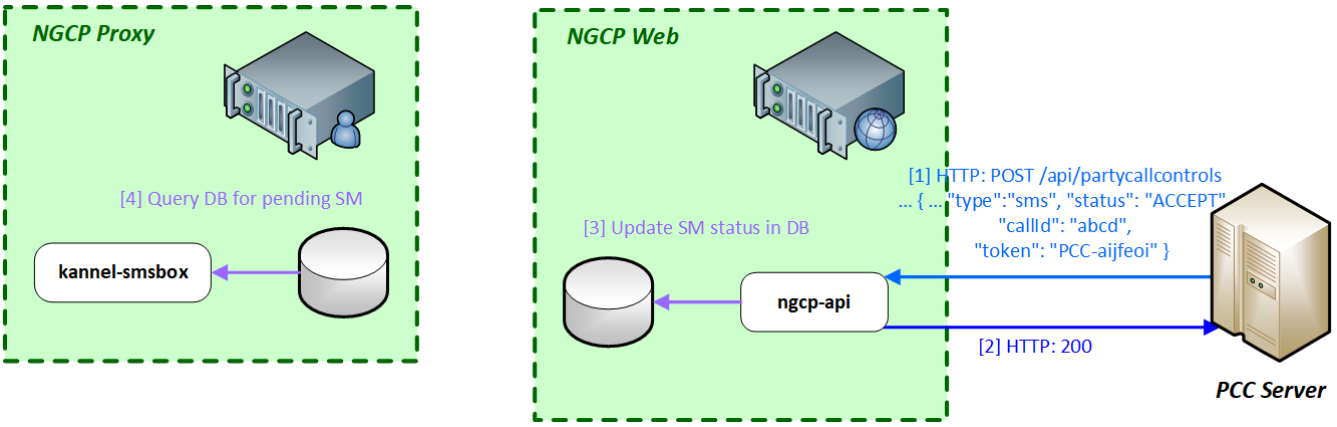


Figure 184. Short Message Accepted by PCC

1. The PCC server sends the request to Sipwise C5 Web component, where *ngcp-api* module will process it. An example:

```
POST /api/partycallcontrols HTTP/1.1
Content-Type: application/json

{
  "type": "sms",
  "caller": 4369912345,
  "callee": 4310001,
  "status": "ACCEPT",
  "callId": "abcd",
  "token": "PCC-aijfeoi"
}
```

   The target URL of the request: /api/**partycallcontrols**. The **type** parameter must have a value of **sms**.

2. The *ngcp-api* module responds with *200 OK* HTTP status if it properly understood the request.
3. The *ngcp-api* updates the status of the SM in the database so that the SM may be forwarded to the recipient.
4. The *kannel-smsbox* module on Sipwise C5 Proxy will query the central database for SMs to be delivered and will forward the SM towards an SMSC, via Sipwise C5 LB.

8.4.5. Configuration of PCC

The configuration of the PCC feature is done via the main configuration file: /etc/ngcp-config/config.yml. The relevant section is: apps.party_call_control, the example below shows the default values of the parameters.

Copyright © Sipwise GmbH - All rights reserved

```
apps:
  party_call_control:
    accepted_reply: 200*
    enable: no
    pcc_server_url:
https://127.0.0.1:9090/pcc/${prefix}${callee}${suffix}
    request_timeout: '30'
    trigger_on_hangup: yes
```

The configuration parameters are:

- accepted_reply: defines the value of status data element (in the PCC server's *POST* request sent to /api/partycallcontrols API resource) that means the "accepted" status of the call. For instance the handbook showed the value `ACCEPT` in previous sections, instead of the default `200*`
- enable: must be set to `yes` in order to enable the PCC feature
- pcc_server_url: the URL, pointing to the PCC server, where HTTP *POST* requests must be sent. The variables `${prefix}, ${callee}` and `${suffix}` will be replaced with actual values when a request is sent. *Please do not change* this part of the URL! Possible values are:

  prefix = calls, suffix = initiate

  prefix = calls, suffix = terminate

  prefix = voicemail, suffix = notify

  prefix = sms, suffix = in

  callee = <called_party_num>
- request_timeout: time in seconds until Sipwise C5 will wait for an HTTP reply from the PCC server, once Sipwise C5 has sent a request to it
- trigger_on_hangup: if set to `yes`, Sipwise C5 will send a "terminate" request to the PCC server at the end of the call

8.4.6. Troubleshooting of PCC

The Sipwise C5 will provide logs of its activities that are very useful for troubleshooting the call processing with PCC feature. This section will provide examples from various log files that can help to find potential problems in call setup.

Kamailio Proxy Log

PCC activation at *sems-b2b* module

```
Oct 17 17:00:45 prx01a proxy[3206]: NOTICE: <script>: Call to PCC (Party
Call Control) - R=sip:2133339@192.168.10.11:5060;user=phone
ID=1849964028_125696279@10.0.0.121 UA='<null>'
```

Call accepted by PCC server

Copyright © Sipwise GmbH - All rights reserved

```
Oct 17 17:00:16 prx01a proxy[3210]: NOTICE: <script>: NAT-Reply - S=301
- Accepted M=INVITE IP=192.168.10.12:5080 (192.168.10.12:5080)
ID=1850250074_83465152@10.0.0.121 UA='<null>'
Oct 17 17:00:16 prx01a proxy[3210]: INFO: <script>: Received 200 OK
(Accepted) from PCC Server, routing the call to its original callee -
ID=1850250074_83465152@10.0.0.121 UA='<null>'
```

SEMS Log

Initiate call at PCC

```
Oct 17 17:10:47 prx01a sems[5059]: [#7f73237f7700] [mod_py_log,
PyDSM.cpp:42] INFO: PCC http request to
http://example.com/pcc/calls/4366811112222/initiate - callid
1851794724_134068006@10.0.0.121
Oct 17 17:10:47 prx01a sems[5059]: [#7f73237f7700] [mod_py_log,
PyDSM.cpp:42] INFO: PCC form data: {'actualMsisdn': '4369933334444',
'actualClir': '0', 'token': 'PCC-12DBBD25-59E61D770001841C-237F7700',
'callingMsisdn': '4366811112222', 'callId':
'1851794724_134068006@10.0.0.121'} - callid
1851794724_134068006@10.0.0.121
Oct 17 17:10:47 prx01a sems[5059]: [#7f73237f7700] [mod_py_log,
PyDSM.cpp:42] INFO: PCC ret: 0 num_handles: 1
Oct 17 17:10:47 prx01a sems[5059]: [#7f73237f7700] [mod_py_log,
PyDSM.cpp:42] INFO: RT: 0 1 0 [] []
...
Oct 17 17:10:47 prx01a sems[5059]: [#7f73237f7700] [mod_py_log,
PyDSM.cpp:42] INFO: RT: 0 0 0 [<pycurl.Curl object at 0x7f7378067c50>]
[]
Oct 17 17:10:47 prx01a sems[5059]: [#7f73237f7700] [mod_py_log,
PyDSM.cpp:42] INFO: PCC reply for callid
1851794724_134068006@10.0.0.121: 200
```

Call accepted by PCC server

```
Oct 17 17:10:51 prx01a sems[5059]: [#7f7323efe700] [execute,
XMLRPC2DI.cpp:714] INFO: XMLRPC2DI 'postDSMEvent': function
'postDSMEvent'
Oct 17 17:10:51 prx01a sems[5059]: [#7f7323efe700] [execute,
XMLRPC2DI.cpp:718] INFO:  params: <['PCC-12DBBD25-59E61D770001841C-
237F7700', [['cmd', 'handleCall'], ['callid',
'1851794724_134068006@10.0.0.121'], ['caller', '4369933334444'],
['callee', '4366811112222'], ['status', 'ACCEPT']]]>
Oct 17 17:10:51 prx01a sems[5059]: [#7f7323efe700] [execute,
XMLRPC2DI.cpp:724] INFO:  result: <[200, 'OK']>
Oct 17 17:10:51 prx01a sems[5059]: [#7f73237f7700] [execute,
DSMCoreModule.cpp:521] INFO: FSM:  'PCC RESULT -- ACCEPT'
```

Copyright © Sipwise GmbH - All rights reserved

Terminate call at PCC

```
Oct 17 17:10:53 prx01a sems[5059]: [#7f73235f5700] [mod_py_log,
PyDSM.cpp:42] INFO: PCC http request to
http://example.com/pcc/calls/4366811112222/terminate - callid
1851794724_134068006@10.0.0.121
Oct 17 17:10:53 prx01a sems[5059]: [#7f73235f5700] [mod_py_log,
PyDSM.cpp:42] INFO: PCC form data: {'actualMsisdn': '4369933334444',
'callId': '1851794724_134068006@10.0.0.121', 'callingMsisdn':
'4366811112222', 'reason': 'CANCEL', 'token': 'PCC-12DBBD25-
59E61D770001841C-237F7700', 'actualClir': '0'} - callid
1851794724_134068006@10.0.0.121
```

Sipwise C5 Panel Log

SM notification at PCC server

```
Oct 18 09:10:16 web01a ngcp-panel: INFO: pcc is set to 1 for prov
subscriber id 18451
Oct 18 09:10:16 web01a ngcp-panel: INFO: >>>> source check for
booking.com passed, continue with time check
Oct 18 09:10:16 web01a ngcp-panel: INFO: >>>> time check for 1508310615
passed, use destination set
Oct 18 09:10:16 web01a ngcp-panel: INFO: >>>> proceed sms forwarding
Oct 18 09:10:16 web01a ngcp-panel: INFO: >>>> forward sms to
4369933334444
Oct 18 09:10:16 web01a ngcp-panel: INFO: sending pcc request for sms
with id 305125 to http://example.com/pcc/sms/4366811112222/in
Oct 18 09:10:16 web01a ngcp-panel: INFO: sending pcc request succeeded
Oct 18 09:10:16 web01a ngcp-panel: INFO: status for pcc sms of 305125 is
BUSY, don't forward sms
```

In the last line the status is BUSY. The purpose of this is to prevent forwarding the SM to the mobile phone of the recipient. Otherwise, in order to let Sipwise C5 forward the message to the recipient, the status is ACCEPT.

REST API Log

Call accepted by PCC server

Copyright © Sipwise GmbH - All rights reserved

```
Oct 18 10:19:39 web01a ngcp-panel: INFO: IP=192.168.10.20
CALLED=API[POST]/api/partycallcontrols/ TX=14EE9C4CD2599A70
USER=username DATA={} MSG=""
LOG="{"type":"pcc","caller":"4365033334444","callee":"4366811112222","st
atus":"ACCEPT","token":"PCC-273C2CDA-59E70E96000BE0C4-
231F1700","callid":"406885946_117428858@10.0.0.121"}"
Oct 18 10:19:39 web01a ngcp-panel: INFO: IP=192.168.10.20
CALLED=API[POST 200]/api/partycallcontrols/ TX=14EE9C4CD2599A70
USER=username DATA={} MSG="" LOG=""
```

SM accepted by PCC server

```
Oct 18 10:20:30 web01a ngcp-panel: INFO: IP=192.168.10.20
CALLED=API[POST]/api/partycallcontrols/ TX=14EE9C58CEA4D960
USER=username DATA={} MSG=""
LOG="{"type":"sms","caller":"15556666","callee":"4366811112222","status"
:"ACCEPT","token":"1482d9e2-a9fc-40ee-bdaf-
de6f7fc239f8","callid":"305175"}"
Oct 18 10:20:30 web01a ngcp-panel: INFO: IP=192.168.10.20
CALLED=API[POST 200]/api/partycallcontrols/ TX=14EE9C58CEA4D960
USER=username DATA={} MSG="" LOG=""
```

Voicemail Notification Log

The voicemail notifier program (ngcp-vmnotify) writes its log messages into the system log (/var/log/syslog). An example:

```
Oct 18 09:53:34 prx01a vmnotify[20072]: Arguments: default 4366811112222
1 0 0 0 4365033334444 2017-10-18T09:53:34+0200 8
```

Where the *Arguments* are:

- default: Asterisk voicemail context
- the voicemail box owner
- 1: number of new messages
- 0: number of old messages
- 0: number of urgent messages
- 0: message ID of the latest message
- who left the message (caller)
- date and time of the message
- 8: duration of the message in seconds

Copyright © Sipwise GmbH - All rights reserved

8.5. 3PCC functionality (TPCC), CSTA sessions and Websocket component

8.5.1. Abbreviations / definitions used in this chapter

- 3PCC - Third Party Call Control. Is a concept of managing calls from the separate network element (for e.g. through ngcp-websocket using CSTA). In a scope of this chapter, 3PCC is the .so module connected via the SEMS component and it implements the third party call control in Sipwise C5.

NOTE Sometimes 3PCC can be also written as TPCC.

- 1PCC - First Party Call Control. Is a usual/manual way of originating/managing calls. In other words, when Alice calls Bob manually using SIP hard- / softphone (she can put Bob on hold and then terminate session if needed), this is then called 1PCC.
- CSTA - Computer Supported Telecommunication Application(s). Is a protocol for voice service invocations using XML-based messages, which works via the WebSocket protocol (that implements a role of a transport mechanism for the CSTA messages). It is implemented by the ngcp-websocket component, which interacts with the Kamailio-Proxy for a purpose of calls monitoring and with the SEMS for calls control (3PCC).

NOTE Standard ECMA-269 - "Services for Computer Supported Telecommunications Applications (CSTA) Phase III" : https://www.ecma-international.org/publications-and-standards/standards/ecma-269/ Standard ECMA-323 - "XML protocol for Computer Supported Telecommunications Applications (CSTA) Phase III" : https://www.ecma-international.org/publications-and-standards/standards/ecma-323/ Standard ECMA-TR/82 - "Scenarios for Computer Supported Telecommunications Applications (CSTA) Phase III" : https://www.ecma-international.org/publications-and-standards/technical-reports/ecma-tr-82/

- FIFO - fifo socket (first in / first out), in other words program's in-queue / out-queue. In a scope of this chapter this is related to the SEMS, which has a dedicated FIFO implemented in the 3PCC (TPCC) module.
- SEMS - this is a sems-b2b component, through which the 3PCC (TPCC) module is connected to whole architecture/stack.
- ngcp-websocket - this is the websocket component and is used to interconnect the sems-b2b and the kamailio-proxy with the CSTA environment. The main purpose is to send/receive commands to the dedicated redis channels, which can trigger some job/action to be implemented either on sems-b2b/kamailio-proxy side or in the CSTA environment (for e.g. change of SIP call status).
- protobuf - protocol buffers is Google's language- / platform-neutral, extensible mechanism for serializing structured data – think XML, but smaller, faster, and simpler.

NOTE https://en.wikipedia.org/wiki/Protocol_Buffers

- WebSocket protocol - is a transport mechanism used for the CSTA messages.

NOTE WebSocket protocol is compatible with RFC6455: https://tools.ietf.org/html/rfc6455

Copyright © Sipwise GmbH - All rights reserved

8.5.2. Introduction

This chapter covers CSTA call flows of the 3PCC model and various extended telephony services based on that. It shows how the relations between different components involved into CSTA processing, make it possible to establish SIP-based sessions/dialogs. It includes different possible call scenarios, which can be done using CSTA/3PCC and how to use API in order to trigger them.

The CSTA model is implemented based on the ECMA-269 Phase III and ECMA TR/82 standards. It uses XML format with ECMA-323 XSD to carry required metadata over the WebSocket protocol.

In practice you can use your own web-based application or any other kind of application capable of sending IP-packets, to realize the following stack:

- web client (web GUI for user)
- CSTA via the WebSocket protocol
- ngcp-websocket component
- 3PCC (TPCC) module of the SEMS-b2b component

A good example of such integration is the Rainbow messenger developed by the Alcatel Lucent, which is used as a superstructure in a combination with Sipwise C5 to give users all capabilities of the CSTA.

8.5.3. CSTA and WebSocket protocol

CSTA session is being established using WebSocket protocol. Which in its turn makes it possible to control sessions via the ngcp-websocket component of Sipwise C5. As already mentioned before, the WebSocket protocol is compatible with the RFC6455.

Devices that are visible or controllable via CSTA are known as CSTA Devices.

CSTA Devices can be either physical devices (such as SIP hardphone) or logical sdevices (such as groups of devices, pilot numbers). CSTA Devices have attributes that allow CSTA to monitor and manipulate them.

The most important CSTA requests/event types used in the setup with Sipwise C5:

- *MakeCall* - originate a call with a remote side
- *HoldCall* - put a remote side on hold
- *ConsultationCall* - initiate a consultation call with a third side
- *ClearConnection* - finalize a connection/terminate a session
- *MonitorStart* - start monitoring of device
- *RequestSystemStatus* - establish a connection with the CSTA system (ngcp-websocket)
- *OriginatedEvent* - indicates to the caller, that an invitation has been sent to the remote device (however not answered yet)
- *ServiceInitiatedEvent* - indicates to the caller, that an invitation has been sent to the originating device (however not answered yet)
- *DeliveredEvent* - indicates that remote device has started to ring (180/183)
- *EstablishedEvent* - indicates that remote device has answered a call (200OK)

Copyright © Sipwise GmbH - All rights reserved

- *HoldEvent* - indicates that the session has been successfully put on hold
- *ConnectionClearedEvent* - indicates a call clearing/termination phase (due to whatever reason)
- *FailedEvent* - indicates that a previously requested event hasn't been delivered as intended (for e.g. on an invitation we've gotten 486 from a remote device)

Some of the most important attributes used in the setup with Sipwise C5:

- *monitorCrossRefID* - this is a monitor cross reference identifier parameter, which associates the event with the established monitor
- *initiatedConnection* - this parameter specifies the connection, at which a service was initiated
- *originatedConnection* - this parameter specifies the connection, at which a call was originated
- *callID* - an identifier which is used to represent a valid call
- *deviceID* - an identifier which is used to represent a device in the domain
- *callingDevice* - this parameter in the Call Control events represents the calling device associated with the call
- *calledDevice* - this parameter specifies an originally called device. In case of prompting, this parameter specifies the device that will be called after the prompt is answered
- *initiatingDevice* - this parameter specifies the initiating device
- *answeringDevice* - this parameter specifies the device that has been connected to the call
- *deviceIdentifier* - this parameter keeps the SIP URI of callingDevice/calledDevice/initiatingDevice/alertingDevice/answeringDevice
- *localConnectionInfo* - this parameter specifies the local connection state of device associated with the 'monitorCrossRefID'
- *cause* - this parameter specifies the reason for the event (for e.g.: 'newCall', 'makeCall' etc.)
- *connection* - this parameter specifies the connection that is alerting (ringing)
- *establishedConnection* - this parameter specifies the connection that has been established
- *alertingDevice* - this parameter specifies the device that is alerting (ringing)
- *lastRedirectionDevice* - this parameter specifies a previously known redirection from device (in most cases will be used with the 'notRequired' value)

A list of important details:

- Subprotocol (Sec-WebSocket-Protocol header) must have the value: 'csta'
- WebSocket frames should be of the 'text' type
- Packets fragmentation has to be not applied on CSTA packets
- WebSocket clients can send WebSocket ping requests periodically and the WebSocket server should respond to it

A connection with the ngcp-websocket

When using your own system with CSTA (presumably this can be a web platform with GUI which sends WebSocket requests of XML format), this system will have to establish a connection or a pool of connections with the ngcp-websocket component.

Copyright © Sipwise GmbH - All rights reserved

NOTE For simplicity reasons, let's name a client/system getting connected to the CSTA/ngcp-websocket as web-interface-1

Each connection between the web-interface-1 (client) and the ngcp-websocket upon establishment, requires the CSTA RequestSystemStatus message to be sent from the web-interface-1. Once a response from the ngcp-websocket is received, a session is considered to be established. Consider:

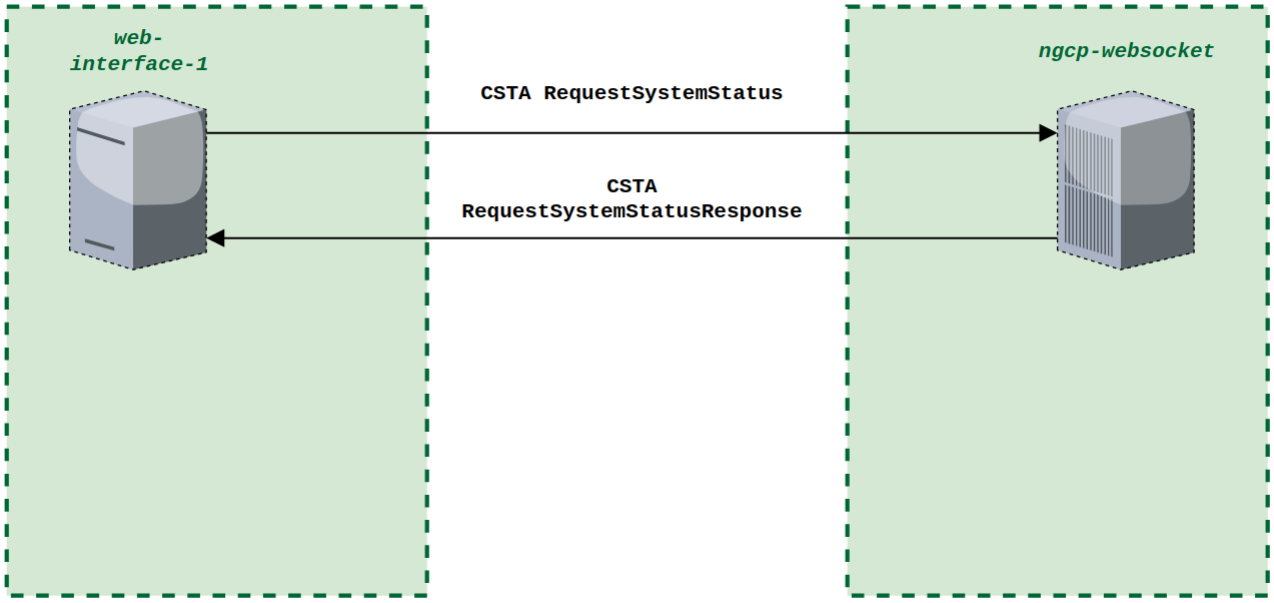


Figure 185. CSTA RequestSystemStatus messages exchange

An example of such messages exchange can be found here: CSTA Appendix, RequestSystemStatus.

Messages format

The content of CSTA messages is regulated with the ECMA-269 standard and is illustrated in the ECMA TR/82. XSD (XML Schema Definition) is described by ECMA-323: https://www.ecma-international.org/publications/standards/Ecma-323.htm

IMPORTANT As described in "5.2 Request/Response Protocol Requirements" of ECMA-323, the XML part doesn't provide mechanism for correlating a CSTA request with a CSTA response.

CSTA has a server/client model (request/answer), this is quite similar to what HTTP or SIP protocols have. The content of such message has headers and has a body. In 100% cases now the body will always be of the XML format of course.

At least two headers are obligatory and must be presented in each request/response of the WebSocket messages exchange:

- 'Content-Type:' - a content type of CSTA body, now it has only one possible value 'application/csta+xml'
- 'X-CSTA-Seq-ID:' - a sequence Id of CSTA request/response transaction. It is recommended that it must be unique within one NGCP/CSTA environment. It is incremented by 1 with every new request.

Copyright © Sipwise GmbH - All rights reserved

For e.g.: ab234_1, ab234_2, ab234_3, ... ab234_N, where ab234 is an unique ID.

8.5.4. Telephony scenarios

In this section of the chapter, it is described, how an interaction of the CSTA and SIP environment builds up 3PCC-controlled SIP sessions. In other words, how to use the CSTA (over the WebSocket protocol) to initiate, modify and properly terminate 3PCC-controlled sessions.

IMPORTANT For a proper work of CSTA, each device included into the 3PCC-controlled SIP session must be monitored.

NOTE In some scenarios, it can happen that it's just impossible to monitor a remote side, for e.g. in case when a call with a remote side is established via the SIP peering/media gateway. In such situation, the monitoring can be applied to the SIP peering/media gateway as the last visible point Sipwise C5 interacts with.

Monitoring of device

The web-interface-1 (client) needs to start a CSTA monitor for a given SIP device in order to observe evaluations on CSTA calls via CSTA events. To start a monitoring, a CSTA MonitorStart service message is being sent (as the body of a WebSocket request). Consider:

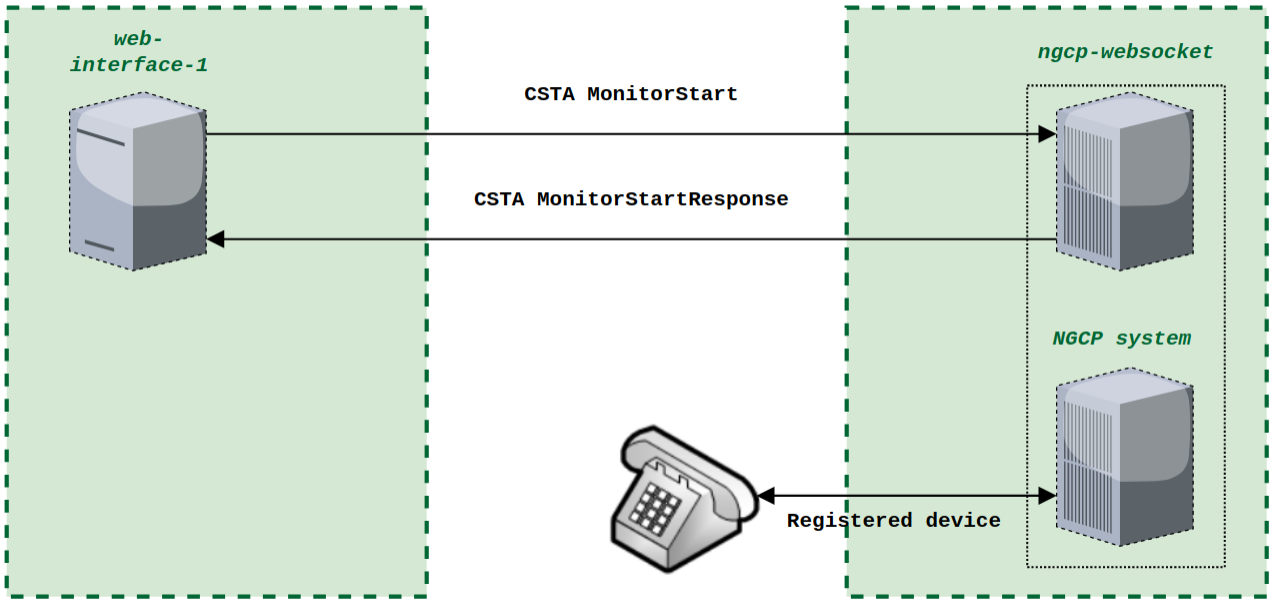


Figure 186. CSTA MonitorStart messages exchange

As you can see on the picture, the sequence is the following:

- Monitoring is started for the device with AOR: sip:1000@sipwise.com, and a message is sent from the web-interface-1 (MonitorStart)
- SIP device gets monitored and an unique monitor ID is generated for a given response (MonitorStartResponse)

Now, any time when certain event happens to this SIP device, ngcp-websocket will notify the web-

Copyright © Sipwise GmbH - All rights reserved

interface-1 about the ongoing event using the WebSocket message (DeliveredEvent).

An example of such messages exchange can be found here: CSTA Appendix MonitorStart.

As well as a monitoring of any device can be started, it can also be stopped using the 'monitorCrossRefID', which was previously given when establishing the monitoring session using the WebSocket MonitorStart request. After it's done no notifications/events will be generated for a given device. Consider:

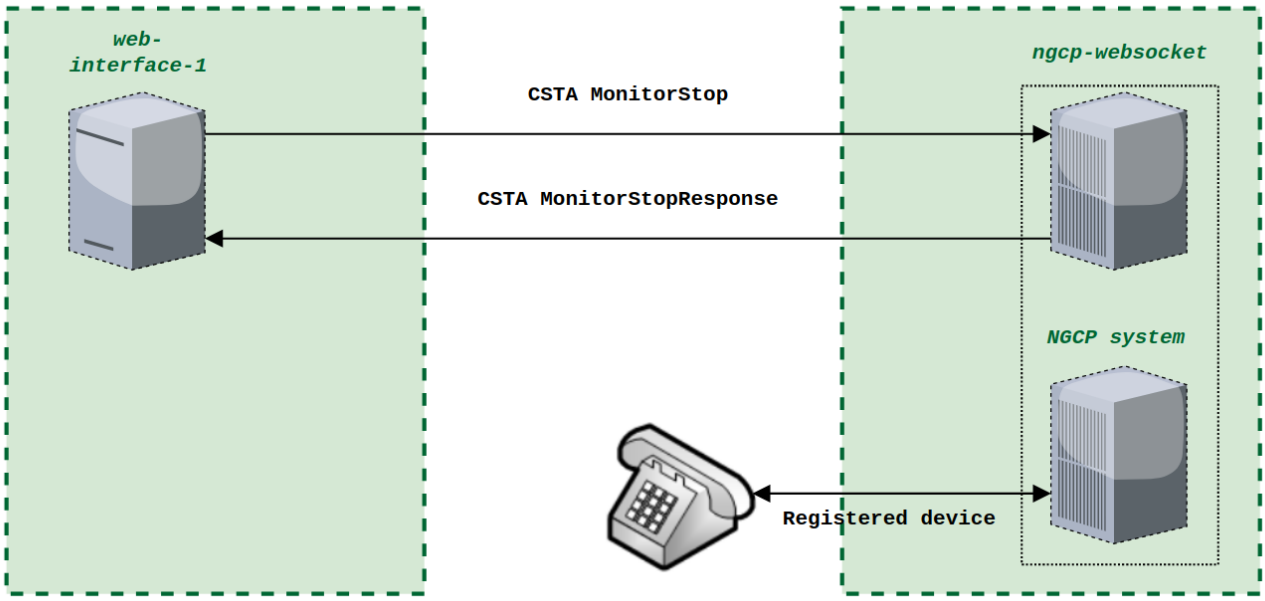


Figure 187. CSTA MonitorStop messages exchange

It could happen that a particular device is monitored by two monitors at the same time, in this situation when one of monitoring events has been stopped, remaining one should keep on delivering CSTA events.

Aspects of user identity

The CSTA requires a user identity, which includes SIP URI as well as a display name (if present) and must be stored using the '<deviceIdentifier>' tag in the XML body of the message, for e.g.:

```
<callingDevice><deviceIdentifier>&lt;Carol&gt;
sip:1002@company1.sipwise.com</deviceIdentifier></callingDevice>
```

It corresponds to a real user identity taken from the SIP message (from the P-Asserted-Identity):

```
P-Asserted-Identity : “Carol” <sip:1002@company1.sipwise.com>
```

NOTE During the call forward type of event, once a call is established, a real/actual user identity has to be reflected in the 'P-Asserted-Identity' header of the SIP 200 OK message going to an initial caller.

Copyright © Sipwise GmbH - All rights reserved

NOTE In case of SIP legs changes (for example a call transfer), a distant identity will likely be changed. In this case INVITE with 'Replaces' parameter should be issued to the remote leg, in order to update current user identity.

For inbound calls to Sipwise C5, the user identity has to be provided in the incoming SIP INVITE request and then be reflected in the WebSocket message (so on the CSTA level) by a proper setting of the '<callingDevice>' tag value. In case of incoming call, such CSTA events will be of 'DeliveredEvent' type and then of 'EstablishedEvent' type (the last one, if a call got connected). The priority for a source of this value is as following:

1. the 'P-Asserted-Identity' header ;
2. the 'From' header (if PAI does not exist) ;

For outbound calls, if we have the 'P-Asserted-Identity' header in the 200 OK response coming from a remote side, it has to be reflected as the value of the '<answeredDevice>' tag in the 'EstablishedEvent' WebSocket message. If PAI is not present, then the 'To' header is a source of the value.

Third-party call control, scenarios

These operations are usually issued by an application (using CSTA services) and known as 3PCC operations. Let's have a look at an amount of examples.

Here is an example of the Make Call successful session, internal call:

This scenario illustrates a successful Make Call from device A to device B.

 Copyright © Sipwise GmbH - All rights reserved

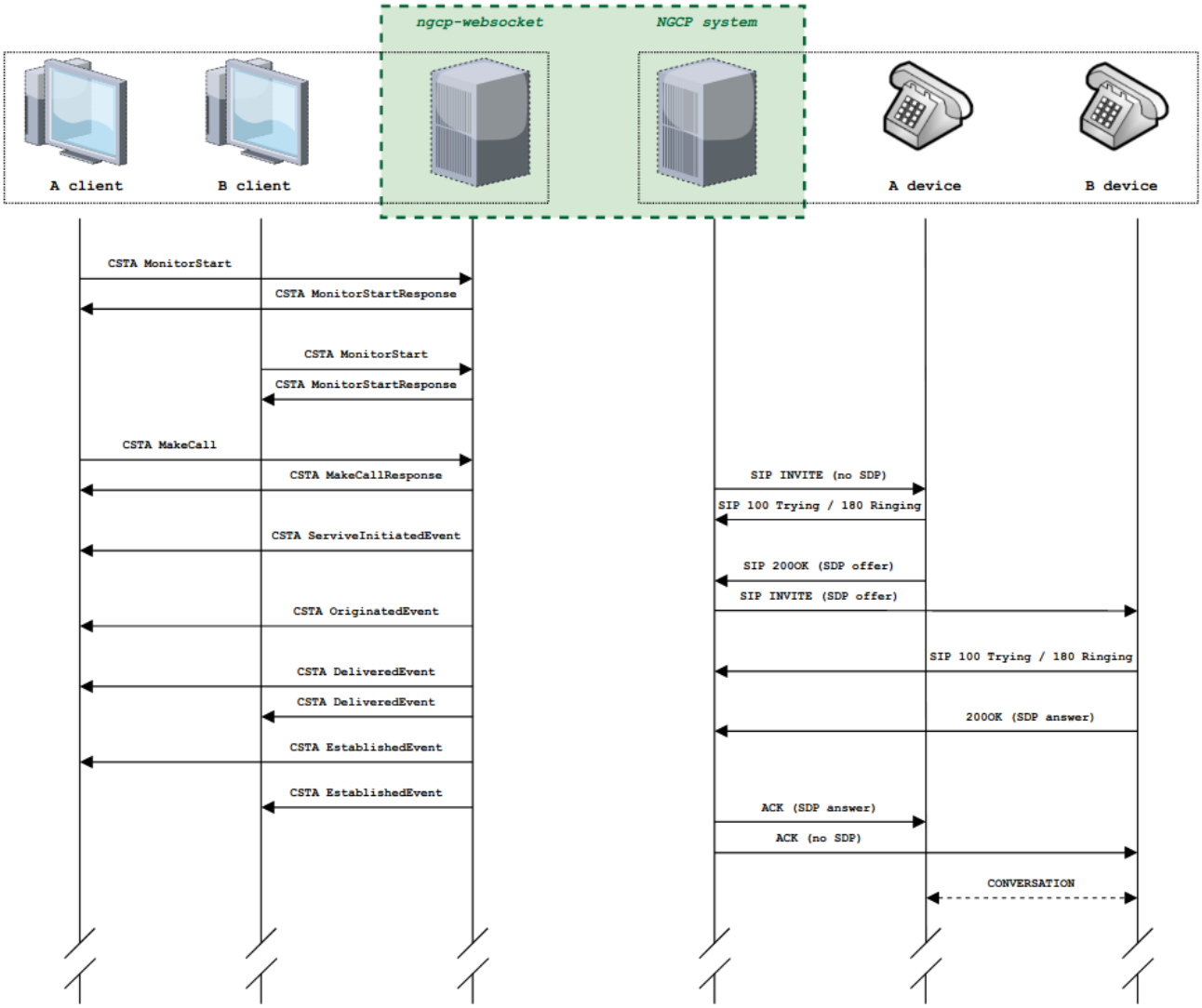


Figure 188. CSTA MakeCall messages exchange

In this scenario both devices are registered and ready to process calls. The client of A sends a request to start a SIP session with B. The client of B is notified about the incoming call. B picks up the call and media session is established between A and B. Both clients of A and B are notified about the delivered event.

The detailed content of CSTA messages is presented in CSTA Appendix MakeCall.

NOTE There is a possibility to choose a specific list of codecs for a session to be established, if the default list of codecs (configured in the config.yml *tpcc.codec_list*) doesn't fit requirements of the calling client. This preference is being transferred using '<mediaSessionInfo>' XML element.

An example of the Make Call failed session, internal call:

There could be different reasons, why the MakeCall scenario can fail, this may happen for e.g. in case, when a destination or source device is busy, or one of them doesn't respond, not found, etc.

In the scenario shown below, device B is busy and cannot receive the call at the time of attempt to

Copyright © Sipwise GmbH - All rights reserved

reach it.

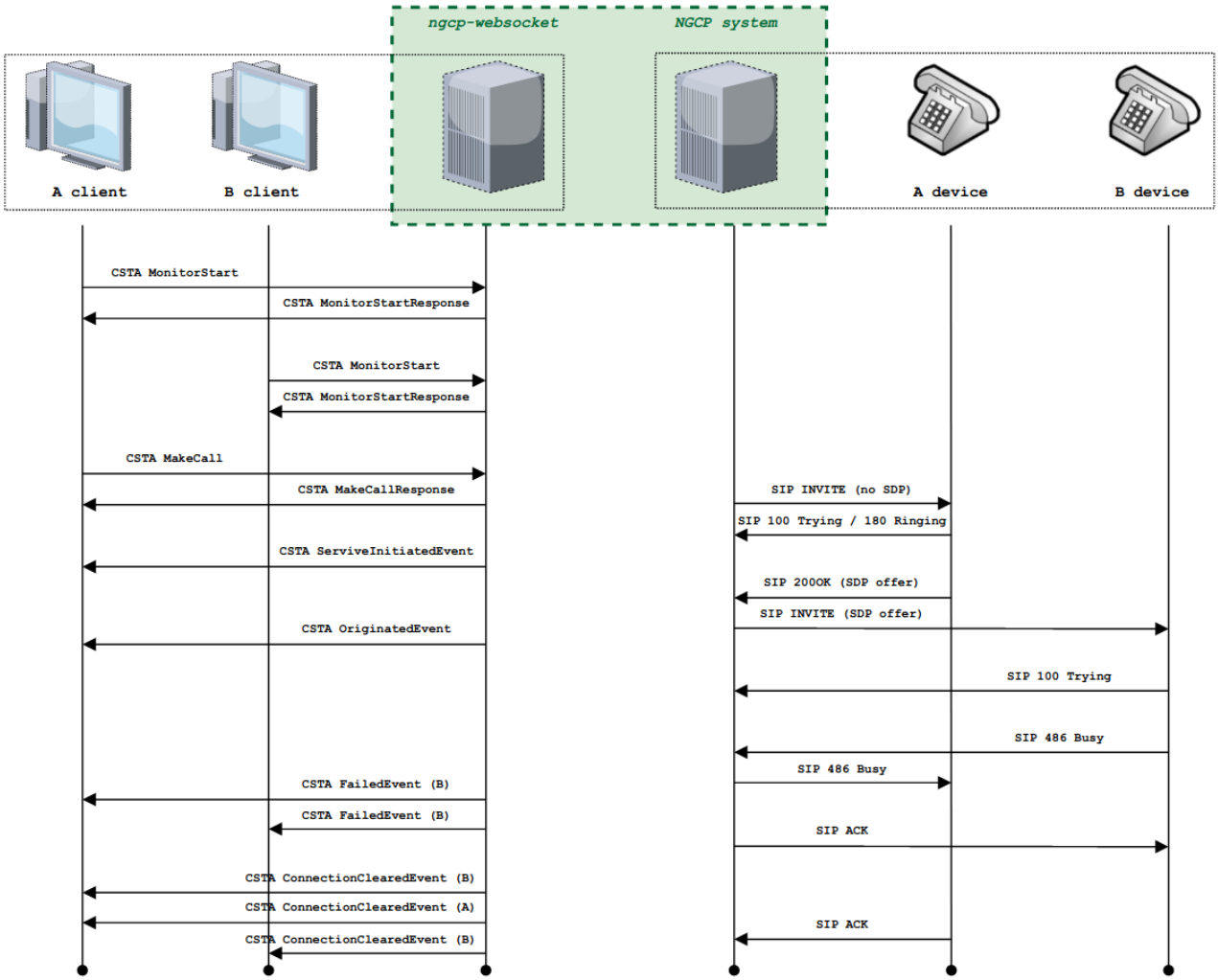


Figure 189. Failed MakeCall scenario (destination is busy)

The content of CSTA messages is presented in in CSTA Appendix MakeCall failed.

A treatment of other errors:

- if 'callingDevice' of MakeCall is unknown – the response to the MakeCall request will be a CSTAErrorCode with the 'invalidSourceDeviceID' error;
- if 'calledDirectoryNumber' of MakeCall is unknown – the response to the MakeCall request will be a CSTAErrorCode with the 'invalidDestinationDeviceID' error.

An example of the Make Call successful session, outbound scenario:

The following scenario illustrates the Make Call service request from behalf of device A to an external device which is outside of the served domain.

Since the destination is located outside this CSTA domain, it cannot be directly monitored via the CSTA API and therefore no events can be generated for the remote side. Since the remote device is reachable via a SIP peering, this SIP peering technically represents a SIP endpoint visible for the system, and hence, can be instead monitored by the CSTA, since this SIP Peering is directly handled by Sipwise C5. But this is not required.

Copyright © Sipwise GmbH - All rights reserved

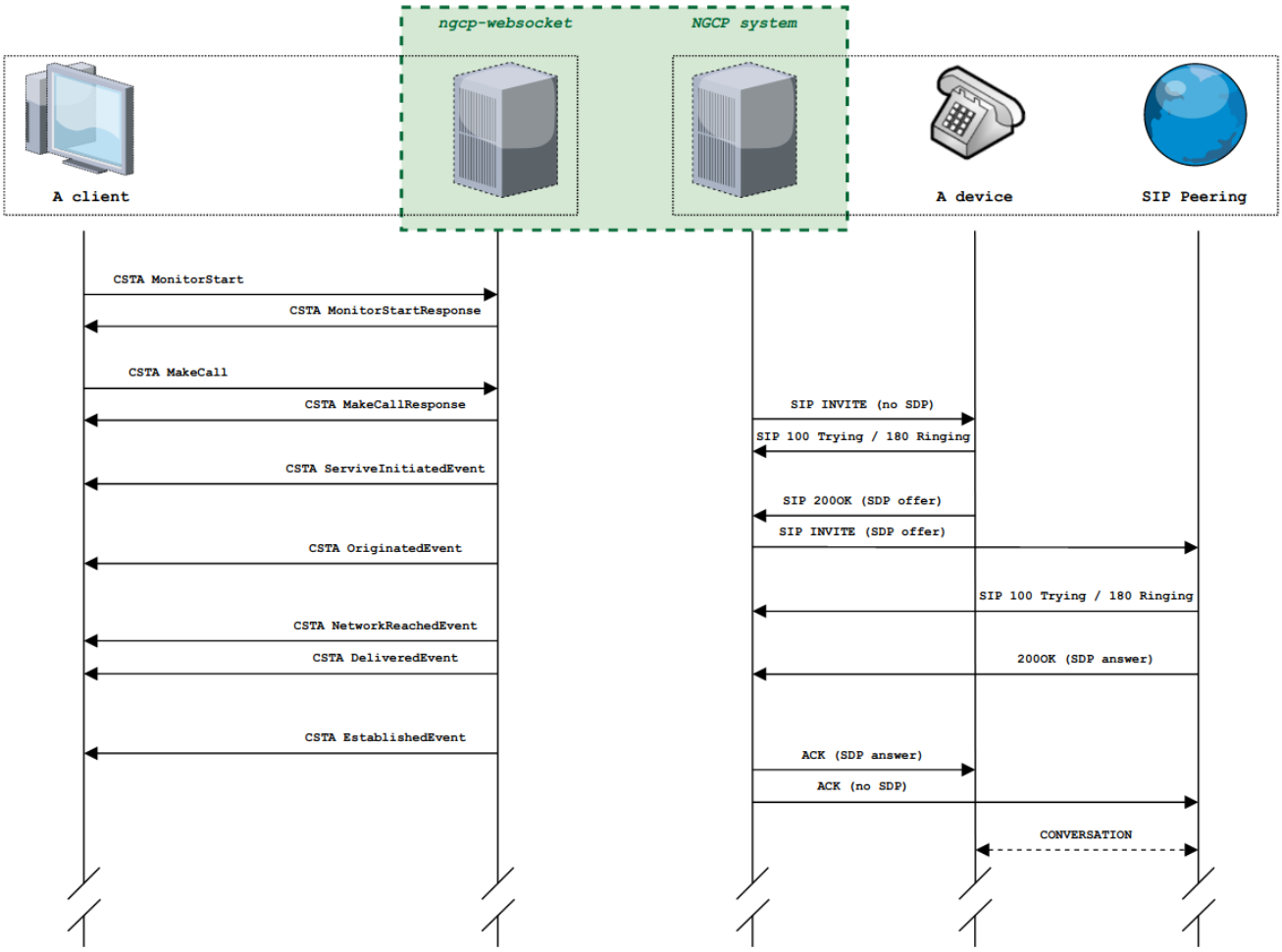


Figure 190. CSTA MakeCall with a SIP Peering, messages exchange

The content of CSTA messages is presented in in CSTA Appendix MakeCall outbound.

A treatment of other errors:

- if the 'callingDevice' of MakeCall is unknown – the response to the MakeCall is 'CSTAErrorCode' with the 'invalidSourceDeviceID' error;
- if the 'calledDirectoryNumber' of MakeCall is unknown – the response to the MakeCall is 'CSTAErrorCode' with the 'invalidDestinationDeviceID' error.

For outbound calls, if there is the 'P-Asserted-Identity' in the SIP 200OK coming from a distant participant, it has to be reflected in the 'answeredDevice' XML element of CSTA 'EstablishedEvent' event message.

An example of the Hold Call scenario:

This scenario illustrates a successful Hold Call request processing. A hold request is issued by a client and then a SIP re-INVITE transaction with the special content of SDP body will be sent to the remote side (sendonly / inactive / 0.0.0.0).

Copyright © Sipwise GmbH - All rights reserved

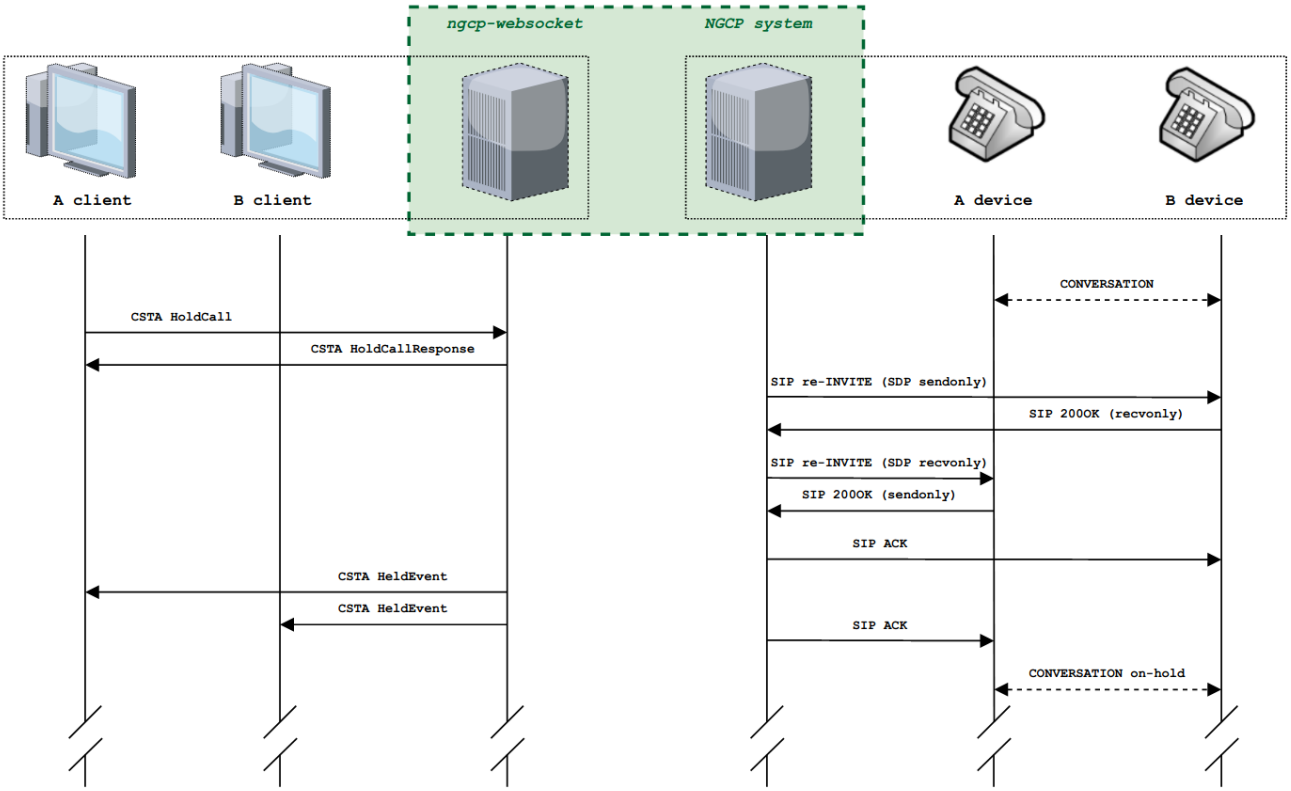


Figure 191. CSTA MakeCall with a SIP Peering, messages exchange

The content of CSTA messages is presented in in CSTA Appendix HoldCall.

A treatment of errors:

- if a connection in the 'callToBeHeld' of HoldCall is unknown – the response to the HoldCall request is the 'CSTAErrorCode' with the 'invalidConnectionID' error ;
- if a connection of HoldCall isn't in the 'connected' state – the response to the HoldCall request is the 'CSTAErrorCode' with the 'invalidConnectionState' error.

An example of the Retrieve call:

This scenario illustrates a successful Retrieve request processing. A prerequisite - SIP call must be on hold. After the retrieve operation, a conversation is to be continued.

 Copyright © Sipwise GmbH - All rights reserved

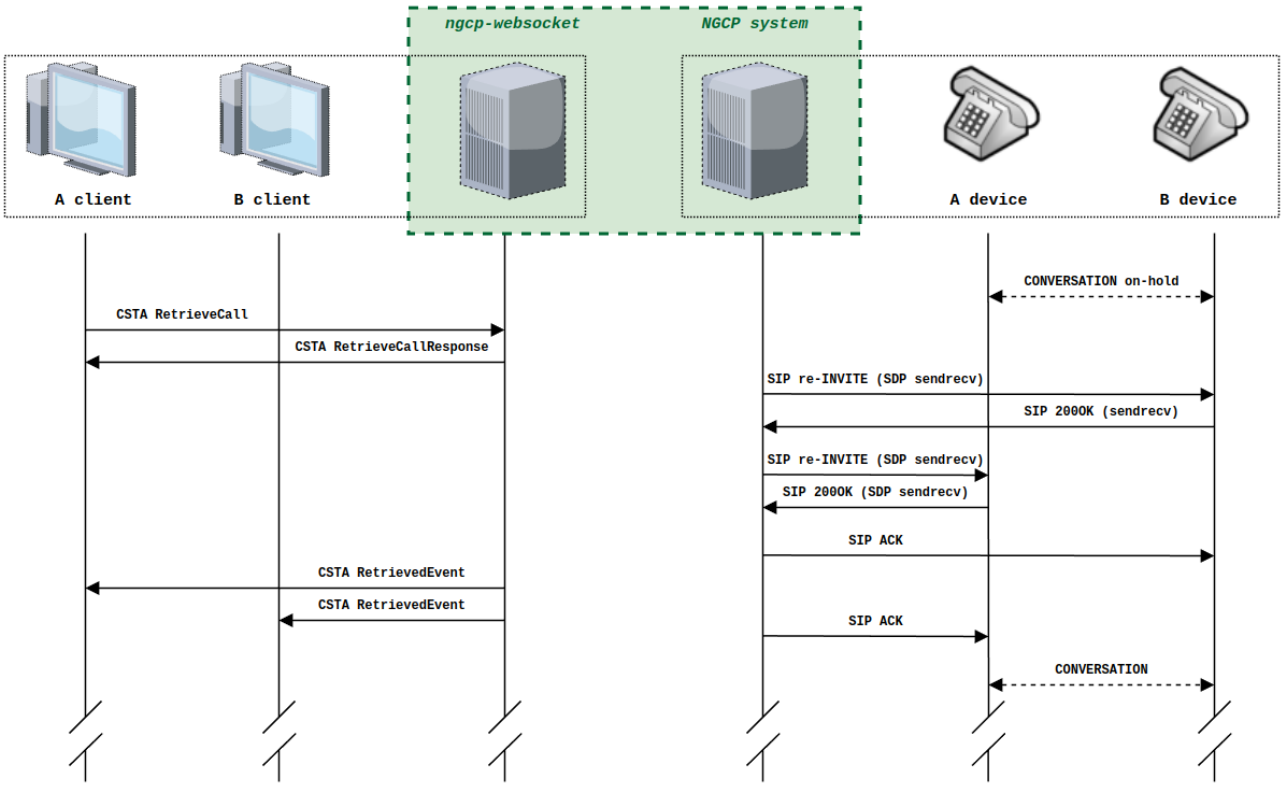


Figure 192. Successful RetrieveCall scenario using SIP re-INVITE

The content of CSTA messages is presented in in CSTA Appendix RetrieveCall.

A treatment of errors:

- if a connection in the 'heldCall' of the 'RetrieveCall' is unknown – a response to the 'RetrieveCall' is the 'CSTAErrorCode' with the 'invalidConnectionID' error ;
- if a connection in the 'heldCall' of the 'RetrieveCall' isn't in the 'held' state – a response to the 'RetrieveCall' is the 'CSTAErrorCode' with the 'invalidConnectionState' error.

An example of the Clear Connection:

This scenario illustrates how a normal call clearing (call termination) is being done. A session with a remaining device is cleared as well.

Copyright © Sipwise GmbH - All rights reserved

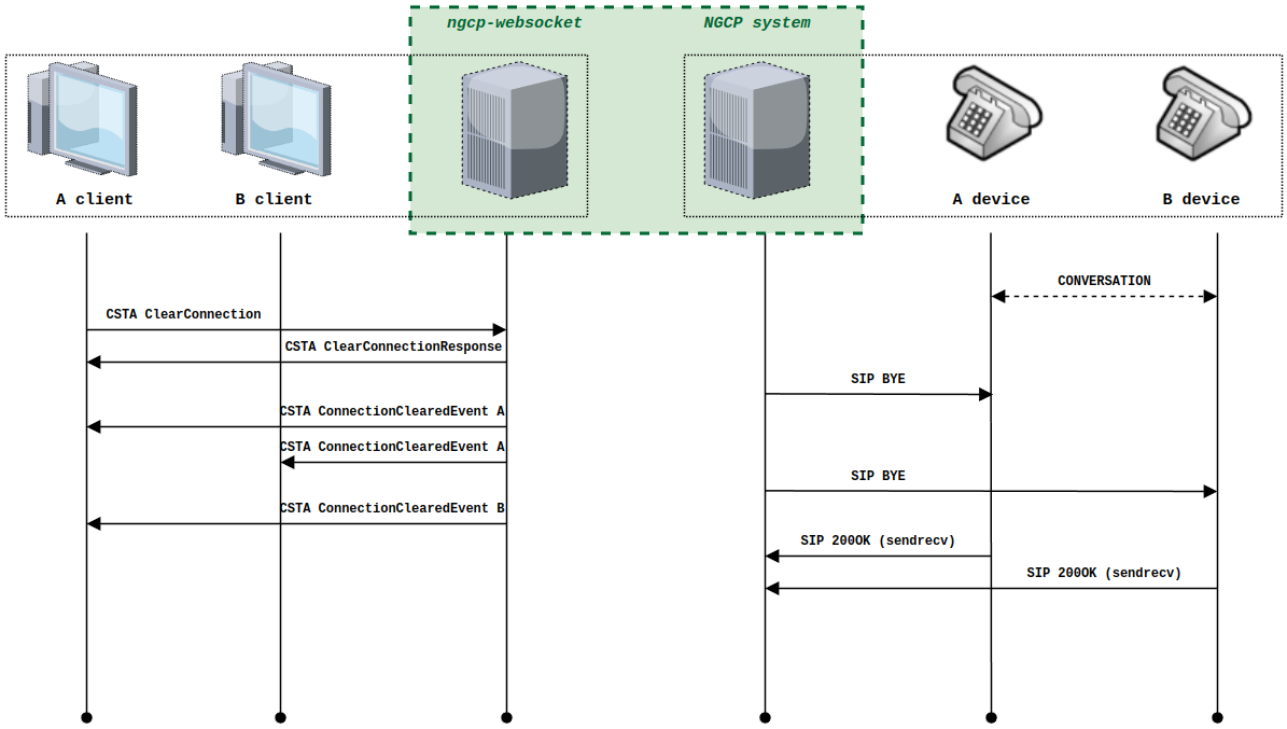


Figure 193. Successful ClearConnection scenario

The content of CSTA messages is presented in in CSTA Appendix ClearConnection.

A treatment of errors:

- if a connection in the 'connectionToBeCleared' of the 'ClearConnection' is unknown – a response to the 'ClearConnection' is the 'CSTAErrorCode' with the 'invalidConnectionID' error ;
- if a connection in the 'connectionToBeCleared' of the 'ClearConnection' is in the 'Null' state – a response to the 'ClearConnection' is the 'CSTAErrorCode with the 'invalidConnectionState' error.

Third-party call control, other scenarios not described in the document

- ClearCall
- DeflectCall
- ConsultationCall
- SingleStepTransferCall
- TransferCall

Third-party call control, retrieve information about device / call

SnapshotCall:

An application uses the SnapshotCall service to obtain information about the CSTA connections of the particular call and supplementary information, like a CallingDevice, CalledDevice, etc.

The ngcp-websocket will provide a positive response with the list of zero or more connections and information about each connection as well as supplementary information.

Copyright © Sipwise GmbH - All rights reserved

A list of common elements in the snapshotDeviceResponseInfo element:

- <*deviceOnCall*> – this mandatory element contains the endpoint device ID ;
- <*connectionIdentifier*> – this mandatory element provides the 'connectionID' of the connection. This is the 'connectionID' that is used in CSTA services that are applied to the connection. The <*connectionIdentifier*> is combined of the *called* and the *deviceID*. The *deviceID '<privateNumber'>'* attribute may be set to true in case of privacy (CLIR/COLR) ;
- <*localCallState*> – this mandatory element specifies a *'compoundCallState'*, which consists of one or more CSTA connection states. The first connection state in the list is the 'local' connection state of the connection being reported. Other connection states that reflect other connections in the same call (rest of devices) may also be provided, if known to the UAC ;
- <*calling*>/<*called*> – a calling called device ;
- <*privateData*> – an additional information about the call state: recorded connections, display names ;

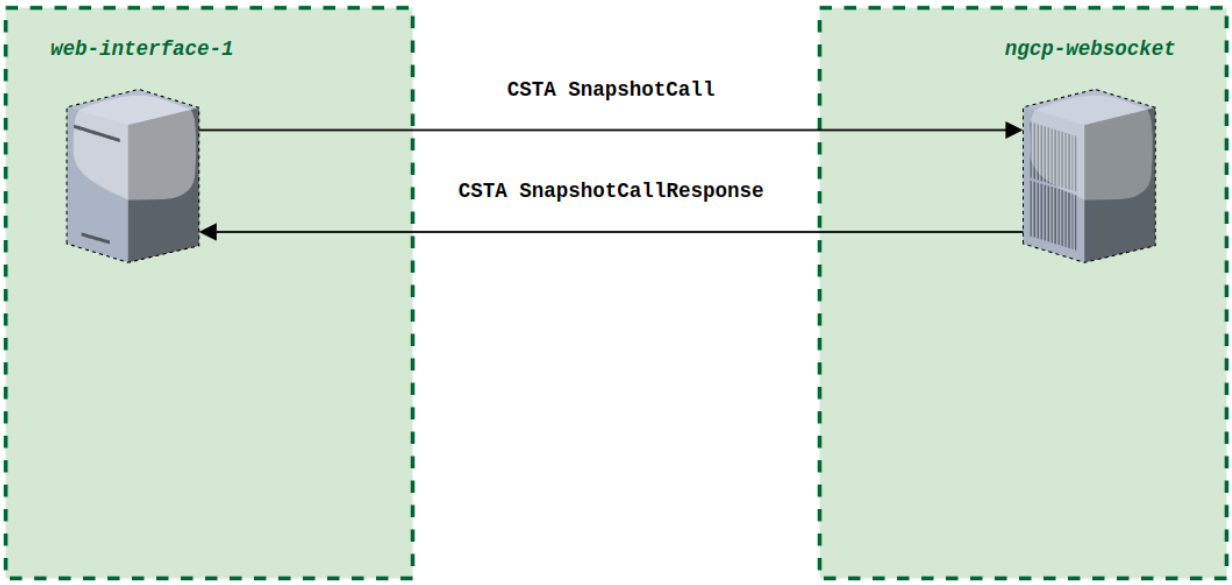


Figure 194. Snapshot Call scenario

If a SIP call doesn't exist, a response with the 'CSTAErrorCode' is returned to the 'SnapshotCall' with the 'invalidCallID' status.

SnapshotDevice:

An application uses the SnapshotCall service to obtain information about the CSTA connections of the particular device.

The ngcp-websocket will provide a positive response with a connection and information it.

- <*deviceOnCall*> – this mandatory element contains the endpoint device ID ;
- <*connectionIdentifier*> – this mandatory element provides the 'connectionID' of the connection. This is the 'connectionID' that is used in CSTA services that are applied to the connection. The <*connectionIdentifier*> is combined of the *called* and the *deviceID*. The *deviceID '<privateNumber'>'* attribute may be set to true in case of privacy (CLIR/COLR) ;

Copyright © Sipwise GmbH - All rights reserved

- <*localCallState*> – this mandatory element specifies a *'compoundCallState'*, which consists of one or more CSTA connection states. The first connection state in the list is the 'local' connection state of the connection being reported. Other connection states that reflect other connections in the same call (rest of devices) may also be provided, if known to the UAC ;
- <*calling*>/<*called*> – a calling called device ;
- <*privateData*> – an additional information about the call state: recorded connections, display names ;

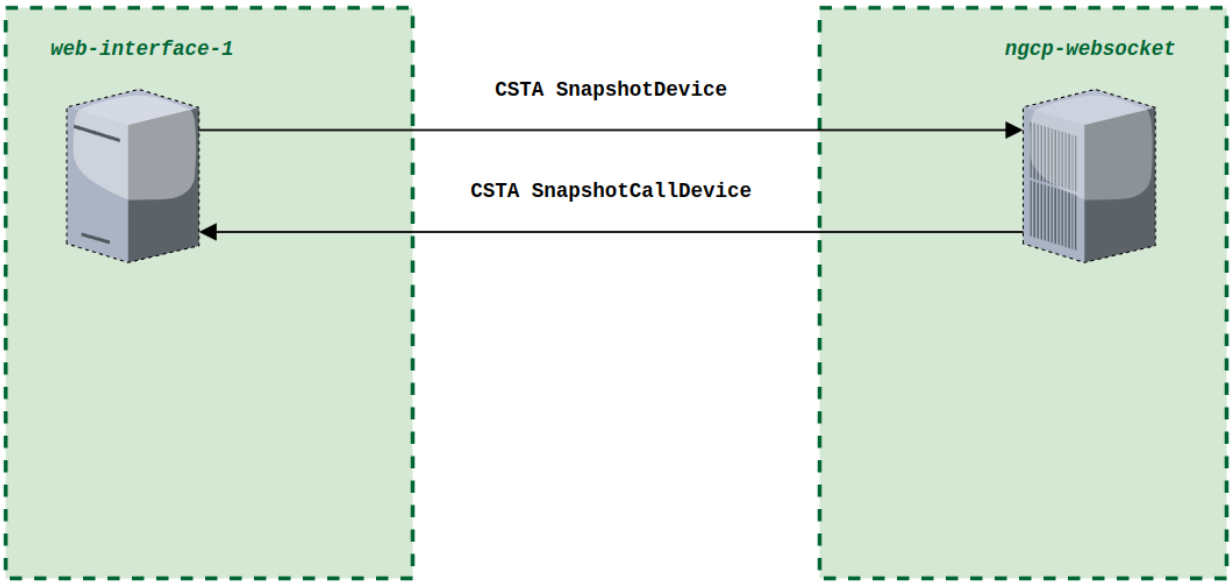


Figure 195. Snapshot Call scenario for a device

If a requested device is for any reason out-of-service, a response with the 'CSTAErrorCode' is returned to the 'SnapshotDevice' with the 'deviceOutOfService' status.

Third-party call control, device got into/out of service

BackInService:

The 'BackInService' event indicates that a device has been returned to service and is available. Back into service means one of the following: SIP device register / configuration object enabled / out-of-service check with success result.

NOTE 'BackInService' doesn't need to be sent upon SIP REGISTER refresh, only upon initial REGISTER.

OufOfService:

The 'OutOfService' event indicates that a device has entered a maintenance or a disabled state (for e.g. has been de-registered) and can no longer accept calls and some other categories of CSTA service requests (for e.g. Call Control services).

Out of service means one of the following: SIP device de-register / configuration object disabled / out-of-service check with failed result.

Copyright © Sipwise GmbH - All rights reserved

8.5.5. 3PCC (TPCC) module of sems-b2b component

This is a module, which allows to establish different call events, such as usual audio session between two points, based on the websocket/CSTA generated commands.

The main goal of the 3PCC (TPCC) SEMS module is to have an interconnection between the internal ngcp-websocket component and the SEMS-b2b in terms of translating actions to real call sessions (create call, hang-up, put on hold, transfer call etc.)

Two abstract (logical) endpoints that we have in such architecture:

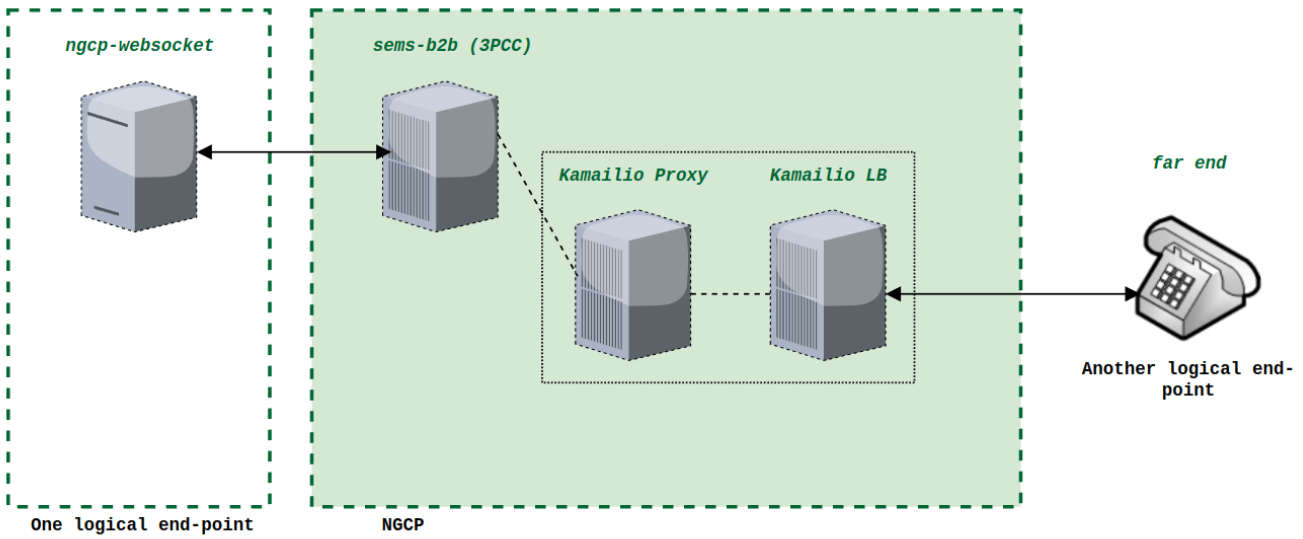


Figure 196. Two logical endpoints in the architecture

Interaction with the ngcp-websocket

The ngcp-websocket communicates with the CSTA environment (CSTA events) and translates this logic into out internal communication environment:

Copyright © Sipwise GmbH - All rights reserved

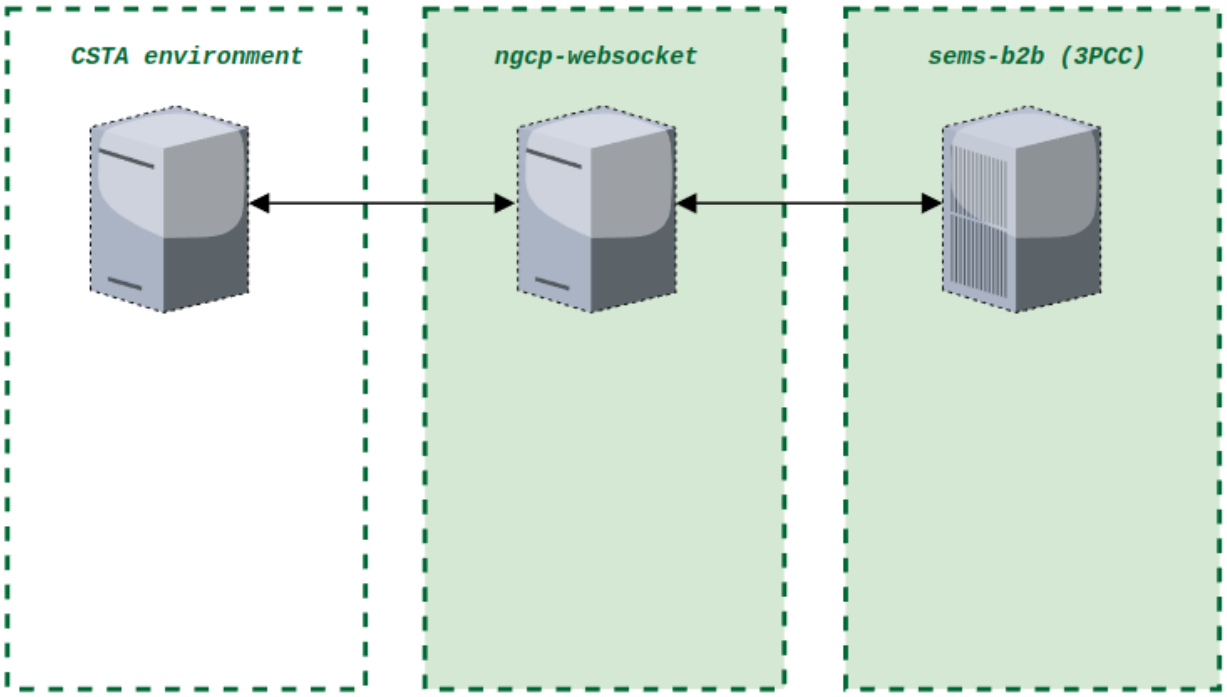


Figure 197. CSTA environment with SEMS

The ngcp-websocket communicates with the SEMS using specified (our internal) protocol (sending/receiving messages to/from SEMS). The communication between the web-socket and the SEMS is implemented using Redis channels (Redis queues).

NOTE The idea is that we can communicate via these channels also with other components, such as Kamailio Proxy.

This approach currently supports multi-nodes distributed environment and depending on a subscription, some particular component will receive, what it is supposed to receive. In scope of this chapter our interest is SEMS-b2b component.

Redis-channels (KeyDB)

The SEMS-b2b component is currently using at least three Redis channels:

- Receiving Data (actions): 1) Subscription Channel (as a command interface), 2) Events Channel (for data)
- Sending Data (updates / brand new SIP events): 1) Publish Channel (responses to the channel, or new requests to the channel)

NOTE Sometimes Redis channels can be also called as Redis queues.

By default it's been set to:

- receiving commands via the 'csta.websocket' channel (separate thread in SEMS)

Copyright © Sipwise GmbH - All rights reserved

- receiving any data additional via the 'sp.sems-b2b' channel (parent thread)
- sending data to the channel 'csta.sems'

IMPORTANT the names of the channels can be controlled accordingly in the tpcc.conf of the sems-b2b component. Parameters: 'subscribe_channel', 'publish_channel' and 'output_dst'. If for any reason there is a rational need to change the names of these channels, please contact the Sipwise Operations team to get a consultation.

It's worth to mention, that in the multi-active setup with multiple SEMS-b2b components working simultaneously the 'subscribe_channel' and the 'output_dst' values will be set based on the hostnames. This allows to differentiate messages being sent from/to Redis channels between different SEMS-b2b instances.

SEMS component of Sipwise C5

The SEMS is working asynchronously (multi-threaded program). Inside the SEMS we have two main queues:

- coming comands > FIFO input > 3PCC (TPCC) Handler (multi-threaded)
- responses / outgoing requests > FIFO output > 3PCC (TPCC) Handler (multi-threaded)

The SEMS-b2b constantly sends keepalive messages (just dummy 'ping') to remote Redis (KeyDB) to keep connections with the Redis (KeyDB) alive.

It also has a multi-threaded processing of incoming/outgoing commands (FIFO). It picks an incoming message (command/metadata) from the input queue and passes it to its events processor. It also picks any pending messages in the output queue, serializes it into the text (JSON) representation and publishes it to an according Redis channel.

The events processor of SEMS will try to undertake something upon receiving of the command. Logically we can divide it into two categories of actions:

- make a call between two participants. That means it calls an initiator (we can call it leg A) and then it calls the recipient of the call (we can call it leg B), and it calls it from behalf of the A side.
- perform an action on an existing session (as a simple example - we can hang-up it)

There will be no digging into code/technical details of the implementation of it, because that would be rather superfluous information in a scope of this chapter.

It's also worth to mention another aspect of 3PCC handling - events happening on the SIP side of 3PCC session.

All things happening within the SIP session (not triggered by the 3PCC handling), will be reflected accordingly on the 3PCC/CSTA. This means, if we receive some SIP message during an existing SIP session (re-INVITE for e.g.), which has been previously initiated by the 3PCC, SEMS-b2b will do all the required work to notify the ngcp-websocket component and will additionally run all own work in regards of 3PCC (on its side of 3PCC session processing).

Here is a usueful sequence description of the 3PCC dialog initiation:

Copyright © Sipwise GmbH - All rights reserved

- the CSTA environment sends some command in the XML to the ngcp-websocket component ;
- the ngcp-websocket translates it to the internal protocol and puts the message in the JSON format into the dedicated Redis channel (Subscription Events / Events Redis) ;
- the SEMS-b2b component receives this JSON message from Redis (KeyDB), parses it and translates it into the internal SEMS-b2b program usage (in other words make it convertible into c++ objects, hence machine code)
- the SEMS-b2b starts processing the command using its events processor mechanism and will undertake all required actions, if they are applicable ;

As a simple example of such event processing is a call A-to-B. In such call scenario SEMS-b2b invites A without an SDP body present (no offer), waits until A responds with 200OK containing an SDP offer with certain media capabilities, and then will use these capabilities to build up an SDP offer to be sent to the B subscriber.

B gets invited into the SIP call, as soon as the call is accepted by B, the SDP answer contained by the 200OK response will be used to build up an SDP answer for the A as well. Eventually this session gets acknowledged by SEMS for B (leg with B), and by A for SEMS (leg with A). A media session is established then through the SEMS-b2b and if needed, transcoding is to be applied.

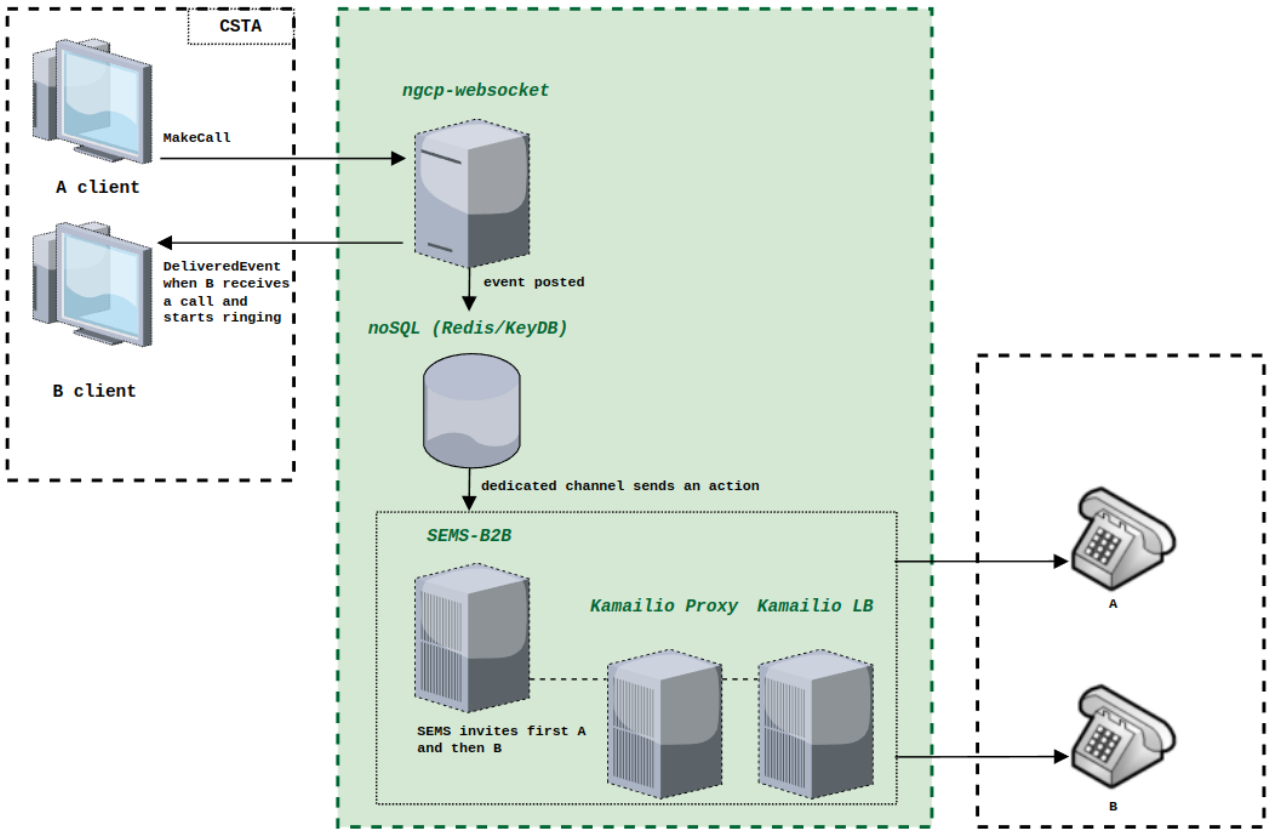


Figure 198. 3PCC dialog initiation

Copyright © Sipwise GmbH - All rights reserved

NOTE

In case there is more than one active SEMS component working in the cluster, for e.g. Carrier grade setup, then the first SEMS who received and started processing this JSON message (new event), is responsible for this particular TPCC session. A very first SEMS-b2b, which has received this message, will RPOP this event from the end of the dedicated Redis list (channel) and will start processing it. Hence other active SEMS-b2b instances in the cluster, will not undertake anything.

8.5.6. Configuration of 3PCC (TPCC)

To start using 3PCC/CSTA functionality of Sipwise C5 the following list of config.yml parameters must be activated:

- *tpcc.enable yes*
- *websocket.enable yes*

Detailed example of 3PCC (TPCC) config.yml block:

```
tpcc:
  add_header_to_detect_device: ''
  codec_list: PCMA,PCMU,g729,opus,g722
  enable: no
  instance_id: '*'
  publish_channel: csta.sems
  use_broadsoft_hold: no
```

Where:

- *add_header_to_detect_device:* - this option defines, which header to add into the 3PCC originated calls in order to notify the caller/callee subscriber, who is the real device involved in the call
- *codec_list:* - a list of preferred audio codecs
- *enable:* - enabling of the 3PCC (TPCC) module
- *instance_id:* - identification of the sems instance, only for the multi-active sems setup
- *publish_channel:* - a channel to be used for publishing 3PCC events
- *use_broadsoft_hold:* - use it to switch between initial and broadsoft-specific hold handling

Copyright © Sipwise GmbH - All rights reserved

```
websocket:
  allow_foreign_monitor_stop: yes
  enable: no
  inactivity_timeout: 0
  loglevel: info
  max_clients: 1000
  max_requests: 100
  mode: development
  port: 3443
  redis:
    csta_monitor_db: 31
    csta_publish_channel: csta.websocket
    csta_subscribe_channels:
    - csta.sems
    - csta.proxy
    csta_subscription_db: 32
  ssl:
    cert: /etc/ngcp-config/shared-files/ssl/myserver.crt
    enable: no
    key: /etc/ngcp-config/shared-files/ssl/myserver.key
```

Where:

- *allow_foreign_monitor_stop:* - enable/disable MonitorStop with the 'monitor_id' belonging to another connection
- *enable:* - an enabling of the websocket component
- *inactivity_timeout:* - a timeout for the remote side in case of actions absence
- *loglevel:* - the log level for the component
- *max_clients:* - a maximum amount of clients can be processed simultaneously
- *max_requests:* - a maximum amount of requests can be processed simultaneously
- *mode:* - possible values 'full', 'development' and 'brief'
- *port:* - the port to listen on
- *redis:* - a block of options related to Redis (KeyDB)
- *redis.csta_monitor_db:* - select needed db for the monitor
- *redis.csta_publish_channel:* - a Redis channel for publishing events
- *redis.csta_subscribe_channels:* - Redis channels to which sems-b2b and kamailio-proxy will subscribe
- *redis.csta_subscription_db:* - select needed db for publishing events
- *ssl:* - a block of options related to the SSL encryption
- *ssl.enable:* - enable SSL encryption
- *ssl.cert:* - a path to the certificate
- *ssl.key:* - a path to the key

After both *'tpcc.enable'* and *'websocket.enable'* config.yml options are set to *yes*, changes must be

 Copyright © Sipwise GmbH - All rights reserved

applied with the:

```
bash$ sudo ngcpcfg apply 'enable tpcc and websocket'
```

IMPORTANT by enabling these options a restart of the sems-b2b and kamailio-proxy services will be triggered, so it's better to plan this maintenance out of business hours. If there are any doubts regarding how to properly enable CSTA/3PCC support in the system, please contact the Operations team of Sipwise.

After both the 3PCC (TPCC) module and the ngcp-websocket component are enabled you can start using API to issue new 3PCC-controlled SIP calls.

8.6. SMS (Short Message Service) on Sipwise C5

Sipwise C5 offers *short messaging service* to its local subscribers. The implementation is based on a widely used software module: *Kannel*, and it needs to interact with a mobile operator's SMSC in order to send and receive SMs for the local subscribers. The data exchange with SMSC uses *SMPP* (Short Message Peer-to-Peer) protocol.

SMS directions:

- incoming / received: the destination of the SM is a local subscriber on the NGCP
- outgoing / sent: the SM is submitted by a local subscriber

NOTE The Sipwise C5 behaves as a short message client towards the SMSC of a mobile operator. This means every outgoing SM will be forwarded to the SMSC, and every incoming SM will reach Sipwise C5 through an SMSC.

The architecture of the SMS components of Sipwise C5 and their interaction with other elements is depicted below:

Copyright © Sipwise GmbH - All rights reserved

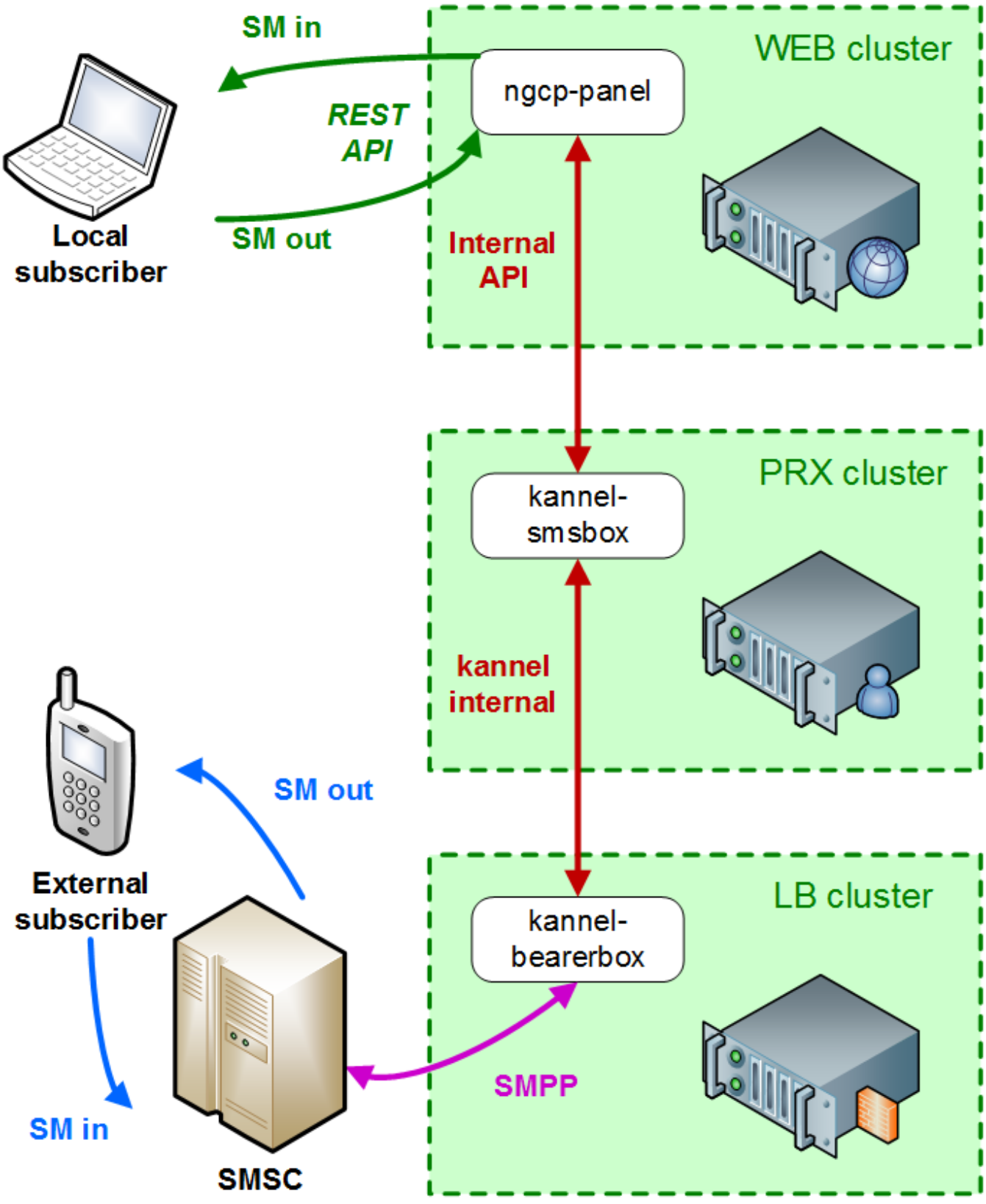


Figure 199. SMS Interaction

NOTE For the *Sipwise C5 CE and PRO* installations: the *Kannel* components and the *ngcp-panel* all run on the same single node. The description of SMS module will continue referring to a *Sipwise C5 CARRIER* installation in the handbook.

Copyright © Sipwise GmbH - All rights reserved

There are 2 components of the SMS module:

- **SMS Box**: this component takes care of handling the messages locally, that means:

  delivering them to subscribers (writing into database for later retrieval)

  picking up the submitted SMs from the database and forwarding them to the *Bearer Box* component
- **Bearer Box**: this component manages the transmission of SMs between Sipwise C5 and the mobile operator's SMSC

8.6.1. Configuration

Main Parameters

The SMS functionality of Sipwise C5 is disabled by default. In order to **enable SMS**, change the value of configuration parameter sms.enable to yes in the main configuration file (/etc/ngcp-config/config.yml).

The second step of configuration is related to the **SMSC** where Sipwise C5 will connect to. Set the following parameters:

- sms.smsc.host: IP address of the SMSC
- sms.smsc.port: Port number of the SMSC
- sms.smsc.username: Username for authentication on the SMSC
- sms.smsc.password: Password for authentication on the SMSC

Other parameters of the SMSC connection may also need to be changed from the default values, but this is specific to each deployment.

Then, as usual, you have to make the new configuration active:

```
$ ngcpcfg apply 'Enabled SMS'
$ ngcpcfg push all
```

Configuration Files of Kannel

There are a few configuration files for the *Kannel* module, namely:

- /etc/default/ngcp-kannel: determines which components of *Kannel* will be started. This is auto-generated from /etc/ngcp-config/templates/etc/default/ngcp-kannel.tt2 file when SMS is enabled.
- /etc/kannel/kannel.conf: contains detailed configuration of *Kannel* components. This is auto-generated from /etc/ngcp-config/templates/etc/kannel/kannel.conf.tt2 file when SMS is enabled.
- /etc/logrotate.d/ngcp-kannel.conf: configuration of *logrotate* for *Kannel* log files. This is auto-generated from /etc/ngcp-config/templates/etc/logrotate.d/ngcp-kannel.conf.tt2 file when SMS is enabled.

Copyright © Sipwise GmbH - All rights reserved

CAUTION Please do not change settings in the above mentioned template files, unless you have to tailor *Kannel* settings to your specific needs!

Finally: see the description of each configuration parameter in the appendix.

Call Forwarding for SMS (CFS)

Any subscriber registered on Sipwise C5 can apply a call forwarding setting for short messages, referred to as "CFS" (Call Forward - SMS). If the CFS feature is enabled, he can receive the SMs on his mobile phone, for example, instead of retrieving the SMs through the REST API. This is much more convenient for users if they do not have an application on their smartphone or computer that could manage the SMs through the REST API.

In order to enable CFS you have to set the forwarding as usual on the admin web interface, or through the REST API. Navigate to *Subscribers* *select one* *Details* *Preferences* *Call Forwards* and press the *Edit* button.

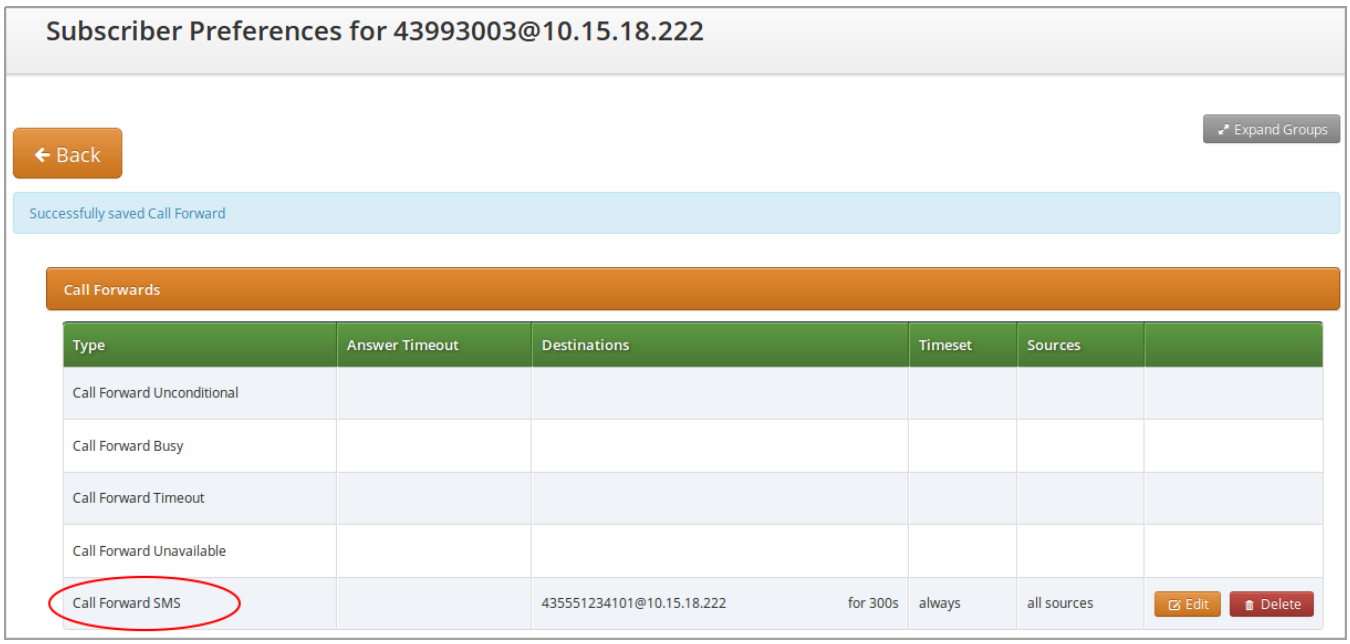


Figure 200. Call Forward for SMS

8.6.2. Monitoring, troubleshooting

Bearer Box (LB node of NGCP)

On the LB node you can see a **process** named **"bearerbox"**. This process has 2 **listening ports** assigned to it:

- 13000: this is the generic *Kannel* administration port, that belongs to the "core" component of Kannel.
- 13001: this is the communication port towards the *SMS Box* component running on PRX nodes of NGCP.

The *ngcp-service* tool also shows the *bearerbox* process in its summary information:

Copyright © Sipwise GmbH - All rights reserved

```
$ ngcp-service summary
Ok Service                         Managed    Started   Status
-- ------------------------------- ---------- --------- ---------
…
   kannel-bearerbox                managed    by-ha     active
…
```

The following log files can provide information about the operation of *Bearer Box*:

- status messages and high level, short entries about sent and received messages: /var/log/ngcp/kannel/kannel.log

```
...
2017-09-26 08:57:32 [15922] [10] DEBUG: boxc_receiver: heartbeat with
load value 0 received
...
2017-09-26 11:12:06 [15922] [10] DEBUG: boxc_receiver: sms received
2017-09-26 11:12:06 [15922] [10] DEBUG: send_msg: sending msg to box:
<192.168.1.4>
2017-09-26 11:12:06 [15922] [11] DEBUG: send_msg: sending msg to box:
<192.168.1.4>
2017-09-26 11:12:06 [15922] [11] DEBUG: boxc_sender: sent message to
<192.168.1.4>
2017-09-26 11:12:06 [15922] [10] DEBUG: boxc_receiver: got ack
...
```

- detailed information and message content of sent and received messages, link enquiries: /var/log/kannel/smsc.log

NOTE Sent and received message examples shown here do not contain the full phone number and content for confidentiality reason.

Example received message:

```
...
2017-09-26 12:09:36 [15922] [6] DEBUG: SMPP[default_smsc]: Got PDU:
2017-09-26 12:09:36 [15922] [6] DEBUG: SMPP PDU 0x7f2274025070
dump:
2017-09-26 12:09:36 [15922] [6] DEBUG:   type_name: deliver_sm
2017-09-26 12:09:36 [15922] [6] DEBUG:   command_id: 5 = 0x00000005
2017-09-26 12:09:36 [15922] [6] DEBUG:   command_status: 0 =
0x00000000
2017-09-26 12:09:36 [15922] [6] DEBUG:   sequence_number: 11867393
= 0x00b51501
2017-09-26 12:09:36 [15922] [6] DEBUG:   service_type: NULL
2017-09-26 12:09:36 [15922] [6] DEBUG:   source_addr_ton: 2 =
0x00000002
2017-09-26 12:09:36 [15922] [6] DEBUG:   source_addr_npi: 1 =
```

Copyright © Sipwise GmbH - All rights reserved

```
0x00000001
2017-09-26 12:09:36 [15922] [6] DEBUG:   source_addr: "0660......."
2017-09-26 12:09:36 [15922] [6] DEBUG:   dest_addr_ton: 1 =
0x00000001
2017-09-26 12:09:36 [15922] [6] DEBUG:   dest_addr_npi: 1 =
0x00000001
2017-09-26 12:09:36 [15922] [6] DEBUG:   destination_addr:
"43668......."
2017-09-26 12:09:36 [15922] [6] DEBUG:   esm_class: 0 = 0x00000000
2017-09-26 12:09:36 [15922] [6] DEBUG:   protocol_id: 0 =
0x00000000
2017-09-26 12:09:36 [15922] [6] DEBUG:   priority_flag: 0 =
0x00000000
2017-09-26 12:09:36 [15922] [6] DEBUG:   schedule_delivery_time:
NULL
2017-09-26 12:09:36 [15922] [6] DEBUG:   validity_period: NULL
2017-09-26 12:09:36 [15922] [6] DEBUG:   registered_delivery: 0 =
0x00000000
2017-09-26 12:09:36 [15922] [6] DEBUG:   replace_if_present_flag: 0
= 0x00000000
2017-09-26 12:09:36 [15922] [6] DEBUG:   data_coding: 3 =
0x00000003
2017-09-26 12:09:36 [15922] [6] DEBUG:   sm_default_msg_id: 0 =
0x00000000
2017-09-26 12:09:36 [15922] [6] DEBUG:   sm_length: 158 =
0x0000009e
2017-09-26 12:09:36 [15922] [6] DEBUG:   short_message:
2017-09-26 12:09:36 [15922] [6] DEBUG:    Octet string at
0x7f2274000f80:
2017-09-26 12:09:36 [15922] [6] DEBUG:      len:  158
2017-09-26 12:09:36 [15922] [6] DEBUG:      size: 159
2017-09-26 12:09:36 [15922] [6] DEBUG:      immutable: 0
2017-09-26 12:09:36 [15922] [6] DEBUG:      data: 5a <14 bytes> 46
2017-09-26 12:09:36 [15922] [6] DEBUG:      data: 72 <14 bytes> 68
2017-09-26 12:09:36 [15922] [6] DEBUG:      data: 61 <14 bytes> 67
2017-09-26 12:09:36 [15922] [6] DEBUG:      data: 20 <14 bytes> 57
2017-09-26 12:09:36 [15922] [6] DEBUG:      data: 65 <14 bytes> 63
2017-09-26 12:09:36 [15922] [6] DEBUG:      data: 68 <14 bytes> 73
2017-09-26 12:09:36 [15922] [6] DEBUG:      data: 2e <14 bytes> 61
2017-09-26 12:09:36 [15922] [6] DEBUG:      data: 6c <14 bytes> 73
2017-09-26 12:09:36 [15922] [6] DEBUG:      data: 3a <14 bytes> 73
2017-09-26 12:09:36 [15922] [6] DEBUG:      data: 4d <14 bytes> 6e
2017-09-26 12:09:36 [15922] [6] DEBUG:    Octet string dump ends.
2017-09-26 12:09:36 [15922] [6] DEBUG: SMPP PDU dump ends.
2017-09-26 12:09:36 [15922] [6] DEBUG: SMPP[default_smsc]: Sending
PDU:
2017-09-26 12:09:36 [15922] [6] DEBUG: SMPP PDU 0x7f2274020790
dump:
2017-09-26 12:09:36 [15922] [6] DEBUG:   type_name: deliver_sm_resp
2017-09-26 12:09:36 [15922] [6] DEBUG:   command_id: 2147483653 =
0x80000005
2017-09-26 12:09:36 [15922] [6] DEBUG:   command_status: 0 =
```


Copyright © Sipwise GmbH - All rights reserved

```
0x00000000
2017-09-26 12:09:36 [15922] [6] DEBUG:   sequence_number: 11867393
= 0x00b51501
2017-09-26 12:09:36 [15922] [6] DEBUG:   message_id: NULL
2017-09-26 12:09:36 [15922] [6] DEBUG: SMPP PDU dump ends.
2017-09-26 12:09:36 [15922] [6] DEBUG: SMPP[default_smsc]:
throughput (0.00,5.00)
...
```

Example sent message:

```
...
2017-09-26 12:04:08 [15922] [6] DEBUG: SMPP[default_smsc]:
throughput (0.00,5.00)
2017-09-26 12:04:08 [15922] [6] DEBUG: SMPP[default_smsc]: Manually
forced source addr ton = 1, source add npi = 1
2017-09-26 12:04:08 [15922] [6] DEBUG: SMPP[default_smsc]: Manually
forced dest addr ton = 1, dest add npi = 1
2017-09-26 12:04:08 [15922] [6] DEBUG: SMPP[default_smsc]: Sending
PDU:
2017-09-26 12:04:08 [15922] [6] DEBUG: SMPP PDU 0x7f2274025070
dump:
2017-09-26 12:04:08 [15922] [6] DEBUG:   type_name: submit_sm
2017-09-26 12:04:08 [15922] [6] DEBUG:   command_id: 4 = 0x00000004
2017-09-26 12:04:08 [15922] [6] DEBUG:   command_status: 0 =
0x00000000
2017-09-26 12:04:08 [15922] [6] DEBUG:   sequence_number: 98163 =
0x00017f73
2017-09-26 12:04:08 [15922] [6] DEBUG:   service_type: NULL
2017-09-26 12:04:08 [15922] [6] DEBUG:   source_addr_ton: 5 =
0x00000005
2017-09-26 12:04:08 [15922] [6] DEBUG:   source_addr_npi: 0 =
0x00000000
2017-09-26 12:04:08 [15922] [6] DEBUG:   source_addr: "any"
2017-09-26 12:04:08 [15922] [6] DEBUG:   dest_addr_ton: 1 =
0x00000001
2017-09-26 12:04:08 [15922] [6] DEBUG:   dest_addr_npi: 1 =
0x00000001
2017-09-26 12:04:08 [15922] [6] DEBUG:   destination_addr:
"43676......."
2017-09-26 12:04:08 [15922] [6] DEBUG:   esm_class: 3 = 0x00000003
2017-09-26 12:04:08 [15922] [6] DEBUG:   protocol_id: 0 =
0x00000000
2017-09-26 12:04:08 [15922] [6] DEBUG:   priority_flag: 0 =
0x00000000
2017-09-26 12:04:08 [15922] [6] DEBUG:   schedule_delivery_time:
NULL
2017-09-26 12:04:08 [15922] [6] DEBUG:   validity_period: NULL
2017-09-26 12:04:08 [15922] [6] DEBUG:   registered_delivery: 0 =
0x00000000
2017-09-26 12:04:08 [15922] [6] DEBUG:   replace_if_present_flag: 0
```

Copyright © Sipwise GmbH - All rights reserved

```
= 0x00000000
2017-09-26 12:04:08 [15922] [6] DEBUG:   data_coding: 0 =
0x00000000
2017-09-26 12:04:08 [15922] [6] DEBUG:   sm_default_msg_id: 0 =
0x00000000
2017-09-26 12:04:08 [15922] [6] DEBUG:   sm_length: 23 = 0x00000017
2017-09-26 12:04:08 [15922] [6] DEBUG:   short_message:
2017-09-26 12:04:08 [15922] [6] DEBUG:    Octet string at
0x7f227400c460:
2017-09-26 12:04:08 [15922] [6] DEBUG:      len:  23
2017-09-26 12:04:08 [15922] [6] DEBUG:      size: 24
2017-09-26 12:04:08 [15922] [6] DEBUG:      immutable: 0
2017-09-26 12:04:08 [15922] [6] DEBUG:      data: 44 <14 bytes> 73
2017-09-26 12:04:08 [15922] [6] DEBUG:      data: 74 <5 bytes> 39
2017-09-26 12:04:08 [15922] [6] DEBUG:    Octet string dump ends.
2017-09-26 12:04:08 [15922] [6] DEBUG: SMPP PDU dump ends.
2017-09-26 12:04:08 [15922] [6] DEBUG: SMPP[default_smsc]:
throughput (1.00,5.00)
...
```

Example link enquiry:

Copyright © Sipwise GmbH - All rights reserved

```
...
2017-09-26 12:13:38 [15922] [6] DEBUG: SMPP[default_smsc]:
throughput (0.00,5.00)
2017-09-26 12:13:38 [15922] [6] DEBUG: SMPP[default_smsc]: Got PDU:
2017-09-26 12:13:38 [15922] [6] DEBUG: SMPP PDU 0x7f2274020790
dump:
2017-09-26 12:13:38 [15922] [6] DEBUG:   type_name: enquire_link
2017-09-26 12:13:38 [15922] [6] DEBUG:   command_id: 21 =
0x00000015
2017-09-26 12:13:38 [15922] [6] DEBUG:   command_status: 0 =
0x00000000
2017-09-26 12:13:38 [15922] [6] DEBUG:   sequence_number: 90764 =
0x0001628c
2017-09-26 12:13:38 [15922] [6] DEBUG: SMPP PDU dump ends.
2017-09-26 12:13:38 [15922] [6] DEBUG: SMPP[default_smsc]: Sending
PDU:
2017-09-26 12:13:38 [15922] [6] DEBUG: SMPP PDU 0x7f2274025070
dump:
2017-09-26 12:13:38 [15922] [6] DEBUG:   type_name:
enquire_link_resp
2017-09-26 12:13:38 [15922] [6] DEBUG:   command_id: 2147483669 =
0x80000015
2017-09-26 12:13:38 [15922] [6] DEBUG:   command_status: 0 =
0x00000000
2017-09-26 12:13:38 [15922] [6] DEBUG:   sequence_number: 90764 =
0x0001628c
2017-09-26 12:13:38 [15922] [6] DEBUG: SMPP PDU dump ends.
2017-09-26 12:13:38 [15922] [6] DEBUG: SMPP[default_smsc]:
throughput (0.00,5.00)
...
```

SMS Box (PRX node of NGCP)

On the PRX node you can see a **process** named **"smsbox"**. This process has a **listening port** assigned to it: 13002, that is the communication port towards the *Bearer Box* component running on LB nodes.

The *ngcp-service* tool also shows the *smsbox* process in its summary information:

```
$ ngcp-service summary
Ok Service                        Managed    Started   Status
-- ------------------------------ ---------- --------- ---------
…
   kannel-smsbox                  managed    by-ha     active
…
```

The following log files can provide information about the operation of *SMS Box*:

- sent and received messages using the API of WEB node: /var/log/kannel/smsbox.log

Copyright © Sipwise GmbH - All rights reserved

NOTE Sent and received message examples shown here do not contain the full phone number and content for confidentiality reason.

Example sent message:

```
...
2017-09-26 12:16:42 [22763] [2] DEBUG: HTTP: Creating HTTPClient
for `192.168.1.2'.
2017-09-26 12:16:42 [22763] [2] DEBUG: HTTP: Created HTTPClient
area 0x7f5dcc000ad0.
2017-09-26 12:16:42 [22763] [3] INFO: smsbox: Got HTTP request
</cgi-bin/sendsms> from <192.168.1.3>
2017-09-26 12:16:42 [22763] [3] INFO: sendsms used by <sipwise>
2017-09-26 12:16:42 [22763] [3] INFO: sendsms
sender:<sipwise:43668.......> (192.168.1.3) to:<43676.......>
msg:<...>
2017-09-26 12:16:42 [22763] [3] DEBUG: Stored UUID ab95eb45-1ec0-
4932-9863-1a95609a025f
2017-09-26 12:16:42 [22763] [3] DEBUG: message length 52, sending 1
messages
2017-09-26 12:16:42 [22763] [3] DEBUG: Status: 202 Answer: <Sent.>
2017-09-26 12:16:42 [22763] [3] DEBUG: Delayed reply - wait for
bearerbox
2017-09-26 12:16:42 [22763] [0] DEBUG: Got ACK (0) of ab95eb45-
1ec0-4932-9863-1a95609a025f
2017-09-26 12:16:42 [22763] [0] DEBUG: HTTP: Destroying HTTPClient
area 0x7f5dcc000ad0.
2017-09-26 12:16:42 [22763] [0] DEBUG: HTTP: Destroying HTTPClient
for `192.168.1.3'.
...
```

Example received message:

```
...
2017-09-26 11:59:45 [22763] [5] INFO: Starting to service
<...message content...> from <+43676-------> to <+43668------->
2017-09-26 11:59:45 [22763] [10] DEBUG: Queue contains 0 pending
requests.
2017-09-26 11:59:45 [22763] [10] DEBUG: HTTPS URL; Using SSL for
the connection
2017-09-26 11:59:45 [22763] [10] DEBUG: Parsing URL
`https://192.168.1.2:1443/internalsms/receive?auth_token=fNLosMgwdN
UrKvEfFMm9
&timestamp=2017-09-26+09:59:45&from=%2B43676-------
&to=%2B43668-------&charset=UTF-8&coding=0&text=...':
2017-09-26 11:59:45 [22763] [10] DEBUG:   Scheme: https://
2017-09-26 11:59:45 [22763] [10] DEBUG:   Host: 192.168.1.2
2017-09-26 11:59:45 [22763] [10] DEBUG:   Port: 1443
2017-09-26 11:59:45 [22763] [10] DEBUG:   Username: (null)
2017-09-26 11:59:45 [22763] [10] DEBUG:   Password: (null)
```


Copyright © Sipwise GmbH - All rights reserved

```
2017-09-26 11:59:45 [22763] [10] DEBUG:   Path:
/internalsms/receive
2017-09-26 11:59:45 [22763] [10] DEBUG:   Query:
auth_token=fNLosMgwdNUrKvEfFMm9&timestamp=2017-09-
26+09:59:45&from=%2B43676-------
&to=%2B43668-------&charset=UTF-8&coding=0&text=...
2017-09-26 11:59:45 [22763] [10] DEBUG:   Fragment: (null)
2017-09-26 11:59:45 [22763] [10] DEBUG: Connecting nonblocking to
<192.168.1.2>
2017-09-26 11:59:45 [22763] [10] DEBUG: HTTP: Opening connection to
`192.168.1.2:1443' (fd=31).
2017-09-26 11:59:45 [22763] [10] DEBUG: Socket connecting
2017-09-26 11:59:45 [22763] [9] DEBUG: Get info about connecting
socket
2017-09-26 11:59:45 [22763] [9] DEBUG: HTTP: Sending request:
2017-09-26 11:59:45 [22763] [9] DEBUG: Octet string at
0x7f5dbc00f470:
2017-09-26 11:59:45 [22763] [9] DEBUG:   len:  382
2017-09-26 11:59:45 [22763] [9] DEBUG:   size: 1024
2017-09-26 11:59:45 [22763] [9] DEBUG:   immutable: 0
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 47 45 54 20 2f 69 6e
74 65 72 6e 61 6c 73 6d 73   GET /internalsms
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 2f 72 65 63 65 69 76
65 3f 61 75 74 68 5f 74 6f   /receive?auth_to
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 6b 65 6e 3d ...
ken=
                                                        ... 20 48
54 54 50 2f 31 2e 31 0d 0a         HTTP/1.1..
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 43 6f 6e 6e 65 63 74
69 6f 6e 3a 20 6b 65 65 70   Connection: keep
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 2d 61 6c 69 76 65 0d
0a 55 73 65 72 2d 41 67 65   -alive..User-Age
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 6e 74 3a 20 4b 61 6e
6e 65 6c 2f 31 2e 34 2e 34   nt: Kannel/1.4.4
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 0d 0a 48 6f 73 74 3a
20 31 39 32 2e 31 36 38 2e   ..Host: 192.168.
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 31 2e 32 3a 31 34 34
33 0d 0a 0d 0a               1.2:1443....
2017-09-26 11:59:45 [22763] [9] DEBUG: Octet string dump ends.
2017-09-26 11:59:45 [22763] [9] DEBUG: HTTP: Status line: <HTTP/1.1
200 OK>
2017-09-26 11:59:45 [22763] [9] DEBUG: HTTP: Received response:
2017-09-26 11:59:45 [22763] [9] DEBUG: Octet string at
0x7f5dbc006970:
2017-09-26 11:59:45 [22763] [9] DEBUG:   len:  333
2017-09-26 11:59:45 [22763] [9] DEBUG:   size: 1024
2017-09-26 11:59:45 [22763] [9] DEBUG:   immutable: 0
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 53 65 72 76 65 72 3a
20 6e 67 69 6e 78 0d 0a 44   Server: nginx..D
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 61 74 65 3a 20 54 75
65 2c 20 32 36 20 53 65 70   ate: Tue, 26 Sep
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 20 32 30 31 37 20 30
```

Copyright © Sipwise GmbH - All rights reserved

```
39 3a 35 39 3a 34 35 20 47    2017 09:59:45 G
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 4d 54 0d 0a 43 6f 6e
74 65 6e 74 2d 54 79 70 65   MT..Content-Type
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 3a 20 74 65 78 74 2f
68 74 6d 6c 3b 20 63 68 61   : text/html; cha
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 72 73 65 74 3d 75 74
66 2d 38 0d 0a 43 6f 6e 74   rset=utf-8..Cont
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 65 6e 74 2d 4c 65 6e
67 74 68 3a 20 30 0d 0a 43   ent-Length: 0..C
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 6f 6e 6e 65 63 74 69
6f 6e 3a 20 6b 65 65 70 2d   onnection: keep-
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 61 6c 69 76 65 0d 0a
53 65 74 2d 43 6f 6f 6b 69   alive..Set-Cooki
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 65 3a 20 6e 67 63 70
5f 70 61 6e 65 6c 5f 73 65   e: ngcp_panel_se
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 73 73 69 6f 6e 3d 34
35 30 32 64 64 66 65 31 62   ssion=4502ddfe1b
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 63 31 65 33 39 30 65
30 64 36 66 39 64 34 37 30   c1e390e0d6f9d470
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 35 30 37 62 64 64 33
61 65 32 36 62 64 63 3b 20   507bdd3ae26bdc;
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 70 61 74 68 3d 2f 3b
20 65 78 70 69 72 65 73 3d   path=/; expires=
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 54 75 65 2c 20 32 36
2d 53 65 70 2d 32 30 31 37   Tue, 26-Sep-2017
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 20 31 30 3a 35 39 3a
34 35 20 47 4d 54 3b 20 48    10:59:45 GMT; H
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 74 74 70 4f 6e 6c 79
0d 0a 58 2d 43 61 74 61 6c   ttpOnly..X-Catal
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 79 73 74 3a 20 35 2e
39 30 30 37 35 0d 0a 53 74   yst: 5.90075..St
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 72 69 63 74 2d 54 72
61 6e 73 70 6f 72 74 2d 53   rict-Transport-S
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 65 63 75 72 69 74 79
3a 20 6d 61 78 2d 61 67 65   ecurity: max-age
2017-09-26 11:59:45 [22763] [9] DEBUG:   data: 3d 31 35 37 36 38 30
30 30 0d 0a 0d 0a            =15768000....
2017-09-26 11:59:45 [22763] [9] DEBUG: Octet string dump ends.
2017-09-26 11:59:45 [22763] [6] WARNING: Tried to set Coding field,
denied.
2017-09-26 11:59:45 [22763] [6] INFO: No reply sent, denied.
2017-09-26 11:59:55 [22763] [9] DEBUG: HTTP: Server closed
connection, destroying it
<192.168.1.2:1443:1::><0x7f5db0000b20><fd:31>.
...
```

- short log of sent/received messages: /var/log/kannel/smsbox-access.log

Copyright © Sipwise GmbH - All rights reserved

```
...
2017-09-26 12:39:18 SMS HTTP-request sender:+43680------- request: ''
url: 'https://192.168.1.2:1443/internalsms/receive?
auth_token=fNLosMgwdNUrKvEfFMm9&timestamp=2017-09-
26+10:39:18&from=%2B43680-------&to=%2B43668-------&charset=UTF-
8&coding=0
&text=<...message content...>' reply: 200 '<< successful >>'
...
2017-09-26 12:41:54 send-SMS request added -
sender:sipwise:43668------- 192.168.1.3 target:43680-------- request:
'<...message content...>'
...
```

8.6.3. REST API

Handling of short messages from the user perspective happens with the help of NGCP's REST API. There is a dedicated resource: `https://<IP of WEB node>:1443/api/sms` that allows you to:

- Get a **list of sent and received messages**. This is achieved by sending a GET request on the /api/sms collection, as in the following example:

```
curl -i -X GET -H 'Connection: close' --cert NGCP-API-client-
certificate.pem \
  --cacert ca-cert.pem
'https://example.org:1443/api/sms/?page=1&rows=10'
```

- **Retrieve an SM** (both sent and received). This is achieved by sending a GET request for a specific /api/sms/id item, as in the following example:

```
curl -i -X GET -H 'Connection: close' --cert NGCP-API-client-
certificate.pem \
  --cacert ca-cert.pem 'https://example.org:1443/api/sms/1'
```

- **Send a new message** from a local subscriber. This is achieved by sending a POST request for the /api/sms collection, as in the following example:

```
curl -i -X POST -H 'Connection: close' -H 'Content-Type:
application/json' \
  --cert NGCP-API-client-certificate.pem --cacert ca-cert.pem \
  'https://example.org:1443/api/sms/'  --data-binary '{"callee" :
"43555666777", \
  "subscriber_id" : 4, "text" : "test"}'
```

As always, the full documentation of the REST API resources is available on the admin web interface of NGCP: `https://<IP of WEB node>:1443/api/#sms`

Copyright © Sipwise GmbH - All rights reserved

Chapter 9. Configuration Framework

The Sipwise C5 provides a configuration framework for consistent and easy to use low level settings management. A basic usage of the configuration framework only needs two actions already used in previous chapters:

- Edit */etc/ngcp-config/config.yml* file.
- Execute *ngcpcfg apply 'my commit message'* command.

Low level management of the configuration framework might be required by advanced users though. This chapter explains the architecture and usage of Sipwise C5 configuration framework. If the basic usage explained above fits your needs, feel free to skip this chapter and return to it when your requirements change.

A more detailed workflow of the configuration framework for creating a configuration file consists of 7 steps:

- Generation or editing of configuration templates and/or configuration values.
- Generation of the configuration files based on configuration templates and configuration values defined in config.yml, constants.yml and network.yml files.
- Execution of *prebuild* commands if defined for a particular configuration file or configuration directory.
- Placement of the generated configuration file in the target directory. This step is called *build* in the configuration framework.
- Execution of *postbuild* commands if defined for that configuration file or configuration directory.
- Execution of *services* commands if defined for that configuration file or configuration directory. This step is called *services* in the configuration framework.
- Saving of the generated changes. This step is called *commit* in the configuration framework.

9.1. Configuration templates

The Sipwise C5 provides configuration file templates for most of the services it runs. These templates are stored in the directory */etc/ngcp-config/templates*.

Example: Template files for */etc/kamailio/proxy/kamailio.cfg* are stored in */etc/ngcp-config/templates/etc/kamailio/proxy/*.

There are different types of files in this template framework, which are described below.

9.1.1. .tt2, .customtt.tt2 and .patchtt.tt2 files

These files are the main template files that will be used to generate the final configuration file for the running service. They contain all the configuration options needed for a running Sipwise C5 system. The configuration framework will combine these files with the values provided by *config.yml*, *constants.yml* and *network.yml* to generate the appropriate configuration file.

Example: Let's say we are changing the IP used by kamailio load balancer on interface *eth0* to IP 1.2.3.4. This will change kamailio's listen IP address, when the configuration file is generated. A quick look to

 Copyright © Sipwise GmbH - All rights reserved

the template file under */etc/ngcp-config/templates/etc/kamailio/lb/kamailio.cfg.tt2* will show a line like this:

```
listen=udp:[% ip %]:[% kamailio.lb.port %]
```

After applying the changes with the *ngcpcfg apply 'my commit message'* command, a new configuration file will be created under */etc/kamailio/lb/kamailio.cfg* with the proper values taken from the main configuration files (in this case *network.yml*):

```
listen=udp:1.2.3.4:5060
```

All the low-level configuration is provided by these .tt2 template files and the corresponding config.yml file. Anyway, advanced users might require a more particular configuration.

Instead of editing .tt2 files, the configuration framework recognises .customtt.tt2 files. These files are the same as .tt2, but they have higher priority when the configuration framework creates the final configuration files. If you need to introduce changes in a template, you must always copy the required .tt2 file to .customtt.tt2, make changes in the latter file one and leave the .tt2 file untouched. This way, the system will use the new custom configuration allowing you to switch back to the original one quickly.

Example: We'll create */etc/ngcp-config/templates/etc/lb/kamailio.cfg.customtt.tt2* and use it for our customized configuration. In this example, we'll append a comment at the end of the template.

```
cd /etc/ngcp-config/templates/etc/kamailio/lb
cp kamailio.cfg.tt2 kamailio.cfg.customtt.tt2
echo '# This is my last line comment' >> kamailio.cfg.customtt.tt2
ngcpcfg apply 'my commit message'
```

The ngcpcfg command will generate */etc/kamailio/lb/kamailio.cfg* from our custom template instead of the general one:

```
tail -1 /etc/kamailio/lb/kamailio.cfg
# This is my last line comment
```

WARNING users have to upgrade all .customtt.tt2 manually every time .tt2 is upgraded, as ngcpcfg completely ignores new code in .tt2 received from new package version.

The huge drawback of .customtt.tt2 files are necessity to keep them up-to-date manually. Keeping them outdated will cause the system misbehaviour as different components will use different code version (as new .tt2 version will be overwritten by old .customtt.tt2).

The .patchtt.tt2 concept should help users here. It will minimise the manual efforts by using the "patch(1)" utility. The ngcpcfg tool is searching for .patchtt.tt2 files every time 'ngcpcfg build' has been called. If .patchtt.tt2 is detected, the ngcpcfg tool will try to apply .patchtt.tt2 on .tt2 and store result in .customtt.tt2 if no conflicts noticed during patching. Further building process happens in a common

Copyright © Sipwise GmbH - All rights reserved

way. Example:

```
root@spce:~# ngcpcfg build /etc/kamailio/lb/kamailio.cfg
spce: yml configs were validated successfully
spce: configs were checked successfully
spce: Validating patch '/etc/ngcp-
config/templates/etc/kamailio/lb/kamailio.cfg.patchtt.tt2'
spce: Applying patch '/etc/ngcp-
config/templates/etc/kamailio/lb/kamailio.cfg.patchtt.tt2'
spce: Successfully created '/etc/ngcp-
config/templates/etc/kamailio/lb/kamailio.cfg.customtt.tt2'
spce: Requested patchtt operation has finished successfully.
Loading /etc/ngcp-config/config.yml in memory: OK
Loading /etc/ngcp-config/network.yml in memory: OK
Loading /etc/ngcp-config/constants.yml in memory: OK
spce: Generating /etc/kamailio/lb/kamailio.cfg: OK
spce: Executing postbuild for /etc/kamailio/lb/kamailio.cfg
root@spce:~#
```

To convert some/all the current .customtt.tt2 users can use command *ngcpcfg patch --from-customtt [<customtt_file>]*:

```
root@spce:~# ngcpcfg patch --from-customtt /etc/ngcp-
config/templates/etc/kamailio/lb/kamailio.cfg.customtt.tt2
spce: Validating customtt '/etc/ngcp-
config/templates/etc/kamailio/lb/kamailio.cfg.customtt.tt2'
spce: Creating patchtt file '/etc/ngcp-
config/templates/etc/kamailio/lb/kamailio.cfg.patchtt.tt2'
spce: Requested customtt operation has finished successfully.
root@spce:~#
```

Here is the example of newly created .patchtt.tt2 file:

```
root@spce:~# cat /etc/ngcp-
config/templates/etc/kamailio/lb/kamailio.cfg.patchtt.tt2
@@ -1799,3 +1799,4 @@
 }

 # vim: ft=cfg
+# This is my last line comment
root@spce:~#
```

See more details about .patchtt.tt2 files below in patchtt section.

TIP The .tt2 files use the Template Toolkit language. Therefore you can use all the feature this excellent toolkit provides within ngcpcfg's template files (all the ones with the .tt2 suffix).

Copyright © Sipwise GmbH - All rights reserved

9.1.2. Using patchtt for generation of a relevant customtt file

Keeping custom modifications directly in the .customtt.tt2 templates is NOT recommended as templates become outdated with every software upgrade.

A better way is to handle custom modifications using .patchtt.tt2 files (e.g. /etc/ngcp-config/templates/etc/cron.d/cleanup-tools.patchtt.tt2). In this case, on every "ngcpcfg patch", a .patchtt.tt2 file will be applied on top of the .tt2 file and the result will be saved into the customtt file and used commonly as described in the previous section. "ngcpcfg patch" is the first step on "ngcpcfg build" that guarantees the latest upstream templates with the availability of the necessary local changes on every configuration apply.

TIP The patch to be applied to the corresponding .tt2 template file is selected in the following order (highest to lowest): *.patchtt.tt2.$HOSTNAME *.patchtt.tt2.$PAIRNAME *.patchtt.tt2.$HA_NODE *.patchtt.tt2

NOTE If a suitable patchtt file is found for a template, then the ngcpcfg patch command will overwrite the corresponding customtt file, if any.

Creating a patchtt file

Let us see how to introduce custom changes into a template through a patchtt file. For example, we need to change the accounting records cleanup time, which is defined in *cleanup-tools.tt2*. Here is how to do this:

- Go to the corresponding templates directory:

```
cd /etc/ngcp-config/templates/etc/cron.d/
```

- Duplicate the required .tt2 file to .customtt.tt2

```
cp ./cleanup-tools.tt2 ./cleanup-tools.customtt.tt2
```

- Introduce the necessary changes to the duplicated file:

```
vim ./cleanup-tools.customtt.tt2
```

- Create the patchtt file from your customtt file and recheck it:

```
ngcpcfg patch --from-customtt ./cleanup-tools.customtt.tt2
cat ./cleanup-tools.patchtt.tt2
```

- Apply and push the changes

Copyright © Sipwise GmbH - All rights reserved

```
ngcpcfg apply "Change acc-cleanup time from 00 to 02 hours"
ngcpcfg push all
```

You will notice that the "ngcpcfg apply" command has generated the customtt file for the corresponding template:

```
root@web01a:/etc/ngcp-config/templates/etc/cron.d# ls -l ./cleanup-
tools*
-rw------- 1 root root 932 Jan  4 11:11 ./cleanup-tools.customtt.tt2
-rw-r--r-- 1 root root 630 Jan  4 11:08 ./cleanup-tools.patchtt.tt2
-rw-r--r-- 1 root root 932 Dec 18 15:09 ./cleanup-tools.tt2
```

Now, even if cleanup-tools.tt2 slightly changes after a software upgrade, "ngcpcfg apply" will still preserve your custom changes.

NOTE If in a new release the .tt2 file gets changed in the same lines where you had introduced custom changes (e.g. your changes were temporary until a feature is implemented properly in a new software release), the apply process will fail and ask you to review the corresponding .patchtt.tt2 file. Then, check it and either correct if it is still required or remove it.

TIP To convert all existing customtt files to patchtt files use the command: **ngcpcfg patch --from-customtt**

9.1.3. .prebuild and .postbuild files

After creating the configuration files, the configuration framework can execute some commands before and after placing that file in its target directory. These commands usually are used for changing the file's owner, groups, or any other attributes. There are some rules these commands need to match:

- They have to be placed in a *.prebuild* or *.postbuild* file in the same path as the original *.tt2* file.
- The file name must be the same as the configuration file, but having the mentioned suffixes.
- The commands must be *bash* compatible.
- The commands must return 0 if successful.
- The target configuration file is matched by the environment variable *output_file*.

Example: We need *www-data* as owner of the configuration file */etc/ngcp-ossbss/provisioning.conf*. The configuration framework will by default create the configuration files with root:root as owner:group and with the same permissions (rwx) as the original template. For this particular example, we will change the owner of the generated file using the *.postbuild* mechanism.

```
echo 'chgrp www-data ${output_file}' \
  > /etc/ngcp-config/templates/etc/ngcp-
ossbss/provisioning.conf.postbuild
```


Copyright © Sipwise GmbH - All rights reserved

9.1.4. .services files

.services files are pretty similar and might contain commands that will be executed after the *build* process. There are two types of *.services* files:

- The particular one, with the same name as the configuration file it is associated to.
  Example: */etc/ngcp-config/templates/etc/asterisk/sip.conf.services* is associated to */etc/asterisk/sip.conf*
- The general one, named *ngcpcfg.services* that is associated to every file in its target directory.
  Example: */etc/ngcp-config/templates/etc/asterisk/ngcpcfg.services* is associated to every file under */etc/asterisk/*

When the *services* step is triggered all *.services* files associated to a changed configuration file will be executed. In case of the general file, any change to any of the configuration files in the directory will trigger the execution of the commands.

TIP If the service script has the execute flags set (*chmod +x $file*) it will be invoked directly. If it doesn't have execute flags set it will be invoked under bash. Make sure the script is bash compatible if you do not set execute permissions on the service file.

These commands are usually service reload/restarts to ensure the new configuration has been loaded by running services.

NOTE The configuration files mentioned in the following example usually already exist on the platform. Please make sure you don't overwrite any existing files if following this example.

Example:

```
echo 'ngcp-service restart mariadb' \
  > /etc/ngcpcfg-config/templates/etc/mysql/my.cnf.services
echo 'ngcp-service restart asterisk' \
  > /etc/ngcpcfg-config/templates/etc/asterisk/ngcpcfg.services
```

In this example we created two *.services* files. Now, each time we trigger a change to */etc/mysql.my.cnf* or to */etc/asterisk/** we'll see that MySQL or Asterisk services will be restarted by the ngcpcfg system.

9.2. config.yml, constants.yml and network.yml files

The */etc/ngcp-config/config.yml* file contains all the user-configurable options, using the YAML (YAML Ain't Markup Language) syntax.

The */etc/ngcp-config/constants.yml* file provides configuration options for the platform that aren't supposed to be edited by the user. Do not manually edit this file unless you really know what you're doing.

The */etc/ngcp-config/network.yml* file provides configuration options for all interfaces and IP addresses on those interfaces. You can use the *ngcp-network* tool for conveniently change settings without having to manually edit this file.

Copyright © Sipwise GmbH - All rights reserved

The */etc/ngcp-ngcpcfg/ngcpcfg.cfg* file is the main configuration file for ngcpcfg itself. Do not manually edit this file unless you really know what you are doing.

9.3. ngcpcfg and its command line options

On a CARRIER the shared storage used by all nodes is the shared storage of the mgmt pair.

The ngcpcfg utility supports the following command line options:

9.3.1. apply

The *apply* option is a short-cut for the options "check && build && services && commit" and also executes *etckeeper* to record any modified files inside /etc. It is the recommended option to use the ngcpcfg framework unless you want to execute any specific commands as documented below.

9.3.2. build

The *build* option generates (and therefore also updates) configuration files based on their configuration (config.yml) and template files (.tt2). Before the configuration file is generated a present .prebuild will be executed, after generation of the configuration file the according .postbuild script (if present) will be executed. If a *file* or *directory* is specified as argument the build will generate only the specified configuration file/directory instead of running through all present templates.

Example: to generate only the file /etc/nginx/sites-available/ngcp-panel you can execute:

```
ngcpcfg build /etc/nginx/sites-available/ngcp-panel
```

Example: to generate all the files located inside the directory /etc/nginx/ you can execute:

```
ngcpcfg build /etc/nginx/
```

9.3.3. commit

The *commit* option records any changes done to the configuration tree inside */etc/ngcp-config*. The commit option should be executed when you've modified anything inside the configuration tree.

9.3.4. decrypt

Decrypt /etc/ngcp-config-crypted.tgz.gpg and restore configuration files, doing the reverse operation of the *encrypt* option. Note: This feature is only available if the ngcp-ngcpcfg-locker package is installed.

9.3.5. diff

Show uncommitted changes between ngcpcfg's Git repository and the working tree inside */etc/ngcp-config*. Iff the tool doesn't report anything it means that there are no uncommitted changes. If the *--addremove* option is specified then new and removed files (iff present) that are not yet (un)registered to the repository will be reported, no further diff actions will be executed then. Note: This option is

 Copyright © Sipwise GmbH - All rights reserved

available since ngcp-ngcpcfg version 0.11.0.

9.3.6. encrypt

Encrypt /etc/ngcp-config and all resulting configuration files with a user defined password and save the result as /etc/ngcp-config-crypted.tgz.gpg. Note: This feature is only available if the ngcp-ngcpcfg-locker package is installed.

9.3.7. help

The *help* options displays ngcpcfg's help screen and then exits without any further actions.

9.3.8. initialise

The *initialise* option sets up the ngcpcfg framework. This option is automatically executed by the installer for you, so you shouldn't have to use this option in normal operations mode.

9.3.9. pull

Retrieve modifications from shared storage.

9.3.10. push

Push modifications to shared storage and remote systems. After changes have been pushed to the nodes the *build* option will be executed on each remote system to rebuild the configuration files (unless the --nobuild has been specified, then the build step will be skipped). If hostname(s) or IP address(es) is given as argument then the changes will be pushed to the shared storage and to the given hosts only. You can use 'all' as a shortcut to push to the other nodes. If no host has been specified then the hosts for this pair specified in */etc/ngcp-config/network.yml* are used.

9.3.11. services

The *services* option executes the service handlers for any modified configuration file(s)/directory.

9.3.12. status

The *status* option provides a human readable interface to check the state of the configuration tree. If you are unsure what should be done as next step or if want to check the current state of the configuration tree, invoke *ngcpcfg status*.

If everything is OK and nothing needs to be done the output should look like:

```
# ngcpcfg status
Checking state of ngcpcfg:
OK:   has been initialised already (without shared storage)
Checking state of configuration files:
OK:   nothing to commit.
Checking state of /etc files
OK:   nothing to commit.
```

Copyright © Sipwise GmbH - All rights reserved

If the output doesn't say "OK", follow the instructions provided by the output of 'ngcpcfg status'.

Further details regarding the ngcpcfg tool are available through 'man ngcpcfg' on the Sipwise Next Generation Platform.

Copyright © Sipwise GmbH - All rights reserved

Chapter 10. Network Configuration

Starting with version 2.7, Sipwise C5 uses a dedicated *network.yml* file to configure the IP addresses of the system. The reason for this is to be able to access all IPs of all nodes for all services from any particular node in case of a distributed system on one hand, and in order to be able the generate */etc/network/interfaces* automatically for all nodes based on this central configuration file.

10.1. General Structure

The basic structure of the file looks like this:

```
hosts:
  self:
    role:
      - proxy
      - lb
      - mgmt
    interfaces:
      - eth0
      - lo
    eth0:
      ip: 192.168.51.213
      netmask: 255.255.255.0
      type:
        - sip_ext
        - rtp_ext
        - web_ext
        - web_int
    lo:
      ip: 127.0.0.1
      netmask: 255.255.255.0
      type:
        - sip_int
        - ha_int
```

Some more complete, sample configuration is shown in network.yml Overview section of the handbook.

The file contains all configuration parameters under the main key: hosts

In Sipwise C5 systems all hosts of the system are defined, and the names are the actual host names instead of *self*, like this:

On a PRO it would look like:

Copyright © Sipwise GmbH - All rights reserved

```
hosts:

  sp1:
    peer: sp2
    role: ...
    interfaces: ...

  sp2:
    peer: sp1
    role: ...
    interfaces: ...
```

On a CARRIER it would look like:

```
hosts:

  web01a:
    peer: web01b
    role: ...
    interfaces: ...

  web01b:
    peer: web01a
    role: ...
    interfaces: ...
```

10.1.1. Available Host Options

There are three different main sections for a host in the config file, which are *role*, *interfaces* and the actual interface definitions.

- *role*: The role setting is an array defining which logical roles a node will act as. Possible entries for this setting are:

  mgmt: This entry means the host is acting as management node for the platform. In a Sipwise C5 system this option must always be set. The management node exposes the admin and CSC panels to the users and the APIs to external applications and is used to export CDRs. Please note: on a CARRIER this is only set on the nodes of the management pairs. This node is also the source of the installations of other nodes via iPXE and has the *approx* service (apt proxy).

  lb: This entry means the host is acting as SIP load-balancer for the platform. In a Sipwise C5 system this option must always be set. Please note: on a CARRIER this is only set on the nodes of the *lb* pairs. The SIP load-balancer acts as an ingress and egress point for all SIP traffic to and from the platform.

  proxy: This entry means the host is acting as SIP proxy for the platform. In a Sipwise C5 system this option must always be set. Please note: on a CARRIER this is only set on the nodes of the *proxy* pairs. The SIP proxy acts as registrar, proxy and application server and media relay, and is responsible for providing the features for all subscribers provisioned on it.

  db: This entry means the host is acting as the database node for the platform. In a Sipwise C5

Copyright © Sipwise GmbH - All rights reserved

system this option must always be set. Please note: on a CARRIER this is only set on the nodes of the *db* pairs. The database node exposes the MySQL and Redis databases.

rtp: This entry means the host is acting as the RTP relay node for the platform. In a Sipwise C5 system this option must always be set. Please note: on a CARRIER this is only set on the nodes of the *RTP relay* pairs. The RTP relay node runs the *rtpengine* Sipwise C5 component.

li: (CARRIER-only) This entry means the host is acting as the interface towards a lawful interception service provider. Note that the virtual *li_dist* role cannot be specified in the configuration file as it is synthesized at run-time.

- *interfaces*: The interfaces setting is an array defining all interface names in the system. The actual interface details are set in the actual interface settings below. It typically includes lo, eth0, eth1 physical and a number of virtual interfaces, like: bond0, vlanXXX
- <*interface name*>: After the interfaces are defined in the *interfaces* setting, each of those interfaces needs to be specified as a separate set of parameters.

Additional main parameters of a node:

- *dbnode*: the sequence number (unique ID) of the node in the database cluster;
- *peer*: the hostname of the peer node within the pair of nodes (e.g. "sp2" for *sp1* host on a PRO; "web01b" for *web01a* host on a CARRIER). The purpose of that: each node knows its companion for providing high availability, data replication etc.
- *status*: one of 'online', 'offline', 'inactive'. 'inactive' means that the node is up but is not ready to work in the cluster (installing process). 'offline' means that the node is not reachable. 'online' is a normal working node.

10.1.2. Interface Parameters

- hwaddr: MAC address of the interface

CAUTION	On a CARRIER this *must* be filled in properly for the interface that is used as type ha_int, because the value of it will be used during the boot process of the installation of nodes via iPXE, if PXE-boot is enabled.

- ip: IPv4 address of the node
- v6ip: IPv6 address of the node; optional
- netmask: IPv4 netmask
- v6netmask: IPv6 netmask
- gateway: IPv4 gateway address
- v6gateway: IPv6 gateway address
- shared_ip: shared IPv4 address of the pair of nodes; this is a list of addresses
- shared_v6ip: shared IPv6 address of the pair of nodes; optional; this is a list of addresses
- shared_ip_only: Boolean switch ('yes' or 'no') to enable usage with only a shared floating IPv4 address, without a static IPv4 address configured. To prevent accidental misconfiguration, this usage mode must be explicitly enabled. This usage mode is disallowed for certain interface types (e.g. 'ssh', 'mon', or 'ha').
- shared_v6ip_only: same as above, but for IPv6

Copyright © Sipwise GmbH - All rights reserved

- advertised_ip: the IP address that is used in SIP messages when Sipwise C5 system is behind NAT/SBC. An example of such a deployment is *Amazon AMI*, where the server doesn't have a public IP, so *load-balancer* component of Sipwise C5 needs to know what his public domain is (advertised_ip).
- type: type of services that the node provides; these are usually the VLANs defined for a particular Sipwise C5 system.

  NOTE You can assign a type only once per node.

  Available types are:

  api_int: internal, API-based communication interface. It is used for the internal communication of such services as faxserver, fraud detection and others.

  aux_ext: interface for potentially insecure external components like remote system log collection service.

  NOTE For example the *CloudPBX* module can use it to provide time services and remote logging facilities to end customer devices. The type 'aux_ext' is assigned to *lo* interface by default. If it is needed to expose this type to the public, it is recommended to assign the type *aux_ext* to a separate VLAN interface to be able to limit or even block the incoming traffic easily via firewalling in case of emergency, like a (D)DoS attack on external services.

  mon_ext: remote monitoring interface (e.g. SNMP)

  rtp_ext: main (external) interface for media traffic

  sip_ext: main (external) interface for SIP signalling traffic between NGCP and other SIP endpoints

  sip_ext_incoming: additional, optional interface for incoming SIP signalling traffic

  sip_int: internal SIP interface used by Sipwise C5 components (*lb, proxy, etc.*)

  ssh_ext: command line (SSH) remote access interface

  ssh_int: command line (SSH) internal NGCP access interface

  web_ext: interface for web-based or API-based provisioning and administration

  web_int: interface for the administrator's web panel, his API and generic internal API communication

  li_int: used for LI (Lawful Interception) traffic routing

  ha_int: HA (High Availability) communication interface between the services

  boot_int: the default VLAN used to install nodes via PXE-boot method

  rtp_int: internal interface for handling RTP traffic among Sipwise C5 nodes that may reside in greater distance from each other, like in case of a specialised NGCP configuration with centralized web / DB / proxy nodes and distributed LB nodes. (Please refer to Cluster Sets section for further details)

  NOTE Please note that, apart from the standard ones described so far, there might be other *types* defined for a particular Sipwise C5 system.

- vlan_raw_device: tells which physical interface is used by the particular VLAN

 Copyright © Sipwise GmbH - All rights reserved

- post_up: routes can be defined here (interface-based routing), for example:

```
post_up:
- route add -host 1.2.3.4 gw 192.168.1.1 dev vlan70
- route add -net 10.11.12.0/21 gw 192.168.1.2 dev vlan300
- route del -host 1.2.3.4 gw 192.168.1.1 dev vlan70
- route del -net 10.11.12.0/21 gw 192.168.1.2 dev vlan300
```

- bond_XY: specific to "bond0" interface only; these contain Ethernet bonding properties

10.2. Advanced Network Configuration

You have a typical deployment now and you are good to go, however you may need to do extra configuration depending on the devices you are using and functionality you want to achieve.

10.2.1. Additional entries in /etc/hosts

The file /etc/hosts is generated by a template, containing entries for basic host configuration (localhost and basic IPv4/IPv6), and the IPs of other nodes in PRO/CARRIER configurations.

To add extra entries in this file, it can be done in several ways:

- etc_hosts_global_extra_entries at the global level, added to all hosts
- etc_hosts_global_extra_entries at the host level, which overrides the global one if for some reason the whole content is undesired for a particular host (e.g. to have some but not all of the "default" global entries)
- etc_hosts_local_extra_entries at the host level, which are added only to the hosts where this entry is present, if for some reason it is desired to have extra entries only visible in some subset of the hosts

The behaviour is the same in all cases, to append the entries directly to /etc/hosts.

Example of both in a configuration file:

Copyright © Sipwise GmbH - All rights reserved

```
---
hosts_common:
  etc_hosts_global_extra_entries:
  - 10.100.1.1 server-1 server-1.internal.example.com
  - 10.100.1.2 server-2 server-2.internal.example.com
hosts:
  db01b:
    etc_hosts_local_extra_entries:
    - 127.0.1.1 local-alias-1.db01b
    - 127.0.2.1 local-alias-2.db01b
    - 172.30.52.180 db01b.example.com
    ...
  web01a:
    etc_hosts_local_extra_entries:
    - 127.0.1.1 local-alias-1.web01a
    - 127.0.2.1 local-alias-2.web01a
    - 172.30.52.168 web01a.example.com
    etc_hosts_global_extra_entries:
    - 10.100.1.1 server-1 server-1.internal.example.com
    ...
```

With this, the additional output in /etc/hosts for db01b will be:

```
# local extra entries for host 'db01b'
127.0.1.1 local-alias-1.db01b
127.0.2.1 local-alias-2.db01b
172.30.52.180 db01b.example.com

# global extra entries
10.100.1.1 server-1 server-1.internal.example.com
10.100.2.1 server-2 server-2.internal.example.com
```

and in web01a:

```
# local extra entries for host 'web01a'
127.0.1.1 local-alias-1.web01a
127.0.2.1 local-alias-2.web01a
172.30.52.168 web01a.example.com

# global extra entries overridden for host 'web01a'
10.100.1.1 server-1 server-1.internal.example.com
```

10.2.2. Extra SIP Sockets

By default, the load-balancer listens on the UDP and TCP ports 5060 (*kamailiolbport*) and TLS port 5061 (*kamailiolbtlsport*). If you need to setup one or more extra SIP listening ports or IP addresses in addition to those standard ports, please edit the *kamailiolbextra_sockets* option in your */etc/ngcp-config/config.yml* file.

Copyright © Sipwise GmbH - All rights reserved

The correct format consists of a label and value like this:

```
    extra_sockets:
      port_5064: udp:10.15.20.108:5064
      test: udp:10.15.20.108:6060
```

The label is shown in the outbound_socket peer preference (if you want to route calls to the specific peer out via specific socket); the value must contain a transport specification as in example above (udp, tcp or tls). After adding execute ngcpcfg apply:

```
ngcpcfg apply 'added extra socket'
ngcpcfg push all
```

The direction of communication through this SIP extra socket is incoming+outgoing. The Sipwise C5 will answer the incoming client registrations and other methods sent to the extra socket. For such incoming communication no configuration is needed. For the outgoing communication the new socket must be selected in the outbound_socket peer preference. For more details read the next section Extra SIP and RTP Sockets that covers peer configuration for SIP and RTP in greater detail.

IMPORTANT In this section you have just added an extra SIP socket. RTP traffic will still use your *rtp_ext* IP address.

10.2.3. Extra SIP and RTP Sockets

If you want to use an additional interface (with a different IP address) for SIP signalling and RTP traffic you need to add your new interface in the */etc/network/interfaces* file. Also the interface must be declared in */etc/ngcp-config/network.yml*.

Suppose we need to add a new SIP socket and a new RTP socket on VLAN 100. You can use the *ngcp-network* tool for adding interfaces without having to manually edit this file: On a PRO system that would be:

```
ngcp-network --set-interface=eth0.100 --host=sp1 --ip=auto
--netmask=auto --hwaddr=auto --type=sip_ext_incoming --type=rtp_int_100
ngcp-network --set-interface=eth0.100 --host=sp2 --ip=auto
--netmask=auto --hwaddr=auto --type=sip_ext_incoming --type=rtp_int_100
```

On a CARRIER system that would be:

Copyright © Sipwise GmbH - All rights reserved

```
ngcp-network --set-interface=eth0.100 --host=lb01a --ip=auto
--netmask=auto --hwaddr=auto --type=sip_ext_incoming
ngcp-network --set-interface=eth0.100 --host=lb01b --ip=auto
--netmask=auto --hwaddr=auto --type=sip_ext_incoming
ngcp-network --set-interface=eth0.100 --host=prx01a --ip=auto
--netmask=auto --hwaddr=auto --type=rtp_int_100
ngcp-network --set-interface=eth0.100 --host=prx01b --ip=auto
--netmask=auto --hwaddr=auto --type=rtp_int_100
```

On a PRO system the generated files would look like:

 Copyright © Sipwise GmbH - All rights reserved

```
sp1:
..
..
    eth0.100:
      hwaddr: ff:ff:ff:ff:ff:ff
      ip: 192.168.1.2
      netmask: 255.255.255.0
      shared_ip:
        - 192.168.1.3
      shared_v6ip: ~
      type:
        - sip_ext_incoming
        - rtp_int_100
..
..
    interfaces:
      - lo
      - eth0
      - eth0.100
      - eth1
..
..
sp2:
..
..
    eth0.100:
      hwaddr: ff:ff:ff:ff:ff:ff
      ip: 192.168.1.4
      netmask: 255.255.255.0
      shared_ip:
        - 192.168.1.3
      shared_v6ip: ~
      type:
        - sip_ext_incoming
        - rtp_int_100
..
..
    interfaces:
      - lo
      - eth0
      - eth0.100
      - eth1
```

On a CARRIER system the generated files would look like:

```
lb01a:
..
..
    eth0.100:
      hwaddr: ff:ff:ff:ff:ff:ff
```

Copyright © Sipwise GmbH - All rights reserved

```
      ip: 192.168.1.2
      netmask: 255.255.255.0
      shared_ip:
        - 192.168.1.3
      shared_v6ip: ~
      type:
        - sip_ext_incoming
..
..
    interfaces:
      - lo
      - eth0
      - eth0.100
      - eth1
..
..
prx01a:
..
..
    eth0.100:
      hwaddr: ff:ff:ff:ff:ff:ff
      ip: 192.168.1.20
      netmask: 255.255.255.0
      shared_ip:
        - 192.168.1.30
      shared_v6ip: ~
      type:
        - rtp_int_100
..
..
    interfaces:
      - lo
      - eth0
      - eth0.100
      - eth1
..
..
lb01b:
..
..
    eth0.100:
      hwaddr: ff:ff:ff:ff:ff:ff
      ip: 192.168.1.4
      netmask: 255.255.255.0
      shared_ip:
        - 192.168.1.3
      shared_v6ip: ~
      type:
        - sip_ext_incoming
..
..
    interfaces:
```

 Copyright © Sipwise GmbH - All rights reserved

```
      - lo
      - eth0
      - eth0.100
      - eth1
..
..
prx01b:
..
..
    eth0.100:
      hwaddr: ff:ff:ff:ff:ff:ff
      ip: 192.168.1.40
      netmask: 255.255.255.0
      shared_ip:
        - 192.168.1.30
      shared_v6ip: ~
      type:
        - rtp_int_100
..
..
    interfaces:
      - lo
      - eth0
      - eth0.100
      - eth1
```

As you can see from the above example, extra SIP interfaces must have type *sip_ext_incoming*. While *sip_ext* should be listed only once per host, there can be multiple *sip_ext_incoming* interfaces. The direction of communication through this SIP interface is incoming only. The Sipwise C5 will answer the incoming client registrations and other methods sent to this address and remember the interfaces used for clients' registrations to be able to send incoming calls to him from the same interface.

In order to use the interface for the outbound SIP communication it is necessary to add it to extra_sockets section in */etc/ngcp-config/config.yml* and select in the outbound_socket peer preference. So if using the above example we want to use the vlan100 IP as source interface towards a peer, the corresponding section may look like the following:

```
    extra_sockets:
      port_5064: udp:10.15.20.108:5064
      test: udp:10.15.20.108:6060
      int_100: udp:192.168.1.3:5060
```

The changes have to be applied:

```
ngcpcfg apply 'added extra SIP and RTP socket'
ngcpcfg push all
```

After applying the changes, a new SIP socket will listen on IP 192.168.1.3 on a CARRIER in the lb01 node and this socket can now be used as source socket to send SIP messages to your peer for example. In

Copyright © Sipwise GmbH - All rights reserved

above example we used label 'int_100'. So the new label "int_100" is now shown in the outbound_socket peer preference.

Also, RTP socket is now listening on 192.168.1.3 on a PRO, and 192.168.1.30 on a CARRIER prx01 node and you can choose the new RTP socket to use by setting parameter rtp_interface to the Label "int_100" in your Domain/Subscriber/Peer preferences.

10.2.4. Alternative RTP Interface Selection Using ICE

Normally, each interface that was configured with a type that starts with 'rtp_' can be selected individually as RTP interface in the Domain/Subscriber/Peer preferences. For example, if the interface types 'rtp_ext', 'rtp_int', and 'rtp_int_100' have been configured, the Domain/Subscriber/Peer preferences will allow the RTP interfaces to be selected as either 'ext', 'int', or 'int_100' in addition to "default".

The same 'rtp_' interface type can be configured on multiple interfaces. If this is the case, and if ICE ('Interactive Connectivity Establishment') is enabled for a Domain/Subscriber/Peer, it is possible to use ICE to automatically negotiate which interface should be used for RTP communications. ICE must be supported by the remote client for this to work.

For example, 'rtp_ext' can be configured on multiple interfaces like so (abbreviated):

```
..
..
    eth0.100:
      type:
        - rtp_ext
..
    eth0.150:
      type:
        - rtp_ext
..
    eth1:
      type:
        - rtp_ext
..
..
```

In this example, the RTP interface 'ext' will be available for selection in the Domain/Subscriber/Peer preferences. If selected and if ICE is enabled, the addresses of all three interfaces will be presented to the remote client, and ICE will be used to negotiate which one of them will be used for communications. This can be useful in multi-homed environments, or when remote clients are on private networks.

10.2.5. Extended RTP Port Range Using Multiple Interfaces

If the RTP port range configured via the config.yml keys rtpengine.minport and rtpengine.maxport is not sufficient to handle all concurrent calls, it is possible to load-balance the RTP ports across multiple interfaces. This is useful if the RTP proxy runs out of ports and if not enough additional ports are available.

 Copyright © Sipwise GmbH - All rights reserved

To enable this, multiple interfaces with different addresses must be configured, and interface types of the format 'rtp_NAME:SUFFIX' must be assigned to them. For example, if the RTP interface named 'ext' should be load-balanced across three interfaces, they can be configured like so (abbreviated):

```
..
..
    eth0.100:
      type:
        - rtp_ext:1
..
    eth0.150:
      type:
        - rtp_ext:2
..
    eth1:
      type:
        - rtp_ext:3
..
..
```

In this example, all three given RTP interface types will be available for selection in the Domain/Subscriber/Peer preferences individually (as 'ext:1' and so on), but in addition to that, an interface named just 'ext' will also be available for selection. If 'ext' is selected, only one of the three RTP interfaces will be selected in a round-robin fashion, thus increasing the number of available RTP ports threefold. The round-robin algorithm only selects an interface if it actually has RTP ports available.

10.2.6. Cluster Sets

In a Sipwise C5 CARRIER system it is possible to have geographically distributed nodes in the same logical Sipwise C5 unit. Such a configuration typically involves the following elements:

- **centralised** management (*web*), database (*db*) and proxy (*prx*) nodes: these provide all higher level functionality, like system administration, subscriber registration, call routing, etc.
- **distributed** load balancer (*lb*) nodes: these serve as SBCs for the whole Sipwise C5 and handle SIP and RTP traffic to / from SIP endpoints (e.g. subscribers); and they also communicate with the central elements of Sipwise C5 (e.g. proxy nodes)

In case of such an Sipwise C5 node configuration it is possible to define *cluster sets* which are collections of Sipwise C5 nodes providing the load balancer functionality.

Cluster sets can be assigned to subscriber *domains* or *SIP peers* and will determine the route of SIP and RTP traffic for those sets of SIP endpoints:

- For *SIP peers* the selected nodes will be used to send outbound SIP traffic through
- For both *SIP peers* and subscriber *domains* the selected nodes will provide RTP relay functionality (the *rtpengine* Sipwise C5 component will run on those nodes)

Configuration of Nodes of Cluster Sets

There are 2 places in NGCP's main configuration files where an entry for cluster sets must be inserted:

Copyright © Sipwise GmbH - All rights reserved

1. Declaration of cluster sets

   This happens in /etc/ngcp-config/config.yml file, see an example below:

```
cluster_sets:
  default:
    dispatcher_id: 50
  default_set: default
  poland:
    dispatcher_id: 51
  type: distributed
```

   Configuration entries are:

   <label>: an arbitrary label of the cluster set; in the above example we have 2 of them: default and poland; the cluster set default is always defined, even if cluster sets are not used

   <label>.dispatcher_id: a unique, numeric value that identifies a particular cluster set

   default_set: selects the default cluster set

   type: the type of cluster set; can be central or distributed

2. Assignment of cluster sets

   This happens in /etc/ngcp-config/network.yml file, see an example below:

```
  .
  .
  lb03a:
    .
    .
    vlan792:
      cluster_sets:
        - poland
      hwaddr: 00:00:00:00:00:00
      ip: 172.30.61.37
      netmask: 255.255.255.240
      shared_ip: 172.30.61.36
      type:
        - sip_int
      vlan_raw_device: bond0
```

   In the network configuration file typically the load balancer (*lb*) nodes are assigned to cluster sets. More precisely: network interfaces of load balancer nodes that have sip_int type—that are used for SIP signalling and NGCP's internal *rtpengine* command protocol—are assigned to cluster sets.

   In order to do such an assignment a cluster set's label has to be added to the cluster_sets parameter, which is a list.

After modifying network configuration with cluster sets, the new configuration must be applied in the

 Copyright © Sipwise GmbH - All rights reserved

usual way:

```
> ngcpcfg apply 'Added cluster sets'
> ngcpcfg push all
```

Configuration of Cluster Sets for SIP and RTP Traffic

For both SIP peers and subscriber domains you can select the cluster set labels predefined in config.yml file.

- **SIP peers**: In order to select a particular cluster set for a SIP peer you have to navigate to *Peerings* *select the peering group* *select the peering server* *Preferences* *NAT and Media Flow Control* and then *Edit* lbrtp_set parameter.

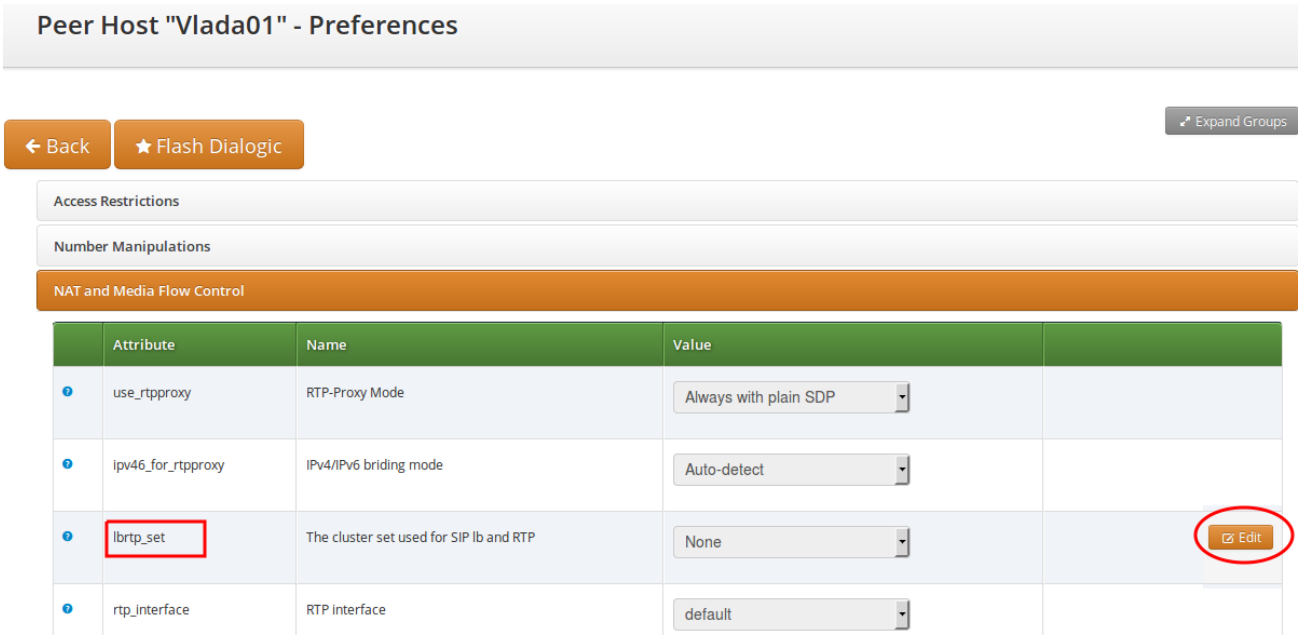


Figure 201. Select Cluster Set for a Peer

- **Domains**: In order to select a particular cluster set for a domain you have to navigate to *Domains* *select the domain* *Preferences* *NAT and Media Flow Control* and then *Edit* lbrtp_set parameter.

Copyright © Sipwise GmbH - All rights reserved

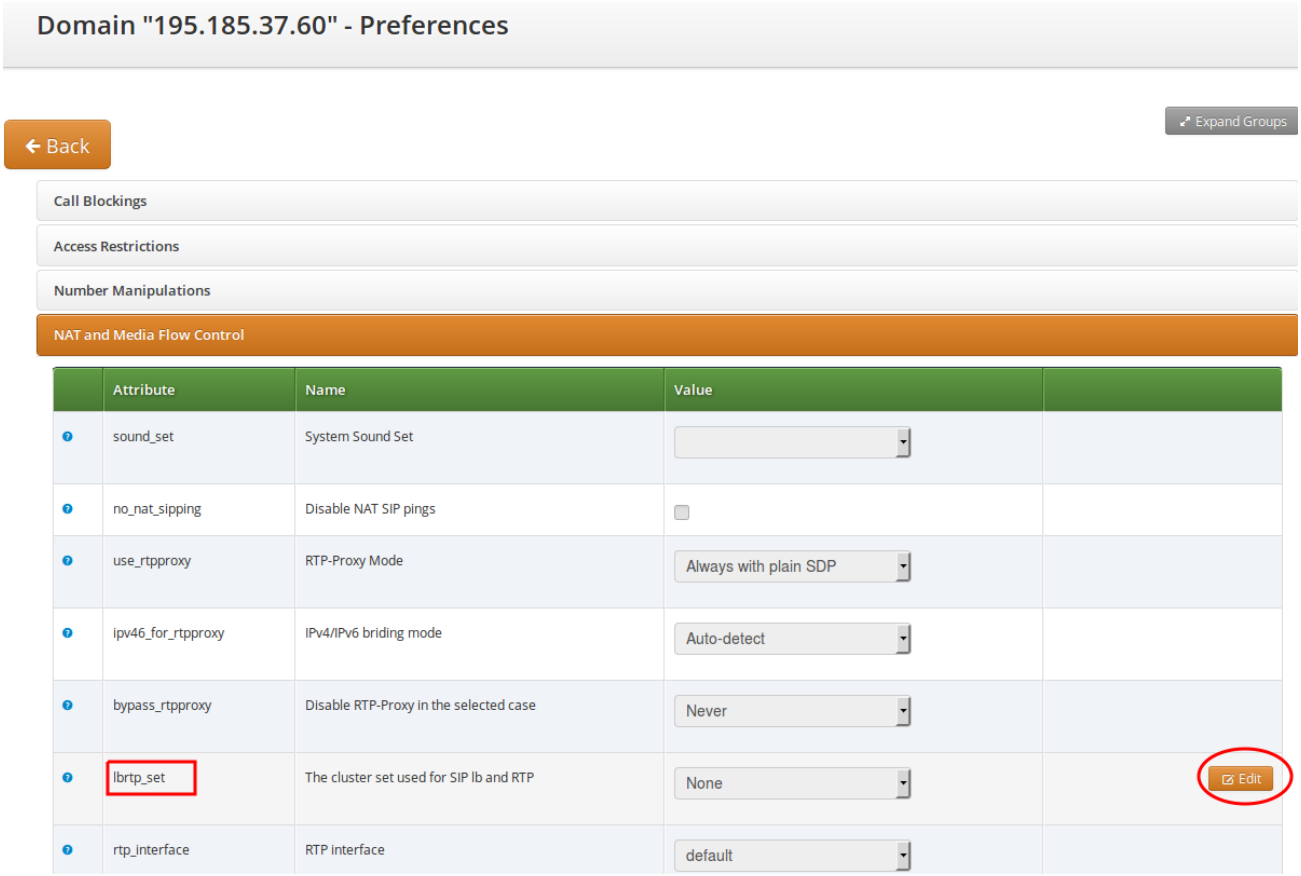


Figure 202. Select Cluster Set for a Domain

Copyright © Sipwise GmbH - All rights reserved

Chapter 11. Instances configuration

11.1. The network.yml structure

Instances are defined in the *network.yml* file. To start, add at the end of the file a new configuration block called 'instances:'.

Let's have a look at the example of instances configuration block below. Here a part of the configuration details is intentionally skipped for simplicity, in order to just create a first, superficial understanding:

```
instances:
  - name: instance_lb_1
    service: kamailio-lb
    host: sp1
    label: lb
    status: online
    interfaces:
      ...
    connections:
      ...
    databases:
      nosql:
        ...
      sql:
        ...
  - name: instance_proxy_1
    ...
  - name: instance_sems_1
    ...
```

Instances are defined as a list of elements, where each element corresponds to an individual instance, regardless of its type. Each element requires the following parameters to be configured:

- name: an arbitrary name to give to the instance. Only letters, digits and '_' chars are allowed.
- service: the type of the service the instance has to run. Choose one from the 'Instances Supported' table.
- host: the name of the host where the instance should run by default (i.e. sp1, lb01a, ...)
- label: a label assigned to the instance. Choose the value listed in the 'Instances Supported' table corresponding to the 'service' you selected.
- status: the status of the instance. Choose one among: 'online' / 'offline' / 'inactive'.
- interfaces: a list of the interfaces assigned to the instance. See Interfaces section for more details.
- connections (optional): a list of the connections to other instances or services. See Connections section for more details.
- databases (optional): a list of the database connections required for the instance. See DB Connections section for more details.

Copyright © Sipwise GmbH - All rights reserved

Once a list of instances is configured, apply these changes and then push to all the other nodes:

```
ngcpcfg apply 'added new instances'
ngcpcfg push all
```

IMPORTANT Please remember that applying this configuration can trigger a short disruption of service. If the instances are being configured on the production platform, plan this as maintenance outside of business hours.

11.2. Instances operation

While the changes are being applied, it will be noticeable that the number of generated configuration files increases, compared to what the system has had before. This is because the ngcpcfg framework generates specialized configuration for each single instance using the same base templates.

The system will also add a new service with a specific instance name and it will set this service in the active state by default, if not explicitly defined as 'inactive'.

For example, if the 'kamailio-lb' instance has been defined with a name 'A' and a host 'sp1', then:

- a new folder with the name '/etc/kamailio/lb/lb_A' is created. This folder will contain the configuration files needed for proper operation of the newly added instance, with specific values for things such as 'listen=', 'alias=' etc.
- a new service 'kamailio-lb-A' is automatically started on host 'sp1' and it will use a list of dedicated configuration files stored under the '/etc/kamailio/lb_A/' folder.

The status of the new instance can be checked using the command:

```
ngcp-service status kamailio-lb-A
```

NOTE A new dedicated folder will be added on all of the nodes/locations of the cluster, in order to provide a possibility for this instance to start anywhere in case of failover.

11.2.1. Template's customization for Instances

Template's customization is available also for instantiated services. To generate a dedicated version of an existing template for a single instance, you have to copy the original .tt2 file to .customtt.tt2.inst-$INSTANCE_NAME, where '$INSTANCE_NAME' refers to the real name of the instance.

Example: We'll create */etc/ngcp-config/templates/etc/lb/kamailio.cfg.customtt.tt2.inst-A* and use it for our customized configuration. In this example, we'll append a comment at the end of the template.

Copyright © Sipwise GmbH - All rights reserved

```
cd /etc/ngcp-config/templates/etc/kamailio/lb
cp kamailio.cfg.tt2 kamailio.cfg.customtt.tt2.inst-A
echo '# This is my last line comment for the instance config' >>
kamailio.cfg.customtt.tt2.inst-A
ngcpcfg apply 'my commit message'
```

The ngcpcfg command will generate */etc/kamailio/lb_A/kamailio.cfg* from our custom template instead of the generic one:

```
tail -1 /etc/kamailio/lb_A/kamailio.cfg
# This is my last line comment for the instance config
```

WARNING similar to traditional customtt file users have to upgrade all .customtt.tt2 manually every time .tt2 is upgraded, as customtt files take precedence over newly unpacked .tt2 files.

IMPORTANT Remember configuration file precedence (highest to lowest): *.customtt.tt2.inst-$INSTANCE_NAME *.customtt.tt2.$NGCP_HOSTNAME *.customtt.tt2.$NGCP_PAIRNAME *.customtt.tt2.$NGCP_NODENAME *.customtt.tt2 *.tt2.inst-$INSTANCE_NAME *.tt2.$NGCP_HOSTNAME *.tt2.$NGCP_PAIRNAME *.tt2.$NGCP_NODENAME *.tt2 Check the ngcpcfg framework documentation or the ngcpcfg script man page for a full description of all the supported template files.

11.3. Interfaces

Each instance always has to be reachable independently, regardless of the node/location it currently takes. Due to that, each instance has dedicated floating IP address(es) that is/are migratable between cluster node pairs.

This floating IP address will always stick to its instance and therefore will provide IP connectivity to it.

IMPORTANT this implies that while migrating to instances, administrators have to reserve a sufficient number of IP addresses in the subnets that the services will be listening on. For example: kamailio-lb service is listening on the 'sip_int' interface to obtain internal SIP traffic and on the 'sip_ext' interface to obtain external traffic. That means a new IP has to be reserved on both the 'sip_int' and 'sip_ext' subnets to be able to start a new kamailio-lb instance. The same approach can be applied to the other types of instances, with the exception of Proxy and Sems-b2b type of instances that will only need private IP addresses.

Here is a list of the interface types to be defined for each instance:

Table 41. Instances Interfaces

Copyright © Sipwise GmbH - All rights reserved

Service name	Interface type	Used for
kamailio-lb	sip_int	internal SIP messages
	sip_ext	external SIP messages
kamailio-proxy	sip_int	internal SIP messages
sems-b2b	sip_int	internal SIP messages
	rtp_int	internal RTP messages
asterisk	sip_int	internal SIP and RTP messages

As mentioned before each instance has a paramater called 'host' to define on which node the instance should has to be run by default.

In a normal state, the system will always try to run the instance on the specified node. In case the specified node is down (for any reason, e.g. maintenance), then the instance will be automatically migrated to the pair node. When the default node is back to normal state, then the instance is migrated back together with the floating IP address belonging to it.

On the node where the instance has to be run (or could be run in case of failover), the interface(s) with the same name and subnetwork must be defined.

This means that, if for example a kamailio-proxy instance with the interface 'neth1' and IP 192.168.1.1 is defined, then on any node that it could possibly run on, an interface with the name 'neth1' and with the including subnet must be defined. For example the sub-network '192.168.1.0/24' would fit this demand.

As shown in the Instances Interfaces table, certain instances could require more than one interface. However, the same interface can be used for two or more types of connections. This mainly depends on the network topology and on the system administrator decision (how to define/interconnect these instance elements).

It is important to define the following parameters for each instance interface:

- name: the name of the host's interface that has to be used
- ip: the IP to assign to this instance's interface
- type: list of types of services assigned to this interface. See Service Types section for more details.

An example on how a definition of the 'instance_lb_1' can look like after adding the interfaces:

Copyright © Sipwise GmbH - All rights reserved

```
- name: instance_lb_1
  service: kamailio-lb
  host: sp1
  label: lb
  status: online
  interfaces:
    - name: neth2
      ip: 192.168.1.211
      type:
        - sip_ext
    - name: neth1
      ip: 192.168.255.211
      type:
        - sip_int
  connections:
    ...
  databases:
    nosql:
      ...
    sql:
      ...
```

11.4. Connections between instances

In a standard Sipwise C5 system, all the services are working in immovable, predefined order. What that means is that there is no possibility to set a specific kamailio-proxy to work with a specific kamailio-lb. The stack will be always: LbProxySems-b2b, regardless of whether this is a PRO or Carrier grade setup.

A decision was made to give more flexibility in this regard:

- to provide mobility and scalability (sharding) to the instance-based services
- to give a possibility to the system administrator to design their own internal topology (in terms of inter-connections between the instances)

This opens up completely new capabilities for the Sipwise C5 system, because it gives a possibility to create a dedicated path for call flow/routing, meaning that it's possible to define which specific LB, Proxy and Sems-b2b instances are engaged into processing of the SIP call.

The list of connections between instances that can be defined:

Table 42. Instances Connections

Service name	Connection	Scope	Multilple links	Fallback
kamailio-lb	proxy	dispacth the call to internal proxies	yes	yes
kamailio-proxy	b2b	dispatch the call to b2b	no	no

Copyright © Sipwise GmbH - All rights reserved

Service name	Connection	Scope	Multilple links	Fallback
	voicemail	dispatch the call to voicemail server	no	no
sems-b2b	lb	select default LB for outbound registration messages	no	no
	proxy	select default proxy for sems generated messages	no	no
asterisk	proxy	dispatch the outgoing fax	no	no

By default, instances will automatically try to connect between each other, looking for other instances running on the same node.

NOTE Even if this method could be useful for the very initial configuration of the system, it could lead to certain obstacles, in particular when more than one instance is active by default on the same node.

11.4.1. Structure of instances connections

This is a list of options/parameters which build up instance connections to other instances/hosts:

- name: a name of the connection to be defined. Available options are: lb, proxy, b2b, voicemail
- algorithm: algorithm to be used in order to dispatch a call in case multiple links are defined and supported, see the 'Instances Connections' table. Available options are: hash, hash_ruri, round_robin, random, serial, weight, parallel.
- links: a list of connections

  type: it defines whether the connection is directed to an instance (type 'instance') or to a standard service (type 'host').

  name: for 'type: instance' this is the name of the instance to connect to, for 'type: host' it is where the service will run on.

  interfaces: a list of interfaces where a remote instance/host can be reached

    name: the name of the instance's/host's interface that has to be used

    type: the interface type

Here an example of how the connections for the new kamailio-lb instance can be configured:

Copyright © Sipwise GmbH - All rights reserved

```
- name: instance_lb_1
  service: kamailio-lb
  host: sp1
  label: lb
  status: online
  interfaces:
    - name: neth2
      ip: 192.168.1.211
      type:
        - sip_ext
    - name: neth1
      ip: 192.168.255.211
      type:
        - sip_int
  connections:
    - name: proxy
      algorithm: random
      links:
        - type: instance
          name: instance_proxy_1
          interfaces:
          - name: neth1
            type: sip_int
        - type: instance
          name: instance_proxy_2
          interfaces:
          - name: neth1
            type: sip_int
        - type: host
          name: prx01
          interfaces:
          - name: neth1
            type: sip_int
  databases:
    nosql:
      ...
    sql:
      ...
```

Where:

- a connection to the 'proxy' has been defined
- since kamailo-lb supports multiple connections and fallback definitions, the 'random' algorithm of selection has been defined
- links allowing to reach the 'proxy' have been defined: two of them towards instances 'instance_proxy_1' and 'instance_proxy_2', and one to the default kamailio-proxy service running on the prx01 host

Copyright © Sipwise GmbH - All rights reserved

NOTE In all of the defined links, two instances and the host are reachable on the 'neth1' interface of type 'sip_int' and all of these links will be used by the 'instance_lb_1' to distribute calls.

11.5. Connections to databases

The concept of instance connections is also applied in the scope of the NoSQL / SQL databases backend usage, and makes it configurable from a dedicated block of instances called 'databases:'. The setup is able to define connections towards NoSQL and SQL using the configuration based on instances (network.yml), therefore pointing to the desired databases a particular instance must be connected with.

NOTE Currently the backend for the SQL database is implemented using MariaDB, and for the NoSQL database using KeyDB (analogous to Redis)

It is important that if instances are enabled templating will try to collect proper values for:

- db.central.${hostname} / nosql.central.${hostname}
- db.replicatedpair.${hostname} / nosql.replicatedpair.${hostname}
- db.replicatedcentral.${hostname}

from the:

- 'instances.${name}.databases.sql'
- 'instances.${name}.databases.nosql'

Which then will be used to build up configuration files for instances as well as the /etc/hosts file, which will contain a list of name translations for the databases. This allows to have host records for connections towards nosql/sql databases, including the local one. Using that approach, the NoSQL database (KeyDB) and the SQL database (Maria DB) can be listening on the internal IP addresses (not on the loopback interfaces) and at the same time not use floating IP addresses, and still be reachable by the instances.

IMPORTANT only one of the databases types can be located standalone: 'db_central'. The 'db_replicated_central' and 'db_replicated_pair' are always local.

If the database connections are not defined, the Active/Active setup with instances will still work. Hence configuration of the databases is not absolutely a must.

A list of the connections towards databases that can be defined:

Table 43. Instances Connections to Databases

Service name	noSQL type	SQL type
kamailio-lb	db_replicated_pair	
kamailio-proxy	db_central	db_central
	db_replicated_pair	db_replicated_pair

 Copyright © Sipwise GmbH - All rights reserved

Service name	noSQL type	SQL type
		db_replicated_central
sems-b2b	db_central	db_central
	db_replicated_pair	db_replicated_pair
		db_replicated_central

NOTE 'db_central' can be located as a separate node serving a role of the central SQL/NoSQL database, while 'db_replicated_pair' / 'db_replicated_central' are working locally on the node.

This is a list of options/parameters which build up instance connections to other instances/hosts:

- nosql:

  name: a name of the connection to be defined, is equal to the node/location value.

  port: port of the NoSQL DB to be connected to

  type: type of the connection: 'db_central', 'db_replicated_pair', 'db_replicated_central'

- sql:

  name: a name of the connection to be defined, is equal to the node/location value.

  port: port of the NoSQL DB to be connected to

  type: type of the connection: 'db_central', 'db_replicated_pair', 'db_replicated_central'

Here is an example, how the databases connections for the new kamailio-lb instance can be configured:

Copyright © Sipwise GmbH - All rights reserved

```
- name: instance_lb_1
  service: kamailio-lb
  host: sp1
  label: lb
  status: online
  interfaces:
    - name: neth2
      ip: 192.168.1.211
      type:
        - sip_ext
    - name: neth1
      ip: 192.168.255.211
      type:
        - sip_int
  connections:
    - name: proxy
      algorithm: random
      links:
        - type: instance
          name: instance_proxy_1
          interfaces:
          - name: neth1
            type: sip_int
        - type: instance
          name: instance_proxy_2
          interfaces:
          - name: neth1
            type: sip_int
        - type: host
          name: prx01
          interfaces:
          - name: neth1
            type: sip_int
  databases:
    nosql:
      - name: sp1
        port: 6379
        type: db_replicated_pair
    sql: []
```

NOTE The kamailio-lb service requires only one NoSQL connection of the 'db_replicated_pair' type.

11.6. Disable default services

When the most important/required steps of the configuration are done and all those migrated standard services are not doing any significant work, they can be safely moved into the offline mode by executing the following commands:

Copyright © Sipwise GmbH - All rights reserved

```
ngcpcfg set /etc/ngcp-config/config.yml "kamailio.lb.status: offline"
ngcpcfg set /etc/ngcp-config/config.yml "kamailio.proxy.status: offline"
ngcpcfg set /etc/ngcp-config/config.yml "b2b.status: offline"
ngcpcfg set /etc/ngcp-config/config.yml "asterisk.status: offline"
ngcpcfg apply "Turn off lb, proxy, b2b, asterisk standard services"
ngcpcfg push all
```

IMPORTANT Please remember that applying this configuration can trigger a short disruption of service. Plan this as maintenance outside of business hours.

Copyright © Sipwise GmbH - All rights reserved

Chapter 12. Software Upgrade

Sipwise C5 can be upgraded to mr10.5.7 from previous LTS release mr9.5, any non-LTS release since the previous LTS-release (mr9.*) or from LTS release mr8.5.

The mr10.5.7 maintenance release uses the new upgrade approach commonly known as 'A/B Upgrade'

ngcp-upgrade requires a special partitioning schema of disk subsystem. The server has 3 separate partitions:

ngcp-data - it stores data files, like databases files, logs, etc.

ngcp-root and ngcp-fallback - they are equal size and contain the software, OS files and NGCP packages. One of them is the current root '/' partition, the another one is mounted as /ngcp-fallback directory.

See more details in The default disk partitions

During the upgrade the second partition is formatted and the target version is installed into it. After the reboot the system will be started from this partition and previous one becomes the /ngcp-fallback directory.

WARNING	Please, pay particular attention that this partitioning schema is mandatory and if your system doesn't have it - create it beforehand. You can reinstall node from the peer and re-join the cluster. WARNING: This is the only supported upgrade schema. The old, in-place upgrade is not supported and technically not possible.

12.1. Release Notes

Please find the complete release notes and changelog of Sipwise C5 version mr10.5.7 on our WEB site.

12.2. Overview

The Sipwise C5 software upgrade procedure to mr10.5.7 will perform several fundamental tasks, and it's split into 2 stages:

First stage:

- do pre-upgrade checks
- format fallback partition
- install Debian to fallback partition
- install NGCP packages to fallback partition
- copy current configuration to fallback partition
- upgrade the NGCP configuration schema in fallback partition
- update grub configuration so after reboot the current fallback partition becomes the active one

Second stage:

Copyright © Sipwise GmbH - All rights reserved

- reboot to new partition. This steps needs to be taken care manually during maintenance window
- upgrade the NGCP database schema

The 1st stage is safe to run outside the maintenance window - it changes nothing on currently running system, only prepares the fallback partition.

Also it can be done in parallel on all the nodes.

WARNING It still can affect the system in term of CPU, network and disk load, to download, unpack and install the packages.

WARNING Grub configuration is changed so in case of any reboot (expected or not) system will be booted to mr10.5.7 partition. It won't affect the installation though as no NGCP services are run there. So if it was unintentionally you can reboot it back to previous partition via Rollback procedure.

The 2nd stage should be run during maintenance window with enabled maintenance mode in configuration.

WARNING Please make sure to have remote access to the system via out-of-band management (like IPMI, iLO, IMM, iDRAC, KVM, etc)

Sipwise C5 CARRIER is a PRO-style system that has "A" and "B" sets of nodes with specific roles. The number of nodes can differ between installations and must be clarified before the upgrade at the planning stage.

The software upgrade is usually performed by Sipwise engineers according to the following steps:

- create the software upgrade plan
- execute pre-upgrade steps: patchtt, customtt, backups, checks
- perform the first stage of upgrade on all the nodes
- make all "B" nodes (on CARRIER) or the sp2 node (on PRO) active
- ensure that all "A" nodes (on CARRIER) or the sp1 node (on PRO) are standby
- enable Maintenance Mode
- perform the second stage of upgrade on all "A" nodes (on CARRIER) or the sp1 node (on PRO)
- schedule and make services switchover to all "A" nodes (on CARRIER) or the sp1 node (on PRO)
- ensure that "A" nodes (on CARRIER) or the sp1 node (on PRO) perform well (otherwise, perform a switch back)
- perform the second stage of upgrade on all "B" nodes (on CARRIER) or the sp2 node (on PRO)
- perform the system post-upgrade testing and cleanup

WARNING The only allowed software upgrade path is the one described above. The nodes sp1/s2 (or a/b) MUST be used as described in this document. All the other theoretically possible upgrade scenarios can lead to unpredictable results.

Copyright © Sipwise GmbH - All rights reserved

NOTE If you upgrade from mr8.5.* release install the latest hotfix for the ngcp-upgrade-propackage.

NOTE Check the current HA stack used. If it's heartbeat switch it to corosync/pacemaker using the Corosync/Pacemaker guide of the current release. Heartbeat v2 is not supported in mr10.5.7.

12.3. Planning a software upgrade

Confirm the following information:

- which system should be upgraded (LAB/LIVE, country, etc.)
- the date and time schedule for each of the steps above (keeping the time zone in mind)
- a confirmed timeframe for the upgrade operation (allowed switchover timeframe)
- the basic functionality test (BFT) to be executed before the start of the software upgrade and after the switchovers to ensure that the new release does not show critical issues (the BFT scenario should be prepared by the customer engineers)
- actions to be taken if the software upgrade operation cannot be completed within the defined maintenance window
- contact persons and ways of communication in case of emergency
- ensure that the customer and/or Sipwise engineers have access to the virtual consoles of the servers: KVM, iDRAC, AMM

12.4. Pre-upgrade checks

It is recommended to execute the preparatory steps in this chapter a few days before the actual software upgrade. They do not cause a service downtime, so it is safe to execute them during peak hours.

12.4.1. Log into the NGCP standby management node

This should be web01a on CARRIER and sp1 on PRO.

TIP Use the static server IP address so you can switch between the nodes.

Run the terminal multiplexer under the *sipwise* user (to reuse the Sipwise .screenrc settings that are convenient for working in multiple windows):

```
screen -S my_screen_name_for_ngcp_upgrade
```

Become root inside your screen session:

```
sudo -s
```


Copyright © Sipwise GmbH - All rights reserved

12.4.2. Check the overall system status

Check the overall system status:

```
ngcp-status --all
```

Make sure that the cluster health status is OK: Check the nodes in parallel, using the clish command:

- **ngcp-clish "ngcp version summary"** - ensure that all cluster nodes have correct/expected from version
- **ngcp-clish "ngcp version package installed ngcp-ngcp-pro"** ensure that the metapackages version is equal to the ngcp version above
- **ngcp-clish "ngcp version package check"** - ensure that all nodes have the identical Debian package installed

NOTE Software must be identical on all nodes (before and after the upgrade!)

- **ngcp-clish "ngcp cluster ssh connectivity"** - check SSH connectivity from the current node to all other nodes
- **ngcp-clish "ngcp cluster ssh crossconnectivity"** - check SSH cross-connectivity from all nodes to all other nodes
- **ngcp-clish "ngcp service summary"** - all required services must be running on corresponding nodes
- **ngcp-clish "ngcp cluster status"** - active node(s) (with all services running) must print "active", the other(s) must print "standby"
- **ngcp-clish "ngcp status collective-check"** - all checks must be OK
- **ngcp-clish "ngcp show date"** - date and time must be in sync on all the servers
- **ngcp-clish "ngcp show dns-servers"** - ensure that the DNS configuration is consistent among the nodes
- **ngcp-clish "ngcp cluster replication"** - check all MySQL replications statuses on all nodes.

NOTE to exit from 'ngcp-clish' press Ctrl+Z (or type *exit*):

```
# ngcp-clish
Entering 'clish-enable' view (press Ctrl+Z to exit)...
# exit
#
```

12.4.3. Check access to license server and license validity

Check from within the system (better *from all nodes*, for extra safety) that the license server is accessible from the network point of view, and that these commands do not end with timeouts or HTTP errors:

Copyright © Sipwise GmbH - All rights reserved

```
ping -c 3 -w 5 license.sipwise.com

curl --head https://license.sipwise.com/
```

Also ensure that:

- `/proc/ngcp/check` contains the string `"ok"` (if not, check logs)
- and that there are no errors or important warnings in `/ngcp-data/logs/licensed.log`

12.4.4. Evaluate and update custom modifications

For the below steps, investigate and make sure you understand why the custom modifications were introduced and if they are still required after the software upgrade. If the custom modifications are not required anymore, remove them (e.g. if a bug was fixed in the target release and the existing patch becomes irrelevant).

Create tickets to Sipwise developers to make relevant custom modifications part of the product in future releases. This allows you to get rid of the customtt files one day.

WARNING If you directly change the working configuration (e.g. add custom templates or change the existing ones) for some reason, then the system must be thoroughly tested after these changes have been applied. Continue with the software upgrade preparation only if the results of the tests are acceptable.

Find the local changes to the template files:

```
ngcp-customtt-diff-helper
```

The script will also ask you if you would like to download the templates for your target release. To download the new templates separately, execute:

```
ngcp-customtt-diff-helper -d
```

In the tmp folder provided by the script, you can review the patchtt files or merge the current customtt with the new tt2 templates, creating the new customtt.tt2 files. Once you do this, archive the new patchtt/customtt files to reapply your custom modifications after the software upgrade:

```
ngcp-customtt-diff-helper -t
```

Find all available script options with the "-h" parameter.

12.4.5. Check system integrity

Log into all the servers.

Open separate windows for all the servers inside your "screen" session. (Press `Ctrl+a + c` to open a

 Copyright © Sipwise GmbH - All rights reserved

new window, `Ctrl+a + a` or `Ctrl+a + [0-9]` to change the window. `Ctrl+a + "` shows the list of all your windows. Use `Ctrl+a + A` to change the window names to corresponding hosts).

Changes made directly in tt2 templates will be lost after the software upgrade. Only custom changes made in customtt.tt2 or added by patchtt.tt2 files will be kept. Hence, check the system for locally modified tt2 files on **all** nodes:

```
ngcp-status --integrity
```

12.4.6. Check the configuration framework status

Check the configuration framework status on **all** nodes. All checks must show the "OK" result and there must be no actions required:

```
ngcpcfg status
```

On a CARRIER, check the replication on both central DB servers and on ports 3306 and 3308 of all the proxy servers. Ensure that all the proxy nodes replicate the read-only DB (127.0.0.1:3308) from the db01a node. Otherwise, discuss a special plan to address your particular configuration.

On a PRO, check the replication on both nodes.

The result must always show:

```
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Seconds_Behind_Master: 0
```

Test the cluster failover to see if everything works fine as well on "B" nodes (on a CARRIER) or the second node (on a PRO). On all the standby nodes execute:

```
ngcp-make-active
```

Create two test subscribers or use the credentials for existing ones. Register subscribers with the platform and perform a test call to ensure that call routing and media flow are working fine.

Run "apt-get update" on **all** nodes and ensure that you do not have any warnings and errors in the output.

WARNING If the installation uses locally specified mirrors, then the mirrors must be switched to the Sipwise APT repositories (at least for the software upgrade). Otherwise, the public Debian mirrors may not provide packages for old Releases anymore or at least provide outdated ones!

Copyright © Sipwise GmbH - All rights reserved

12.4.7. Check access to deb.sipwise.com

Ensure that both management nodes have access to deb.sipwise.com by executing the following commands on a management node:

```
ngcp-approx-cache --check --node sp1

ngcp-approx-cache --check --node sp2
```

The checks must show only *200 OK* results. If you see *cannot connect!*, *Received 404* or any other error, check these possible causes:

- A node does not have a connection to deb.sipwise.com
- The deb.sipwise.com whitelist does not have the node's public IP (IPv6) address.

12.4.8. License check

The Sipwise C5 enforce *software licensing* restrictions in form of a regular comparison of the licensed services and capacities against the actual usage patterns of the platform. In case some functionalities are enabled but not licensed, an error in *syslog* will be reported and the impacted services will be automatically deactivated.

Before proceeding with the upgrade, please take some time to check that all the modules not licensed are actually disabled in *config.yml* file. To verify if they are enabled execute the following commands (versions before mr10.5):

```
ngcpcfg get sems.prepaid.enable
ngcpcfg get sems.prepaid.inew.enable
ngcpcfg get sems.sbc.xfer.enable
ngcpcfg get pbx.enable
ngcpcfg get pushd.enable
ngcpcfg get intercept.enable
ngcpcfg get voisniff.admin_panel
ngcpcfg get voisniff.daemon.li_x1x2x3.enable
ngcpcfg get voisniff.daemon.start
ngcpcfg get tpcc.enable
ngcpcfg get websocket.enable
```

To verify if they are enabled execute the following commands (all version including mr10.5 and later):

Copyright © Sipwise GmbH - All rights reserved

```
ngcpcfg get b2b.prepaid.enable
ngcpcfg get b2b.prepaid.inew.enable
ngcpcfg get b2b.sbc.xfer.enable
ngcpcfg get pbx.enable
ngcpcfg get pushd.enable
ngcpcfg get intercept.enable
ngcpcfg get voisniff.admin_panel
ngcpcfg get voisniff.daemon.li_x1x2x3.enable
ngcpcfg get voisniff.daemon.start
ngcpcfg get tpcc.enable
ngcpcfg get websocket.enable
```

If the output of one of the commands is 'yes' but the module is not licensed, you have to deactivate it. For example, in case of *pre-paid billing* module execute (versions before mr10.5):

```
ngcpcfg set /etc/ngcp-config/config.yml sems.prepaid.enable=no
ngcpcfg apply 'Disable prepaid module'
ngcpcfg push all
```

For example, in case of *pre-paid billing* module execute (all version including mr10.5 and later):

```
ngcpcfg set /etc/ngcp-config/config.yml b2b.prepaid.enable=no
ngcpcfg apply 'Disable prepaid module'
ngcpcfg push all
```

WARNING Please, pay particular attention to *pre-paid billing* module because it is enabled by default.

12.5. Pre-upgrade steps

12.5.1. Download new package metadata into the approx cache (on the standby node only)

WARNING Customers with far-sighted software upgrade policies usually have pre-production installations to test the services in their environment before upgrading the production platform. In this case, the approx cache should be updated on both platforms simultaneously to synchronize the package versions between them, hence consider carefully before executing this step.

To download the latest packages metadata into the approx cache, execute the following command on the **standby management** node. This action ensures that all nodes within the platform have identical packages after the software upgrade:

Copyright © Sipwise GmbH - All rights reserved

```
ngcp-approx-cache --update --skip-all-repos --extra-ngcp-release
mr10.5.7 --extra-debian-release bullseye
```

Run the following command node to install the package responsible for upgrading Sipwise C5 to a newer release:

```
ngcp-prepare-upgrade mr10.5.7
```

NOTE Don't worry, ngcp-upgrade-carrier does not exist, ngcp-upgrade-pro is used in Carrier too.

12.5.2. Run upgrade checks

There is a list of checks before the actual upgrade so it's wise to run them beforehand to detect and fix all the issues.

```
ngcp-upgrade-pre-checks mr10.5.7
```

NOTE If there is an error during the upgrade, the ngcp-upgrade script will request you to solve it. Once you've fixed the problem, execute `ngcp-upgrade` again and it will continue from the previous step.

The upgrade script will ask you to confirm that you want to start. Read the given information **carefully**, and if you agree, proceed with *y*.

The upgrade process will take several minutes, depending on your network connection and server performance. After everything has been updated successfully, it will finally ask you to reboot your system. Confirm to let the system reboot (it will boot with an updated kernel).

12.5.3. `ngcp-upgrade` options

The following options in `ngcp-upgrade` can be specially useful in some instances of upgrade:

- **`--step-by-step`**: confirm before proceeding to next step. With this option the upgrade operation is performed confirming every step before execution, with the possibility to instruct to continue without confirming further steps until the end (if confirmation is only needed for some steps at the beginning).
- **`--pause-before-step STEP_NAME`**: pause execution before step, given by the name of the script (e.g. "backup_mysql_db"). This option can be useful in several scenarios, for example:

  to help to debug problems or work around known problems during upgrades. In this case the operator can pause at a given step known to be problematic or right before a problematic set, perform some manual checks or changes, then continue the upgrade until another step (with confirmation like with the recent option `--step-by-step`), or continue without stop until the end

  another use might be to help to speed up upgrades when it involves several nodes: they can all

Copyright © Sipwise GmbH - All rights reserved

proceed in parallel when it's known to be safe to do so; then perform some parts in lock-step (some nodes waiting until others finish with some stage); then continue in parallel until the end

- **`--skip-db-backup`**: This will speed-up the process in cases where it's deemed unnecessary, and this is very likely in the upgrade of nodes other than the first.

12.6. Upgrading Sipwise C5 CARRIER

Log in to all nodes and execute the checks from Pre-upgrade checks again. This will ensure that nothing was broken since the preparation steps were finished. Also, execute **ngcpcfg show** and **ngcpcfg status** to check the latest configuration changes.

Perform the BFT test.

12.6.1. The custom modification handling (optional)

During the execution of ngcp-upgrade on the 1st node in step **place_customtt_files**, you will be asked to place customtt/patchtt files for the new system to */ngcp-fallback/etc/ngcp-config*.

When one node is fully upgraded, you won't be asked about customtt/patchtt files in the upgrade of the following nodes.

If, after upgrading all "A" nodes and promoting them to active ones, you found that you need to change something in customtt/patchtt files - do it, execute `ngcpcfg apply`, and push commit to shared storage. Do not push it to all the nodes but upgraded ones only (all "A" nodes).

12.6.2. First stage of upgrade

You can run 1st stage outside the maintenance window. Also you can run this stage in parallel on all the nodes ("A" and "B" ones).

To do this run the upgrade script on the nodes as *root*:

```
ngcp-upgrade --target mr10.5.7
```

There is "stop" step in upgrade scenario, ngc-upgrade stops there. At this time the new system in fallback partition is ready.

12.6.3. Enable Maintenance Mode

The maintenance mode of Sipwise C5 will disable some background services (for instance: *ngcp-mediator*) during the software upgrade. It thus prevents the system from getting into an inconsistent state while the upgrade is being performed. You can activate maintenance mode by applying a simple configuration change as described later.

- Pull pending configuration (if any):

```
ngcpcfg pull
```

Copyright © Sipwise GmbH - All rights reserved

- Enable maintenance mode:

```
ngcpcfg set /etc/ngcp-config/config.yml "general.maintenance=yes"
```

- Apply configuration changes by executing:

```
ngcpcfg apply 'Enabling maintenance mode before the upgrade to mr10.5.7'
ngcpcfg push all
```

12.6.4. Second stage of upgrade

Upgrading the standby management node "A" (web01a/db01a)*

NOTE Sometimes the DB and MGMT roles are assigned to the same host. This is OK.

WARNING Do NOT run the second stage on management and db node in parallel!

Before reboot switch boot record, run as *root*:

```
/usr/share/ngcp-upgrade/ngcp-switch-root-partition
```

Reboot the management node and run as *root*:

```
ngcp-upgrade --target mr10.5.7
```

Upgrading the standby database node "A" (db*a)

NOTE If the DB and MGMT roles are assigned to the same host, then skip this step as you have already upgraded the standby MGMT node "A" above.

Before reboot switch boot record, run as *root*:

```
/usr/share/ngcp-upgrade/ngcp-switch-root-partition
```

Reboot the database node and run as *root*:

```
ngcp-upgrade --target mr10.5.7
```

NOTE It is important to upgrade db01a node *before* upgrading any proxy nodes. Otherwise, the "local" MySQL (127.0.0.1:3308) on proxy nodes may become out of sync in case the new release has `_not_replicated.up` DB statements.

Copyright © Sipwise GmbH - All rights reserved

Upgrading other standby nodes "A" (lb*a/prx*a)

This part of the upgrade can be done in parallel on other "A" nodes.

Before reboot switch boot record, run as *root*:

```
/usr/share/ngcp-upgrade/ngcp-switch-root-partition
```

Reboot the node and run as *root*:

```
ngcp-upgrade --target mr10.5.7
```

Useful options in `ngcp-upgrade`

The following options in `ngcp-upgrade` can be useful for this phase of upgrades, because it is very likely that the backup was already performed:

- **`--skip-db-backup`**: This will speed-up the process in cases where it's deemed unnecessary.

See a more detailed description of the options in: ngcp-upgrade options

Promote ALL standby nodes "A" to active.

WARNING Ensure that all standby nodes "A" are: * upgraded to the new release (check /etc/ngcp_version or use `ngcp-clish`)

Prior to the promotion, the DB has to be updated with information from the new configuration. This should be done as close as possible to the activation of the upgraded nodes, to minimize the changes to the active service with the old release.

Execute on one of the standby nodes as *root*, for example on db01a:

```
UPGRADE_VERSION=mr10.5.7 MYSQL_VALUES_UPDATE_BEFORE_SWITCHOVER=true
/etc/ngcp-config/templates/etc/ngcp-provisioning-
tools/mysql_values.cfg.services
```

On all "A" nodes run:

```
ngcp-make-active
```

Ensure that the "A" nodes became active, by executing the *'ngcp-status'* and *'ngcp-clish'* commands described above.

Ensure that ALL "B" nodes are standby now!

Copyright © Sipwise GmbH - All rights reserved

Upgrading ALL standby nodes "B" (web*b/db*b/lb*b/prx*b)

NOTE You can upgrade all standby "B" nodes simultaneously (including the ones with the mgmt and db roles).

Before reboot switch boot record, run as *root*:

```
/usr/share/ngcp-upgrade/ngcp-switch-root-partition
```

Reboot the node and run as *root*:

```
ngcp-upgrade --target mr10.5.7
```

Useful options in `ngcp-upgrade`

The following options in `ngcp-upgrade` can be useful for this phase of upgrades:

- **`--step-by-step`**: confirm before proceeding to next step.
- **`--pause-before-step STEP_NAME`**: pause execution before step, given by the name of the script (e.g. "backup_mysql_db").

See a more detailed description of the options in: ngcp-upgrade options

12.7. Upgrading Sipwise C5 PRO

Make sure you are prepared to spend about two hours upgrading the system. Note that a short service downtime is possible during the services switchover to the upgraded node.

Start with the software upgrade on the standby sp1 node. Then, switch the services over to the upgraded node and upgrade the other (now standby) sp2 node, as described in the steps below.

12.7.1. The custom modification handling (optional)

During the execution of ngcp-upgrade on the 1st node in step **place_customtt_files**, you will be asked to place customtt/patchtt files for the new system to */ngcp-fallback/etc/ngcp-config*.

When one node is fully upgraded, you won't be asked about customtt/patchtt files in the upgrade of the 2nd node.

If, after upgrading the 1st node and promoting it to active one, you found that you need to change something in customtt/patchtt files - do it, execute `ngcpcfg apply`, and push commit to shared storage. But do not push changes to 2nd node as it still running old system.

12.7.2. First stage of upgrade

You can run 1st stage outside the maintenance window. Also you can run this stage in parallel on all the nodes (sp1 and sp2).

To do this run the upgrade script on the nodes as *root*:

Copyright © Sipwise GmbH - All rights reserved

```
ngcp-upgrade --target mr10.5.7
```

There is "stop" step in upgrade scenario, ngcp-upgrade stops there. At this time the new system in fallback partition is ready.

Enable Maintenance Mode

The maintenance mode of Sipwise C5 will disable some background services (for instance: *ngcp-mediator*) during the software upgrade. It thus prevents the system from getting into an inconsistent state while the upgrade is being performed. You can activate maintenance mode by applying a simple configuration change as described later.

- Pull pending configuration (if any):

```
ngcpcfg pull
```

- Enable maintenance mode:

```
ngcpcfg set /etc/ngcp-config/config.yml "general.maintenance=yes"
```

- Apply configuration changes by executing:

```
ngcpcfg apply 'Enabling maintenance mode before the upgrade to mr10.5.7'
ngcpcfg push all
```

12.7.3. Second stage of upgrade

Before reboot switch boot record, run as *root*:

```
/usr/share/ngcp-upgrade/ngcp-switch-root-partition
```

Reboot the standby node and run as *root*:

```
ngcp-upgrade --target mr10.5.7
```

Useful options in `ngcp-upgrade`

The following options in `ngcp-upgrade` can be useful for this phase of upgrades:

- **`--step-by-step`**: confirm before proceeding to next step.
- **`--pause-before-step STEP_NAME`**: pause execution before step, given by the name of the script (e.g. "backup_mysql_db").

Copyright © Sipwise GmbH - All rights reserved

See a more detailed description of the options in: ngcp-upgrade options

12.7.4. Promote the upgraded standby node to active

Prior to the promotion, the DB has to be updated with information from the new configuration. This should be done as close as possible to the activation of the upgraded node, to minimize the changes to the active service with the old release.

Execute on the current standby node as *root*:

```
UPGRADE_VERSION=mr10.5.7 MYSQL_VALUES_UPDATE_BEFORE_SWITCHOVER=true
/etc/ngcp-config/templates/etc/ngcp-provisioning-
tools/mysql_values.cfg.services
ngcp-make-active
```

12.7.5. Upgrade the second PRO node

Before reboot switch boot record, run as *root*:

```
/usr/share/ngcp-upgrade/ngcp-switch-root-partition
```

Reboot the new standby node and run as *root*:

```
ngcp-upgrade --target mr10.5.7
```

Useful options in `ngcp-upgrade`

The following options in `ngcp-upgrade` can be useful for this phase of upgrades, because it is very likely that the backup was already performed in the upgrade of the first node:

- **`--skip-db-backup`**: This will speed-up the process in cases where it's deemed unnecessary.

See a more detailed description of the options in: ngcp-upgrade options

12.8. Post-upgrade steps

12.8.1. Migrate from RedisDB to KeyDB

Starting from mr10.3.1 we introduced a new noSQL database called KeyDB. KeyDB is powerful drop-in alternative to Redis adding many interesting features like multithreading and master-master replica. All new deployments already have KeyDB activated by default, but existing deployments are probably still set to use RedisDB and therefore have to be migrated. The procedure can be done in a later time but it is mandatory because all tests and fixes done for this and future versions will consider KeyDB only. To check if your system is already running with KeyDB or not, execute the following command:

```
ngcpcfg get "database.key_value.flavor"
```


Copyright © Sipwise GmbH - All rights reserved

If the command returns 'keydb' as output, the following steps can be skipped and it is possible to jump to Disabling maintenance mode. Otherwise if the command returns nothing or 'redis' then the migration must still be performed. The necessary steps to perform the migration are:

Perform the switch on the standby management node (web01 on Carrier)

```
ngcp-redis-flavor keydb
```

Perform the switch on all other standby nodes

ATTENTION: remember to use the '-a' option as shown below

```
ngcp-redis-flavor -a keydb
```

Make all the current standby nodes active:

```
ngcp-make-active
```

Perform the switch on all new standby nodes

ATTENTION: remember to use the '-a' option as shown below

```
ngcp-redis-flavor -a keydb
```

(optional) Bring back to active all the current standby nodes:

```
ngcp-make-active
```

12.8.2. Disabling maintenance mode

In order to disable the *maintenance mode*, do the following:

- Pull outstanding ngcpcfg changes (if any):

```
ngcpcfg pull
```

- Disable the maintenance mode:

```
ngcpcfg set /etc/ngcp-config/config.yml "general.maintenance=no"
```

- Apply the changes to configuration templates:

Copyright © Sipwise GmbH - All rights reserved

```
ngcpcfg apply 'Disable the maintenance mode after the upgrade to
mr10.5.7'
ngcpcfg push all
```

12.8.3. Post-upgrade checks

When everything has finished successfully, check that replication is running. Check `ngcp-status --all`. Finally, do a basic functionality test. Check the web interface, register two test subscribers and perform a test call between them to ensure call routing works.

NOTE You can find a backup of some important configuration files of your existing installation under */ngcp-data/backup/ngcp-mr10.5.7-** (where * is a place holder for a timestamp) in case you need to roll back something at any time. A log file of the upgrade procedure is available at */ngcp-data/ngcp-upgrade/$FROM-mr10.5.7/logs/*.

12.9. Applying the Latest Hotfixes

If your current release is already the latest or you prefer to be on the LTS release, we still suggest applying the latest hotfixes and critical bug fixes.

Execute all steps as described in Pre-upgrade checks. They include the system checks, customtt/patchtt preparation and others. It is important to execute all the steps from the above chapter.

On a CARRIER it is suggested to promote B-nodes to active and start the update with A-nodes.

12.9.1. Update the approx cache on the standby management node

The main goal of the following command is to download the new packages into the approx cache. So all the nodes in the cluster will get identical packages.

```
ngcp-approx-cache --auto --node localhost
```

12.9.2. Apply hotfixes on the standby management node

```
ngcp-update
```

12.9.3. If there are custom configuration templates

Merge/add the custom configuration templates if needed.

Apply the changes to configuration templates:

```
ngcpcfg apply 'apply customtt/patchtt after installing the latest
packages'
```


Copyright © Sipwise GmbH - All rights reserved

Send the new templates to the shared storage and the other nodes.

```
ngcpcfg push --shared-only
```

12.9.4. Apply hotfixes on all other standby nodes (CARRIER-only)

```
ngcp-update
```

12.9.5. Promote the standby nodes to active

Execute on the **standby** nodes as *root*:

```
ngcp-make-active
```

Check in a minute that the nodes became active:

```
ngcp-check-active
```

12.9.6. Apply hotfixes on the second node

```
ngcp-update
```

Execute the final checks as described in the **Post-upgrade checks** section.

Copyright © Sipwise GmbH - All rights reserved

Chapter 13. Backup, Recovery and Maintenance

13.1. Sipwise C5 Backup

For any service provider it is important to maintain a reliable backup policy as it enables prompt services restoration after any force majeure event. Although the design of Sipwise C5 implies data duplication and high availability of services, we still strongly suggest you to configure a backup procedure. The Sipwise C5 has a built-in solution that can help you back up the most crucial data. Alternatively, it can be integrated with any Debian compatible backup software.

13.1.1. What data to back up

- The database

This is the most important data in the system. All subscriber and billing information, CDRs, user preferences, etc. are stored in the MySQL server. It is strongly recommended to have up-to-date dumps of all the databases on corresponding Sipwise C5 nodes.

- System configuration

The system configuration folder */etc/ngcp-config/* must be included in the backup as well. It contains the system specific configuration (like SSL keys). Also you might have some local modifications. We suggest backing up the whole */etc* folder to preserve the *etckeeper* history to be able to answer when and who changed particular configuration files in the past.

- Exported CDRs (optional)

The */home/jail/home/cdrexport* directory contains the exported CDRs. It depends on your call data retention policy whether or not to remove these files after exporting them to an external system.

13.1.2. The built-in backup solution

The Sipwise C5 comes with an easy-to-use solution that creates everyday backups of the most important data:

- The system configuration files. The whole /etc directory is backed up.
- Exported CDRs. The */home/jail/home/cdrexport* directory with csv files.
- All required databases on corresponding servers.

This functionality is disabled by default and can be enabled and configured in the backuptools subsection in the config.yml file. Please, refer to the “C.1.3 backup tools” section of the “Sipwise C5 configs overview” chapter for the backup configuration options.

Once you set the required configuration options, apply the changes:

```
ngcpcfg apply 'enable the backup feature'
ngcpcfg push all
```

Once you activate the feature, Sipwise C5 will create backups in the off-peak time on the standby

 Copyright © Sipwise GmbH - All rights reserved

nodes and put them to the */ngcp-data/backup/ngcp_backup* directory, namespaced by a timestamp and the node pairname. By default it will also copy them to its peer node. It can also be configured to copy them to the 'mgmt' nodes, so that they keep a consolidated backup of the entire Carrier. You can copy these files to your backup server using scp or ftp.

NOTE make sure that you have enough free disk space to store the backups for the specified number of days.

13.2. Recovery

In the worst case scenario, when the system needs to be recovered from a total loss, you only need 4 steps to get the services back online:

- Install Sipwise C5 as explained in chapter 2.
- Restore the */etc/ngcp-config/* directory from the backup, overwriting your local files.
- Restore 'mysql.encryption.key' in constants.yml if new MariaDB instance/binlogs were encrypted (DB is encrypted by default). See the detailed information in MariaDB data restoration remarks.
- Restore the database from the latest MySQL dump.
- Apply the changes to bring the original configuration into effect:

```
ngcpcfg apply 'restored the system from the backup'
ngcpcfg push all
```

13.3. Reset Database

IMPORTANT All existing data will be wiped out! Use this script only if you want to clear all previously configured services and start configuration from scratch.

To reset database to its original state you can use a script provided by CE: * Execute *ngcp-reset-db*. It will assign new unique passwords for Sipwise C5 services and reset all services. The script will also create dumps for all Sipwise C5 databases.

13.4. Synchronize database

In case of unresolvable database replication issues or to copy mysql data between a pair of hosts (usually a pair of sp1 and sp2 nodes).

There is a script for that: *ngcp-sync-db*.

To synchronize databases you need to run the script on your target host.

- Definitions:

  'master' - remote/master host (the database is dumped from there)

  'local' - target/local host (the database is imported onto)

- Usage:

Copyright © Sipwise GmbH - All rights reserved

IMPORTANT	Your existing database on 'local' will be completely wiped. The script provides a possibility to backup both 'master' and 'local' databases during the procedure.

You can run the script with -h or --help to check its options or use man ngcp-sync_db

If you run it without any options it automatically calculates 'master' hostname (e.g. if you run it on 'sp2' then 'sp2'=='local' and 'sp1'=='master').

The script also requires mysql credentials and if none is provided it uses the ones from the file */etc/mysql/sipwise_extra.cnf*. You can specify user and/or password for both 'master' and 'local'.

Before the actual start it produces a summary with settings used to the procedure and a confirmation prompt to prevent accidental usage. Making use of --force option" however suppresses the confirmation prompt. By default no messages are printed on STDOUT (compliant to be integrated into another tools) and with -v or --verbose options you enable debugging where all the ongoing steps will be printed to STDOUT.

There are 2 modes available for synchronization, 'online' and 'backup'. By default 'online' is used where the procedure does not create any backups and everything goes on the fly. That is useful for large databases where creating backups would require solid amounts of available free disk space. With the 'backup' mode 'master' db is dumped into a backup file on 'local' first (default directory: '/ngcp-data/backup/ngcp-sync-db') and imported upon the backup completion.

Mysql database connection to the 'master' db and the 'local' db is the essential part and by default the script tries to establish direct mysql connection however that may not be possible due to the access restrictions. To overcome that you can use --ssh-tunnel option and specifying there a local custom free port (e.g. --ssh-tunnel=33125) in this case an ssh tunnel will be created to 'master' and used to establish the db connection on the 'localhost' behalf (NOTE: Public key based ssh negotiation is required for the tunnel as the script does not suppot ssh credentials for security reasons).

Backups may be a subject to create during synchronization for possible rollbacks. To create the 'local' db backup you should add --local-backup. The 'master' db backup is automatically created only using --sync-mode=backup. Upon completion all those created backups are deleted and if you need to keep them please use --keep-backups option (NOTE: In case of errors during synchronization and when backups are created they are NOT automatically deleted. Therefore, if the script had failed with an error and afterwards completed successfully you may want to manually remove the remaining backups from */ngcp-data/backup/ngcp-sync-db*).

- Examples:

Normal online mode synchronization 'sp1' 'sp2'.

```
sp2> ngcp-sync-db
```

Normal backup mode synchronization 'sp1' 'sp2'.

```
sp2> ngcp-sync-db --sync-mode=backup
```

Forced online mode synchronization 'sp1' 'sp2'. USE WITH CARE as there will be no confirmation prompts.

 Copyright © Sipwise GmbH - All rights reserved

```
sp2> ngcp-sync-db --force
```

Direct mysql db access is not possible. SSH tunnel is initialised to local port 33125 and forwards all connections 127.0.0.1:33125 sp1:3306.

```
sp2> ngcp-sync-db --ssh-tunnel=33125
```

Custom mysql credentials for the 'master' db connection (by default: */etc/mysql/sipwise_extra.cnf*)

```
sp2> ngcp-sync-db --master-user=frank --master-pass=dbconnect
```

Normal online mode synchronization 'sp1' 'sp2' with the 'local' db backup and retaining the backup. (no 'master' backup in this case as it is only available with --sync-mode=backup).

```
sp2> ngcp-sync-db --local-backup --keep-backups
```

Normal online mode synchronization 'custom-node' 'sp2' with ssh tunnel

```
sp2> ngcp-sync-db --master-host=custom-node --ssh-tunnel=45001
```

Forced syncrhonization 'custom-node' 'sp2' with ssh tunnel, backup sync mode, local backup, custom 'master' and 'local' db credentials and ports as well as a different backup dir

```
sp2> ngcp-sync-db --force --sync-mode=backup --master-host=custom-node
--master-port=3308 --ssh-tunnel=45001 --master-user=frank --master
-pass=dbconnect --local-user=john --local-pass=dblocal --local-backup
--keep-backups --backup-dir=/home/barry/backups
```

13.5. Accounting Data (CDR) Cleanup

Sipwise Sipwise C5 offers ways to cleanup, backup or archive old accounting data—i.e. CDRs—that is not necessary for further processing any more, or must be deleted according to the law. There are some Sipwise C5 components designed for this purpose and they are commonly called *cleanuptools*. These are configurable scripts that interact with NGCP's accounting and kamailio databases, or remove exported CDR files in order to clean or archive the unnecessary data.

13.5.1. Cleanuptools Configuration

The configuration parameters of *cleanuptools* are located in the main Sipwise C5 configuration file: /etc/ngcp-config/config.yml. Please refer to the config.yml file description: Cleanuptools Configuration Data for configuration parameter details.

In case the system administrator needs to modify some configuration value, the new configuration

Copyright © Sipwise GmbH - All rights reserved

must be activated in the usual way, by running the following commands:

```
> ngcpcfg apply 'Modified cleanuptools config'
> ngcpcfg push all
```

As a result new configuration files will be generated for the accounting database and the exported CDR cleanup tools. Please read detailed description of those tools in subsequent sections of the handbook.

The Sipwise C5 system administrator can also select the time when cleanup scripts are run, by modifying the schedule here: /etc/cron.d/cleanup-tools

13.5.2. Accounting Database Cleanup

The script responsible for cleaning up the database is: ngcp-cleanup-acc

The configuration file used by the script is: /etc/ngcp-cleanup-tools/acc-cleanup.conf

An extract from a sample configuration file is provided here:

```
############

batch = 10000
archive-target = /ngcp-data/backup/cdr
compress = gzip

username = dbcleaner
password = rcKamRdHhx7saYRbkJfP
host = localhost
port = 3306

redis-batch = 10000
redis-port = 6379

connect accounting
keep-months = 2
use-partitioning = yes
timestamp-column = cdr_start_time
backup cdr_cash_balance_data
backup cdr_time_balance_data
backup cdr_relation_data
backup cdr_tag_data
backup cdr_mos_data
backup cdr_export_status_data
backup cdr_group
timestamp-column = first_cdr_start_time
backup cdr_period_costs
timestamp-column = start_time
backup cdr

archive-months = 2
```


Copyright © Sipwise GmbH - All rights reserved

```
archive cdr_cash_balance_data
archive cdr_time_balance_data
archive cdr_relation_data
archive cdr_tag_data
archive cdr_mos_data
archive cdr_export_status
archive cdr_group
archive cdr_period_costs
archive cdr

cleanup-days = 1
use-partitioning = no
timestamp-column = cdr_start_time
cleanup int_cdr_cash_balance_data
cleanup int_cdr_time_balance_data
cleanup int_cdr_relation_data
cleanup int_cdr_tag_data
cleanup int_cdr_group
cleanup int_cdr_export_status
timestamp-column = start_time
cleanup int_cdr

connect kamailio
time-column = time
cleanup-days = 90
cleanup acc

connect-redis 21
connect kamailio
time-column = time_hires
cleanup-days = 3
cleanup-mode = mysql
cleanup-redis acc:entry::*

# Clean up after mediator by deleting old leftover acc entries and
deleting
# old entries out of acc_trash and acc_backup
connect kamailio
time-column = time
cleanup-days = 30
cleanup acc_trash
cleanup acc_backup

maintenance = no
```

The configuration file itself contains a detailed description of how database cleanup script works. It consists of a series of statements, one per line, which are going to be executed in sequence. A statement can either only set a variable to some value, or perform an action.

There are 4 types of actions the database cleanup script can take:

- backup database tables

Copyright © Sipwise GmbH - All rights reserved

- archive database tables
- cleanup database tables
- cleanup redis databases

These actions are discussed in following sections.

A generic action is connecting to the proper database: connect <database name>

Backup Database Tables

The database cleanup tool can create *monthly backups* of data in the accounting database tables by moving old records to separate tables named: cdr_YYYYMM. The instruction in the configuration file looks like: backup <table name>, by default and typically it is: backup cdr

Configuration values that govern the backup procedure are:

- time-column: The name of the column in the table to use for determining which month a record belongs to. Must be a "datetime" column.
- timestamp-column: The name of the column in the table to use for determining which month a record belongs to. Must be a "decimal(13,3)" column.
- use-partitioning: If a table is partitioned using the time-column (timestamp-column) and the value is set to "yes", then moving/deleting records are instant operations, done by managing the partitions. Otherwise the usual method is used to delete/move records by chunks.
- batch: How many rows to include per transaction when processing in chunks. If unset or 0, it does them all at once.
- keep-months: How many months worth of records to keep in the table and not move into the monthly backup tables.

  IMPORTANT: Months are always processed as a whole and this specifies how many months to keep AT MOST. In other words, if the script is started on December 15th and this value is set to "2", then all of December and November is kept, and all of October will be moved out.

Archive Database Tables

The database cleanup tool can archive (dump) old monthly tables. The statement used for this purpose is: archive <table name>, by default and typically it is: archive cdr

This creates an SQL dump out of older tables created by the backup statement and drop them from database afterwards. Archiving uses the following configuration values:

- archive-months: Uses the same logic as the "keep-months" variable. If set to "12" and the script was started on December 15th, it will start archiving with the December table of the previous year. Archiving continues month by month, going backwards in time, until the script encounters a missing table.
- archive-target: Target directory for writing the SQL dump files. If explicitly specified as "/dev/null", then no actual archiving will be performed, but instead the tables will only be dropped.
- compress: If set to "gzip", then gzip the dump files after creation. If unset, do not compress.
- host, "username" and "password": As dumping is performed by an external command, those variables are reused from the "connect" statement.

 Copyright © Sipwise GmbH - All rights reserved

Cleanup Database Tables

The database cleanup tool may do database table cleanup without performing backup. In order to do that, the statement: cleanup <table name> is used. Typically this has to be done in kamailio database, examples:

- cleanup acc
- cleanup acc_trash
- cleanup acc_backup

The cleanup statement works exactly like the backup statement, but doesn't actually backup anything, but rather only deletes old records. Additional configuration parameters required by the cleanup procedure:

- cleanup-days: Any record older than these many days will be deleted.

Cleanup Redis Databses

With the advent of persisting kamailio.acc record data in a redis keystore, a separate cleanup-redis <key pattern> statement was introduced. It will remove old redis entries, whose keys match the given redis SCAN pattern. Typically this has to be done for the redis 21 database (acc records), e.g.:

- cleanup-redis acc:entry::*

connect-redis <redis database number> has to be used instead of connect +<mariadb database name>, which is needed to initially connect before any <backup>, <archive> and <cleanup> operations.

The cleanup-redis statement works similar to the cleanup statement, with some additional options below:

- time-column: The name of the field in a redis entry denoting the record timestamp in epoch seconds.
- cleanup-mode: If set to "delete", aged entires will be simply removed. If set to "mysql", they will be inserted into a database table first. The latter requires connect to open the database additionally.
- redis-batch: Chunk size of redis entries with matching keys to look at. Note that redis processing always works in chunks, there is no partitioning.
- cleanup-days: Any entry older than these many days will be deleted.

13.5.3. Exported CDR Cleanup

The script responsible for cleaning up exported CDR files is: ngcp-cleanup-cdr-files

The configuration file used by exported CDR cleanup script is: /etc/ngcp-cleanup-tools/cdr-files-cleanup.yml

A sample configuration file is provided here:

Copyright © Sipwise GmbH - All rights reserved

```
enable: no
max_age_days: 30
paths:
  -
    path: /home/jail/home/*/20[0-9][0-9][0-9][0-9]/[0-9][0-9]
    wildcard: yes
    remove_empty_directories: yes
    max_age_days: ~
  -
    path: /home/jail/home/cdrexport/resellers/*/20[0-9][0-9][0-9][0-
9]/[0-9][0-9]
    wildcard: yes
    remove_empty_directories: yes
    max_age_days: ~
  -
    path: /home/jail/home/cdrexport/system/20[0-9][0-9][0-9][0-9]/[0-
9][0-9]
    wildcard: yes
    remove_empty_directories: yes
    max_age_days: ~
```

The exported CDR cleanup tool deletes CDR files in the directories provided in the configuration file, if those have already expired.

Configuration values that define the files to be deleted:

- enable: Enable (yes) or disable (no) exported CDR cleanup.
- max_age_days: Gives the expiration time of the exported CDR files in days. There is a general value which may be overridden by a local value provided at a specific path. The local value is valid for the particular path only.
- paths: an array of path definitions

  path: a path where CDR files are to be found and deleted; this may contain wildcard characters

  wildcard: Enable (yes) or disable (no) using wildcards in the path

  remove_empty_directories: Enable (yes) or disable (no) removing empty directories if those are found in the given path

  max_age_days: the local expiration time value for files in the particular path

13.6. Managing packages

The Sipwise C5 uses Debian packages to deliver the code to servers. Therefore it is important to keep Debian packages installation state and version consistent across nodes in the cluster. To achieve it, Sipwise C5 uses the Approx component.

Approx is a proxy server for Debian archive files. It fetches files from remote repositories on demand, and caches them for local use. All files are always being delivered from the approx cache at the same version as they were delivered previously to all the other cluster nodes.

All approx cache files are stored in '/var/cache/approx/' on the first management type (MGMT) Carrier

 Copyright © Sipwise GmbH - All rights reserved

nodes (normally 'web01' on Carrier) and are shared between 'web01a' and 'web01b' pair for high availability.

The following tools are available on the platform to maintain the Approx cache (call them with '--help' options to see all the possible functionality):

- ngcp-approx-cache
- ngcp-approx-snapshots

To provide all the necessary functionality, Approx distinguishes between two main types of files:

- Repository metadata (Indexes files)
- Repository packages (packages *.deb and source *.tar.gz files)

The first type of files 'Repository metadata' is always 'frozen' and always returned from the Approx cache. Freezing them is enough to provide the same Debian packages for all the cluster nodes. To update approx cache ('Repository metadata') users should use 'ngcp-approx-cache', to sync approx cache ('Repository metadata') between installations (LAB and PROD) users can use 'ngcp-approx-snapshots'.

The second type of files 'Repository packages' is a local cache/mirror of remote Debian servers. Every time the server requests some package, it is being checked in the Approx cache storage and returned if available (for performance reasons). If the package files are missing, they will be requested from the remote server, returned to the client and stored locally for future usage. Such an approach speeds up the installation stage (all packages are available from the LAN) and make possible to reinstall old packages state in case a cluster node needs reinstallation (e.g. disaster recovery) as all packages are available locally even if they have disappeared from the Debian servers.

To provide such a separation, Sipwise C5 has two TCP ports to use:

- Approx read-write (RW) port (by default '9999'). Managed by Approx itself.
- Approx read-only (RO) port (by default '9998'). Managed by Nginx.

All requests towards Approx RW port will overwrite the Approx cache (access to the Approx RW port is limited to the 'ha_int' interface only). Normally ngcp-approx-cache only uses the RW port. APT source files '/etc/apt/sources.list.d/*.list' only uses the RO port. (the Approx host is 'web01' for Carrier and 'sp' for PRO installations):

Copyright © Sipwise GmbH - All rights reserved

```
root@web01a:~# grep -H 9998 /etc/apt/sources.list.d/*
/etc/apt/sources.list.d/debian.list:deb http://web01:9998/debian/
bullseye main contrib non-free
/etc/apt/sources.list.d/debian.list:deb http://web01:9998/debian-
security/ bullseye-security main contrib non-free
/etc/apt/sources.list.d/debian.list:deb http://web01:9998/debian/
bullseye-updates main contrib non-free
/etc/apt/sources.list.d/debian.list:deb http://web01:9998/debian-debug/
bullseye-debug main contrib non-free
/etc/apt/sources.list.d/sipwise.list:deb [arch=amd64]
http://web01:9998/autobuild/ release-trunk-bullseye main
/etc/apt/sources.list.d/sipwise.list:#deb-src
http://web01:9998/autobuild/ release-trunk-bullseye main
root@web01a:~#
```

Approx does NOT support a secure HTTP protocol (HTTPS), all connections from NGCP servers towards Approx should use the plain HTTP transport.

All connections from the Approx to external servers use HTTPS by default and can be fine-tuned in case of a HTTP/HTTPS proxy in use. The proxy servers should be configured in config.yml:

```
bootenv:
  http_proxy: ''
  https_proxy: ''
```

The custom Approx repositories can be defined using the following section in config.yml:

```
bootenv:
  custom_repos:
  - enable: no
    name: my-example-repo
    url: https://example.com/debian
  - enable: yes
    name: my-example-repo2
    url: https://example.com/myrepo
```

13.6.1. Maintaining the Approx cache

NOTE It is recommended to create an Approx snapshot before updating the Approx cache.

The script 'ngcp-approx-cache' is designed to update the Approx cache and clean/manage it.

To update the Approx cache please use the following command:

```
ngcp-approx-cache --auto # you can add --force to skip all confirmations
```


Copyright © Sipwise GmbH - All rights reserved

The command above will update all the 'Repository metadata' to the latest available versions from the remote servers (pointed by APT source list files in '/etc/apt/sources.list.d/*.list').

The tool ngcp-approx-cache should be called on the management type (MGMT) Carrier node. It is enough to execute it only once (web01a/sp1 or web01b/sp2).

To update all packages on the server from the Approx cache it is enough to call the ususal Debian command:

```
apt update && apt upgrade
```

WARNING	Do not forget to update DB/ngcp-config if NGCP packages have been updated, and apply configuration changes: ngcp-update-db-schema && ngcp-update-cfg-schema && ngcpcfg apply 'new packages'.
WARNING	To prevent unnecessary downtime always upgrade Debian packages on inactive HA node.

Also, the tool 'ngcp-approx-cache' allows to check the Approx cache consistency, clean stale packages and/or NGCP releases from the Approx cache, and more. See all the available functionality using '--help' option:

```
ngcp-approx-cache --help
```

13.6.2. Maintaining the Approx snapshots

An approx snapshot is the concept of having the approx cache state of a particular point in time. The approx snapshots only contain the 'Repository metadata' parts. The tool 'ngcp-approx-snapshots' is designed to create/manage/export/import approx snapshots between different installations (e.g. LAB and PROD). It allows syncing Debian packages versions between systems and guarantees installation state consistency between production and the code tested by QA in the LAB, etc.

Usage example:

On the LAB system, update approx cache, install and test new packages:

Copyright © Sipwise GmbH - All rights reserved

```
root@web01a:~# ngcp-approx-cache --auto --force
...
root@web01a:~# ngcp-approx-snapshots --create
Creating approx snapshot '20210319002625'...
root@web01a:~#
root@web01a:~# apt update && apt upgrade && ... && ngcp-config apply ...
...
root@web01a:~# # HACK/FIX/TEST
...
root@web01a:~# ngcp-approx-snapshots --export 20210319002625
Exporting approx snapshot '20210319002625'...
Successfully exported approx snapshot: /tmp/tmp.71CS2yjrli/ngcp-approx-
snapshot-20210319002625.gzip
root@web01a:~#
```

Copy snapshot from LAB to PROD (~40MB for mr9.4/buster):

```
root@web01a:~# scp /tmp/tmp.71CS2yjrli/ngcp-approx-snapshot-
20210319002625.gzip PROD:/tmp/
```

Check the versions of packages on PROD, import new approx snapshot, switch to new approx snapshot:

Copyright © Sipwise GmbH - All rights reserved

```
root@sp1:/var/cache# apt-cache policy ngcp-templates-pro
ngcp-templates-pro:
  Installed: 9.4.1.1+0~mr9.4.1.1
  Candidate: 9.4.1.1+0~mr9.4.1.1
  Version table:
 *** 9.4.1.1+0~mr9.4.1.1 990
        990 http://sp:9998/sppro/mr9.4.1 buster/main amd64 Packages
        100 /var/lib/dpkg/status
root@sp1:/var/cache#

root@sp1:/var/cache# ngcp-approx-snapshots --import /tmp/ngcp-approx-
snapshot-20210319002625.gzip
Importing approx snapshot '/tmp/ngcp-approx-snapshot-
20210319002625.gzip'...
Successfully imported approx snapshot.
root@sp1:/var/cache#

root@sp1:/var/cache# ngcp-approx-snapshots --list
List of locally available snapshots:
20210319002625 | Created at 2021-03-19 00:26:25 on web01a
root@sp1:/var/cache#

root@sp1:/var/cache# ngcp-approx-snapshots --switch 20210319002625
Switching to approx snapshot...
WARNING: the active approx cache will be removed! (Snapshots created?)
Should we switch to approx snapshot '20210319002625'? (yes/no): yes
Removing the active approx cache '/var/cache/approx'...
Switching to the approx snapshot '20210319002625'
root@sp1:/var/cache#

root@sp1:/var/cache# ngcp-approx-snapshots --apt-update # to force local
repository metadata update
...
root@sp1:/var/cache# apt-cache policy ngcp-templates-pro
ngcp-templates-pro:
  Installed: 9.4.1.1+0~mr9.4.1.1
  Candidate: 9.4.1.13+0~mr9.4.1.13
  Version table:
     9.4.1.13+0~mr9.4.1.13 990
        990 http://sp:9998/sppro/mr9.4.1 buster/main amd64 Packages
 *** 9.4.1.1+0~mr9.4.1.1 100
        100 /var/lib/dpkg/status
root@sp1:/var/cache#

root@web01a:~# apt update && apt upgrade && ... && ngcp-config apply
```

It is also useful to see the list of available approx snapshots (the tool ngcp-approx-snapshots has to be executed on the MGMT node):

Copyright © Sipwise GmbH - All rights reserved

```
root@web01a:~# ngcp-approx-snapshots --list
List of locally available approx snapshots:
 * 20210319000519 | Created at 2021-03-19 00:05:19 on sp1
   20210325081717 | Created at 2021-03-25 08:17:17 on web01b
   20210325101739 | Created at 2021-03-25 10:17:39 on web01a [LOCKED]
root@web01a:~#
```

The symbol "*" above shows the currently active snapshot (it implies the content in /var/cache/approx/snapshot-info/created and /mnt/glusterfs/mgmt-share/approx_snapshots/20210319000519/snapshot-info/created is identical).

Example for locking/unlocking snapshots (to prevent deleting by mistake):

```
root@web01a:~# ngcp-approx-snapshots --list
List of locally available approx snapshots:
 * 20210319000519 | Created at 2021-03-19 00:05:19 on sp1
   20210325081717 | Created at 2021-03-25 08:17:17 on web01b
   20210325101739 | Created at 2021-03-25 10:17:39 on web01a [LOCKED]

root@web01a:~# ngcp-approx-snapshots --delete 20210325101739
ERROR: cannot remove locked snapshots '20210325101739'

root@web01a:~# ngcp-approx-snapshots --unlock 20210325101739
Unlocked snapshot '20210325101739'

root@web01a:~# ngcp-approx-snapshots --delete 20210325101739
Deleting approx snapshot '20210325101739'

root@web01a:~# ngcp-approx-snapshots --list
List of locally available approx snapshots:
 * 20210319000519 | Created at 2021-03-19 00:05:19 on sp1
   20210325081717 | Created at 2021-03-25 08:17:17 on web01b
root@web01a:~#
```

Also, it is possible to search and show all packages versions in all snapshots:

```
root@sp1:~# ngcp-approx-snapshots --search ngcp-templates-pro
Search results for the package 'ngcp-templates-pro':
   20210326204026 : 9.4.1.1+0~mr9.4.1.1 (mr9.4.1)
   20210326224730 : 9.4.1.2+0~mr9.4.1.2 (mr9.4.1)
    active approx : 9.4.1.3+0~mr9.4.1.3 (mr9.4.1)
    apt installed : 9.4.1.2+0~mr9.4.1.2
    apt candidate : 9.4.1.3+0~mr9.4.1.3
root@sp1:~#
```

See more details in 'ngcp-approx-snapshots --help'.

Copyright © Sipwise GmbH - All rights reserved

Chapter 14. Security, Performance and Troubleshooting

Once Sipwise C5 is in production, security and maintenance becomes really important. In this chapter, we'll go through a set of best practices for any production system.

14.1. Sipwise SSH access to Sipwise C5

The Sipwise C5 provides SSH access to the system for Sipwise operational team for debugging and final tuning. Operational team uses user 'sipwise' which can be logged in through SSH key only (password access is disabled) from dedicated access server 'jump.sipwise.com' only.

To completely remove Sipwise access to your system, please execute as user root:

```
root@myserver:~# ngcp-support-access --disable && apt-get install ngcp-
support-noaccess
```

NOTE you have to execute the command above on each node of your Sipwise C5 system!

WARNING please ensure that the script complete successfully:

```
  * Support access successfully disabled.
```

If you need to restore Sipwise access to the system, please execute as user root:

```
root@myserver:~# apt-get install ngcp-support-access && ngcp-support-
access --enable
```

WARNING please ensure that the script complete successfully:

```
  * Support access successfully enabled.
```

14.2. Firewalling

14.2.1. Firewall framework

The Sipwise C5 runs a wide range of services. In order to secure the platform while allowing access to Sipwise C5 , Sipwise C5 configuration framework provides a set of predefined network zones. Services are aggregated into appropriate zones by default. Zones are assigned to network interfaces (and VLANs if applicable) in /etc/ngcp-config/network.yml.

Copyright © Sipwise GmbH - All rights reserved

CAUTION Though the default firewall setup provided by Sipwise C5 configuration framework provides a safe setup for Sipwise C5, security audits of the platform performed by qualified engineers before commissioning the platform into service are strongly recommended. Customization of the setup requires in-depth knowledge of firewalling principles in general and the 'netfilter' facility in particular.

Table 44. Sipwise C5 network zones

Zone name	Description
ha_int	Internal HA (High Availability) communication interface between the services
mon_ext	Interface to connect external monitoring appliances (SNMP)
rtp_ext	Interface for external RTP media relay between Sipwise C5 and endpoints (e.g. user agents, peers)
sip_ext	Interface for external SIP signalling between Sipwise C5 and endpoints (e.g. user agents, peers)
sip_int	Interface for internal signalling, e.g. between load-balancers, proxies and applications servers
ssh_ext	Interface providing external access to Sipwise C5 command line interface
ssh_int	Interface providing internal access to Sipwise C5 command line interface (necessary for ngcp-installer)
web_ext	Interface providing access to the customers' self-care Web panel
web_int	Interface for access to the administrative Web panel, its REST APIs and internal API communications

NOTE Additional custom zones may be configured, but will not be automatically integrated into the firewall configuration.

To facilitate firewall functionality, Sipwise C5 uses the Kernel's 'netfilter' facility and 'iptables-persistent' as an interface to 'netfilter'. 'Netfilter' is using 'tables' and within that 'chains' to store rules in this hierarchy: 'table' 'chain' 'rule'. Default firewall setups of Sipwise C5 do not use netfilter tables 'nat' and 'raw', but only default table 'filter'.

NOTE Custom 'nat' rules for IPv4 and IPv6 may be added in file /etc/ngcp-config/config.yml in sections 'securityfirewallnat_rules4' and 'securityfirewallnat_rules6'.

Each 'chain' deploys a 'default policy' handling packets which did not trigger any rule in a particular 'chain'.

Table 45. Sipwise C5 'netfilter' default policies

Copyright © Sipwise GmbH - All rights reserved

Chain	Default policy	Description
INPUT	DROP	Handling all packets directly destined for a Sipwise C5 node (only packets matching a rule are allowed)
FORWARD	DROP	Handling all packets received by a Sipwise C5 node and destined for another, non-local IP destination (no default rules added)
OUTPUT	ACCEPT	Handling all packets originating on a Sipwise C5 node (no default rules added)
rtpengine	N/A	Container for rptengine rule to allow the rule to persist even when the Kernel module is unloaded (e.g. during upgrades)

The default firewall setup provided by Sipwise C5:

- adds rules to INPUT to secure access to platform and services
- blocks all traffic from and to FORWARD
- allows all OUTPUT traffic

14.2.2. Sipwise C5 firewall configuration

The Sipwise C5 comes with a preconfigured set of firewall rules, which can be enabled and configured in /etc/ngcp-config/config.yml in section security->firewall. Refer to security for available configuration options.

Firewall configuration is applied by running ngcpcfg apply. However, this will not activate new rules automatically to avoid inadvertent self-lockout. To finally activate new firewall rules run iptables-apply. This will prompt for another system logon to verify access remains available. If the prompt is not confirmed, firewall rules will automatically be reverted to the previous state re-enabling access to the command line.

IMPORTANT iptables-apply needs to be enforced on each active and standby Sipwise C5 node providing the 'mgmt', 'lb', or 'rtp' roles (Please, refer to Available Host Options for further details). Additionally, any changes made into firewall rules will require to execute this command again on the corresponding nodes.

CAUTION The Sipwise C5 firewall subsystem by default is disabled in '/etc/ngcp-config/config.yml' key security.firewall.enable: no. This is to avoid blocking any traffic inadvertently during installation. After the firewall subsystem has been configured appropriately, it needs to be enabled by setting security.firewall.enable: yes in '/etc/ngcp-config/config.yml'.

14.2.3. IPv4 System rules

The following set of rules is added by the system upon activation of the firewall subsystem. Individual system rules are configured in '/etc/ngcp-config/templates/etc/iptables/rules.v4.tt2' and '/etc/ngcp-config/templates/etc/iptables/rules.v6.tt2'

Table 46. Firewall system rules

Copyright © Sipwise GmbH - All rights reserved

Zone	Chain	Target	Rule	Description
all	INPUT	rtpengine	-p udp -j rtpengine	Redirects all incoming UDP packets to chain 'rtpengine' (putting RTPENGINE rule into a dedicated chain allows for the rule to persist even when the Kernel module gets unloaded, e.g. during upgrades)
all	rtpengine	RTPENGINE	-p udp -j RTPENGINE --id 0	Feeds all RTP packets to RTPENGINE Kernel module
n/a	INPUT	ACCEPT	-i lo -j ACCEPT	Accept all packets received by local loopback interface
all	INPUT	ACCEPT	-m state --state RELATED,ESTABLISHED -j ACCEPT	Accept all incoming packets tied to 'related' or 'established' connections
all	INPUT (IPv4)	ACCEPT	-p icmp -m icmp --icmp-type 8 -j ACCEPT	Accept all ICMP 'echo' messages
all	INPUT (IPv4)	ACCEPT	-p icmp -m icmp --icmp-type 0 -j ACCEPT	Accept all ICMP 'echo reply' messages
all	INPUT (IPv6)	ACCEPT	-A INPUT -p ipv6-icmp -j ACCEPT	Accept all ICMPv6 messages
all	INPUT	cluster	-j cluster	Divert all incoming packets to the 'cluster' chain
all	cluster	ACCEPT	-s '<node_ip>' -j ACCEPT	Set of rules white-listing all IP-addresses owned by Sipwise C5 platform for incoming traffic
api_int	INPUT	ACCEPT	-p tcp --dport '<ossbss.port>' -j ACCEPT	Set of rules for all 'api_int' interfaces accepting all incoming packets for API port defined in '/etc/ngcp-config/config.yml' with key 'ossbss.port'
mon_ext	INPUT	ACCEPT	+-p udp -s '<snmpclient_ip>' --dport 161 -j ACCEPT	Set of rules for all 'mon_ext' interfaces based on a list of IPs for all SNMP communities configured in 'snmpd.communities'
rtp_ext	INPUT	ACCEPT/'name'	-p udp --dport '<rtpengine.minport>':'<rtpengine.maxport>' -j ACCEPT/'name'	Set of rules for all 'rtp_ext' interfaces accepting all incoming packets for RTP port range defined in '/etc/ngcp-config/config.yml' with keys 'rtpengine.minport' and 'rtpengine.maxport' (see note below for custom options)

Copyright © Sipwise GmbH - All rights reserved

Zone	Chain	Target	Rule	Description
sip_ext	INPUT	ACCEPT	-p udp --dport '<kamailio.lb.port>' -j ACCEPT	Set of rules for all 'sip_ext' interfaces accepting all packets on the loda balancer's SIP signalling port defined in '/etc/ngcp-config/config.yml' with key 'kamailio.lb.port' (UDP)
sip_ext	INPUT	ACCEPT	-p tcp --dport '<kamailio.lb.port>' -j ACCEPT	Set of rules for all 'sip_ext' interfaces accepting all packets on the loda balancer's SIP signalling port defined in '/etc/ngcp-config/config.yml' with key 'kamailio.lb.port' (TCP)
sip_ext	INPUT	ACCEPT	-p tcp --dport '<kamailio.lb.tls.port>' -j ACCEPT	Set of rules for all 'sip_ext' interfaces accepting all packets on the loda balancer's SIP signalling port defined in '/etc/ngcp-config/config.yml' with key 'kamailio.lb.tls.port' (TCP/TLS)
sip_ext	INPUT	ACCEPT	-p tcp --dport 5222 -j ACCEPT	Set of rules for all 'sip_ext' interfaces accepting all packets on TCP port 5222 (XMPP client)
sip_ext	INPUT	ACCEPT	-p tcp --dport 5269 -j ACCEPT	Set of rules for all 'sip_ext' interfaces accepting all packets on TCP port 5269 (XMPP server)
sip_ext	INPUT	ACCEPT	-p tcp --dport '<pushd.port>' -j ACCEPT	Set of rules for all 'sip_ext' interfaces accepting all packets incoming for the 'pushd' server port configured in '/etc/ngcp-config/config.yml' with key 'pushd.port'
ssh_ext	INPUT	ACCEPT	-A INPUT -i '<ssh_ext_interface>' -p tcp -s '<sshd.permit_support_from>' --dport 'sshd.port' -j ACCEPT	List of rules to accept incoming packets for SSH on all 'ssh_ext' interfaces from hosts configured in '/etc/ngcp-config/config.yml' with key 'sshd.permit_support_from'

Copyright © Sipwise GmbH - All rights reserved

Zone	Chain	Target	Rule	Description
web_ext	INPUT	ACCEPT	-p tcp --dport '<www_admin.http_csc.port>' -j ACCEPT	List of rules to accept incoming packets for the 'Customer Self Care' interface defined in '/etc/ngcp-config/config.yml' with key 'www_admin.http_csc.port' on all 'web_ext' interfaces
web_int	INPUT	ACCEPT	-p tcp --dport '<www_admin.http_admin.port>' -j ACCEPT	List of rules to accept incoming packets for the 'Admin Panel' interface defined in '/etc/ngcp-config/config.yml' with key 'www_admin.http_admin.port' on all 'web_int' interfaces

CAUTION

To function correctly, the *rtpengine* requires an additional *iptables* rule installed. This rule (with a target of RTPENGINE) is automatically installed and removed when the rtpengine starts and stops, so normally you don't need to worry about it. However, any 3rd party firewall solution can potentially flush out all existing iptables rules before installing its own, which would leave the system without the required RTPENGINE rule and this would lead to decreased performance. It is imperative that any 3rd party firewall solution either leaves this rule untouched, or installs it back into place after flushing all rules out. The complete parameters to install this rule (which needs to go into the INPUT chain of the filter table) are: -p udp -j RTPENGINE --id 0

NOTE

Some of the parameters used to populate the firewall rules automatically may contain hostnames instead of IP addresses. Since firewall rules need to be configured based on IP addresses by design, Sipwise C5 configuration framework will lookup such hostnames during 'ngcpcfg apply' and expand them to the IP addresses as returned by 'gethostbyname'. If DNS resolving changes for such hostnames due to changes to DNS the rules will not update automatically. Another run of 'ngcpcfg apply' will be needed to re-perform the lookup and update the rules to reflect changes in DNS. If this step is omitted, clients may be locked out of the system.

NOTE

By default, the rules for the 'rtp_ext' zone are created with a target of ACCEPT. It is optionally possible to create these rules with another *iptables* chain as target, and instruct the RTP proxy to dynamically manage individual rules for each running call in this chain. If this is enabled, the chain with the name given in the /etc/ngcp-config/config.yml key rtpengine->firewall_iptables_chain will be created as empty, leaving the effective target for UDP packets within the RTP port range as the table's default policy (normally DROP). The RTP proxy will then dynamically created one ACCEPT rule for each open RTP media port in the given chain when a call starts, and delete it when the call is finished. It should be noted that dynamically creating and deleting iptables rules can incur a significant performance overhead, especially in scenarios with high call volumes, and it is therefore not recommended to enable this feature in such cases.

Copyright © Sipwise GmbH - All rights reserved

14.2.4. Custom rules

The Sipwise C5 configuration framework makes it possible to add custom rules to the firewall setup in '/etc/ngcp-config/config.yml'. The custom rules are added after the system rules. Hence, they apply for packets not matched by the systems rules only.

Example custom rule to whitelist all IPv4 traffic from network interface eth1.301 effectively making VLAN 301 a trusted network:

```
rules4:
  - '-A INPUT -i eth1.301 -j ACCEPT'
```

Example custom rule to accept incoming traffic from monitoring station 203.0.113.93 for an optionally installed check_mk agent:

```
rules4:
  - '-A INPUT -p tcp -s 203.0.113.93 --dport 6556 -j ACCEPT'
```

To add hosts or networks to the SSH whitelist they can be either added to key 'sshd.permit_support_from' in '/etc/ngcp-config/config.yml' or a custom rule may be used:

```
rules4:
  - '-A INPUT -s 198.51.100.0/24 --dport 22 -j ACCEPT'
  - '-A INPUT -s 203.0.113.93 --dport 22 -j ACCEPT'
```

NOTE In custom rules keys from '/etc/ngcp-config/config.yml' cannot be referenced. Thus, the values need to be manually looked up, hard coded, and kept in sync manually. This is by design of YAML.

14.2.5. Example firewall configuration section

An example for Sipwise C5 firewall configuration in '/etc/ngcp-config/config.yml' enabling both the firewall subsystem and the logging facility may look like:

Copyright © Sipwise GmbH - All rights reserved

```
security:
  firewall:
    enable: yes
    logging:
      enable: yes
      file: '/var/log/firewall.log'
      tag: 'NGCPFW'
    policies:
      input: 'DROP'
      forward: 'DROP'
      output: 'ACCEPT'
    rules4:
      - '-A INPUT -i eth0 -j ACCEPT'
```

14.3. Password management

The Sipwise C5 comes with some default passwords the user should change during the deployment of the system. They have been explained in the previous chapters of this handbook.

IMPORTANT Many Sipwise C5 services use MySQL backend. Users and passwords for these services are created during the installation. These passwords are unique for each installation, and the connections are restricted to localhost. You should not change these users and passwords.

14.3.1. The "root" account

The Sipwise C5's super-user account comes with a preconfigured password. It is imperative that this password is changed by the operator immediately after Sipwise C5 is shipped and before it is connected to any potentially unsecure public or private network using a secure password in compliance with existing password policies of the operator. The "root" password must not be shared outside of the operator's organization including Sipwise engineers. The "root" password must not be shared in any publicly accessible communications including e-mail or ticketing systems.

To change the root password log into the freshly deployed system as "root" using the preconfigured password and execute:

```
root@myserver:~# passwd
```

Then follow the prompts to change the password.

14.3.2. The "administrator" account

The Sipwise C5 Web-interface comes with a preconfigured "administrator" account deployed with a default password. This account can be considered Sipwise C5 application super-user and has far-reaching access to application specific settings via the Web-interface. It is imperative that the password for this account is changed by the operator immediately after Sipwise C5 is shipped and before it is connected to any potentially unsecure public or private network using a secure password in compliance with existing password policies of the operator. The "administrator" password must not be

 Copyright © Sipwise GmbH - All rights reserved

shared outside of the operator's organization including Sipwise engineers. The "administrator" password must not be shared in any publicly accessible communications including e-mail or ticketing systems.

The password for the "administrator" account can be changed via the Web-interface.

14.3.3. The "cdrexport" account

The login for the system account *cdrexport* is disabled by default. Although this is a jailed account, it has access to sensitive information, namely the Call Detail Records of all calls. SSH keys should be used to login this user, or alternatively a really strong password should be used when setting the password via *passwd cdrexport*.

14.3.4. The MySQL "root" user

The *root* user in MySQL has no default password. A password should be set using the *mysqladmin password* command.

14.3.5. The "ngcpsoap" account

Generate new password for user *ngcpsoap* to access the provisioning interfaces, see the details in REST API.

14.4. Remote 'root' logins via SSH

To mitigate possible system intrusions from the outside, it is commonly held best practise to disable remote 'root' (administrator) logins. With remote root logins disabled, it's still possible to log into the system via SSH to a regular, non-privileged user account, and then use 'sudo' or 'su' to elevate account privileges to 'root' administrator level.

WARNING	For system setup purposes, the default setting on the Sipwise C5 is to permit remote 'root' logins via SSH. It is strongly recommended to change this default setting as soon as possible.

Once you've created a user account for yourself that can be used to gain 'root' privileges, remote 'root' logins can be disabled via the config switch *sshdpermit_root_login* in */etc/ngcp-config/config.yml*.

```
sshd:
  permit_root_login: yes
```

Valid options are:

- 'yes' - permit remote root logins via SSH. Not recommended.
- 'no' - prohibit all remote root logins via SSH.
- 'prohibit-password' - permit remote root logins, except when password authentication is used. This is a good setting if you still wish to access the 'root' account directly using public-key authentication, but also want to prohibit remote brute-force attacks against the 'root' account password.
- 'forced-commands-only' - identical to 'prohibit-password' but with the additional restriction that

Copyright © Sipwise GmbH - All rights reserved

only SSH config sections that have a forced command (via *command=* or *ForceCommand*) are permitted to log in. For advanced users.

If either 'no' or 'forced-commands-only' is chosen, the Sipwise C5 will generate special SSH config sections that still allow root logins (using 'prohibit-password') from IP addresses that are internal to the Sipwise C5. This is necessary to ensure that the scripts handling the Sipwise C5 config framework continue to function and does not compromise security.

14.5. 'sudo-io': logging input/output of commands run through 'sudo'

It is possible to enable logging for the input and output of commands run through 'sudo', along other meta-information like timing, to be able to reproduce sessions.

The logs are saved in directories with a special structure, the numbers are for a so-called "session" or sequence:

```
$BASE_DIR/$USERNAME/00/00/00
```

For example:

```
/var/log/sudo-io/root/00/00/00
```

and within them, several files for tty input and output, stdin/stdout/stderr, and other ancillary files.

New session use *00/00/01*, *00/00/02* and so on.

This is disabled by default, but can be configured in */etc/ngcp-config/config.yml* with the following options:

```
sudo:
  logging:
    enable: no
    exclude_users: []
    max_sessions: 0
```

Valid options are:

- 'enable': 'no' (default) or 'yes'; to use the feature or not
- 'exclude_users': list of users whose commands are excluded from logging
- 'max_sessions': 0 by default, which means unlimited. If set to a positive integer, the sequence returns to 0 after reaching it, and starts to overwrite the files in the subdir '00/00/00' instead of increasing indefinitely.

 Copyright © Sipwise GmbH - All rights reserved

WARNING This sudo plugin saves all input and output including passwords typed in shell prompts (even when they are not echoed to the screen), and this is unencrypted, so please consider the security implications.

WARNING Even if in some versions the compression of these files is enabled (releases mr9.5 and after), some commands can use huge amounts of data and fill-up the available disk space very quickly, and full disk will cause serious problems (failures in services, or even preventing external access to the system to fix the problems). Please consider setting up monitoring of free space, or mitigating measures like setting a small amount of sessions saved, taking data out the system often for preservation and deleting sessions from previous days, etc.

WARNING Additionally, this has a performance impact (both CPU and I/O) on every command run with sudo that triggers logging, negligible for most commands but significant when the input or output is large.

14.6. SSL certificates

The Sipwise C5 provides default, self-signed SSL certificates for SSL connections. These certificates are common for every installation. Before going to production state, the system administrator should provide SSL certificates for the web services. These certificates can either be shared by all web interfaces (*provisioning*, *administrator interface* and *customer self care interface*), or separate ones for each them can be used.

- Generate the certificates. The *customer self care interface* certificate should be signed by a certification authority to avoid browser warnings.
- Upload the certificates to the system
- Set the path to the new certificates in */etc/ngcp-config/config.yml*:

  ossbssapacheautoprovsslcertfile and *ossbssapacheautoprovsslcertkeyfile* for the *provisioning interface*.

  ossbssapacherestapisslcertfile and *ossbssapacherestapisslcertkeyfile* for the *REST interface*.

  www_adminhttp_adminsslcertfile and *www_adminhttp_adminsslcertkeyfile* for the *admin interface*.

  www_adminhttp_cscsslcertfile and *www_adminhttp_cscsslcertkeyfile* for the *customer self care interface*.
- Apply the configuration changes with *ngcpcfg apply 'added web ssl certs'*.

The Sipwise C5 also provides the self-signed SSL certificates for SIP over TLS services. The system administrator should replace them with certificates signed by a trusted certificate authority if he is going to enable it for the production usage (config.yml section: *kamailiolbtls* (TLS is enabled by default and uses self-signed SSL certificates)).

- Generate the certificates.
- Upload the certificates to the system
- Set the path to the new certificates in */etc/ngcp-config/config.yml*:

  kamailiolbtlssslcertfile and *kamailiolbtlssslcertkeyfile* .

Copyright © Sipwise GmbH - All rights reserved

- Apply the configuration changes with *ngcpcfg apply 'added kamailio certs'*.

14.7. Securing your Sipwise C5 against SIP attacks

The Sipwise C5 allows you to protect your VoIP system against SIP attacks, in particular **Denial of Service** and **brute-force attacks**. Let's go through each of those attacks and let's see how to configure your system in order to face such situations and react against them.

14.7.1. Denial of Service

As soon as you have packets arriving on your Sipwise C5 server, it will require a bit of time of your CPU. Denial of Service attacks are aimed to break down your system by sending floods of SIP messages in a very short period of time and keep your system busy to handle such huge amount of requests. Sipwise C5 allows you to block such kind of attacks quite easily, by configuring the following section in your /etc/ngcp-config/config.yml :

```
kamailio:
  lb:
    security:
      dos_ban_enable: yes
      dos_ban_time: 3600
      dos_reqs_density_per_unit: 50
      dos_sampling_time_unit: 2
      dos_whitelisted_ips: []
      dos_whitelisted_subnets: []
```

As soon as Sipwise C5 receives more than 50 messages from the same IP in a time window of 2 seconds, that IP will be blocked for 3600 sec, and you will see in the kamailio-lb.log a line saying:

```
Nov 9 00:11:53 sp1 lb[41958]: WARNING: <script>: IP '1.2.3.4' is blocked
and banned - R=<null> ID=304153-3624477113-19168@tedadg.testlab.local
```

The banned IP will be stored in kamailio memory, you can check the list via web interface or via the following command:

```
# ngcp-kamctl lb fifo htable.dump ipban
```

IMPORTANT On CARRIER systems, this command must be executed on the **active** load balancer node.

Excluding SIP endpoints from banning

There may be some SIP endpoints that send a huge traffic towards Sipwise C5 from a specific IP address. A typical example is a *SIP Peering Server*.

Copyright © Sipwise GmbH - All rights reserved

CAUTION Sipwise C5 supports handling such situations by excluding all defined *SIP Peering Servers* from DoS protection mechanism.

The Sipwise C5 platform administrator may also add whitelisted IP addresses manually in /etc/ngcp-config/config.yml at kamailio.lb.security.dos_whitelisted_ips and kamailio.lb.security.dos_whitelisted_subnets parameters.

14.7.2. Bruteforcing SIP credentials

This is a very common attack that can be checked via the */var/log/ngcp/kamailio-proxy.log* file. There will be INVITE/REGISTER messages coming in with strange usernames. Attackers are trying to spoof/guess subscriber's credentials, which allow them to call out. The very first protection against these attacks is: **ALWAYS USE STRONG PASSWORD**. Nevertheless, Sipwise C5 allows you to detect and block such attacks quite easily by configuring the following parameters in /etc/ngcp-config/config.yml file:

```
kamailio:
  lb:
    security:
      failed_auth_attempts: 3
      failed_auth_ban_enable: yes
      failed_auth_ban_time: 3600
```

It may be required to increase the number of failed attempts by adjusting the ban time (e.g. some users can be banned accidentally because they are not writing the right password). If a user tries to authenticate an INVITE/REGISTER (or more in general, any request containing an *Authorization* or *Proxy-Authorization* SIP header) and if it fails more than 3 times, the *user@domain* (not the IP as for Denial of Service attack) will be blocked for 3600 seconds (see *failed_auth_ban_time* on */etc/ngcp-config/config.yml*). In this case, you will see the following lines in */var/log/ngcp/kamailio-lb.log* file:

```
Nov 9 13:31:56 sp1 lb[41952]: WARNING: <script>: Consecutive
Authentication Failure for 'sipvicous@mydomain.com' UA='sipvicous-
client' IP='1.2.3.4' - R=<null> ID=313793-3624525116-
589163@testlab.local
```

Both the banned IPs and banned users will be shown in the Admin web interface by accessing the *SettingsSecurity Bans* menu. To retrieve the same information from *Kamailio* memory, use the following command:

```
# ngcp-kamctl lb fifo htable.dump auth
```

IMPORTANT On CARRIER systems, this command must be executed on the **active** load balancer node.

Copyright © Sipwise GmbH - All rights reserved

14.8. Topology Hiding

14.8.1. Introduction to Topology Hiding on NGCP

The term "topology hiding" in SIP is used to describe the measures taken by typically an SBC (Session Border Controller) to hide detailed information of the internal network at the border of which it is located. Pieces of information such as IP addresses and port numbers used by SIP endpoints and intermediaries within the network are considered sensitive, as these can give some hints to potential attackers about the topology of the network.

In a typical SIP session the mandatory headers may carry that sensitive information, for example: *Contact, Via, Record-Route, To, From, Call-ID*. An SBC applying topology hiding will mangle the content of those headers.

14.8.2. Topology Masking Mechanism

Concealment of sensitive information using this mechanism is achieved through encoding the original content of selected SIP headers. Then Sipwise C5 will create a new SIP URI using a preselected IP address and the encoded content as URI parameter, finally re-assembling the SIP header.

Examples for encoded SIP headers:

```
Record-Route: <sip:127.0.0.8;line=sr-NvaAlWtecghucEhu6WtAcu...>
Contact: <sip:127.0.0.8;line=sr-NvaAli-1VeL.kRxLcbN86W...>
```

The *load-balancer* element of Sipwise C5 has an SBC role, from the SIP peers point of view. The *LB* offers topology masking function that can be activated through a configuration change. By default the function is disabled.

Configuration of Topology Masking

Activating topology masking function is possible through the modification of the following configuration parameters in /etc/ngcp-config/config.yml file (shown below with default values of parameters):

```
kamailio:
  lb:
    security:
      topoh:
        enable: no
        mask_callid: no
        mask_ip: 127.0.0.8
```

Meaning of the configuration parameters:

- enable: if set to `yes`, the topology mask will be activated
- mask_callid: if set to `yes`, the SIP Call-ID header will also be encoded
- mask_ip: an IP address that will be used to create valid SIP URIs, after encoding the real/original header content.

 Copyright © Sipwise GmbH - All rights reserved

TIP Any valid, preferably private network address can be used. The suggestion is however to use an address that is not used by any other SIP endpoint or intermediary element in the network.

Considerations for Topology Masking

Although masking sensitive information about a VoIP provider's network is desired, there are some potential side effects caused by topology masking.

The most common example is the consequence that **SIP message size may grow** when applying topology masking. The fact that SIP messages become larger may even prevent Sipwise C5 from communicating successfully with another SIP entity (a peer SBC, for example). This can be expected under following circumstances:

- SIP transport protocol is UDP
- SIP messages have more *Via and Record-Route* headers
- IP packets of SIP messages without the topology masking feature already have a size close to the MTU

In such a case the IP packets carrying SIP messages with encoded headers will have a size exceeding the MTU, that will cause loss of data in some networks.

The recommended solution in such a case is to use TCP transport for SIP messages.

14.8.3. Topology Hiding Mechanism

This mechanism achieves topology hiding by stripping the SIP routing headers that show topology details and storing those data in the associative data structure (hash) in the Redis DB so that it can look it up when a reply or in-dialog SIP message comes in. From the signaling perspective it simulates a SBC (Session Border Controller) on the LB.

Considerations for Topology Hiding

This mechanism offers some benefits over the older topology masking approach:

- It enables the Sipwise C5 to interconnect with SIP endpoints that are not capable of operating through a SIP proxy.
- The message size is decreased because of stripping the SIP Record-Route, Route and Via header fields.
- It solves the interoperability issues with SIP ALG in some cases.
- It retains also the lightweight nature and the efficient operation.

The module uses the auto-expiration of the Redis keys so it can cause temporary spikes in the memory usage and redis keys count until produced data is cleaned up by redis.

Configuration of Topology Hiding

Activation of the topology hiding function is done through the modification of the following configuration parameters in /etc/ngcp-config/config.yml file (shown below with default values of parameters):

Copyright © Sipwise GmbH - All rights reserved

```
topos:
  enable: no
  redis_db: 24
```

In order to activate the function, you should set *enable: 'yes'* in */etc/ngcp-config/config.yml* and leave the Redis DB number unchanged, then execute *ngcpcfg apply "activated topos"*.

14.9. System Requirements and Performance

The Sipwise C5 is a very flexible system, capable of serving from hundreds to several tens of thousands of subscribers in a single node. The system comes with a default configuration, capable of serving up to 50.000 subscribers in a *normal* environment. But there is no such thing as a *normal* environment. And Sipwise C5 has sometimes to be tuned for special environments, special hardware requirements or growing traffic.

NOTE If you have performance issues with regards to disk I/O please consider enabling the 'noatime' mount option for the root filesystem. Sipwise recommends the usage of 'noatime', though remove it if you use software which conflicts with its presence.

In this section some parameters will be explained to allow Sipwise C5 administrator tune the system requirements for optimum performance.

Table 47. Requirement_options

Option	Default value	Requirement impact
cleanuptoolsbinlog_days	15	Heavy impact on the harddisk storage needed for mysql logs. It can help to restore the database from backups or restore broken replication.
databasebufferpoolsize	1/2 * Total system RAM	The installer will calculate the total system RAM and dedicate 50% to the mysql innodb buffer. This value won't be changed in case the system RAM changes so it's up to the administrator to adjust it. For test systems or low RAM systems, lowering this setting is one of the most effective ways of releasing RAM. The administrator can check the innodb buffer hit rate on production systems; a hit rate over 99% is desired to avoid bottlenecks.
kamailiolbpkg_mem	16	This setting affects the amount of RAM the system will use. Each kamailio-lb worker will have this amount of RAM reserved. Lowering this setting up to 8 will help to release some memory depending on the number of kamailio-lb workers running. This can be a dangerous setting as the lb process could run out of memory. Use with caution.

 Copyright © Sipwise GmbH - All rights reserved

Option	Default value	Requirement impact
kamailiolbshm_mem	1/16 * Total System RAM	The installer will set this value to 1/16 of the total system RAM. This setting does not change even if the system RAM does so it's up to the administrator to tune it. It has been calculated that 1024 (1GB) is a good value for 50K subscriber environment. For a test environment, setting the value to 64 should be enough. "Out of memory" messages in the kamailio log can indicate that this value needs to be raised.
kamailiolbtcp_children	8	Number of TCP workers kamailio-lb will spawn per listening socket. The value should be fine for a mixed UDP-TCP 50K subscriber system. Lowering this setting can free some RAM as the number of kamailio processes would decrease. For a test system or a pure UDP subscriber system 2 is a good value. 1 or 2 TCP workers are always needed.
kamailiolbtlsenable	yes	Enable or not TLS signaling on the system. Setting this value to "no" will prevent kamailio to spawn TLS listening workers and free some RAM.
kamailiolbudp_children	8	See *kamailiolbtcp_children* explanation
kamailioproxychildren	8	See *kamailiolbtcp_children* explanation. In this case the proxy only listens udp so these children should be enough to handle all the traffic. It could be set to 2 for test systems to lower the requirements.
kamailioproxy*_expires		Set the default and the max and min registration interval. The lower it is more REGISTER requests will be handled by the lb and the proxy. It can impact in the network traffic, RAM and CPU usage.
kamailioproxynatping_interval	30	Interval for the proxy to send a NAT keepalive OPTIONS message to the nated subscriber. If decreased, this setting will increase the number of OPTIONS requests the proxy needs to send and can impact in the network traffic and the number of natping processes the system needs to run. See *kamailioproxynatping_processes* explanation.
kamailioproxynatping_processes	7	Kamailio-proxy will spawn this number of processes to send keepalive OPTIONS to the nated subscribers. Each worker can handle about 250 messages/second (depends on the hardware). Depending the number of nated subscribers and the *kamailioproxynatping_interval* parameter the number of workers may need to be adjusted. The number can be calculated like nated_subscribers/natping_interval/pings_per_second_per_process. For the default options, assuming 50K nated subscribers in the system the parameter value would be 50.000/30/250 = (6,66) 7 workers. 7 is the maximum number of processes kamailio will accept. Raising this value will cause kamailio not to start.

Copyright © Sipwise GmbH - All rights reserved

Option	Default value	Requirement impact
kamailioproxyshm_mem	1/16 * Total System RAM	See *kamailiolbshm_mem* explanation.
rateomatenable	yes	Set this to no if the system shouldn't perform rating on the CDRs. This will save CPU usage.
rsyslogexternal_log	0	If enabled, the system will send the log messages to an external server. Depending on the *rsyslogexternal_loglevel* parameter this can increase dramatically the network traffic.
rsyslogngcp_logs_preserve_days	93	This setting will set the number of days ngcp logs under */var/log/ngcp* will be kept in disk. Lowering this setting will free a high amount of disk space.

TIP In case of using virtualized environment with limited amount of hardware resources, you can use the script *ngcp-toggle-performance-config* to adjust Sipwise C5 configuration for high/low performance:

```
root@spce:~# /usr/sbin/ngcp-toggle-performance-config
/usr/sbin/ngcp-toggle-performance-config - tool to adjust Sipwise C5
configuration for low/high performance

  --help                Display this usage information
  --high-performance    Adjust configuration for system with normal/high
performance
  --low-performance     Adjust configuration for system with low
performance (e.g. VMs)

root@spce:~#
```

14.10. Troubleshooting

The Sipwise C5 platform provides detailed logging and log files for each component included in the system via rsyslog. The main folder for log files is */var/log/ngcp/*, it contains a list of self explanatory log files named by component name.

The Sipwise C5 is a high performance system which requires compromise between traceability (maximum amount of debug information being written to hard drive) and productivity (minimum load on IO subsystem). This is the reason why different log levels are configured for the provided components by default.

Most log files are designed for debugging Sipwise C5 by Sipwise operational team while main log files for daily routine usage are:

 Copyright © Sipwise GmbH - All rights reserved

Log file	Content	Estimated size
/var/log/ngcp/api.log	API logs providing type and content of API requests and responses as well as potential errors	medium
/var/log/ngcp/panel.log /var/log/ngcp/panel-debug.log	Admin Web UI logs when performing operational tasks on the ngcp-panel	medium
/var/log/ngcp/cdr.log	mediation and rating logs, e.g. how many CDRs have been generated and potential errors in case of CDR generation or rating fails for particular accounting data	medium

Copyright © Sipwise GmbH - All rights reserved

Log file	Content	Estimated size
/var/log/ngcp/ha.log	fail-over related logs in case a node in a pair loses connection to the other side, when a standby node takes over or an active node goes standby due to intra-node communication issues or external ping node connection issues	small
/var/log/ngcp/kamailio-proxy.log	Overview of SIP requests and replies between lb, proxy and sems processes. It's the main log file for SIP overview	huge
/var/log/ngcp/kamailio-lb.log	Overview of SIP requests and replies along with network source and destination information flowing through the platform	huge

Copyright © Sipwise GmbH - All rights reserved

Log file	Content	Estimated size
/var/log/ngcp/sems-b2b.log	Overview of SIP requests and replies between lb, proxy and sems processes	small
/var/log/ngcp/rtp.log	rtpengine related log, showing information about RTP communication	small

WARNING it is highly NOT recommended to change default log levels as it can cause system IO overloading which will affect call processing.

NOTE the exact size of log files depend on system type, system load, system health status and system configuration, so cannot be estimated with high precision. Additionally operational network parameters like ASR and ALOC may impact the log files' size significantly.

14.10.1. Collecting call information from logs

The easiest way to fetch information about a single call among the log files is the search for the SIP CallID (a unique identifier for a SIP dialog). The call ID is used as call marker in almost all the VoIP related log file, such as */var/log/ngcp/kamailio-lb.log*, */var/log/ngcp/kamailio-proxy.log*, */var/log/ngcp/sems.log*, */var/log/ngcp/sems-b2b.log* or */var/log/ngcp/rtp.log*. Example of kamailio-proxy.log line:

```
Nov 19 00:35:56 sp1 proxy[7475]: NOTICE: <script>: New request on proxy
- M=REGISTER R=sip:sipwise.local
F=sip:jdoe@sipwise.local T=sip:jdoe@sipwise.local IP=10.10.1.10:5060
(127.0.0.1:5060) ID=364e467677662103497793 4e055d19ea@127.0.0.1 UA='SIP-
UA 1.2.3.4'
```

The above line shows the SIP information you can find in a general line contained in */var/log/ngcp/kamailio-**:

- M=REGISTER : The SIP Method
- R=sip:sipwise.local : The SIP Request URI
- F=sip:jdoe@sipwise.local : The SIP From header
- T=sip:jdoe@sipwise.local : The SIP To header

Copyright © Sipwise GmbH - All rights reserved

- IP=10.10.1.10:5060 (127.0.0.1:5060) : The source IP where the message is coming from. Between brackets it is shown the local internal IP where the message come from (in this case Load Balancer)
- ID=364e4676776621034977934e055d19ea@127.0.0.1 : The SIP CallID.
- UAIP=10.10.1.10 : The User Agent source IP
- UA='SIP-UA 1.2.3.4' : The SIP User Agent header

In order to collect the full log related to a single call, it's necessary to "grep" the */var/log/ngcp/kamailio-proxy.log* using the **ID=** string, for example:

```
# grep "364e4676776621034977934e055d19ea@127.0.0.1"
/var/log/ngcp/kamailio-proxy.log
```

14.10.2. Collecting SIP traces

The Sipwise C5 platform provides several tools to collect SIP traces. It can be used Sipwise C5 *ngrep-sip* tool to collect SIP traces, for example to fetch traffic in text format from outbound and among load balancer, proxy and sems :

```
# ngrep-sip b
```

see the manual to know all the options:

```
# man ngrep-sip
```

The *ngrep* debian tool can be used in order to make a SIP trace and save it into a *.pcap* file :

```
# ngrep -s0 -Wbyline -d any -O /tmp/SIP_trace_file_name.pcap port 5062
or port 5060
```

The *sngrep* debian graphic tool as well can be used to visualize SIP trace and save them in a *.pcap* file :

```
# sngrep
```

The Sipwise C5 PRO platform provides also the native VoIP sniffer, called *ngcp-voisniff*, which provide a graphic view of all the calls passing through the platform. It can be enabled via __/etc/ngcp-config/config.yml:

 Copyright © Sipwise GmbH - All rights reserved

```
voisniff:
  admin_panel: yes
  daemon:
    custom_bpf: ''
    filter:
      exclude:
      - active: '0'
        case_insensitive: '1'
        pattern: '\ncseq: *\d+ +(register|notify|options)'
      include: []
      sip_ports:
      - 5060
      - 5062
    interfaces:
      extra: []
      types:
      - sip_int
      - sip_ext
      - rtp_ext
    li_x1x2x3:
      call_id:
        del_patterns:
        - _pbx\-1(?:_[0-9]{1,10})?$
        - _b2b\-1(?:_[0-9]{1,10})?$
        - _xfer\-1(?:_[0-9]{1,10})?$
      captagent:
        cin_max: '3000'
        cin_min: '0'
        x2:
          threads: 20
      client_certificate: ''
      enable: no
      fix_checksums: no
      fragmented: no
      interface:
        excludes: []
      local_name: sipwise
      x1:
        port: '18090'
      x23:
        protocol: sipwise
    mysql_dump:
      enable: yes
      max_query_len: 67108864
      num_threads: '4'
    rtp_filter: yes
    start: yes
    threads_per_interface: '2'
  partitions:
    increment: '700000'
    keep: '10'
```

Copyright © Sipwise GmbH - All rights reserved

admin_panel should be set to 'yes' as well as start and mysql_dump.enable. Also filter.exclude.active should be set to '1' in order to avoid to sniff REGISTER, NOTIFY, OPTIONS and SUBSCRIBE messages. Then run:

```
ngcpcfg apply 'enable voisniff' && ngcpcfg push
```

WARNING Please notice that enabling voisniff, specially under a huge amount of traffic, may affect the system performance due to the fact that voisniff needs to save all the traffic into the database.

14.11. Log file obfuscation

As many of the log files produced by Sipwise C5 contain sensitive and private data, and as various jurisdictions around the world have placed restrictions on who can view whose private data (e.g. GDPR in the EU), Sipwise C5 provides a mechanism to safely view log files in a partially obfuscated and anonymised (pseudonymised) fashion.

This obfuscated view is provided by the system service ngcp-logfs and is enabled by default. This service provides a read-only, partially obfuscated view of the Sipwise C5 log files in a separate folder, which by default is */var/log/mirror-ngcp/*. The actual log files in */var/log/ngcp/* are normally readable only by the system administrator (root), while the obfuscated view of them in */var/log/mirror-ngcp/* is readable even by non-administrator system users by default.

Log files produced by Sipwise C5 contain special markers that identify data fields that correspond to private data belonging to 3rd parties. When accessing the log files through ngcp-logfs, the data contained in these fields will be replaced by other, seemingly random strings. However, this replacement is deterministic, meaning that the same original string will always be replaced with the same obfuscated string, making it still possible to correlate log lines belonging to the same entity, even across log files from different applications. Examples of such obfuscated data fields are user names, phone numbers, IP addresses, and other uniquely identifiable data fields.

The ngcp-logfs service also provides the same kind of access to archived (rotated) log files contained in */var/log/ngcp/old/*. While these log files are compressed (.gz) on disk, they appear as uncompressed, plain text files when viewed through ngcp-logfs, as the service decompresses them in the background on demand.

14.11.1. Configuration

The service can be configured via its respective section in /etc/ngcp-config/config.yml:

Copyright © Sipwise GmbH - All rights reserved

```
logfs:
  cache_db: /usr/lib/ngcp-logfs/cache.db
  chmod_dirs: '0555'
  chmod_files: '0444'
  disk_retention_timeout: 365
  enable: yes
  file_cache_timeout: 2
  gid: 0
  log_dir: /var/log/ngcp
  max_mem_usage: 500
  mem_cache_timeout: 24
  mountpoint: /var/log/mirror-ngcp
  obfuscation_prefix: GDPR
  suffix: \.\d+$|-\d{8}$|-\d{8}-\d+$
  uid: 0
```

- cache_db: path and file name of the on-disk cache for obfuscated strings and their replacements. Does not normally need to be changed.
- chmod_dirs: Unix file mode for directories visible through ngcp-logfs in octal. Defaults to octal 0555 (world readable).
- chmod_files Unix file mode for log files visible through ngcp-logfs in octal. Defaults to octal 0444 (world readable).
- disk_retention_timeout: how long to store obfuscated strings in the on-disk cache before they get deleted, in days. Defaults to 365 (one year).
- enable: master switch for the service itself, yes or no.
- file_cache_timeout: how long to cache obfuscated files (portions or entirely) in memory, in hours. Defaults to 2 hours.
- gid: numeric Unix group ID for presented files and directories. Defaults to 0 (root).
- log_dir: root directory of the log files that should be mirrored. Does not normally need to be changed.
- max_mem_usage: upper limit in megabytes for the in-memory cache for obfuscated files. If this limit is hit, the in-memory cache will start to get aggressively emptied, even if file_cache_timeout isn't yet reached. Defaults to 500 MB.
- mem_cache_timeout: how long to cache obfuscated strings and their replacements in memory, in hours. Defaults to 24 hours.
- mountpoint: where to make obfuscated log files visible in the file system.
- obfuscation_prefix: optional prefix that obfuscated strings are guaranteed to have, provided the string is at least twice as long as the prefix. The prefix therefore should be kept short. Can be empty to disable this feature.
- suffix: regular expression to match the file name suffix for archived (rotated) log files. Does not normally need to be changed.
- uid: numeric Unix user ID for presented files and directories. Defaults to 0 (root).

Copyright © Sipwise GmbH - All rights reserved

TIP Access to obfuscated log files can be further restricted by setting chmod_files, chmod_dirs and uid and/or gid. For example, to make log files accessible only to users belonging to the system group adm with group ID 4, set gid to 4, set chmod_files to 0440, and chmod_dirs to 0550, followed by executing *ngcpfg apply*.

14.11.2. Forward and reverse lookup

In some cases, for example for troubleshooting purposes, it can be necessary to determine the underlying unobfuscated plain text string from its obfuscated version, or perhaps even vice versa. For this purpose, the tool ngcp-lookup-obfuscated is provided, which can only be used by the system administrator (root). To perform a forward lookup (obfuscated string to unobfuscated), call it with the obfuscated string as its first argument. To perform a reverse lookup (unobfuscated to obfuscated), add the -r option.

For example, take the following sample log line provided by ngcp-logfs:

```
Mar 28 06:00:27 sp1 proxy[2544]: NOTICE: <script>: Sending reply S=200
Alive fs='127.0.0.1:5062' du='127.0.0.1:5060' -
R=«GDPRcarPrUHeRvvWF2JjGei2Lu0bUjQIgI» ID=«GDPRqOB2kvuAm0JC7zQN6E1w4K»
UA='sipsak 0.9.7pre'
```

The following commands would be used to perform both forward and reverse lookups on the call ID:

```
root@sp1:~# ngcp-lookup-obfuscated GDPRqOB2kvuAm0JC7zQN6E1w4K
1899127565@192.168.255.251
root@sp1:~# ngcp-lookup-obfuscated -r 1899127565@192.168.255.251
GDPRqOB2kvuAm0JC7zQN6E1w4K
```

NOTE These lookups, in particular the reverse lookup, only work on strings that were actually processed by ngcp-logfs. You cannot use the reverse lookup procedure to obfuscate any arbitrary string that wasn't previously provided by ngcp-logfs. The lookup must also be performed on the same host on which ngcp-logfs performed the obfuscation. Lookups don't necessarily work on other hosts.

14.12. NGCP Panel passwords encryption

Encryption is used in case of administrator users passwords and subscribers web passwords.

In case you have upgraded from an earlier version where the passwords where plain text, the ngcp-bcrypt-webpassword can be used, which will encrypt all subscribers web passwords.

Copyright © Sipwise GmbH - All rights reserved

Chapter 15. Monitoring and Alerting

15.1. Internal Monitoring

15.1.1. Service monitoring

The platform uses both *systemd* and *monit* daemons to monitor all essential services. Since Sipwise C5 runs in an active/standby mode, not all services are always running on both nodes, some of them will only run on the active node and be stopped on the standby node. The following commands show the most critical services on the platform:

- ngcp-service summary - to get the list of services and their current status,
- systemctl status - to get a tree of the services running,
- systemctl list-units - to get a list of the service states,
- monit summary - to get the list of services known to monit and their current status,
- monit status - to get the list of services known to monit with detailed status.

IMPORTANT When you perform a stop/start/monitor/unmonitor operation on a service, *monit* affects other services that depend on the initial one. Hence, if you stop or unmonitor a service all services that depend on it will be stopped or unmonitored as well.

For example, monit stop mysql operation will stop kamailio, sbc, asterisk, prosody and some other services. Although the recommended way to operate on services is via the ngcp-service wrapper which will take care of abstracting the underlying process monitoring implementation.

If any service ever fails for whatever reason either the *systemd* or *monit* daemons will quickly restart it. When that happens, the daemon will send a notification email to the address specified in the config.yml file under the general.adminmail key. It will also send warning emails to this address under certain abnormal conditions, such as high memory consumption (> 75% is used) or high CPU load.

IMPORTANT In order for *monit* to be able to send emails to the specified address, the local MTA (*exim4*) must be configured correctly. The CE edition's handbook contains more information about this in the *Installation* chapter.

15.1.2. System monitoring backend

The platform uses the *Prometheus* monitoring backend on new installations and on upgraded systems that have been migrated.

The platform uses various monitoring backend services to monitor many aspects of the system, including CPU, memory, swap, disk, filesystem, network, processes, NTP, Nginx, Redis and MySQL.

The gathered information is stored in *VictoriaMetrics* which is a long-term storage backend for *Prometheus*. NOTE: Both *VictoriaMetrics* and *Prometheus* can act as the prometheus server implementation, and are mutually exclusive in their execution.

Copyright © Sipwise GmbH - All rights reserved

15.1.3. Sipwise C5 specific monitoring via ngcp-witnessd

The platform uses the internal *ngcp-witnessd* service to monitor Sipwise C5 specific metrics or system metrics currently not tracked by the monitoring backend (via *Prometheus* exporters), including HA status, MTA, Kamailio, SIP and MySQL.

The gathered information is stored in *VictoriaMetrics* in the *ngcp* namespace on its time-series database.

TIP Some of the data gathering can be disabled (most are enabled by default) through the config.yml file, and those data points will then either be missing from the database or be initialized with a stub value. This will then cascade into other subsystems using this monitoring information, such as *Grafana* dashboards or SNMP OIDs. The enable/disable flags can be found in the witnessd.gather section.

15.1.4. Monitoring data in the monitoring backend

The platform uses *VictoriaMetrics* as a long-term *Prometheus* time series database to store most of the metrics collected in the system.

On a Sipwise C5 each node stores its own metrics and the ones for their peer node, and in addition on CARRIER systems the management nodes store the metrics for all the nodes in the cluster. On new installations and migrated ones this is done with *Prometheus* instances on each peer, and a *VictoriaMetrics* instance on the management node which uses its *Prometheus* federation and scrapping support.

The monitoring data is used by various components of the platform, including *ngcp-collective-check*, *ngcp-snmp-agent* and by the statistics dashboard powered by *Grafana*.

The monitoring data can also be accessed directly by various means. On new installations by using the *promtool* command-line tool; or by using the HTTP API with *curl* (or other HTTP fetchers), or with the *NGCP::Prometheus::HTTP* perl module.

Monitoring metrics

See Prometheus monitoring metrics for detailed information about the list of *ngcp* namespaced metrics stored in the *Prometheus* monitoring database.

PromQL

See https://prometheus.io/docs/prometheus/latest/querying/basics/ for information about PromQL, the query language used by *Prometheus*.

TIP To get the list of all metrics for a specific namespace the following query can be used {__name__=~"^namespace_.+"}.

15.2. Statistics Dashboard

The platform's administration interface (described in Kick-off) provides a graphical overview based on *Grafana* of the most important system health indicators, such as memory usage, load averages and disk usage. VoIP statistics, such as the number of concurrent active calls, the number of provisioned and registered subscribers, etc. is also present.

 Copyright © Sipwise GmbH - All rights reserved

15.3. External Monitoring Using SNMP

15.3.1. Overview and Initial Setup

The Sipwise C5 exports a variety of cluster health data and statistics over the standard SNMP interface. By default, the SNMP interface can only be accessed locally. To make it possible to provide the SNMP data to an external system, the config.yml file needs to be edited and the list of allowed community names and allowed hosts/IP ranges must be populated. This list can be found under the snmpd.communities key and it consists of one or more hashes of name and sources key/values. The community name is the allowed community name, while sources is a list of IP address or IP blocks where to allow the requests from.

The SNMP notifications (or traps) can also be configured in a similar way, to send them to an external system, by populating the snmpd.trap_communities key with name and targets key/values. The community trap name is the value that will be used when sending the trap, while the targets is a list of IP addresses where to send the trap.

The public communities with the localhost source and target are used for local testing of SNMP functionality. It is recommended that you leave these entries in place. Other legal sources can be formed as single IP addresses or IP blocks in IP/prefix notation, for example 192.168.115.0/24. Other targets can be formed as single IP addresses.

The origin of the SNMP notifications for the SIPWISE MIBs can also be configured with the snmpagent.traps.origin. The supported modes are:

- legacy: The node triggering the condition and its peer (if available) will emit the trap, in addition the management node pair (if distinct) will also emit the trap. This was the original behavior.
- mgmt: Only the active management node will emit the trap. This is the current default.
- distributed: Only the node triggering the condition will emit the trap. For cluster-wide conditions (those that are not node-specific), this mode is equivalent to the mgmt mode.

The Sipwise C5 supports two types of SNMP traps. Event-based, sent whenever a state changes, with a single trap per tracked state. Alarm-based, sent on problematic conditions arising or clearing, with a different trap per state group, where the trap severity is included as part of the trap itself (in addition to the usual convention of documenting it in the MIB). These two types can be enabled or disabled independently (both are enabled by default), depending on the type of monitoring intended.

TIP The event-based traps have been supported for longer, while the alarm-based traps were introduced in mr9.5, so if there are interoperability requirements with various Sipwise C5 versions, event-based traps have better availability, although with different properties.

TIP To locally check if SNMP is working correctly, execute the command snmpwalk -v2c -cpublic localhost . (note the trailing dot). This will generate a long list of raw SNMP OIDs and their values, provided that the default SNMP community key has been left in place. Alternatively the ngcp-systems-tests program checks whether several expected SNMP OIDs are present.

Copyright © Sipwise GmbH - All rights reserved

TIP To locally check if SNMP notifications (or traps) are working correctly, enable the *snmptrapd* daemon, which will be configured by default to catch the traps sent by the localhost SNMP agent. The traps will show up on `/var/log/ngcp/snmp-trap.log`, and a couple of traps can be generated by running ngcp-service restart snmpd. Even though traps generated by *ngcp-snmp-agent* are logged on `/var/log/ngcp/snmp-agent.log`, because the service responsible for sending out the traps is *snmpd*, checking with *snmptrapd* is always an additional safety check in case of problems.

TIP To get information from SNMP tables, you can use the command snmptable -v2c -cpublic -Ci localhost TABLE-OID, where `TABLE-OID` could be for example *procTable*.

TIP When using snmptable you might want to use the `-S` option in *less* (either when calling it or typing it on its prompt) to get proper tabular output that does not fold on terminal end.

NOTE SNMP version 1 and version 2c are supported.

15.3.2. Details

There are two kinds of information that can be retrieved from SNMP OIDs (Object Identifiers). The first one is the native Sipwise C5 cluster overview from Sipwise C5 MIBs (Management Information Bases), which is available from the management nodes. The second is from the stock snmpd implementing the UCD (University of California, Davis) MIBs, which requires querying each individual node.

Sipwise C5 OIDs

The entire Sipwise C5 cluster can be monitored from the management nodes by using the SIPWISE-NGCP-MIB, SIPWISE-NGCP-MONITOR-MIB and SIPWISE-NGCP-ALARMS-MIB (SIPWISE-NGCP-STATS-MIB is deprecated and should not be used anymore). These OIDs are rooted at Sipwise C5 slot .1.3.6.1.4.1.34274.1.*.

The MIBs are self-documented, and can be found as part of the *ngcp-snmp-mibs* package (running dpkg -S 'SIPWISE*MIB' will list their pathnames). The Sipwise C5 SNMP Agent is a part of the *ngcp-snmp-agent* package, which is installed by default and works out-of-the-box as long as the snmpd has been properly configured.

The SIPWISE-NGCP-MIB acts as the root MIB and provides information about the cluster licensing and layout (which is mostly static data about each node, such as node name, its IP address, its roles, etc.) and information required to access the OIDs from the other MIBs. The *clusterTable* defines the nodes layout of the cluster, and its cluster node index (*cnIndex*) is used by many of the other tables to index entries within that specific cluster node (for example within the *procTable* in the monitor MIB).

The SIPWISE-NGCP-MONITOR-MIB provides current monitoring information, global health conditions, the number of provisioned and registered subscribers and devices. It also provides per node information (independently of the number of nodes or their names) on their filesystem, processes, databases, system load, memory, HA status, MTA queues, etc. In addition it defines the event-based traps.

The SIPWISE-NGCP-ALARMS-MIB defines the alarm-based traps.

The SIPWISE-NGCP-STATS-MIB is deprecated and has been superseded by the SIPWISE-NGCP-

 Copyright © Sipwise GmbH - All rights reserved

MONITOR-MIB.

NOTE OIDs under the following trees are not yet implemented: *ngcpMonitorFraud*, *ngcpMonitorPerformance.sipStatsTable.sipCallAttemptsPerSecond*. Deprecated OIDs are currently implemented but will eventually be obsoleted. Obsolete OIDs are not implemented and won't be in the future.

NOTE The Sipwise C5 SNMP Agent uses *Redis* and *Prometheus* as data sources. This data is essential for accurate and complete monitoring data in the SNMP OID tree. In addition, the *Redis* database must be available on a shared IP address, so that *ngcp-witnessd* can always write to it.

UCD OIDs

All basic system health variables (such as memory, disk, swap, CPU usage, network statistics, process lists, etc.) for every node can also be found in standard OID slots from standard MIBs from each node. For example, memory statistics can be found through the *UCD-SNMP-MIB* in OIDs such as memTotalSwap.0, memAvailSwap.0, memTotalReal.0, memAvailReal.0, etc., which translate to numeric OIDs .1.3.6.1.4.1.2021.4.*. In fact, UCD-SNMP-MIB is a useful MIB for overall non-centralized system health checks.

Additionally, there is a list of specially monitored processes, also found through the UCD-SNMP-MIB. UCD-SNMP-MIB::prNames (.1.3.6.1.4.1.2021.2.1.2) gives the list of monitored processes, prCount (.1.3.6.1.4.1.2021.2.1.5) is how many of each process are running and prErrorFlag (.1.3.6.1.4.1.2021.2.1.100) gives a 0/1 error indication (with prErrMessage (.1.3.6.1.4.1.2021.2.1.101) providing an explanation of any error).

TIP Some of these processes are not supposed to be running on the standby node, so you will see the error flag raised there. A possible solution is to run these SNMP checks against the shared service IP of the cluster. See in High Availability and Fail-Over below for more information.

IMPORTANT Furthermore, Sipwise C5 used to provide platform specific information via the UCD-SNMP-MIB custom external extension OIDs, which have been superseded by the Sipwise MIBs, and need to be migrated to use the latter. The names of these OIDs could be found under the UCD-SNMP-MIB::extNames (.1.3.6.1.4.1.2021.8.1.2) tree, with extOutput (.1.3.6.1.4.1.2021.8.1.101) providing the output (one line) from each check and extResult (.1.3.6.1.4.1.2021.8.1.100) the exit code from each check. The following table gives a rough mapping for that migration:

UCD OID name	UCD check name	SIPWISE-NGCP OID name
UCD-SNMP-MIB::extNames.1	collective_check	SIPWISE-NGCP-MONITOR-MIB::ngcpCollectiveCheckResult and SIPWISE-NGCP-MONITOR-MIB::ngcpCollectiveCheckOutput
UCD-SNMP-MIB::extNames.2	sip_check_sp1	SIPWISE-NGCP-MONITOR-MIB::sipResponsiveness.*

Copyright © Sipwise GmbH - All rights reserved

UCD OID name	UCD check name	SIPWISE-NGCP OID name
UCD-SNMP-MIB::extNames.3	sip_check_sp2	SIPWISE-NGCP-MONITOR-MIB::sipResponsiveness.*
UCD-SNMP-MIB::extNames.4	mysql_check_sp1	SIPWISE-NGCP-MONITOR-MIB::dbQueryRate.*
UCD-SNMP-MIB::extNames.5	mysql_check_sp2	SIPWISE-NGCP-MONITOR-MIB::dbQueryRate.*
UCD-SNMP-MIB::extNames.6	mysql_replication_check_sp1	SIPWISE-NGCP-MONITOR-MIB::dbReplDelay.*
UCD-SNMP-MIB::extNames.7	mysql_replication_check_sp2	SIPWISE-NGCP-MONITOR-MIB::dbReplDelay.*
UCD-SNMP-MIB::extNames.8	mpt_check_sp1	Obsolete
UCD-SNMP-MIB::extNames.9	mpt_check_sp2	Obsolete
UCD-SNMP-MIB::extNames.10	exim_queue_check_sp1	SIPWISE-NGCP-MONITOR-MIB::mailQueue.*
UCD-SNMP-MIB::extNames.11	exim_queue_check_sp2	SIPWISE-NGCP-MONITOR-MIB::mailQueue.*
UCD-SNMP-MIB::extNames.12	provisioned_subscribers_check_sp1	SIPWISE-NGCP-MONITOR-MIB::ngcpClusterProvSubs
UCD-SNMP-MIB::extNames.13	provisioned_subscribers_check_sp2	SIPWISE-NGCP-MONITOR-MIB::ngcpClusterProvSubs
UCD-SNMP-MIB::extNames.14	kam_dialog_active_check_sp1	SIPWISE-NGCP-MONITOR-MIB::sipDialogActive.*
UCD-SNMP-MIB::extNames.15	kam_dialog_active_check_sp2	SIPWISE-NGCP-MONITOR-MIB::sipDialogActive.*
UCD-SNMP-MIB::extNames.16	kam_dialog_early_check_sp1	SIPWISE-NGCP-MONITOR-MIB::sipEarlyMedia.*
UCD-SNMP-MIB::extNames.17	kam_dialog_early_check_sp2	SIPWISE-NGCP-MONITOR-MIB::sipEarlyMedia.*
UCD-SNMP-MIB::extNames.18	kam_dialog_type_local_check_sp1	SIPWISE-NGCP-MONITOR-MIB::sipDialogLocal.*
UCD-SNMP-MIB::extNames.19	kam_dialog_type_local_check_sp2	SIPWISE-NGCP-MONITOR-MIB::sipDialogLocal.*
UCD-SNMP-MIB::extNames.20	kam_dialog_type_relay_check_sp1	SIPWISE-NGCP-MONITOR-MIB::sipDdialogRelay.*
UCD-SNMP-MIB::extNames.21	kam_dialog_type_relay_check_sp2	SIPWISE-NGCP-MONITOR-MIB::sipDdialogRelay.*
UCD-SNMP-MIB::extNames.22	kam_dialog_type_incoming_check_sp1	SIPWISE-NGCP-MONITOR-MIB::sipDdialogIncoming.*
UCD-SNMP-MIB::extNames.23	kam_dialog_type_incoming_check_sp2	SIPWISE-NGCP-MONITOR-MIB::sipDdialogIncoming.*

Copyright © Sipwise GmbH - All rights reserved

UCD OID name	UCD check name	SIPWISE-NGCP OID name
UCD-SNMP-MIB::extNames.24	kam_dialog_type_outgoing_check_sp1	SIPWISE-NGCP-MONITOR-MIB::sipDdialogOutgoing.*
UCD-SNMP-MIB::extNames.25	kam_dialog_type_outgoing_check_sp2	SIPWISE-NGCP-MONITOR-MIB::sipDdialogOutgoing.*
UCD-SNMP-MIB::extNames.26	kam_usrloc_regusers_check_sp1	SIPWISE-NGCP-MONITOR-MIB::ngcpClusterRegSubs
UCD-SNMP-MIB::extNames.27	kam_usrloc_regusers_check_sp2	SIPWISE-NGCP-MONITOR-MIB::ngcpClusterRegSubs
UCD-SNMP-MIB::extNames.28	kam_usrloc_regdevices_check_sp1	SIPWISE-NGCP-MONITOR-MIB::ngcpClusterRegDevs
UCD-SNMP-MIB::extNames.29	kam_usrloc_regdevices_check_sp2	SIPWISE-NGCP-MONITOR-MIB::ngcpClusterRegDevs
UCD-SNMP-MIB::extNames.30	mysql_replication_discrepancies_check_sp1	Obsolete
UCD-SNMP-MIB::extNames.31	mysql_replication_discrepancies_check_sp2	Obsolete
UCD-SNMP-MIB::extNames.32	sip_check_self	SIPWISE-NGCP-MONITOR-MIB::sipResponsiveness.*
UCD-SNMP-MIB::extNames.33	mysql_check_self	SIPWISE-NGCP-MONITOR-MIB::dbQueryRate.*
UCD-SNMP-MIB::extNames.34	mysql_replication_check_self	SIPWISE-NGCP-MONITOR-MIB::dbReplDelay.*
UCD-SNMP-MIB::extNames.35	mpt_check_self	Obsolete
UCD-SNMP-MIB::extNames.36	exim_queue_check_self	SIPWISE-NGCP-MONITOR-MIB::mailQueue.*
UCD-SNMP-MIB::extNames.37	provisioned_subscribers_check_self	SIPWISE-NGCP-MONITOR-MIB::ngcpClusterProvSubs
UCD-SNMP-MIB::extNames.38	kam_dialog_active_check_self	SIPWISE-NGCP-MONITOR-MIB::sipDialogActive.*
UCD-SNMP-MIB::extNames.39	kam_dialog_early_check_self	SIPWISE-NGCP-MONITOR-MIB::sipEarlyMedia.*
UCD-SNMP-MIB::extNames.40	kam_dialog_type_local_check_self	SIPWISE-NGCP-MONITOR-MIB::sipDialogLocal.*
UCD-SNMP-MIB::extNames.41	kam_dialog_type_relay_check_self	SIPWISE-NGCP-MONITOR-MIB::sipDialogRelay.*
UCD-SNMP-MIB::extNames.42	kam_dialog_type_incoming_check_self	SIPWISE-NGCP-MONITOR-MIB::sipDialogIncoming.*
UCD-SNMP-MIB::extNames.43	kam_dialog_type_outgoing_check_self	SIPWISE-NGCP-MONITOR-MIB::sipDialogOutgoing.*
UCD-SNMP-MIB::extNames.44	kam_usrloc_regusers_check_self	SIPWISE-NGCP-MONITOR-MIB::ngcpClusterRegSubs

Copyright © Sipwise GmbH - All rights reserved

UCD OID name	UCD check name	SIPWISE-NGCP OID name
UCD-SNMP-MIB::extNames.45	kam_usrloc_regdevices_check_self	SIPWISE-NGCP-MONITOR-MIB::ngcpClusterRegDevs
UCD-SNMP-MIB::extNames.46	mysql_replication_discrepancies_check_self	Obsolete
UCD-SNMP-MIB::extNames.47	kam_dialog_type_local_check_prx0X	SIPWISE-NGCP-MONITOR-MIB::sipDialogLocal.*
UCD-SNMP-MIB::extNames.48	kam_dialog_type_relay_check_prx0X	SIPWISE-NGCP-MONITOR-MIB::sipDialogRelay.*
UCD-SNMP-MIB::extNames.49	kam_dialog_type_incoming_check_prx0X	SIPWISE-NGCP-MONITOR-MIB::sipDialogIncoming.*
UCD-SNMP-MIB::extNames.50	kam_dialog_type_outgoing_check_prx0X	SIPWISE-NGCP-MONITOR-MIB::sipDialogOutgoing.*
UCD-SNMP-MIB::extNames.51	kam_dialog_active_check_prx0X	SIPWISE-NGCP-MONITOR-MIB::sipDialogActive.*
UCD-SNMP-MIB::extNames.52	kam_dialog_early_check_prx0X	SIPWISE-NGCP-MONITOR-MIB::sipEarlyMedia.*

Copyright © Sipwise GmbH - All rights reserved

Chapter 16. Licenses

The Sipwise C5 -implements *software licensing* in form of a regular comparison of the licensed services and capacities against the actual usage patterns of the platform. The purpose of this function is to monitor system usage and to raise warnings to the platform operator if the thresholds of commercially agreed license parameters (like number of provisioned subscribers or number of concurrent calls) are exceeded.

16.1. What is Subject to Licensing?

Sipwise C5 licenses determine 2 groups of system parameters which are regularly compared with actual values gathered from the system:

- **performance parameters**:
  - number of provisioned subscribers
  - number of registered subscribers
  - number of concurrent calls
- **feature parameters**: additional features / services that are subject to commercial agreement:
  - pre-paid billing
  - CPBX (Cloud PBX) services
  - Push notifications (mobile SIP clients on iOS and Android)
  - Lawful Interception services
  - SIP capturing via ngcp-voisniff

NOTE Please remember, that the parameter 'number of concurrent calls' by default doesn't affect emergency type of calls. If you would like to change this behavior globally, you can set the config.yml option 'b2b.sbc.skip_cpslimit_license_check_emergency' to 'no' (which is by default set to 'yes'). Remember, by applying the changes the SEMS-B2B component will have to restart.

16.2. How Licensing Works

Sipwise operates a *licensing server* that is the source of license data for each deployed Sipwise C5 node. The nodes themselves request licensing data from the license server regularly and compare them with actual system performance indicators, check the activated features against the licensed ones. The presence and activity of the *license client* module ("licensed" process) may be confirmed by checking e.g. the output of "ngcp-service summary" command. It should contain a line showing:

```
ngcp-license-client           managed    on-boot   active
```

All nodes of a single Sipwise C5 installation share the same license key. This is also valid for geographically distributed setups. This license key is referred by an ID that has to be configured in the main Sipwise C5 configuration file (config.yml), and that ID will be used to request license data from the license server.

Copyright © Sipwise GmbH - All rights reserved

In order for the license validation to work each node of an Sipwise C5 installation must be able to connect to the Sipwise license server via standard HTTPS protocol (TCP, port 443). Alternatively the nodes may use a local, system-wide proxy server and only that proxy server needs to access the Sipwise license server.

16.3. How to Configure Licenses

The Sipwise C5 operator can set the **license key** in the main configuration file (/etc/ngcp-config/config.yml). The correct license key has to be entered in the configuration file, at the **general.license_key** configuration parameter, so that licensing works as expected.

TIP You always have to add the license key before being able to upgrade Sipwise C5 to release mr5.5.x or above. The upgrade script will look for the license key and will stop if it does not find the key.

The license key is also shown in the /etc/ngcp-license-key file once the key has been added to the configuration file and the new configuration has been applied.

NOTE There is another configuration parameter related to licenses: general.anonymous_usage_statistics that has an effect on Sipwise C5 CE installations only. This parameter enables / disables sending anonymous usage statistics to Sipwise.

Although not strictly related to Sipwise C5 configuration, the platform operator has to keep in mind that all Sipwise C5 nodes need to have **access to Sipwise license server:** license.sipwise.com

The operator has to ensure that there is no firewall rule or other network configuration that prevents Sipwise C5 nodes from connecting to Sipwise license server via HTTPS protocol (TCP, port 443).

16.4. How to Monitor License Client

As mentioned earlier in this chapter, the presence of license client can be monitored using the built-in utility ngcp-service.

The other way to observe the behaviour of the license client is looking into the log file of "licensed" process: /var/log/ngcp/licensed.log

The Sipwise C5 operator may find entries like the below ones in case of normal operation:

```
Dec 12 16:20:42 sp1 ngcp-licensed[2205]: Valid license:
[ABCDEFGHI_123456789_a1b2c3d4e5f6]:
  10000 calls, 1000000 subscribers, 2000000 registered subscribers,
valid until Tue Jan  1
  00:00:00 2030 (signature valid until Tue Dec 26 16:20:43 2017)
Dec 12 16:22:41 sp1 ngcp-licensed[2205]: Usage report: 0 calls, 18
subscribers, 0 registered subscribers
```

where:

1. The first line shows *the licensed capacities*

Copyright © Sipwise GmbH - All rights reserved

2. The second line shows *the actual system usage indicators*

Copyright © Sipwise GmbH - All rights reserved

Chapter 17. Customer Self-Care

17.1. Customer Self-Care User Interface (CSC UI)

There are two ways for end users to maintain their subscriber settings: via the *Customer Self-Care Web Interface* (old/Perl or new/JS versions) and via *Vertical Service Codes* using their SIP phones.

17.2. New (default) Vue.JS-based CSC UI

The Sipwise C5 has been migrated to the new Vue.JS-based CSC UI. It provides a list of new features, is also based on modern technologies and allows Sipwise to deliver all the modern features to end users including WebRTC calls and conferencing (available on commercial PRO/Carrier installations only).

The new CSC UI is technically a Single Page Application, that is fully client side rendered and builds on top of the NCGP REST API as part of a Service Oriented Architecture. The new CSC UI source code is published under a GPL license on https://github.com/sipwise/ngcp-csc-ui and can be used as an example for the customised CSC UI development if necessary (using the same REST API methods).

17.3. Old (deprecated) Perl-based CSC UI

You can reconfigure Sipwise C5 to use the old CSC UI using:

```
ngcpcfg set /etc/ngcp-config/config.yml
www_admin.http_csc.csc_js_enable=no
ngcpcfg apply 'Use old and deprecated CSC UI'
ngcpcfg push-parallel all
```

NOTE it is impossible to have both new and old CSC UI enabled simultaneously.

17.4. The Customer Self-Care Web Interface

The Sipwise C5 provides a web panel for end users (CSC panel) to maintain their subscriber accounts, which is running on https://*ngcp-ip*. Every subscriber can log in there, change subscriber feature settings, view their call lists, retrieve voicemail messages and trigger calls using the click-to-dial feature.

17.4.1. Login Procedure

To log into the CSC panel, the end user has to provide his full web username (e.g. user1@1.2.3.4) and the "web password" defined in Creating a Subscriber. Once logged in, he can change his web password in the *Account* section. This will NOT change his SIP password, so if you control the end user devices, you can auto-provision the SIP password into the device and keep it secret, and hand over the web password to the customer. This way, the end user will only be able to place calls with this auto-provisioned device and not with an arbitrary soft-phone, but can nonetheless manage his account via the CSC panel.

Copyright © Sipwise GmbH - All rights reserved

17.4.2. Site Customization

NOTE it is available on the old CSC UI only.

As an operator (as well as a Reseller), you can change the branding logo of the Customer Self-Care (CSC) panel and the available languages on the CSC panel. This is possible via the admin web interface.

Changing the Logo

NOTE it is available on the old CSC UI only.

For changing the branding logo on a reseller's admin web page and on the CSC panel you need to access the web interface **as Administrator** and navigate to *Reseller* menu. Once there click on the *Details* button for your selected reseller, finally select *Branding*.

In order to do the same **as Reseller**, login on the admin web interface with the reseller's web credentials, then access the *Panel Branding* menu.

The web panel customisation happens as follows:

1. Press the *Edit Branding* button to start the customisation process.
2. Press the *Browse* button to select an image for the new logo:

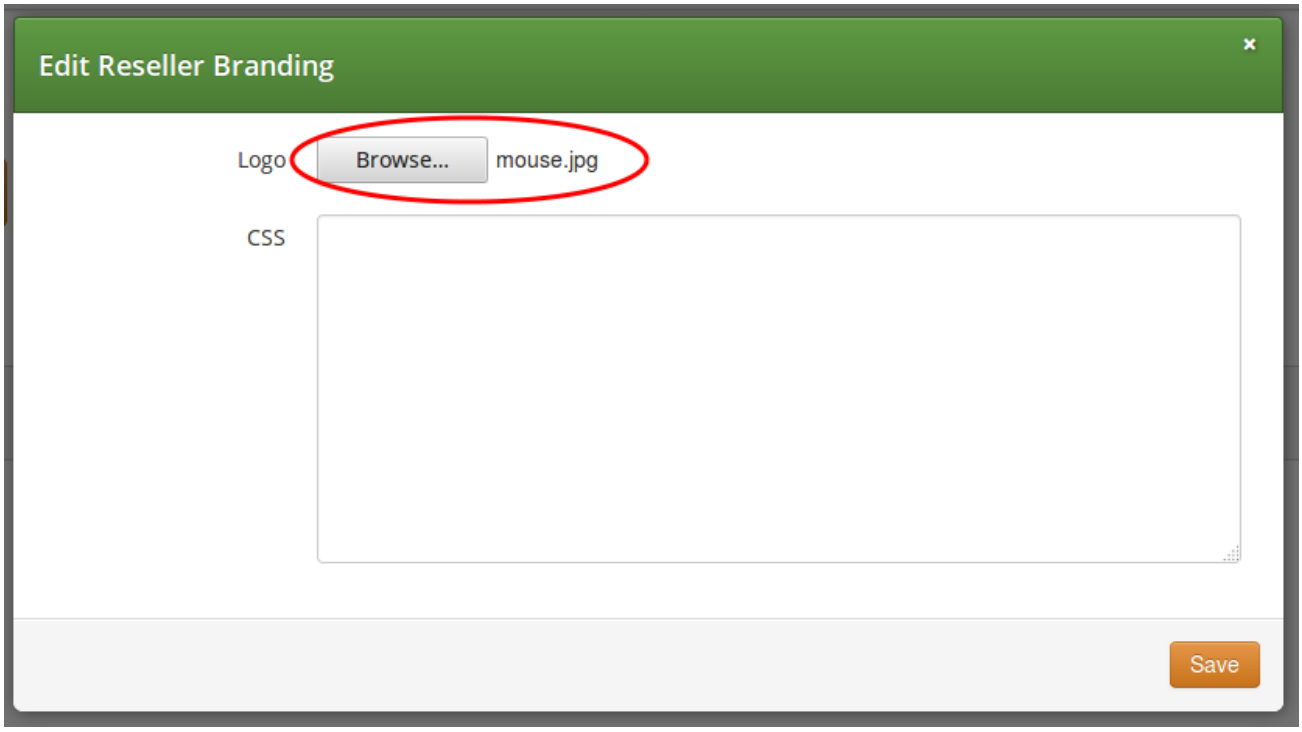


Figure 203. CSC Customisation Step 1: Select an image

3. Press the *Save* button to save changes.
4. Select and copy the auto-generated CSS code from the text box below the uploaded image:

Copyright © Sipwise GmbH - All rights reserved

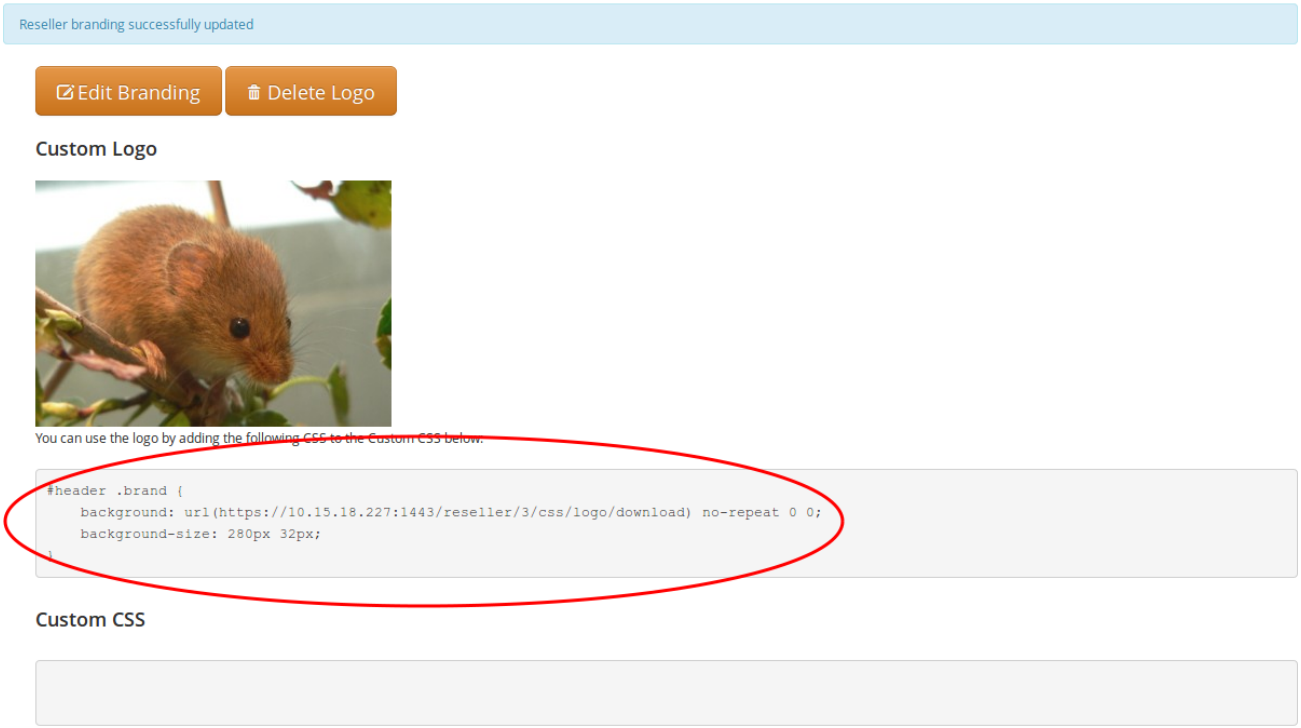


Figure 204. CSC Customisation Step 2: Copy CSS code

5. Press the *Edit Branding* button again.
6. Paste the CSS code into *CSS* text box and *Save* the changes:

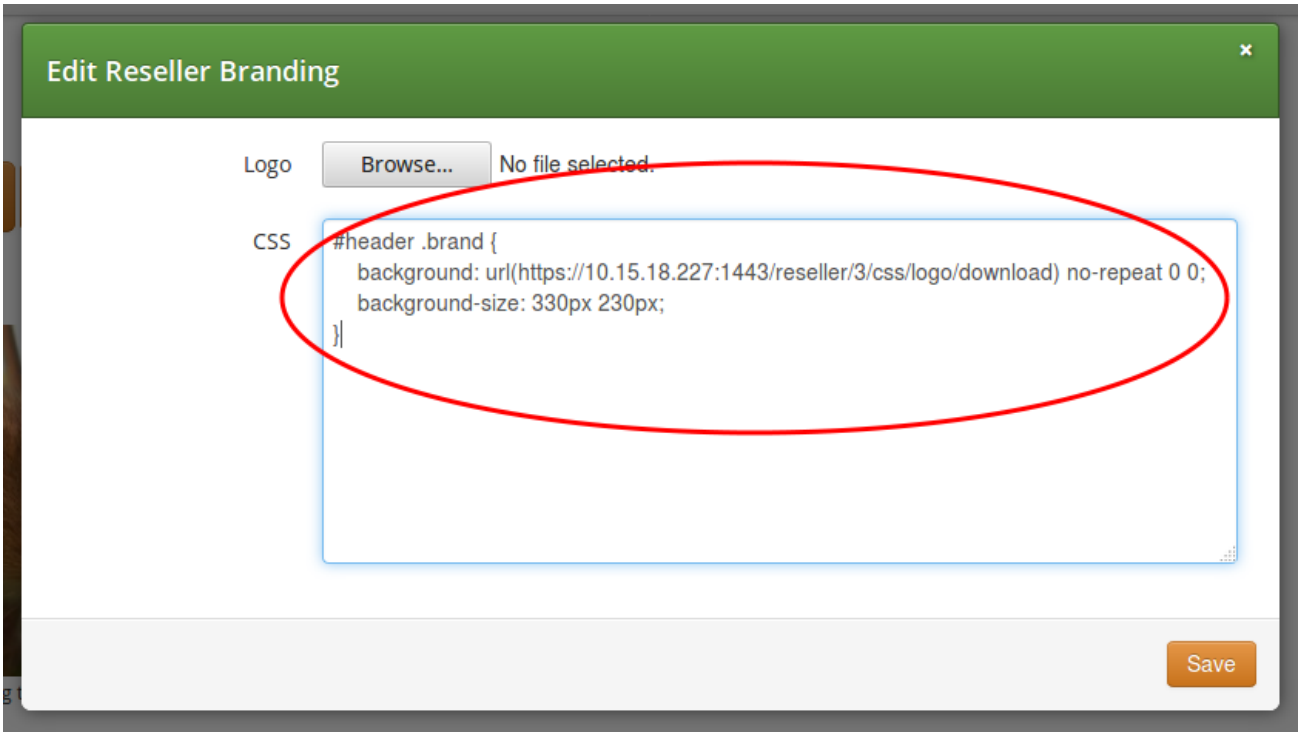


Figure 205. CSC Customisation Step 3: Paste CSS code

7. Now the new logo is already visible on the admin / CSC panel. If you want to hide the Sipwise copyright notice at the bottom of the web panels, add a line of CSS code as shown here:

Copyright © Sipwise GmbH - All rights reserved

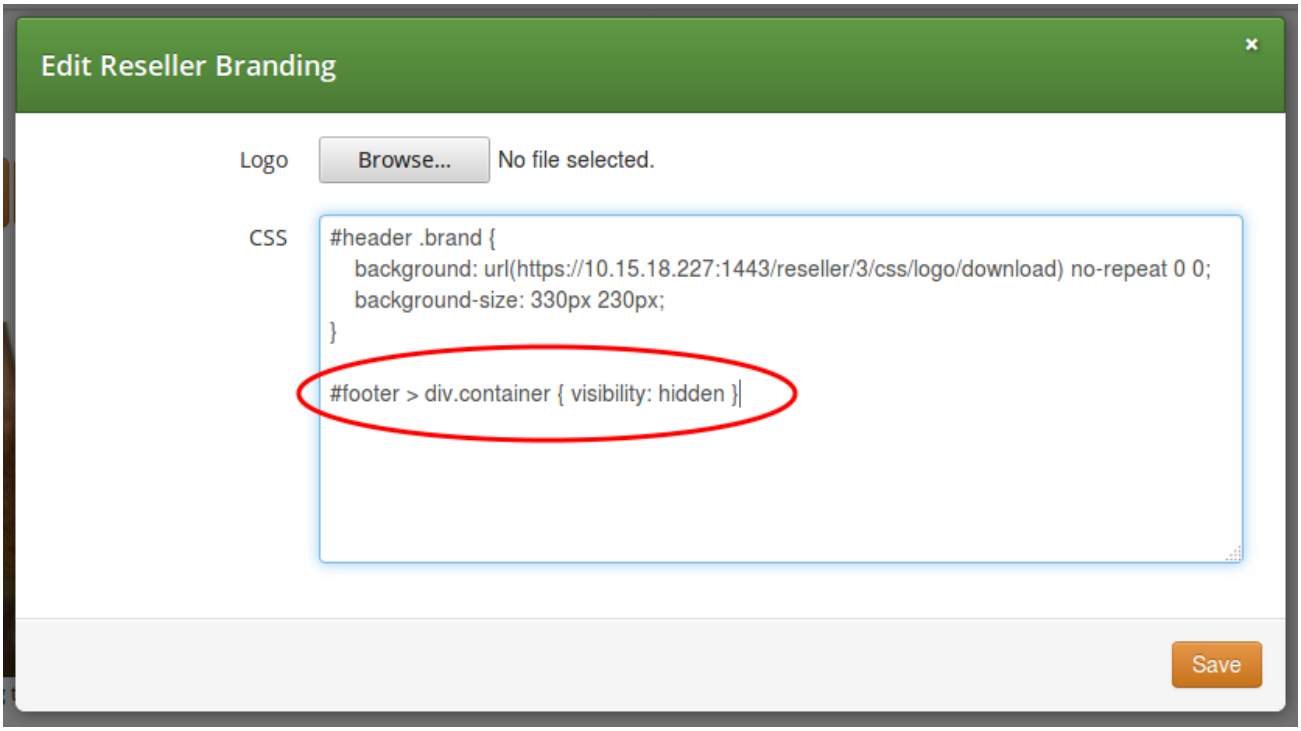


Figure 206. CSC Customisation: Hide copyright notice

8. The final branding data is shown on the admin web panel:



Figure 207. CSC Customisation: Custom data on panel

Copyright © Sipwise GmbH - All rights reserved

Other Website Customisations

NOTE it is available on the old CSC UI only.

The layout and style of NGCP's admin and CSC web panel is determined by a single CSS file: /usr/share/ngcp-panel/static/css/application.css

More complex changes, like replacing colour of some web panel components, is possible via the modification of the CSS file.

WARNING Only experienced users with profound CSS knowledge are advised to change web panel properties in the main CSS file. *Sipwise does not recommend and also does not support the modification of the main CSS file.*

Selecting Available Languages

You can also enable/disable specific languages a user can choose from in the CSC panel. Currently, English (en), German (de), Italian (it) and Spanish (es) are supported, and the default language is the same as the browser's preferred one.

You can select the *default language* provided by CSC by changing the parameter www_admin.force_language in /etc/ngcp-config/config.yml file. An example to set the English language as default:

```
ngcpcfg set /etc/ngcp-config/config.yml www_admin.force_language=en
ngcpcfg apply 'Set English as default on CSC'
ngcpcfg push-parallel all
```

17.5. The Voicemail Menu

Sipwise C5 offers several ways to access the Voicemail box.

The CSC panel allows your users to listen to voicemail messages from the web browser, delete them and call back the user who left the voice message. User can setup voicemail forwarding to the external email and the PIN code needed to access the voicebox from any telephone also from the CSC panel.

To manage the voice messages from SIP phone: dial internal voicemail access number `2000`.

To change the access number: look for the parameter *voicemail_number* in */etc/ngcp-config/config.yml* in the section *semsvsc*. After the changes, execute *ngcpcfg apply 'changed voicebox number'*.

TIP To let the callers leave a voice message when user is not available he should enable Call Forward to Voicebox. The Call Forward can be provisioned from the CSC panel as well as by dialing Call Forward VSC with the voicemail number. E.g. when parameter *voicemail_number* is set to *9999*, a Call Forward on Not Available to the Voicebox is set if the user dials `*93*9999`. As a result, all calls will be redirected to the Voicebox if SIP phone is not registered.

To manage the voice messages from any phone:

Copyright © Sipwise GmbH - All rights reserved

- As an operator, you can setup some DID number as external voicemail access number: for that, you should add a special rewrite rule (Inbound Rewrite Rule for Callee, see Configuring Rewrite Rule Sets.) on the incoming peer, to rewrite that DID to "voiceboxpass". Now when user calls this number the call will be forwarded to the voicemail server and he will be prompted for mailbox and password. The mailbox is the full E.164 number of the subscriber account and the password is the PIN set in the CSC panel.
- The user can also dial his own number from PSTN, if he setup Call Forward on Not Available to the Voicebox, and when reaching the voicemail server he can interrupt the "user is unavailable" message by pressing '*' key and then be prompted for the PIN. After entering PIN and confirming with '#' key he will enter own voicemail menu. PIN is random by default and must be kept secret for that reason.

17.6. Company Hours

NOTE it is available on the new CSC UI only.

The subsection "Company Hours" under the CSC UI "Call Forward" allows the user to specify Call Forwarding Rules for a specific period of time. The time defined in this subsection represents the office hours of the company. The first step is to define the actual days and the according times. Each time entry consists of day, start-time, and end-time. You can define multiple times for the same day, to get a different behavior with the breaks and the office hours. If the current times are not in one of the defined periods, the system falls back to the subsection "Always". Specific dates or date ranges can not be defined. The functionality is limited to weekdays and its timing.

The feature "Company Hours" is a virtual Sipwise C5 function which is based on Sipwise C5 core functionality Subscribers "Unconditional Call Forwarding" which is enhanced with the "Time sets" and "Sounds set" to achieve the necessary functionality.

The feature "Company Hours" can be managed using REST API methods CFDestinationSet, CFMapping, CFSourceSet and CFTimeSet. Please check all available options for those methods in a public REST API documentation.

Copyright © Sipwise GmbH - All rights reserved

Chapter 18. REST API

The Sipwise C5 provides the REST API interface for interconnection with 3rd party tools.

The Sipwise C5 provides a REST API to provision various functionality of the platform. The entry point - and at the same time the official documentation - is at *https://<your-ip>:1443/api*. It allows both administrators and resellers (in a limited scope) to manage the system.

You can either authenticate via username and password of your administrative account you're using to access the admin panel, or via SSL client certificates. Find out more about client certificate authentication in the online API documentation.

18.1. API Workflows for Customer and Subscriber Management

The typical tasks done on the API involve managing customers and subscribers. The following chapter focuses on creating, changing and deleting these resources.

The standard life cycle of a customer and subscriber is:

1. Create customer contact
2. Create customer
3. Create subscribers within customer
4. Modify subscribers
5. Modify subscriber preferences (features)
6. Terminate subscriber
7. Terminate customer

The boiler-plate to access the REST API is described in the online API documentation at */api/#auth*. A simple example in Perl using password authentication looks as follows:

```
#!/usr/bin/perl -w
use strict;
use v5.10;

use LWP::UserAgent;
use JSON qw();

my $uri = 'https://ngcp.example.com:1443';
my $ua = LWP::UserAgent->new;
my $user = 'myusername';
my $pass = 'mypassword';
$ua->credentials('ngcp.example.com:1443', 'api_admin_http', $user,
$pass);
my ($req, $res);
```

For each customer you create, you need to assign a billing profile id. You either have the ID stored

Copyright © Sipwise GmbH - All rights reserved

somewhere else, or you need to fetch it by searching for the billing profile handle.

```
my $billing_profile_handle = 'my_test_profile';
$req = HTTP::Request->new('GET',
"$uri/api/billingprofiles/?handle=$billing_profile_handle");
$res = $ua->request($req);
if($res->code != 200) {
    die "Failed to fetch billing profile: ".$res->decoded_content."\n";
}
my $billing_profile = JSON::from_json($res->decoded_content);
my $billing_profile_id = $billing_profile->{_embedded}-
>{'ngcp:billingprofiles'}->{id};
say "Fetched billing profile, id is $billing_profile_id";
```

A customer is mainly a billing container for subscribers without a real identification other than the *external_id* property you might have stored somewhere else (e.g. the ID of the customer in your CRM). To still easily identify a customer, a customer contact is required. It is created using the */api/customercontacts/* resource.

```
$req = HTTP::Request->new('POST', "$uri/api/customercontacts/");
$req->header('Content-Type' => 'application/json');
$req->content(JSON::to_json({
    firstname => 'John',
    lastname => 'Doe',
    email => 'john.doe\@example.com'
}));
$res = $ua->request($req);
if($res->code != 201) {
    die "Failed to create customer contact: ".$res-
>decoded_content."\n";
}
my $contact_id = $res->header('Location');
$contact_id =~ s/^.+\/(\d+)$/$1/; # extract the ID from the Location
header
say "Created customer contact, id is $contact_id";
```

IMPORTANT To get the ID of the recently created resource, you need to parse the *Location* header. In future, this approach will be changed for POST requests. The response will also optionally return the ID of the resource. It will be controlled via the *Prefer: return=representation* header as it is already the case for PUT and PATCH.

WARNING The example above implies the fact that you access the API via a reseller user. If you are accessing the API as the admin user, you also have to provide a *reseller_id* parameter defining the reseller this contact belongs to.

Once you have created the customer contact, you can create the actual customer.

Copyright © Sipwise GmbH - All rights reserved

```
$req = HTTP::Request->new('POST', "$uri/api/customers/");
$req->header('Content-Type' => 'application/json');
$req->content(JSON::to_json({
    status => 'active',
    contact_id => $contact_id,
    billing_profile_id => $billing_profile_id,
    type => 'sipaccount',
    external_id => undef, # can be set to your crm's customer id
}));
$res = $ua->request($req);
if($res->code != 201) {
    die "Failed to create customer: ".$res->decoded_content."\n";
}
my $customer_id = $res->header('Location');
$customer_id =~ s/^.+\/(\d+)$/$1/; # extract the ID from the Location
header
say "Created customer, id is $customer_id";
```

Once you have created the customer, you can add subscribers to it. One customer can hold multiple subscribers, up to the *max_subscribers* property which can be set via */api/customers/*. If this property is not defined, a virtually unlimited number of subscribers can be added.

Copyright © Sipwise GmbH - All rights reserved

```
$req = HTTP::Request->new('POST', "$uri/api/subscribers/");
$req->header('Content-Type' => 'application/json');
$req->content(JSON::to_json({
    status => 'active',
    customer_id => $customer_id,
    primary_number => { cc => 43, ac => 9876, sn => 10001 }, # the main
number
    alias_numbers => [ # as many alias numbers the subscriber can be
reached at (or skip param if none)
        { cc => 43, ac => 9877, sn => 10001 },
        { cc => 43, ac => 9878, sn => 10001 }
    ],
    username => 'test_10001'
    domain => 'ngcp.example.com',
    password => 'secret subscriber pass',
    webusername => 'test_10001',
    webpassword => undef, # set undef if subscriber shouldn't be able to
log into sipwise csc
    external_id => undef, # can be set to the operator crm's subscriber
id
}));
$res = $ua->request($req);
if($res->code != 201) {
    die "Failed to create subscriber: ".$res->decoded_content."\n";
}
my $subscriber_id = $res->header('Location');
$subscriber_id =~ s/^.+\/(\d+)$/$1/; # extract the ID from the Location
header
say "Created subscriber, id is $subscriber_id";
```

IMPORTANT A domain must exist before creating a subscriber. You can create the domain via */api/domains/*.

At that stage, the subscriber can connect both via SIP and XMPP, and can be reached via the primary number, all alias numbers, as well as via the SIP URI.

If you want to set call forwards for the subscribers, then perform an API call as follows.

Copyright © Sipwise GmbH - All rights reserved

```
$req = HTTP::Request->new('PUT',
"$uri/api/callforwards/$subscriber_id");
$req->header('Content-Type' => 'application/json');
$req->header('Prefer' => "return=minimal"); # use return=representation
to get full json response
$req->content(JSON::to_json({
    cfna => { # set a call-forward if subscriber is not registered
        destinations => [
            { destination => "4366610001", timeout => 10  }, # ring this
for 10s
            { destination => "4366710001", timeout => 300 }, # if no
answer, ring that for 300s
        ],
        times => undef # no time-based call-forward, trigger cfna always
    }
}));
$res = $ua->request($req);
if($res->code != 204) { # if return=representation, it's 200
    die "Failed to set cfna for subscriber: ".$res-
>decoded_content."\n";
}
```

You can set cfu, cfna, cfb, cft, cfs, cfr and cfo via this API call, also all at once. Destinations can be hunting lists as described above or just a single number. Also, a time set can be provided to trigger call forwards only during specific time periods.

To provision certain features of a subscriber, you can manipulate the subscriber preferences. You can find a full list of preferences available for a subscriber at */api/subscriberpreferencedefs/*.

Copyright © Sipwise GmbH - All rights reserved

```
$req = HTTP::Request->new('GET',
"$uri/api/subscriberpreferences/$subscriber_id");
$res = $ua->request($req);
if($res->code != 200) {
    die "Failed to fetch subscriber preferences: ".$res-
>decoded_content."\n";
}
my $prefs = JSON::from_json($res->decoded_content);
delete $prefs->{_links}; # not needed in update

$prefs->{prepaid_library} = 'libinewrate'; # switch to inew billing
$prefs->{block_in_clir} = JSON::true; # reject incoming anonymous calls
$prefs->{block_in_list} = [ # reject calls from the following numbers:
    '4366412345', # this particular number
    '431*', # all vienna/austria numbers
];
$req = HTTP::Request->new('PUT',
"$uri/api/subscriberpreferences/$subscriber_id");
$req->header('Content-Type' => 'application/json');
$req->header('Prefer' => "return=minimal"); # use return=representation
to get full json response
$req->content(JSON::to_json($prefs));
$res = $ua->request($req);
if($res->code != 204) {
    die "Failed to update subscriber preferences: ".$res-
>decoded_content."\n";
}
say "Updated subscriber preferences";
```

Modifying numbers assigned to a subscriber, changing the password, locking a subscriber, etc. can be done directly on the subscriber resource.

Copyright © Sipwise GmbH - All rights reserved

```
$req = HTTP::Request->new('GET', "$uri/api/subscribers/$subscriber_id");
$res = $ua->request($req);
if($res->code != 200) {
    die "Failed to fetch subscriber: ".$res->decoded_content."\n";
}
my $sub = JSON::from_json($res->decoded_content);
delete $sub->{_links}; # not needed in update
push @{ $sub->{alias_numbers} }, { cc => 1, ac => 5432, sn => $t }; #
add this number
push @{ $sub->{alias_numbers} }, { cc => 1, ac => 5433, sn => $t }; #
add another number

$req = HTTP::Request->new('PUT', "$uri/api/subscribers/$subscriber_id");
$req->header('Content-Type' => 'application/json');
$req->header('Prefer' => "return=minimal"); # use return=representation
to get full json response
$req->content(JSON::to_json($sub));
$res = $ua->request($req);
if($res->code != 204) {
    die "Failed to update subscriber: ".$res->decoded_content."\n";
}
say "Updated subscriber";
```

At the end of a subscriber life cycle, it can be terminated. Once terminated, you can NOT recover the subscriber anymore.

```
$req = HTTP::Request->new('DELETE',
"$uri/api/subscribers/$subscriber_id");
$res = $ua->request($req);
if($res->code != 204) {
    die "Failed to terminate subscriber: ".$res->decoded_content."\n";
}
say "Terminated subscriber";
```

Note that certain information is still available in the internal database to perform billing/rating of calls done by this subscriber. Nevertheless, the data is removed from the operational tables of the database, so the subscriber is not able to connect to the system, login or make calls/chats.

Resources modification can be done via the GET/PUT combination. Alternatively, you can add, modify or delete single properties of a resource without actually fetching the whole resource. See an example below where we terminate the status of a customer using the PATCH method.

Copyright © Sipwise GmbH - All rights reserved

```
$req = HTTP::Request->new('PATCH', "$uri/api/customers/$customer_id");
$req->header('Content-Type' => 'application/json-patch+json');
$req->header('Prefer' => "return=minimal"); # use return=representation
to get full json response
$req->content(JSON::to_json([
    { op => 'replace', path => '/status', value => 'terminated' }
]));
$res = $ua->request($req); # this will also terminate all still active
subscribers
if($res->code != 204) {
    die "Failed to terminate customer: ".$res->decoded_content."\n";
}
say "Terminated customer";
```

18.2. API performance considerations

The REST API is designed with pagination support built-in. It is mandatory, to implement pagination in your API clients. If you circumvent pagination by setting the number of rows requested in one API call to a very high number the following side effects may appear:

1. An HTTP timeout at the gateway may occur. The default timeout limit is set to 60s. You can change it by creating a patchtt file for the following template: */etc/ngcp-config/templates/etc/nginx/sites-available/ngcp-panel_admin_api.tt2*.
2. Other parts of the system might become unresponsive due to mysql table locks. This especially applies to endpoints related to the Customers entity.

Copyright © Sipwise GmbH - All rights reserved

Appendix

Copyright © Sipwise GmbH - All rights reserved

Appendix A: Corosync/Pacemaker

The Corosync/Pacemaker pair is the successor of the long obsolete and unsupported Heartbeat v2 software package. While Heartbeat v2 was playing the role of both the **Group Communication System** (GCS) and the **Cluster Resource Manager** (CRM), these roles are split under the new system. Corosync is the GCS and in charge of communication, while Pacemaker sits on top of the GCS and plays the role of the CRM, managing the resources and responding to changes in the cluster status.

Migration

IMPORTANT Starting with Sipwise C5 mr10.5 only Corosync/Pacemaker is supported as the High Availability system. The migration should happen before upgrading to that release.

Rollback

IMPORTANT Because only Corosync/Pacemaker is supported, no rollback is available any longer. The only option is to downgrade to an Sipwise C5 release that still supports Heartbeat v2.

Corosync

Corosync is the **Group Communication System** (GCS). Its configuration resides in `/etc/corosync/corosync.conf` and describes the following details:

- Shared cluster name of `sp`
- Quorum config as a two-node cluster (see below)
- Config details for both nodes:

  Name (`sp1` and `sp2`, or `a` and `b` node names)

  Node ID (`1` and `2` respectively)

  Local IP address for communication

Quorum

Corosync uses a voting system to determine the state of the cluster. Each configured node in the cluster receives one vote. A quorum is defined as a majority presence within the cluster, meaning at least 50% of the configured nodes plus one, or `q = n / 2 + 1`. For example, if 8 nodes were configured, a quorum would be present if at least 5 nodes are communicating with each other. In this state, the cluster is said to be quorate, which means it can operate normally. (Any remaining nodes, 3 in the worst case, would see the cluster as inquorate and would relinquish all their resources.)

A two-node cluster is a special case as under the formula above, a quorum would consist of 2 functioning nodes. The Corosync config setting `two_node: 1` overrides this and artificially sets the quorum to 1. This means that under a split-brain scenario (each node seeing only 1 vote), both nodes would see the cluster as quorate and try to become active, instead of both nodes going standby.

In addition to this, Pacemaker itself also uses an internal scoring system for individual resources. This mechanism is described below and not directly related to the quorum.

Copyright © Sipwise GmbH - All rights reserved

Pacemaker

Pacemaker uses the communication service provided by Corosync to manage local resources. All status and configuration information is shared between all Pacemaker instances within the cluster as long as communication is up. This means that any configuration change done on any node will immediately and automatically be propagated to all other nodes in the cluster.

Pacemaker internally uses an XML document to store its configuration, called "CIB" stored in `/var/lib/pacemaker/cib/cib.xml`. However, this XML document **must never** be edited or modified directly. Instead, a shell-like interface `crm` is provided to talk to Pacemaker, query status information, alter cluster state, view and modify configuration, etc. Any configuration change done through `crm` is immediately reflected in the CIB XML, locally as well as on all other nodes.

WARNING To repeat, do not ever directly modify Pacemaker's XML configuration.

As an added bonus, to make things more awkward, the syntax used by `crm` is not XML at all, but rather uses a Cisco-like hierarchy.

Commands can be issued to `crm` either directly from the shell as command-line arguments, or interactively by entering a Cisco-like shell. So for example, the current config can be viewed either from the shell with:

```
root@sp1:~# crm config show
...
```

Or interactively, as either:

```
root@sp1:~# crm
crm(live/sp1)# config
crm(live/sp1)configure# show
...
```

or

```
root@sp1:~# crm
crm(live/sp1)# config show
...
```

Interactive online help is provided by the `ls` command to list commands valid in the current context, or using the `help` command for a more verbose help output.

Query Status

The current cluster status can be viewed with the top-level `status` command:

Copyright © Sipwise GmbH - All rights reserved

```
crm(live/sp1)# status
Stack: corosync
Current DC: sp2 (version unknown) - partition with quorum
Last updated: Fri Nov 22 18:38:06 2019
Last change: Fri Nov 22 18:25:28 2019 by hacluster via crmd on sp1
```

```
2 nodes configured
7 resources configured
```

```
Online: [ sp1 sp2 ]
```

```
Full list of resources:
```

```
Resource Group: g_vips
    p_vip_eth1_v4_1 (ocf::heartbeat:IPaddr):    Started sp1
    p_vip_eth2_v4_1 (ocf::heartbeat:IPaddr):    Started sp1
Resource Group: g_ngcp
    p_monit_services    (ocf::ngcp:monit-services): Started sp1
Clone Set: c_ping [p_ping]
    Started: [ sp1 sp2 ]
Clone Set: fencing [st-null]
    Started: [ sp1 sp2 ]
```

If the status is queried from `sp2` instead, the output will be the same. Most importantly, the resources will **not** show up as "stopped" on `sp2` but instead will be reported as running on `sp1`.

The resources reported are described in the configuration section below.

Config Management

The NGCP templates do not operate on Pacemaker's CIB XML directly, but instead produce a file in CRM syntax in `/etc/pacemaker/cluster.crm`. This file is not handled by Pacemaker directly, but instead is loaded into Pacemaker via the `crm` command `config load replace`. It shouldn't be necessary to do this manually, as the script `ngcp-ha-crm-reload` handles this automatically, which is called from the config file's postbuild script.

Changes to the config don't need to be saved explicitly. This is done automatically by Pacemaker, as well as sharing any changes with all other members of the cluster.

In `crm`, changes made to the config are cached until made active with `commit`, or discarded with `refresh`. Changes to resource status can be avoided by enabling maintenance mode (see below).

Copyright © Sipwise GmbH - All rights reserved

IMPORTANT	However, since our config is loaded from a template, any changes done to the config through `crm` manually are transient and will be lost the next time a config reload happens.

The currently active config can be shown with `config show` and should be logically identical to the contents of `/etc/pacemaker/cluster.crm`:

Copyright © Sipwise GmbH - All rights reserved

```
crm(live/sp1)# config show
node 1: sp1
node 2: sp2
primitive p_monit_services ocf:ngcp:monit-services \
    meta migration-threshold=20 \
    meta failure-timeout=800 \
    op monitor interval=20 timeout=60 on-fail=restart \
    op_params on-fail=restart
primitive p_ping ocf:pacemaker:ping \
    params host_list="10.15.20.30 192.168.211.1" multiplier=1000
dampen=5s \
    meta failure-timeout=800 \
    op monitor interval=1 timeout=60 on-fail=restart \
    op_params timeout=60 on-fail=restart
primitive p_vip_eth1_v4_1 IPaddr \
    params ip=192.168.255.250 nic=eth1 cidr_netmask=24 \
    op monitor interval=5 timeout=60 on-fail=restart \
    op_params on-fail=restart
primitive p_vip_eth2_v4_1 IPaddr \
    params ip=192.168.1.161 nic=eth2 cidr_netmask=24 \
    op monitor interval=5 timeout=60 on-fail=restart \
    op_params on-fail=restart
primitive st-null stonith:null \
    params hostlist="sp1 sp2"
group g_ngcp p_monit_services
group g_vips p_vip_eth1_v4_1 p_vip_eth2_v4_1
clone c_ping p_ping
clone fencing st-null
location l_ngcp g_ngcp \
    rule pingd: defined pingd
colocation l_ngcp_with_vip inf: g_ngcp g_vips
location l_vips g_vips \
    rule pingd: defined pingd
order o_vip_then_ngcp Mandatory: g_vips g_ngcp
property cib-bootstrap-options: \
    have-watchdog=false \
    cluster-infrastructure=corosync \
    cluster-name=sp \
    stonith-enabled=yes \
    no-quorum-policy=ignore \
    startup-fencing=yes \
    maintenance-mode=false \
    last-lrm-refresh=1574443528
rsc_defaults rsc-options: \
    resource-stickiness=100
```

General Concepts

- The configuration consists of a collection of objects of various types with various attributes.
- Each object has a unique identifying name that can be used to refer to it.

Copyright © Sipwise GmbH - All rights reserved

- Usually the type of the object is the first word and the identifying name is the second. For example, `clone c_ping p_ping` defines a `clone` type object with the name `c_ping`.
- The unique name is used e.g. when deleting an object (`config del …`), when starting or stopping a resource, when referring to resources from a group, etc.

Resources

Resources are the primary type of objects that Pacemaker handles. A resource is anything that can be started or stopped, and a resource is normally allowed to run on one node only. A resource is defined as a `primitive` type object.

Pacemaker supports many types of resources, all of which have different options that can be given to them. The config syntax defines that options given to a resource itself are prefixed with `params`, while options that influence how a resource should be managed are prefixed with `meta`. Options that are relevant to operations that can be performed on a resource are prefixed with `op`.

Resources are grouped into classes, providers, and types. Details about them (e.g. which options they support) can be obtained through the `ra` menu.

```
crm(live/sp1)ra# info IPaddr
Manages virtual IPv4 addresses (portable version) (ocf:heartbeat:IPaddr)
...
```

Shared IP Addresses

```
primitive p_vip_eth1_v4_1 IPaddr \
    params ip=192.168.255.250 nic=eth1 cidr_netmask=24 \
    op monitor interval=5 timeout=60 on-fail=restart \
    op_params on-fail=restart
```

This defines a resource of type `IPaddr` with name `p_vip_eth1_v4_1` and the given parameters (address, netmask, interface). Pacemaker will check for the existence of the address every 5 seconds, with an action timeout of 60 seconds. If the monitor action fails, the resource is restarted.

System Services

```
primitive p_monit_services ocf:ngcp:monit-services \
    meta migration-threshold=20 \
    meta failure-timeout=800 \
    op monitor interval=20 timeout=60 on-fail=restart \
    op_params on-fail=restart
```

While Pacemaker has support for native systemd services, for the time being we're still relying on monit to manage our services. Therefore, services are defined in Pacemaker virtually identical to how they were defined in Heartbeat v2, through a `monit-services` start/stop script. The old Heartbeat script was `/etc/ha.d/resource.d/monit-services` and the new script used by Pacemaker is `/etc/ngcp-ocf/monit-services`.

Copyright © Sipwise GmbH - All rights reserved

NOTE

The primary difference between the two scripts is the support for a `monitor` action for Pacemaker, which can be configured via the `config.yml` variable `ha.monitor_services`. It can be set to 'full' to periodically use the output of `ngcp-service summary` to determine whether all services are running or not. The default value is 'none', which preserves backwards compatibility with the behavior of Heartbeat v2, by performing no periodic checks of the status of the services.

- `meta migration-threshold=20` means that the resource will be migrated away (instead of restarted) after 20 failures. See the discussion on failure counts below.
- `meta failure-timeout=800` means that the failure count should be reset to zero if the last failure occurred more than 800 seconds ago. (However, the actual timer depends on the `cluster-recheck-interval`.)
- Run the monitor action every 20 seconds with a timeout of 60 seconds and restart the resource on failure.

Ping Nodes

```
primitive p_ping ocf:pacemaker:ping \
    params host_list="10.15.20.30 192.168.211.1" multiplier=1000
dampen=5s \
    meta failure-timeout=800 \
    op monitor interval=1 timeout=60 on-fail=restart \
    op_params timeout=60 on-fail=restart
```

The builtin `pingd` service, using a resource name that intelligently is not named `pingd` but rather `ping`, replaces Heartbeat's ping nodes. It supports multiple ping backends, and uses `fping` by default.

Each configured ping node (each entry in `host_list`) produces a score of 1 if that ping node is up. The scores are summed up and multiplied by the `multiplier`. So in the example above, a score of 2000 is generated if both ping nodes are up. Pacemaker will then prefer the node which produces the higher score.

- `dampen=5s` means to wait 5 seconds after a change occurred to prevent transient glitches from causing service flapping.
- Other options are the same as described above.

Fencing/STONITH

```
primitive st-null stonith:null \
    params hostlist="sp1 sp2"
```

Pacemaker will generate a warning if no fencing mechanism is configured, therefore we configure the `null` fencing mechanism.

Pacemaker supports several proper fencing mechanism and these might eventually get supported in the future.

Copyright © Sipwise GmbH - All rights reserved

Groups

```
group g_ngcp p_monit_services
group g_vips p_vip_eth1_v4_1 p_vip_eth2_v4_1
```

To manage, control, and restrict multiple resources at the same time, resources can be grouped into single objects. The group `g_ngcp` is pointless for the time being (it contains only a single other resource) but will become useful once native systemd resources are in use. The group `g_vips` ensures that all shared IP addresses are active at the same time.

Clones

```
clone c_ping p_ping
clone fencing st-null
```

Since a single resource normally only runs on one node, a clone can be defined to allow a resource to run on all nodes. We want the `pingd` service and the fencing service to always run on all nodes.

Constraints

```
colocation l_ngcp_with_vip inf: g_ngcp g_vips
```

This tells Pacemaker that we want to force the `g_ngcp` resource on the same node that is running the `g_vips` resource.

```
location l_ngcp g_ngcp \
    rule pingd: defined pingd
location l_vips g_vips \
    rule pingd: defined pingd
```

This tells Pacemaker that these resources depend on the `pingd` service being healthy. If `pingd` fails on one node (ping nodes are unavailable), then Pacemaker will shut down the constrained resources.

```
order o_vip_then_ngcp Mandatory: g_vips g_ngcp
```

This tells Pacemaker that the shared IP addresses must be up and running before the system services can be started.

Cluster Options

Copyright © Sipwise GmbH - All rights reserved

```
property cib-bootstrap-options: \
    have-watchdog=false \
    cluster-infrastructure=corosync \
    cluster-name=sp \
    stonith-enabled=yes \
    no-quorum-policy=ignore \
    startup-fencing=yes \
    maintenance-mode=false \
    last-lrm-refresh=1574443528
```

Relevant options are:

- `have-watchdog=false` indicates that no external watchdog service such as `SBD` is in use.
- `cluster-name=sp` is to match the configuration of Corosync.
- `stonith-enabled=yes` is required to suppress a warning message, even though no real STONITH (`null` fencing mechanism) is in use.
- `no-quorum-policy=ignore` tells Pacemaker to continue normally if quorum is lost. This is the only setting that makes sense in a two-node cluster.
- `startup-fencing=yes` is also needed to suppress a warning even though no real fencing is in use. This tells Pacemaker to shoot nodes that are not present immediately after startup.
- `maintenance-mode=false` tells Pacemaker to actually perform resource actions. If maintenance mode is enabled, Pacemaker will continue to run, but will not start or stop any services. This should be enabled before loading a new config, and then disabled afterwards. The script `ngcp-ha-crm-reload` does this.

Failure Counts

Pacemaker keeps a failure count for each resource, which is somewhat hidden from view, but can largely influence its behaviour. Each time a service fails (either during runtime or during startup), the failure count is increased by one. If the failure count exceeds the configured `migration-threshold`, Pacemaker will cease trying to start the service and will migrate the service away to another node. In `crm status` this shows up as `stopped`.

Failure counts can be cleared automatically if the `failure-timeout` setting is configured for a resource. This timeout is counted after the last time the resource has failed, and is checked periodically according to the `cluster-recheck-interval`. In other words, a very short failure timeout won't have any effect unless the recheck interval is also very short.

IMPORTANT	If no faiure timeout is configured, any existing failure count must be cleared manually.

Checking Failure Counts

The failure count for a resource can be checked from the shell via `crm_failcount`, for example:

```
root@sp1:~# crm_failcount -G -r p_monit_services
scope=status  name=fail-count-p_monit_services value=0
```

Copyright © Sipwise GmbH - All rights reserved

The failure count on a different node can also be examined:

```
root@sp1:~# crm_failcount -G -r g_ngcp -N sp2
scope=status  name=fail-count-g_ngcp value=0
```

The same can be done via `crm`:

```
crm(live/sp1)resource# failcount g_vips show sp1
scope=status  name=fail-count-g_vips value=0
crm(live/sp1)resource# failcount c_ping show sp2
scope=status  name=fail-count-c_ping value=0
```

As a shortcut, the script `ngcp-ha-show-failcounts` is provided:

```
root@sp1:~# ngcp-ha-show-failcounts
p_vip_eth1_v4_1:
  sp1: 0
  sp2: 0
p_vip_eth2_v4_1:
  sp1: 0
  sp2: 0
p_monit_services:
  sp1: 0
  sp2: 0
```

Clearing Failure Counts

Analogous to checking a failure count, it can be cleared using any one of these methods:

```
root@sp1:~# crm_failcount -D -r p_monit_services
Cleaned up p_monit_services on sp1
root@sp1:~# crm resource failcount g_ngcp delete sp2
Cleaned up p_monit_services on sp2
root@sp1:~# crm
crm(live/sp1)# resource
crm(live/sp1)resource# failcount c_ping delete sp1
Cleaned up p_ping:0 on sp1
Cleaned up p_ping:1 on sp1
crm(live/sp1)resource# bye
root@sp1:~# ngcp-ha-clear-failcounts
Cleaned up p_monit_services on sp2
Cleaned up p_monit_services on sp1
```

In addition, the `crm` command `resource cleanup` also resets failure counts.

Copyright © Sipwise GmbH - All rights reserved

Resource Scores

Pacemaker uses an internal scoring system to determine which resources to run where. A resource will be run on the node on which it received the highest score. If a resource has a negative score, that resource will not be run at all. If a resource has the same score on multiple nodes, then the resource will be run on any one of those nodes. Scores can be calculated and acted upon through various config settings.

A score value of `infinity` (and negative infinity) to force certain states is provided, which evaluates to not infinity at all, but rather to a static value of one million. This can be used to artificially manipulate resource scores to force running a resource on a particular node, or forbid a resource from running on particular nodes.

Scores can be inspected through the `crm` command `resource scores`.

Common Tasks

Takeover and Standby

The commands `ngcp-make-active` and `ngcp-make-standby` work normally. Under Pacemaker, they function through the `crm` command `resource move` to create a temporary location constraint on `g_vips`. This can be done manually through:

```
crm resource move g_vips sp1 30
```

The lifetime of 30 seconds is needed because `g_ngcp` depends on the location of `g_vips`, and therefore `g_ngcp` needs to be stopped before `g_vips` can be stopped. The location constraint must remain active until `g_ngcp` has been completely and successfully stopped.

NOTE These commands only effect the status of the running resources, and not the status of the node itself. This means that after going standby, Pacemaker will immediately be ready to take over the resources again if needed. See below for a discussion on node status.

Similarly, `ngcp-check-active` uses the output of `crm resource status g_vips` to determine whether the local node is active or not.

Node Status (Online/Standby)

In addition to the status and location of individual resources, nodes themselves can also go into standby mode. The submenu `node` in `crm` has the relevant options.

A node in standby mode will not only give up all of its resources, but will also refuse to take them over until it's back online. Therefore, it's possible to set both nodes to standby mode and shut down all resources on both nodes.

NOTE A node in standby mode will still participate in GCS communications and remain visible to the rest of the cluster.

Use `crm node standby` to set the local node to standby mode. A remote node can be set to standby

Copyright © Sipwise GmbH - All rights reserved

using e.g. `crm node standby sp2`.

By default, the standby mode remains active until it's cancelled manually (a lifetime of `forever`). Alternatively, a lifetime of `reboot` can be specified to tell Pacemaker that after the next reboot, the node should automatically come back online. Example: `crm node standby sp2 reboot`

To cancel standby mode, use `crm node online`, optionally followed by the node name.

To show the current status of all nodes, use `crm node show`. The top-level `crm status` also shows this.

Maintenance Mode

If Pacemaker's maintenance mode is enabled, it will continue to operate normally, i.e. continue to run and monitor resources, but will refuse to stop or start any resources. This is useful to make changes to the running config, and is done automatically by `ngcp-ha-crm-reload`.

To enable and disable maintenance mode:

```
crm maintenance on
crm maintenance off
```

or using the more lower level method:

```
crm configure property maintenance-mode=true
crm configure property maintenance-mode=false
```

CLI Alternatives

Several of the commands available through `crm` are also available through standalone CLI tools, such as `crm_failcount`, `crm_standby`, `crm_resource`, etc. They generally have less friendly syntax and so researching them is left as an exercise to the reader.

Copyright © Sipwise GmbH - All rights reserved

Appendix B: Basic Call Flows

General Call Setup

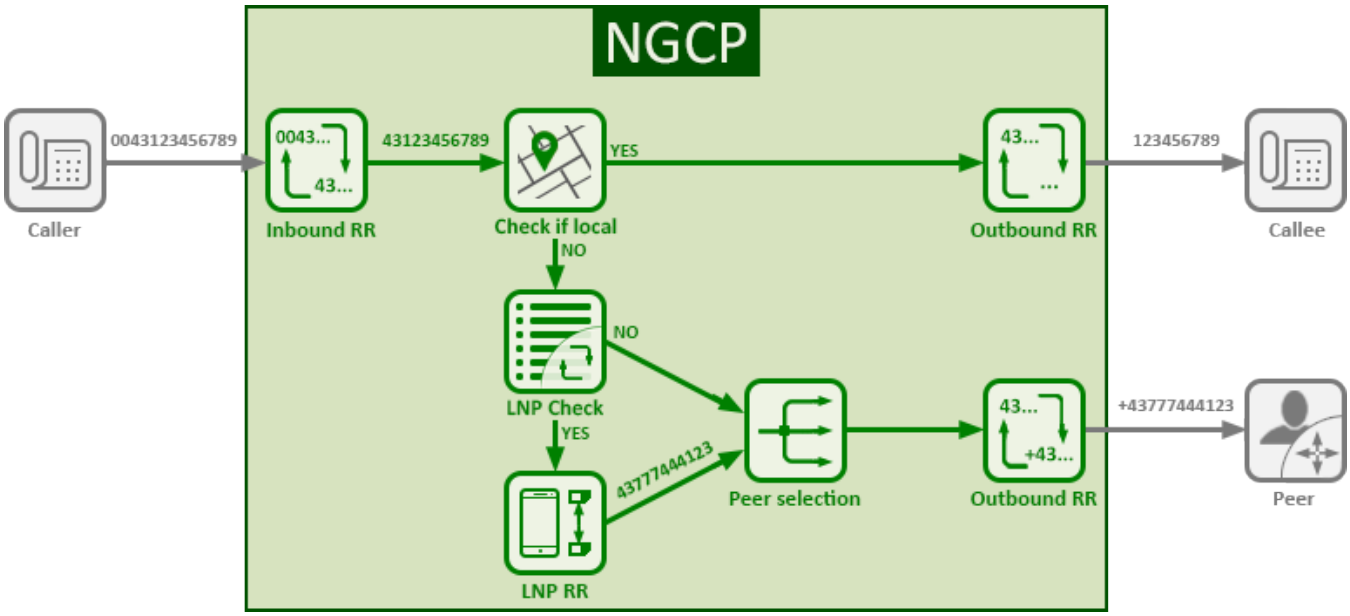


Figure 208. General Call Setup

Sipwise C5 performs the following checks when processing a call coming from a subscriber and terminated at a peer:

- Checks if the IP address where the request came from is in the list of trusted IP addresses. If yes, this IP address is taken as the identity for authentication. Otherwise, Sipwise C5 performs the digest authentication.
- When the subscriber is authorized to make the call, Sipwise C5 applies the Inbound Rewrite Rules for the caller and the callee assigned to the subscriber (if any). If there are no Rewrite Rules assigned to the subscriber, the ones assigned to the subscriber's domain are applied. On this stage the platform normalises the numbers from the subscriber's format to E.164.
- Matches the callee (called number) with local subscribers.

  If it finds a matching subscriber, the call is routed internally. In this case, Sipwise C5 applies the Outbound Rewrite Rules associated with the callee (if any). If there are no Rewrite Rules assigned to the callee, the ones assigned to the callee's domain are applied.

  If it does not find a matching subscriber, the call goes to a peer as described below.
- Queries the LNP database to find out if the number was ported or not.For details of LNP queries refer to the Local Number Porting chapter.

  If it was ported, Sipwise C5 applies the LNP Rewrite Rules to the called number.
- Based on the priorities of peering groups and peering rules (see Routing Order Selection for details), Sipwise C5 selects peering groups for call termination and defines their precedence.
- Within every peering group the weight of a peering server defines its probability to receive the call for termination. Thus, the bigger the weight of a server, the higher the probability that Sipwise C5 will send the call to it.
- Applies the Outbound Rewrite Rules for the caller and the callee assigned to a peering server when

Copyright © Sipwise GmbH - All rights reserved

sending the call to it.

Endpoint Registration

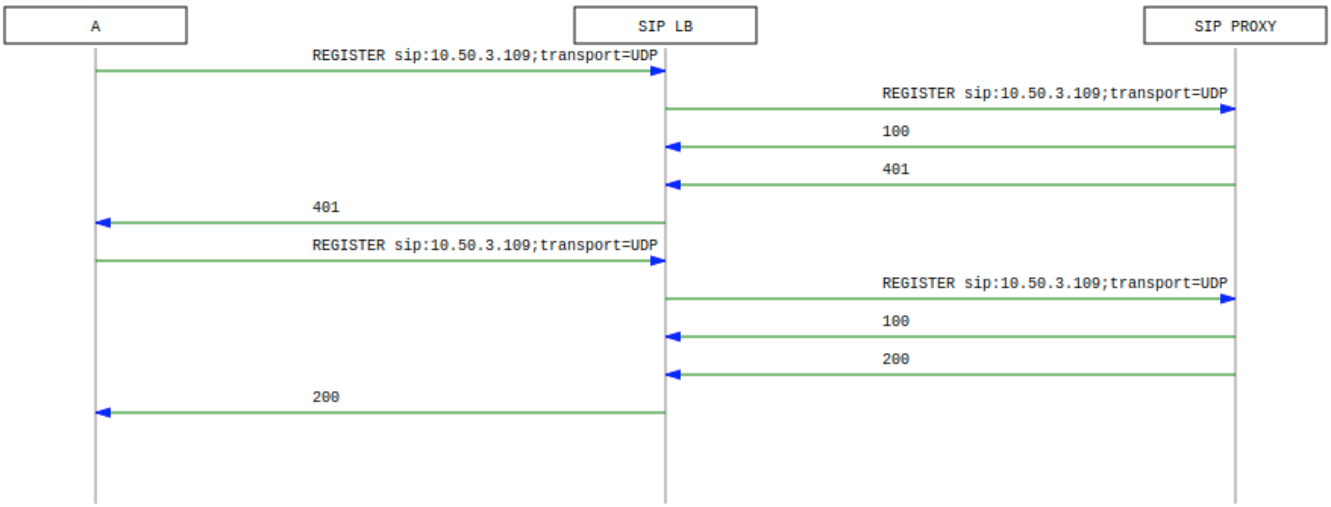


Figure 209. Registration Call-Flow

The subscriber endpoint starts sending a REGISTER request, which gets challenged by a 401. After calculating the response of the authentication challenge, it sends the REGISTER again, including the authentication response. The SIP proxy looks up the credentials of the subscriber in the database,does the same calculation, and if the result matches the one from the subscriber, the registration is granted.

The SIP proxy writes the content of the Contact header (e.g. sip:me@1.2.3.4:1234;transport=UDP) into its location table (in case of NAT the content is changed by the SIP load-balancer to the IP/port from where the request was received), so it knows where to reach a subscriber in case on an inbound call to this subscriber (e.g. sip:someuser@example.org is mapped to sip:me@1.2.3.4:1234;transport=UDP and sent out to this address).

If NAT is detected, the SIP proxy sends a OPTIONS message to the registered contact every 30 seconds, in order to keep the NAT binding on the NAT device open. Otherwise, for subsequent calls to this contact, Sipwise C5 wouldn't be able to reach the endpoint behind NAT (NAT devices usually drop a UDP binding after not receiving any traffic for ~30-60 seconds).

By default, a subscriber can register 5 contacts for an Address of Record (AoR, e.g. sip:someuser@example.org).

Basic Call

 Copyright © Sipwise GmbH - All rights reserved

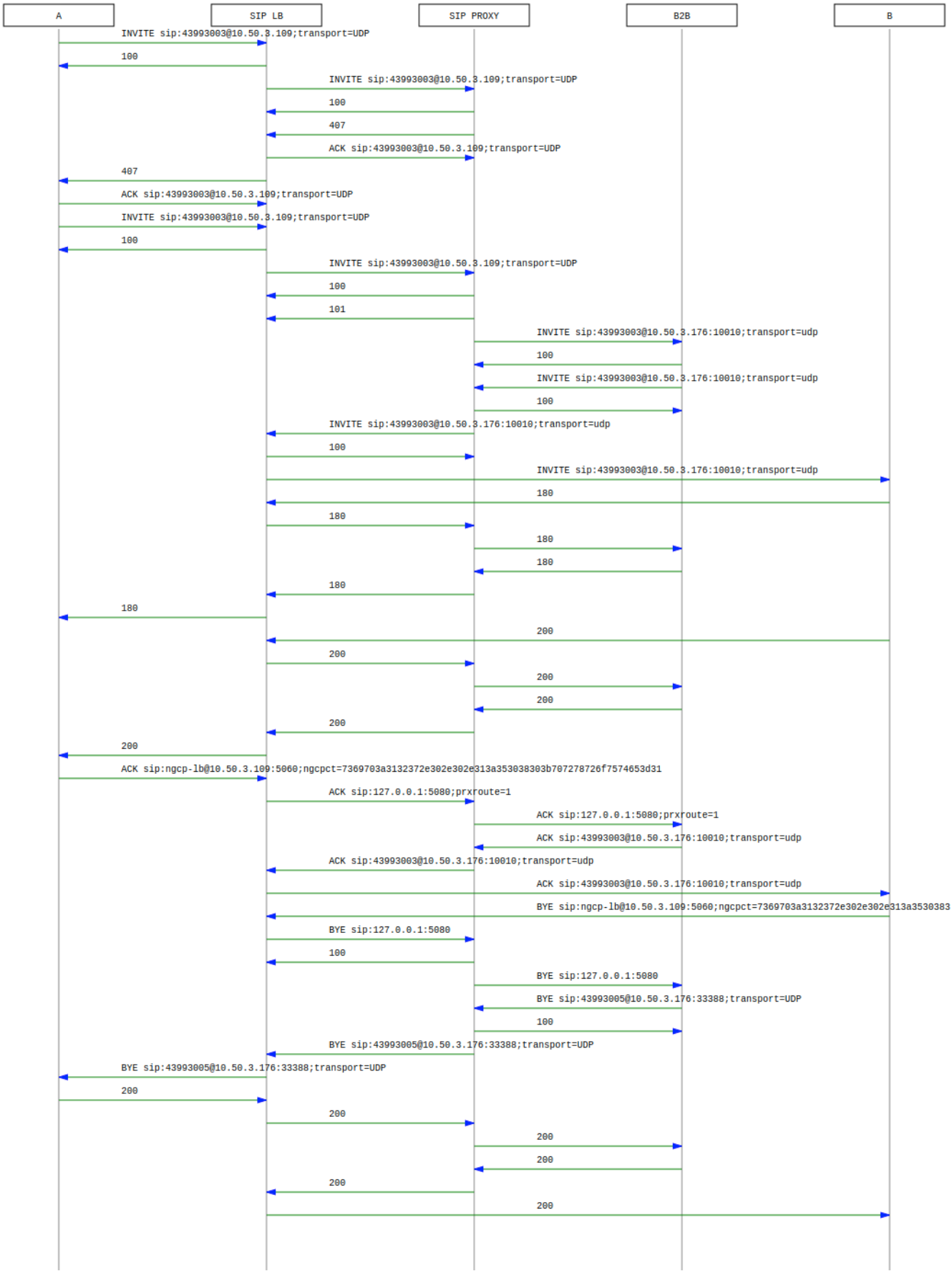


Figure 210. Basic Call Call-Flow

The calling party sends an INVITE (e.g. sip:someuser@example.org) via the SIP load-balancer to the SIP

Copyright © Sipwise GmbH - All rights reserved

proxy. The proxy replies with an authorization challenge in the 407 response, and the calling party sends the INVITE again with authentication credentials. The SIP proxy checks if the called party is a local user. If it is, and if there is a registered contact found for this user, then (after various feature-related tasks for both the caller and the callee) the Request-URI is replaced by the URI of the registered contact (e.g. sip:me@1.2.3.4:1234;transport=UDP). If it's not a local user but a numeric user, a proper PSTN gateway is being selected by the SIP proxy, and the Request-URI is rewritten accordingly (e.g. sip:+43123456789@2.3.4.5:5060).

Once the proxy has finished working through the call features of both parties involved and has selected the final destination for the call, and - optionally - has invoked the Media Relay for this call, the INVITE is sent to the SIP B2BUA. The B2BUA creates a new INVITE message from scratch (using a new Call-ID and a new From-Tag), copies only various and explicitly allowed SIP headers from the old message to the new one, filters out unwanted media capabilities from the SDP body (e.g. to force audio calls to use G.711 as a codec) and then sends the new message back to the SIP proxy that forwards it to the SIP load-balancer to reach to the called party.

SIP replies from the called party are passed through the elements back to the calling party (replacing various fields on the B2BUA to match the first call leg again). If a reply with an SDP body is received by the SIP proxy (e.g. a 183 or a 200), the Media Relay is invoked again to prepare the ports for the media stream.

Once the 200OK is routed from the called party to the calling party, the media stream is fully negotiated, and the endpoints can start sending traffic to each other (either end-to-end or via the Media Relay). Upon reception of the 200OK, the SIP proxy writes a start record for the accounting process. The 200OK is also acknowledged with an ACK message from the calling party to the called party, according to the SIP 3-way handshake.

Either of the parties can tear down the media session at any time by sending a BYE, which is passed through to the other party. Once the BYE reaches the SIP proxy, it instructs the Media Relay to close the media ports, and it writes a stop record for accounting purposes. Both the start- and the stop-records are picked up by the *ngcp-mediator* service in a regular interval and are converted into a Call Detail Record (CDR), which will be rated by the *ngcp-rate-o-mat* process and can be billed to the calling party. For calls made by subscribers on a prepaid plan, rating occurs at call runtime and is actually done by the B2BUA (which is necessary to properly support multiple parallel calls by the same subscriber). The final rating data is then passed on to *ngcp-rate-o-mat* which will update the CDRs accordingly.

Session Keep-Alive

The SIP B2BUA acts as refresher for the Session-Timer mechanism as defined in RFC 4028. If the endpoints indicate support for session timers during call-setup, then the SIP B2BUA will use an UPDATE or re-INVITE message if enabled per peer, domain or subscriber via Provisioning to check if the endpoints are still alive and responsive. Both endpoints can renegotiate the timer within a configurable range. All values can be tuned using the Admin Panel or the APIs using Peer-, Domain- and Subscriber-Preferences.

TIP Keep in mind that the values being used in the signaling are always half the value being configured. So if you want to send a keep-alive every 300 seconds, you need to provision *sst_expires* to 600.

If one of the endpoints doesn't respond to the keep-alive messages or answers with 481 Call/Transaction Does Not Exist, then the call is torn down on both sides. This mechanism prevents excessive over-billing of calls if one of the endpoints is not reachable anymore or "forgets" about the

 Copyright © Sipwise GmbH - All rights reserved

call. The BYE message sent by the B2BUA triggers a stop-record for accounting and also closes the media ports on the Media Relay to stop the call.

Beside the Session-Timer mechanism to prevent calls from being lost or kept open, there is a **maximum call length** of 21600 seconds per default defined in the B2BUA. This is a security/anti-fraud mechanism to prevent overly long calls causing excessive costs.

Voicebox Calls



Figure 211. Voicebox Call-Flow

Calls to the Voicebox (both for callers leaving a voicemail message and for voicebox owners managing it via the IVR menu) are passed directly from the SIP proxy to the App-Server without a B2BUA. The App-Server maintains its own timers, so there is no risk of over-billing or overly long calls.

In such a case where an endpoint talks via the Media Relay to a system-internal endpoint, the Media Relay bridges the media streams between the public in the system-internal network.

In case of an endpoint leaving a new message on the voicebox, the Message-Waiting-Indication (MWI) mechanism triggers the sending of a unsolicited NOTIFY message, passing the number of new messages in the body. As soon as the voicebox owner dials into his voicebox (e.g. by calling sip:voicebox@example.org from his SIP account), another NOTIFY message is sent to his devices, resetting the number of new messages.

Copyright © Sipwise GmbH - All rights reserved

IMPORTANT

The Sipwise C5 does not require your device to subscribe to the MWI service by sending a SUBSCRIBE (it would rather reject it). On the other hand, the endpoints need to accept unsolicited NOTIFY messages (that is, a NOTIFY without a valid subscription), otherwise the MWI service will not work with these endpoints.

Copyright © Sipwise GmbH - All rights reserved

Appendix C: Configuration Overview

config.yml Overview

`/etc/ngcp-config/config.yml` is the main configuration YAML file used by Sipwise C5. After every changes it need to run the command ngcpcfg apply "my commit message" to apply changes (followed by ngcpcfg push in the PRO version to apply changes to sp2). The following is a brief description of the main variables contained into /etc/ngcp-config/config.yml file.

apps

This section contains parameters for the additional applications that may be activated on Sipwise C5.

```
apps:
  malicious_call: no
  party_call_control:
    accepted_reply: 200*
    enable: no
    pcc_server_url:
https://127.0.0.1:9090/pcc/${prefix}${callee}${suffix}
    request_timeout: '30'
    trigger_on_hangup: yes
```

- malicious_call: If set to 'yes', the Malicious Call Identification (MCID) application will be enabled.
- party_call_control.accepted_reply: Defines the value of status data element that means the "accepted" status of the call.
- party_call_control.enable: Must be set to `yes` in order to enable the PCC feature.
- party_call_control.pcc_server_url: The URL, pointing to the PCC server, where HTTP *POST* requests must be sent. Do not change the variable references `${prefix}, ${callee}` and `${suffix}`!
- party_call_control.request_timeout: Time in seconds until Sipwise C5 will wait for an HTTP reply from the PCC server, once Sipwise C5 has sent a request to it.
- party_call_control.trigger_on_hangup: If set to `yes`, Sipwise C5 will send a "terminate" request to the PCC server at the end of the call.

TIP See the Configuration of PCC section of the handbook for more details on PCC configuration.

asterisk

The following is the asterisk section:

Copyright © Sipwise GmbH - All rights reserved

```
asterisk:
  log:
    facility: local6
  rtp:
    maxport: 20000
    minport: 10000
  sip:
    bindport: 5070
    dtmfmode: rfc2833
  voicemail:
    enable: no
    fromstring: 'Voicemail server'
    greeting:
      busy_custom_greeting: '/home/user/file_no_extension'
      busy_overwrite_default: no
      busy_overwrite_subscriber: no
      unavail_custom_greeting: '/home/user/file_no_extension'
      unavail_overwrite_default: no
      unavail_overwrite_subscriber: no
    mailbody: 'You have received a new message from ${VM_CALLERID} in
voicebox ${VM_MAILBOX} on ${VM_DATE}.'
    mailsubject: '[Voicebox] New message ${VM_MSGNUM} in voicebox
${VM_MAILBOX}'
    max_msg_length: 180
    maxgreet: 60
    maxmsg: 30
    maxsilence: 0
    min_msg_length: 3
    normalize_match: '^00|\+([1-9][0-9]+)$'
    normalize_replace: '$1'
    serveremail: voicebox@sip.sipwise.com
```

- log.facility: rsyslog facility for asterisk log, defined in /etc/asterisk/logger.conf.
- rtp.maxport: RTP maximum port used by asterisk.
- rtp.minport: RTP minimum port used by asterisk.
- sip.bindport: SIP asterisk internal bindport.
- voicemail.greetings.*: set the audio file path for voicemail custom unavailable/busy greetings
- voicemail.mailbody: Mail body for incoming voicemail.
- voicemail.mailsubject: Mail subject for incoming voicemail.
- voicemail.max_msg_length: Sets the maximum length of a voicemail message, in seconds.
- voicemail.maxgreet: Sets the maximum length of voicemail greetings, in seconds.
- voicemail.maxmsg: Sets the maximum number of messages that may be kept in any voicemail folder.
- voicemail.min_msg_length: Sets the minimum length of a voicemail message, in seconds.
- voicemail.maxsilence: Maxsilence defines how long Asterisk will wait for a contiguous period of silence before terminating an incoming call to voice mail. The default value is 0, which means the

Copyright © Sipwise GmbH - All rights reserved

silence detector is disabled and the wait time is infinite.

- voicemail.serveremail: Provides the email address from which voicemail notifications should be sent.
- voicemail.normalize_match: Regular expression to match the From number for calls to voicebox.
- voicemail.normalize_replace: Replacement string to return, in order to match an existing voicebox.

autoprov

The following is the autoprovisioning section:

```
autoprov:
  hardphone:
    skip_vendor_redirect: no
  server:
    bootstrap_port: 1445
    ca_certfile: '/etc/ngcp-config/shared-files/ssl/client-auth-ca.crt'
    host: localhost
    port: 1444
    server_certfile: '/etc/ngcp-config/shared-files/ssl/myserver.crt'
    server_keyfile: '/etc/ngcp-config/shared-files/ssl/myserver.key'
    ssl_enabled: yes
  softphone:
    config_lockdown: 0
    webauth: 0
```

- autoprov.skip_vendor_redirect: Skip phone vendor redirection to the vendor provisioning web site.

sems-b2b (some paramenters are only used with additional Cloud PBX module activated)

The following is the B2B section:

```
b2b:
  bindport: 5080
  dialog_publish_expires: 3600
  enable: yes
  highport: 19999
  lowport: 15000
  media_processor_threads: 10
  moh_codecs:
    codecs_list: PCMA,PCMU,telephone-event
    enable: no
    mode: whitelist
  permit_ext_dial_when_no_prompt_exists: no
  session_processor_threads: 10
  xmlrpcport: 8090
```

- b2b.enable: Enable sems-b2b service.

Copyright © Sipwise GmbH - All rights reserved

backuptools

The following is the backup tools section:

```
backuptools:
  cdrexport_backup:
    enable: no
  etc_backup:
    enable: no
  mail:
    address: noc@company.org
    error_subject: '[ngcp-backup] Problems detected during daily backup'
    log_subject: '[ngcp-backup] Daily backup report'
    send_errors: no
    send_log: no
  mysql_backup:
    enable: no
    exclude_dbs: 'syslog sipstats information_schema'
  replicas:
    peer: yes
    mgmt: no
  rotate_days: 7
  storage_dir: '/ngcp-data/backup/ngcp_backup'
  temp_backup_dir: '/ngcp-data/backup/ngcp_backup/tmp'
```

- backuptools.cdrexport_backup.enable: Enable backup of cdrexport (.csv) directory.
- backuptools.etc_backup.enable: Enable backup of /etc/* directory.
- backuptools.mail.address: Destination email address for backup emails.
- backuptools.mail.error_subject: Subject for error emails.
- backuptools.mail.log_subjetc: Subject for daily backup report.
- backuptools.mail.send_error: Send daily backup error report.
- backuptools.mail.send_log: Send daily backup log report.
- backuptools.mysql_backup.enable: Enable daily mysql backup.
- backuptools.mysql_backup.exclude_dbs: exclude mysql databases from backup.
- backuptools.replicas.peer: Enable or disable copying the backups to the peer node, for additional safety.
- backuptools.replicas.mgmt: Enable or disable copying the backups to the mgmt nodes, for additional safety, and so that the management nodes have consolidated backups for the entire cluster.
- backuptools.rotate_days: Number of days backup files should be kept. All files older than specified number of days are deleted from the storage directory.
- backuptools.storage_dir: Storage directory of backups.
- backuptools.storage_group: Name of the group that backup files should be owned by.
- backuptools.storage_user: Name of the user that backup files should be owned by.

Copyright © Sipwise GmbH - All rights reserved

- backuptools.temp_backup_dir: Temporary storage directory of backups.

bootenv

The following is the bootenv section:

```
bootenv:
  custom_repos:
  - enable: no
    name: my-example-repo
    url: https://example.com/debian
  - enable: yes
    name: my-example-repo2
    url: https://example.com/debian-security
  dhcp:
    boot: '/srv/tftp/pxelinux.0'
    enable: yes
    end: 192.168.1.199
    expire: 12h
    start: 192.168.1.101
    uefiboot: /srv/tftp/ipxe.efi
  http_port: 3000
  http_proxy: ''
  https_proxy: ''
  netscript:
    debug: yes
    fallbackfssize: 10G
    pxebootoption: ''
    rootfssize: 10G
    swapfilesize: 2048M
  ppa: []
  ro_port: 9998
  rw_port: 9999
  tftp:
    enable: yes
    root: '/srv/tftp'
```

- bootenv.custom_repo: The list of custom apt repos in approx
- bootenv.custom_repo.0.enable: The flag to enable/disable the repo
- bootenv.custom_repo.0.name: The approx name for the repo
- bootenv.custom_repo.0.url: The approx URL for the repo
- bootenv.dhcp.enable: enable dnsmasq DHCP server
- bootenv.dhcp.boot: PXE image boot location
- bootenv.dhcp.start: first IP of DHCP scope
- bootenv.dhcp.end: last IP of DHCP scope
- bootenv.dhcp.expire: DHCP leasing expiration
- bootenv.dhcp.uefiboot: The location of the UEFI boot file

Copyright © Sipwise GmbH - All rights reserved

- bootenv.http_port: HTTP port for iPXE boot files/configs
- bootenv.http_proxy: HTTP proxy to access Sipwise Debian repositories
- bootenv.https_proxy: HTTPS proxy to access Sipwise Debian repositories
- bootenv.netscript: The section for netscript (installer) options
- bootenv.netscript.debug: The flag to enable debug for iPXE boot/installer
- bootenv.netscript.fallbackfssize: The size of 'fallback' partition created by netscript (installer)
- bootenv.netscript.pxebootoption: The list of a custom iPXE boot options for netscript (installer)
- bootenv.netscript.rootfssize: The size of 'root' partition created by netscript (installer)
- bootenv.netscript.swapfilesize: The size of a swapfile created by netscript (installer)
- bootenv.ppa: The list of a custom NGCP PPA repositories (automatically filled by the tool 'ngcp-ppa')
- bootenv.ppa.0.name: The name of a custom NGCP PPA repositories
- bootenv.ppa.0.priority: The priority of a custom NGCP PPA repositories (optional)
- bootenv.ppa.0.url: The URL of a custom NGCP PPA repositories (optional)
- bootenv.ro_port: HTTP port for read-only access to Approx cache
- bootenv.rw_port: HTTP port for read-write access to Approx cache
- bootenv.tftp.enable: enable tftp server for PXE boot
- bootenv.tftp.root: root folder for tftp server

cdrexport

The following is the cdr export section:

```
cdrexport:
  daily_folder: yes
  export_failed: no
  export_incoming: no
  export_nodes:
    roles:
      - mgmt
    hosts:
  exportpath: '/home/jail/home/cdrexport'
  full_names: yes
  monthly_folder: yes
```

- cdrexport.daily_folder: Set 'yes' if you want to create a daily folder for CDRs under the configured path.
- cdrexport.export_failed: Export CDR for failed calls.
- cdrexport.export_incoming: Export CDR for incoming calls.
- cdrexport.export_nodes: Export CDRs to specific nodes based on role or hostnames.
- cdrexport.exportpath: The path to store CDRs in .csv format.

Copyright © Sipwise GmbH - All rights reserved

- cdrexport.full_names: Use full names for CDRs instead of short ones.
- cdrexport.monthly_folder: Set 'yes' if you want to create a monthly folder (ex. 201301 for January 2013) for CDRs under configured path.

cleanuptools

The following is the cleanup tools section:

```
cleanuptools:
  acc_cleanup_days: 90
  archive_targetdir: '/ngcp-data/backups/cdr'
  binlog_days: 15
  cdr_archive_months: 2
  cdr_keep_months: 2
  compress: gzip
  delete_old_cdr_files:
    enable: no
    max_age_days: 30
    paths:
      -
        max_age_days: ~
        path: '/home/jail/home/*/20[0-9][0-9][0-9][0-9]/[0-9][0-9]'
        remove_empty_directories: yes
        wildcard: yes
      -
        max_age_days: ~
        path: '/home/jail/home/cdrexport/resellers/*/20[0-9][0-9][0-
9][0-9]/[0-9][0-9]'
        remove_empty_directories: yes
        wildcard: yes
      -
        max_age_days: ~
        path: '/home/jail/home/cdrexport/system/20[0-9][0-9][0-9][0-
9]/[0-9][0-9]'
        remove_empty_directories: yes
        wildcard: yes
  sql_batch: 10000
  trash_cleanup_days: 30
```

- cleanuptools.acc_cleanup_days: CDR records in acc table in kamailio database will be deleted after this time
- cleanuptools.binlog_days: Time after MySQL binlogs will be deleted.
- cleanuptools.cdr_archive_months: How many months worth of records to keep in monthly CDR backup tables, instead of dumping them into archive files and dropping them from database.
- cleanuptools.cdr_keep_months: How many months worth of records to keep in the current *cdr* table, instead of moving them into the monthly CDR backup tables.
- cleanuptools.delete_old_cdr_files:

  enable: Enable (yes) or disable (no) exported CDR cleanup.

Copyright © Sipwise GmbH - All rights reserved

max_age_days: Gives the expiration time of the exported CDR files in days. There is a general value which may be overridden by a local value provided at a specific path. The local value is valid for the particular path only.

paths: an array of path definitions

path: a path where CDR files are to be found and deleted; this may contain wildcard characters

wildcard: Enable (yes) or disable (no) using wildcards in the path

remove_empty_directories: Enable (yes) or disable (no) removing empty directories if those are found in the given path

max_age_days: the local expiration time value for files in the particular path

- cleanuptools.sql_batch: How many records to process within a single SQL statement.
- cleanuptools.trash_cleanup_days: Time after CDRs from acc_trash and acc_backup tables in kamailio database will be deleted.

For the description of *cleanuptools* please visit Cleanuptools Description section of the handbook.

cluster_sets

The following is the cluster sets section:

```
cluster_sets:
  default:
    dispatcher_id: 50
  default_set: default
  type: central
```

- cluster_sets.<label>: an arbitrary label of the cluster set; in the above example we have default
- cluster_sets.<label>.dispatcher_id: a unique, numeric value that identifies a particular cluster set
- cluster_sets.default_set: selects the default cluster set
- cluster_sets.type: the type of cluster set; can be central or distributed

database

The following is the database section:

```
database:
  bufferpoolsize: 24768M
```

- database.bufferpoolsize: Innodb_buffer_pool_size value in /etc/mysql/my.cnf

faxserver

The following is the fax server section:

Copyright © Sipwise GmbH - All rights reserved

```
faxserver:
  enable: yes
  fail_attempts: '3'
  fail_retry_secs: '60'
  mail_from: 'Sipwise C5 FaxServer <voipfax@ngcp.sipwise.local>'
```

- faxserver.enable: 'yes'/'no' to enable or disable ngcp-faxserver on the platform respectively.
- faxserver.fail_attempts: Amount of attempts to send a fax after which it is marked as 'failed'.
- faxserver.fail_retry_secs: Amount of seconds to wait between "fail_attemts".
- faxserver.mail_from: Sets the e-mail From Header for incoming fax.

general

The following is the general section:

```
general:
  adminmail: adjust@example.org
  companyname: sipwise
  lang: en
  maintenance: no
  production: yes
  timezone: localtime
```

- general.adminmail: Email address used by monit to send notifications to.
- general.companyname: Label used in SNMPd configuration.
- general.lang: Sets sounds language (e.g: 'de' for German)
- general.production: Label to hint self-check scripts about installation mode.
- general.maintenance: maintenance mode necessary for safe upgrades.
- general.timezone: Sipwise C5 Timezone

ha

The following is the High Availability (ha) section:

```
ha:
  gcs: corosync
  crm: pacemaker
  pingnodes:
    - 10.60.1.1
    - 192.168.3.4
  pingnodes_add_gw: yes
  pingnodes_add_dns: yes
  monitor_services: none
```

Copyright © Sipwise GmbH - All rights reserved

- ha.crs: Group Communication System (GCS). Supported: 'corosync'.
- ha.crm: Cluster Resource Manager (CRM). Supported: 'pacemaker'.
- ha.pingnodes: List of HA pingnodes. Minimum 2 entries, otherwise by default Sipwise C5 will set the default gateway and DNS servers as pingnodes.
- ha.pingnodes_add_gw: Enable whether to add the gateway IP to the HA ping nodes list (only if less than 3 ping nodes are defined in the list already).
- ha.pingnodes_add_dns: Enable whether to add the DNS IPs to the HA ping nodes list (only if less than 3 ping nodes are defined in the list already).
- ha.monitor_services: Whether the HA system should periodically monitor the active services and take that into account as part of considering whether the node can act as the active one (only supported with 'pacemaker').

haproxy

The following is the haproxy section:

```
haproxy:
  admin: no
  admin_port: 8080
  admin_pwd: iKNPFuPFHMCHh9dsXgVg
  enable: no
```

- haproxy.enable: enable haproxy

intercept

The following is the legal intercept section:

```
intercept:
  enable: no
```

- intercept.enable: Enable ngcp-voisniff for Lawful Interception (additional Sipwise C5 module).

kamailio

The following is the kamailio section:

```
kamailio:
  lb:
    block_useragents:
      action: reject
      block_empty: no
      block_absent: no
      enable: no
      mode: blacklist
      ua_patterns: []
    cfgt: no
```


Copyright © Sipwise GmbH - All rights reserved

```
    debug:
      enable: no
      modules:
      - level: '1'
        name: core
      - level: '3'
        name: xlog
    debug_level: '1'
    debug_uri:
      enable: no
      redis_db: 27
      htable_idx_size: 4
    dns:
      dns_sctp_pref: 1
      dns_tcp_pref: 1
      dns_tls_pref: 1
      dns_try_naptr: no
      dns_udp_pref: 1
      use_dns_cache: on
    external_sbc: []
    extra_sockets: ~
    filter_content_type:
      enable: yes
      action: filter
      content_type_list:
      - content_type: application/vnd.etsi.cug+xml
        direction: all
      - content_type: application/isup
        direction: reply
      - content_type: application/xml
        direction: request
    max_forwards: '70'
    mem_log: '1'
    mem_summary: '12'
    max_inv_lifetime: '180000'
    nattest_exception_ips:
    - 1.2.3.4
    - 5.6.7.8
    nattest_exception_nets:
    - 192.168.10.0/24
    - 192.168.11.0/24
    pkg_mem: '16'
    port: '5060'
    sdp_line_filter:
      enable: no
      remove_line_startswith: []
    security:
      dos_ban_enable: yes
      dos_ban_time: '300'
      dos_reqs_density_per_unit: '50'
      dos_sampling_time_unit: '5'
      dos_whitelisted_ips: []
```

Copyright © Sipwise GmbH - All rights reserved

```
      dos_whitelisted_subnets: []
      failed_auth_attempts: '3'
      failed_auth_ban_enable: yes
      failed_auth_ban_time: '3600'
      topoh:
        enable: no
        mask_callid: no
        mask_ip: 127.0.0.8
      topos:
        enable: no
        redis_db: 24
    shm_mem: '64'
    skip_contact_alias_for_ua_when_tcp:
      enable: no
      user_agent_patterns: []
    start: yes
    strict_routing_safe: no
    syslog_options: yes
    tcp_children: 1
    tcp_max_connections: '2048'
    tls:
      enable: no
      port: '5061'
      sslcertfile: /etc/ngcp-config/shared-files/ssl/myserver.crt
      sslcertkeyfile: /etc/ngcp-config/shared-files/ssl/myserver.key
    udp_children: 1
  proxy:
    allow_cf_to_itself: no
    allow_info_method: no
    allow_msg_method: no
    allow_peer_relay: no
    allow_refer_method: no
    always_anonymize_from_user: no
    authenticate_bye: no
    cf_depth_limit: '10'
    cfgt: no
    check_prev_forwarder_as_upn: no
    children: 1
    decode_utu_header: no
    debug:
      enable: no
      modules:
      - level: '1'
        name: core
      - level: '3'
        name: xlog
    debug_level: '1'
    default_expires: '3600'
    default_expires_range: '30'
    dlg_timeout: '43200'
    early_rejects:
      block_admin:
```


Copyright © Sipwise GmbH - All rights reserved

```
      announce_code: '403'
      announce_reason: Blocked by Admin
    block_callee:
      announce_code: '403'
      announce_reason: Blocked by Callee
    block_caller:
      announce_code: '403'
      announce_reason: Blocked by Caller
    block_contract:
      announce_code: '403'
      announce_reason: Blocked by Contract
    block_in:
      announce_code: '403'
      announce_reason: Block in
    block_out:
      announce_code: '403'
      announce_reason: Blocked out
    block_override_pin_wrong:
      announce_code: '403'
      announce_reason: Incorrect Override PIN
    callee_busy:
      announce_code: '486'
      announce_reason: Busy Here
    callee_offline:
      announce_code: '480'
      announce_reason: Offline
    callee_tmp_unavailable:
      announce_code: '480'
      announce_reason: Temporarily Unavailable
    callee_tmp_unavailable_gp:
      announce_code: '480'
      announce_reason: Unavailable
    callee_tmp_unavailable_tm:
      announce_code: '408'
      announce_reason: Request Timeout
    callee_unknown:
      announce_code: '404'
      announce_reason: Not Found
    cf_loop:
      announce_code: '480'
      announce_reason: Unavailable
    emergency_invalid:
      announce_code: '404'
      announce_reason: Emergency code not available in this region
    emergency_unsupported:
      announce_code: '403'
      announce_reason: Emergency Calls Not Supported
    invalid_speeddial:
      announce_code: '484'
      announce_reason: Speed-Dial slot empty
    locked_in:
      announce_code: '403'
```

Copyright © Sipwise GmbH - All rights reserved

```
        announce_reason: Callee locked
      locked_out:
        announce_code: '403'
        announce_reason: Caller locked
      max_calls_in:
        announce_code: '486'
        announce_reason: Busy
      max_calls_out:
        announce_code: '403'
        announce_reason: Maximum parallel calls exceeded
      no_credit:
        announce_code: '402'
        announce_reason: Insufficient Credit
      peering_unavailable:
        announce_code: '503'
        announce_reason: PSTN Termination Currently Unavailable
      reject_vsc:
        announce_code: '403'
        announce_reason: VSC Forbidden
      relaying_denied:
        announce_code: '403'
        announce_reason: Relaying Denied
      unauth_caller_ip:
        announce_code: '403'
        announce_reason: Unauthorized IP detected
    emergency_priorization:
      enable: no
      register_fake_200: yes
      register_fake_expires: '3600'
      reject_code: '503'
      reject_reason: Temporary Unavailable
      retry_after: '3600'
    enum_suffix: e164.arpa.
    expires_range: '30'
    filter_100rel_from_supported: no
    filter_failover_response: 408|500|503
    foreign_domain_via_peer: no
    fritzbox:
      enable: no
      prefixes:
      - 0$avp(caller_ac)
      - $avp(caller_cc)$avp(caller_ac)
      - \+$avp(caller_cc)$avp(caller_ac)
      - 00$avp(caller_cc)$avp(caller_ac)
      special_numbers:
      - '112'
      - '110'
      - 118[0-9]{2}
    ignore_auth_realm: no
    ignore_subscriber_allowed_clis: no
    keep_original_to: no
    lcr_stopper_mode: 0
```


Copyright © Sipwise GmbH - All rights reserved

```
    latency_limit_action: '100'
    latency_limit_db: '500'
    latency_log_level: '1'
    latency_runtime_action: 1000
    lnp:
      add_reply_headers:
        enable: no
        number: P-NGCP-LNP-Number
        status: P-NGCP-LNP-Status
      api:
        add_caller_cc_to_lnp_dst: no
        invalid_lnp_routing_codes:
        - ^EE00
        - ^DD00
        keepalive_interval: '3'
        lnp_request_blacklist: []
        lnp_request_whitelist: []
        port: '8991'
        reply_error_on_lnp_failure: no
        request_timeout: '1000'
        server: localhost
        tcap_field_fci: end.components.0.invoke.parameter
        tcap_field_lnp: ConnectArg.destinationRoutingAddress.0
        tcap_field_opcode: end.components.0.invoke.opCode
      enable: no
      execute_ncos_block_out_before_lnp: no
      skip_callee_lnp_lookup_from_any_peer: no
      strictly_check_ncos: no
      type: api
    lookup_peer_destination_domain_for_pbx: no
    loop_detection:
      enable: no
      expire: '1'
      max: '5'
    max_expires: '43200'
    max_gw_lcr: '128'
    max_registrations_per_subscriber: '5'
    mem_log: '1'
    mem_summary: '12'
    min_expires: '60'
    nathelper:
      sipping_from: sip:pinger@sipwise.local
    nathelper_dbro: no
    natping_interval: '30'
    natping_processes: 1
    nonce_expire: '300'
    pbx:
      hunt_display_fallback_format: '[H %s]'
      hunt_display_fallback_indicator: $var(cloud_pbx_hg_ext)
      hunt_display_format: '[H %s]'
      hunt_display_indicator: $var(cloud_pbx_hg_displayname)
      hunt_display_maxlength: 8
```

Copyright © Sipwise GmbH - All rights reserved

```
    ignore_cf_when_hunting: no
  peer_probe:
    available_treshold: '1'
    enable: yes
    from_uri_domain: probe.ngcp.local
    from_uri_user: ping
    interval: '10'
    method: OPTIONS
    reply_codes: class=2;class=3;code=403;code=404;code=405
    timeout: '5'
    unavailable_treshold: '1'
  perform_peer_failover_on_tm_timeout: yes
  perform_peer_lcr: no
  pkg_mem: '32'
  port: '5062'
  presence:
    enable: yes
    max_expires: '3600'
    reginfo_domain: example.org
  proxy_lookup: no
  push:
    apns_alert: New call
    apns_sound: incoming_call.xaf
    code_18x: 180
    reason_18x: 'Ringing'
    reply_18x: 'no'
  report_mos: yes
  set_ruri_to_peer_auth_realm: no
  shm_mem: '125'
  skip_pbx_loop: no
  start: yes
  stir:
    cache_dir: /var/cache/kamailio/stir/
    cache_expire: 3600
    domains:
    - name: <domain_name>
      private_key: <path_to_a_private_key_related_to_domain>
    enable: yes
    expire: 300
    libopt: []
    shaken:
      attestation_name: verstat
      attestation_values:
        failed: TN-Validation-Failed
        no_validation: No-TN-Validation
        not_present: TN-Validation-Not-Present
        passed: TN-Validation-Passed
        passed_A: TN-Validation-Passed-A
        passed_B: TN-Validation-Passed-B
        passed_C: TN-Validation-Passed-C
    timeout: 5
  store_recentcalls: no
```


Copyright © Sipwise GmbH - All rights reserved

```
    syslog_options: yes
    tcp_children: 1
    tm:
      fr_inv_timer: '180000'
      fr_timer: '9000'
      max_inv_lifetime: '180000'
    treat_600_as_busy: yes
    use_enum: no
    usrloc_dbmode: '1'
    voicebox_first_caller_cli: yes
    xfer_other_party_from: no
```

- kamailio.lb.block_useragents.action: one of [`drop`, `reject`] - Whether to silently drop the request from matching User-Agent or reject with a 403 message.
- kamailio.lb.block_useragents.block_empty: Enable/disable a rejection of messages with an empty User-Agent header (header present, but no value given).
- kamailio.lb.block_useragents.block_absent: Enable/disable a rejection of messages with an absent User-Agent header.
- kamailio.lb.block_useragents.enable: Enable/disable the User-Agent blocking.
- kamailio.lb.block_useragents.mode: one of [`whitelist`, `blacklist`] - Sets the mode of ua_patterns list evaluation (whitelist: block requests coming from all but listed User-Agents, blacklist: block requests from all listed User-Agents).
- kamailio.lb.block_useragents.ua_patterns: List of User-Agent string patterns that trigger the block action.
- kamailio.lb.cfgt: Enable/disable unit test config file execution tracing.
- kamailio.lb.debug.enable: Enable per-module debug options.
- kamailio.lb.debug.modules: List of modules to be traced with respective debug level.
- kamailio.lb.debug_uri.enable: Enable/disable sending SIP messages From/To specific subscriber to an inactive proxy node in order to debug/trace calls. Only makes sense on Sipwise C5 CARRIER appliance environment.
- kamailio.lb.debug_uri.redis_db: A number of internal Redis DB used by htable module to keep the subscribers values
- kamailio.lb.debug_uri.htable_idx_size: number to control how many slots (buckets) to create for the hash table (2^size). See kamailio htable docs for details.
- kamailio.lb.debug_level: Default debug level for `kamailio-lb`.
- kamailio.lb.dns.use_dns_cache: Enable/disable use of internal DNS cache.
- kamailio.lb.dns.dns_udp_pref: Set preference for each protocol when doing NAPTR lookups.In order to use remote site preferences set all dns_*_pref to the same positive value (e.g. dns_udp_pref=1, dns_tcp_pref=1, dns_tls_pref=1, dns_sctp_pref=1). To completely ignore NAPTR records for a specific protocol, set the corresponding protocol preference to -1.
- kamailio.lb.dns.dns_tcp_pref: See above.
- kamailio.lb.dns.dns_tls_pref: See above.
- kamailio.lb.dns.dns_sctp_pref: See above.

Copyright © Sipwise GmbH - All rights reserved

- kamailio.lb.dns.dns_try_naptr: Enable NAPTR support according to RFC 3263.
- kamailio.lb.external_sbc: SIP URI of external SBC used in the Via Route option of peering server.
- kamailio.lb.extra_sockets: Add here extra sockets for Load Balancer.
- kamailio.lb.max_forwards: Set the value for the Max Forwards SIP header for outgoing messages.
- kamailio.lb.mem_log: Specifies on which log level the memory statistics will be logged.
- kamailio.lb.mem_summary: Parameter to control printing of memory debugging information on exit or SIGUSR1 to log.
- kamailio.lb.max_inv_lifetime: Set INVITE transaction timeout per the whole transaction if no final reply for an INVITE arrives after a provisional message was received (whole transaction ringing timeout). It has to be equals or greater than kamailio.proxy.tm.fr_inv_timer.
- kamailio.lb.nattest_exception_ips: List of IPs that don't need the NAT test.
- kamailio.lb.nattest_exception_nets: List of IP networks (sub-nets) that don't need the NAT test. The format is network/mask, see an example of the 'kamailio' section.
- kamailio.lb.shm_mem: Shared memory used by Kamailio Load Balancer.
- kamailio.lb.pkg_mem: PKG memory used by Kamailio Load Balancer.
- kamailio.lb.port: Default listen port.
- kamailio.lb.remove_isup_body_from_replies: Enable/disable stripping of ISUP part from the message body.
- kamailio.lb.sdp_line_filter.enable: Enable/Disable filter of SDP lines in all the SIP messages.
- kamailio.lb.sdp_line_filter.remove_line_startswith: List of the SDP lines that should be removed. Attention: it removes all SDP attribute lines beginning with the listed strings in all media streams.
- kamailio.lb.security.dos_ban_enable: Enable/Disable DoS Ban.
- kamailio.lb.security.dos_ban_time: Sets the ban time.
- kamailio.lb.security.dos_reqs_density_per_unit: Sets the requests density per unit (if we receive more then * lb.dos_reqs_density_per_unit within dos_sampling_time_unit the user will be banned).
- kamailio.lb.security.dos_sampling_time_unit: Sets the DoS unit time.
- kamailio.lb.security.dos_whitelisted_ips: Write here the whitelisted IPs.
- kamailio.lb.security.dos_whitelisted_subnets: Write here the whitelisted IP subnets.
- kamailio.lb.security.failed_auth_attempts: Sets how many authentication attempts allowed before ban.
- kamailio.lb.security.failed_auth_ban_enable: Enable/Disable authentication ban.
- kamailio.lb.security.failed_auth_ban_time: Sets how long a user/IP has be banned.
- kamailio.lb.topoh.enable: Enable topology masking module (see the Topology Masking Mechanism subchapter for a detailed description).
- kamailio.lb.topoh.mask_callid: if set to `yes`, the SIP Call-ID header will also be encoded.
- kamailio.lb.topoh.mask_ip: an IP address that will be used to create valid SIP URIs, after encoding the real/original header content.
- kamailio.lb.topos.enable: Enable topology hiding module (see the Topology Hiding Mechanism subchapter for a detailed description).

Copyright © Sipwise GmbH - All rights reserved

- kamailio.lb.topos.redis_db: A number of internal Redis DB used by the topology hiding module.
- kamailio.lb.start: Enable/disable kamailio-lb service.
- kamailio.lb.strict_routing_safe: Enable strict routing handle feature.
- kamailio.lb.syslog_options: Enable/disable logging of SIP OPTIONS messages to `kamailio-options-lb.log`.
- kamailio.lb.tcp_children: Number of TCP worker processes.
- kamailio.lb.tcp_max_connections: Maximum number of open TCP connections.
- kamailio.lb.tls.enable: Enable TLS socket.
- kamailio.lb.tls.port: Set TLS listening port.
- kamailio.lb.tls.sslcertificate: Path for the SSL certificate.
- kamailio.lb.tls.sslcertkeyfile: Path for the SSL key file.
- kamailio.lb.udp_children: Number of UDP worker processes.
- kamailio.proxy.allow_cf_to_itself: Specify whether or not a Call Forward to the same subscriber (main number to an alias or viceversa) is allowed. To stop the CF loop a source number or a b-number have to be defined in the CF configuration.
- kamailio.proxy.allow_info_method: Allow INFO method.
- kamailio.proxy.allow_msg_method: Allow MESSAGE method.
- kamailio.proxy.allow_peer_relay: Allow peer relay. Call coming from a peer that doesn't match a local subscriber will try to go out again, matching the peering rules.
- kamailio.proxy.allow_refer_method: Allow REFER method. Enable it with caution.
- kamailio.proxy.always_anonymize_from_user: Enable anonymization of full From URI (as opposed to only From Display-name part by default), has same effect as enabling the preference `anonymize_from_user` for all peers.
- kamailio.proxy.authenticate_bye: Enable BYE authentication.
- kamailio.proxy.cf_depth_limit: CF loop detector. How many CF loops are allowed before drop the call.
- kamailio.proxy.cfgt: Enable/disable unit test config file execution tracing.
- kamailio.proxy.check_prev_forwarder_as_upn: Enable/disable validation of the forwarder's number taken from the `Diversion` or `History-Info` header.
- kamailio.proxy.children: Number of UDP worker processes.
- kamailio.proxy.decode_utu_header: Default 'no'. If set to 'yes', the content of the User-to-User field received in 200Ok is decoded and saved in a dedicated field of the ACC records. The decoding consists in few steps: discard everything after the first occurrence of ';', remove the initial '04', hex decode the remaining part.
- kamailio.proxy.debug.enable: Enable per-module debug options.
- kamailio.proxy.debug.modules: List of modules to be traced with respective debug level.
- kamailio.proxy.debug_level: Default debug level for `kamailio-proxy`.
- kamailio.proxy.default_expires: Default expires value in seconds for a new registration (for REGISTER messages that contains neither Expires HFs nor expires contact parameters).
- kamailio.proxy.default_expires_range: This parameter specifies that the expiry used for the

Copyright © Sipwise GmbH - All rights reserved

registration should be randomly chosen in a range given by `default_expires` +/- `default_expires_range` percent. For instance, if default_expires is 1200 seconds and default_expires_range is 50, the expiry is randomly chosen between [600,1800] seconds. If set to '0', default_expires is left unmodified.

- kamailio.proxy.dlg_timeout: Dialog timeout in seconds (by default 43200 sec - 12 hours).
- kamailio.proxy.early_rejects: Customize here the response codes and sound prompts for various reject scenarios. See the subchapter Configuring Early Reject Sound Sets for a detailed description.
- kamailio.proxy.emergency_prioritization.enable: Enable an emergency mode support.
- kamailio.proxy.emergency_prioritization.register_fake_200: When enabled, generates a fake 200 response to REGISTER from non-prioritized subscriber in emergency mode.
- kamailio.proxy.emergency_prioritization.register_fake_expires: Expires value for the fake 200 response to REGISTER.
- kamailio.proxy.emergency_prioritization.reject_code: Reject code for the non-emergency request.
- kamailio.proxy.emergency_prioritization.reject_reason: Reject reason for the non-emergency request.
- kamailio.proxy.emergency_prioritization.retry_after: Retry-After value when rejecting the non-emergency request.

TIP In order to learn about details of *emergency priorization* function of NGCP please refer to Emergency Priorization part of the handbook.

- kamailio.proxy.enum_suffix: Sets ENUM suffix - don't forget '.' (dot).
- kamailio.proxy.expires_range: Set randomization of expires for REGISTER messages (similar to default_expires_range but applies to received expires value).
- kamailio.proxy.filter_100rel_from_supported: Enable filtering of '100rel' from Supported header, to disable PRACK.
- kamailio.proxy.filter_failover_response: Specify the list of SIP responses that trigger a failover on the next available peering server.
- kamailio.proxy.foreign_domain_via_peer: Enable/disable of routing of calls to foreign SIP URI via peering servers.
- kamailio.proxy.fritzbox.enable: Enable detection for Fritzbox special numbers. Ex. Fritzbox add some prefix to emergency numbers.
- kamailio.proxy.fritzbox.prefixes: Fritzbox prefixes pattern to check. Ex. '0$avp(caller_ac)'.
- kamailio.proxy.fritzbox.special_numbers: Specifies Fritzbox special number patterns. They will be checked with the prefixes defined. Ex. '112', so the performed check will be 'sip:0$avp(caller_ac)112@' if prefix is '0$avp(caller_ac)'.
- kamailio.proxy.ignore_auth_realm: Ignore SIP authentication realm.
- kamailio.proxy.ignore_subscriber_allowed_clis: Set to 'yes' to ignore the subscriber's `allowed_clis` preference so that the User-Provided CLI is only checked against customer's `allowed_clis` preference.
- kamailio.proxy.lcr_stopper_mode: 0, default mode first rule to match will stop the gw matching process. 1, lcr will keep matching gws even if the rule has stopper value and after ordering gws by priority it will obey the first stopper value and discard the rest.

Copyright © Sipwise GmbH - All rights reserved

- kamailio.proxy.latency_limit_action: Limit of runtime in ms for config actions. If a config action executed by cfg interpreter takes longer than this value, a message is printed in the logs.
- kamailio.proxy.latency_limit_db: Limit of runtime in ms for DB queries. If a DB operation takes longer than this value, a warning is printed in the logs.
- kamailio.proxy.latency_log_level: Log level to print the messages related to latency. Default is 1 (INFO).
- kamailio.proxy.latency_runtime_action: Limit of runtime in ms for SIP message processing cycle. If the SIP message processing takes longer than this value, a warning is printed in the logs.
- kamailio.proxy.keep_original_to: Not used now.
- kamailio.proxy.lnp.add_reply_headers.enable: Enable/disable dedicated headers to be added after LNP lookup.
- kamailio.proxy.lnp.add_reply_headers.number: Name of the header that will contain the LNP number.
- kamailio.proxy.lnp.add_reply_headers.status: Name of the header that will contain the LNP return code (200 if OK, 500/480/... if an error/timeout is occurred).
- kamailio.proxy.lnp.api.add_caller_cc_to_lnp_dst: Enable/disable adding of caller country code to LNP routing number of the result ('no' by default, LNP result in E.164 format is assumed).
- kamailio.proxy.lnp.api.invalid_lnp_routing_codes [only for `api` type]: number matching pattern for routing numbers that represent invalid call destinations; an announcement is played in that case and the call is dropped.
- kamailio.proxy.lnp.api.keepalive_interval: Not used now.
- kamailio.proxy.lnp.api.lnp_request_whitelist [only for `api` type]: list of matching patterns of called numbers for which LNP lookup must be done.
- kamailio.proxy.lnp.api.lnp_request_blacklist [only for `api` type]: list of matching patterns of called numbers for which LNP lookup must not be done.
- kamailio.proxy.lnp.api.port: Not used now.
- kamailio.proxy.lnp.api.reply_error_on_lnp_failure: Specifies whether platform should drop the call in case of LNP API server failure or continue routing the call to the original callee without LNP.
- kamailio.proxy.lnp.api.request_timeout [only for `api` type]: timeout in milliseconds while Proxy waits for the response of an LNP query from *Sipwise LNP daemon*.
- kamailio.proxy.lnp.api.server: Not used now.
- kamailio.proxy.lnp.api.tcap_field_fci: path of the FCI INFO in the received tcap message
- kamailio.proxy.lnp.api.tcap_field_lnp: path of the LNP NUMBER in the received tcap/inap message
- kamailio.proxy.lnp.api.tcap_field_opcode: path of the FCI OPCODE in the received tcap message
- kamailio.proxy.lnp.enable: Enable/disable LNP (local number portability) lookup during call setup.
- kamailio.proxy.lnp.execute_ncos_block_out_before_lnp: if set to 'yes', the NCOS and BLOCK_OUT checks will be executed before the LNP lookup. Default is 'no', therefore the check are done after the LNP evaluation and rewriting.
- kamailio.proxy.lnp.skip_callee_lnp_lookup_from_any_peer: if set to 'yes', the destination LNP lookup is skipped (has same effect as enabling preference skip_callee_lnp_lookup_from_any_peer for all peers).
- kamailio.proxy.lnp.strictly_check_ncos: specify whether the NCOS LNP should be evaluated even if

Copyright © Sipwise GmbH - All rights reserved

the LNP lookup was not previously executed or if it didn't return any occurrence. If set to *yes*, a whitelist NCOS will fail if the LNP lookup doesn't return any match. The parameter has no impact on blacklist NCOS.

- kamailio.proxy.lnp.type: method of LNP lookup; valid values are: `local` (local LNP database) and `api` (LNP lookup through external gateways). *PLEASE NOTE:* the `api` type of LNP lookup is only available for Sipwise C5 PRO / CARRIER installations.
- kamailio.proxy.lookup_peer_destination_domain_for_pbx: one of [yes, no, peer_host_name] - Sets the content of destination_domain CDR field for calls between CloudPBX subscribers. In case of 'no' this field contains name of CloudPBX domain; 'yes': peer destination domain; 'peer_host_name': human-readable name of the peering server.
- kamailio.proxy.loop_detection.enable: Enable the SIP loop detection based on the combination of SIP-URI, To and From header URIs.
- kamailio.proxy.loop_detection.expire: Sampling interval in seconds for the incoming INVITE requests (by default 1 sec).
- kamailio.proxy.loop_detection.max: Maximum allowed number of SIP requests with the same SIP-URI, To and From header URIs within sampling interval. Requests in excess of this limit will be rejected with 482 Loop Detected response.
- kamailio.proxy.max_expires: Sets the maximum expires in seconds for registration. If set to '0', the check is disabled.
- kamailio.proxy.max_gw_lcr: Defines the maximum number of gateways in lcr_gw table
- kamailio.proxy.max_registrations_per_subscriber: Sets the maximum registration per subscribers.
- kamailio.proxy.mem_log: Specifies on which log level the memory statistics will be logged.
- kamailio.proxy.mem_summary: Parameter to control printing of memory debugging information on exit or SIGUSR1 to log.
- kamailio.proxy.min_expires: Sets the minimum expires in seconds for registration. If set to '0', the check is disabled.
- kamailio.proxy.nathelper.sipping_from: Set the From header in OPTIONS NAT ping.
- kamailio.proxy.nathelper_dbro: Default is "no". This will be "yes" on CARRIER in order to activate the use of a read-only connection using LOCAL_URL
- kamailio.proxy.natping_interval: Sets the NAT ping interval in seconds.
- kamailio.proxy.natping_processes: Set the number of NAT ping worker processes.
- kamailio.proxy.nonce_expire: Nonce expire time in seconds.
- kamailio.proxy.pbx.hunt_display_fallback_format: Default is '[H %s]'. Sets the format of the hunt group indicator that is sent as initial part of the From Display Name when subscriber is called as a member of PBX hunt group if the preferred format defined by the `hunt_display_format` and `hunt_display_indicator` can not be used (as in the case of not provisioned subscriber settings). The '%s' part is replaced with the value of the `hunt_display_fallback_indicator` variable.
- kamailio.proxy.pbx.hunt_display_fallback_indicator: The internal kamailio variable that sets the number or extension of the hunt group. Default is `$var(cloud_pbx_hg_ext)` which is populated during call routing with the extension of the hunt group.
- kamailio.proxy.pbx.hunt_display_format: Default is '[H %s]'. Sets the format of hunt group indicator that is sent as initial part of the From Display Name when subscriber is called as a member of PBX hunt group. This is the preferred (default) indicator format with Display Name, where the '%s' part is replaced with the value of the `hunt_display_indicator` variable.

Copyright © Sipwise GmbH - All rights reserved

- kamailio.proxy.pbx.hunt_display_indicator: The internal kamailio variable that contains the preferred identifier of the hunt group. Default is `$var(cloud_pbx_hg_displayname)` which is populated during call routing with the provisioned Display Name of the hunt group.
- kamailio.proxy.pbx.hunt_display_maxlength: Default is '8'. Sets the maximum length of the variable used as the part of hunt group indicator in Display Name. The characters beyond this limit are truncated in order for hunt group indicator and calling party information to fit on display of most phones.
- kamailio.proxy.pbx.ignore_cf_when_hunting: Default is 'no'. Whether to disregard all individual call forwards (CFU, CFB, CFT and CFNA) of PBX extensions when they are called via hunt groups. Note that call forwards configured to local services such as Voicebox or Conference are always skipped from group hunting.
- kamailio.proxy.peer_probe.enable: Enable the peer probing, must be also checked per individual peer in the panel/API.
- kamailio.proxy.peer_probe.interval: Peer probe interval in seconds.
- kamailio.proxy.peer_probe.timeout: Peer probe response wait timeout in seconds.
- kamailio.proxy.peer_probe.reply_codes: Defines the response codes that are considered successful response to the configured probe request, e.g. `class=2;class=3;code=403;code=404;code=405`, with class defining a code range.
- kamailio.proxy.peer_probe.unavailable_treshold: Defines after how many failed probes a peer is considered unavailable.
- kamailio.proxy.peer_probe.available_treshold: Defines after how many successful probes a peer is considered available.
- kamailio.proxy.peer_probe.from_uri_user: From-userpart for the probe requests.
- kamailio.proxy.peer_probe.from_uri_domain From-hostpart for the probe requests.
- kamailio.proxy.peer_probe.method: [OPTIONS|INFO] - Request method for probe request.

> **TIP** You can find more information about peer probing configuration in Configuration of Peer Probing of the handbook.

- kamailio.proxy.perform_peer_failover_on_tm_timeout: Specifies the failover behavior when maximum ring timeout (fr_inv_timer) has been reached. In case it is set to 'yes': failover to the next peer if any; in case of 'no' stop trying other peers.
- kamailio.proxy.perform_peer_lcr: Enable/Disable Least Cost Routing based on peering fees.
- kamailio.proxy.pkg_mem: PKG memory used by Kamailio Proxy.
- kamailio.proxy.shm_mem: Shared memory used by Kamailio Proxy.
- kamailio.proxy.port: SIP listening port.
- kamailio.proxy.presence.enable: Enable/disable presence feature
- kamailio.proxy.presence.max_expires: Sets the maximum expires value for PUBLISH/SUBSCRIBE message. Defines expiration of the presentity record.
- kamailio.proxy.presence.reginfo_domain: Set FQDN of Sipwise C5 domain used in callback for mobile push.
- kamailio.proxy.push.apns_alert: Set the content of 'alert' field towards APNS.
- kamailio.proxy.push.apns_sound: Set the content of 'sound' field towards APNS.

Copyright © Sipwise GmbH - All rights reserved

- kamailio.proxy.push.code_18x: code to be sent if reply_18x is 'yes'. Default: 180.
- kamailio.proxy.push.reason_18x: reason phrase to be sent if reply_18x is 'yes'. Default: 'Ringing'.
- kamailio.proxy.push.reply_18x: If set to 'yes' proxy will send a 18x message using code_18x and reason_18x config values after sending the PUSH notification. So caller will hear a fake ringing while the app is wakening.
- kamailio.proxy.report_mos: Enable MOS reporting in the log file.
- kamailio.proxy.set_ruri_to_peer_auth_realm: Set R-URI using peer auth realm.
- kamailio.proxy.start: Enable/disable kamailio-proxy service.
- kamailio.proxy.stir.cache_dir: A path to the directory where to store cached public keys. This directory must be r/w for kamailio user.
- kamailio.proxy.stir.cache_expire: An interval in seconds after which a cached public key is considered expired.
- kamailio.proxy.stir.domains: A list of domains for which STIR is enabled, includes the 'name' - domain name (FQDN), the 'private_key' - a path to the private key.
- kamailio.proxy.stir.enable: Enable or disable STIR/SHAKEN support in Sipwise C5.
- kamailio.proxy.stir.libopt: Optional, set a libsecsipid option. The value has to be a list of options: *name=value*.
- kamailio.proxy.stir.shaken: A sub-block of options related to a treatment of incoming calls (mostly is used to define what are the values to be used in PAI header). For now Sipwise C5 only controls with it an optional parameter 'verstat' for the PAI header.
- kamailio.proxy.stir.timeout: An interval in seconds after which the HTTP GET operation to download the public key times out.
- kamailio.proxy.skip_pbx_loop: Enable or disable the sems pbx loop for pbx subscribers
- kamailio.proxy.store_recentcalls: Store recent calls to redis (used by Malicious Call Identification application and VSCs related to recent calls redial).
- kamailio.proxy.syslog_options: Enable/disable logging of SIP OPTIONS messages to `kamailio-options-proxy.log`.
- kamailio.proxy.tcp_children: Number of TCP worker processes.
- kamailio.proxy.tm.fr_inv_timer: Set INVITE transaction timeout per branch if no final reply for an INVITE arrives after a provisional message was received (branch ringing timeout).
- kamailio.proxy.tm.fr_timer: Set INVITE transaction timeout if the destination is not responding with provisional response message.
- kamailio.proxy.tm.max_inv_lifetime: Set INVITE transaction timeout per the whole transaction if no final reply for an INVITE arrives after a provisional message was received (whole transaction ringing timeout). It has to be equals or greater than kamailio.proxy.tm.fr_inv_timer.
- kamailio.proxy.treat_600_as_busy: Enable the 6xx response handling according to RFC3261. When enabled, the 6xx response should stop the serial forking. Also, CFB will be triggered or busy prompt played as in case of 486 Busy response.
- kamailio.proxy.use_enum: Enable/Disable ENUM feature.
- kamailio.proxy.usrloc_dbmode: Set the mode of database usage for persistent contact storage.
- kamailio.proxy.voicebox_first_caller_cli: When enabled the previous forwarder's CLI will be used as caller CLI in case of chained Call Forwards.

Copyright © Sipwise GmbH - All rights reserved

- kamailio.proxy.xfer_other_party_from: If set to 'yes' transferred calls will have the number of the transferred party in the From header. Default is 'no', thus transferred calls have the number of the transferrer party in the From header.

ngcp-lnpd

The following section defines configuration of LNP daemon, that is used when LNP queries are served by external gateways the so called LNP API mode.

```
lnpd:
  config:
    daemon:
      foreground: 'false'
      json-rpc:
        ports:
          - '8095'
      loglevel: '6'
      sip:
        port: '5095'
      threads: '4'
    instances:
      default:
        module: sigtran
        destination: 0.0.0.0
        from-domain: voip.example.com
        headers:
          - header: INAP-Service-Key
            value: '2'
        reply:
          tcap: raw-tcap
  enable: no
```

- lnpd.enable: Enable/disable LNP daemon
- lnpd.config: details are shown in Configuration of LNP daemon

ngcp-logfs

The following section configures the log obfuscation service.

Copyright © Sipwise GmbH - All rights reserved

```
logfs:
  cache_db: /usr/lib/ngcp-logfs/cache.db
  chmod_dirs: '0555'
  chmod_files: '0444'
  disk_retention_timeout: 365
  enable: yes
  file_cache_timeout: 2
  gid: 0
  log_dir: /var/log/ngcp
  max_mem_usage: 500
  mem_cache_timeout: 24
  mountpoint: /var/log/mirror-ngcp
  suffix: \.\d+$|-\d{8}$|-\d{8}-\d+$
  uid: 0
```

- logfs: details are shown in the section on Log file obfuscation

ngcp-mediator

The following is the ngcp-mediator section:

```
mediator:
  interval: 10
```

- mediator.interval: Running interval of *ngcp-mediator*.

modules

The following is the modules section:

```
modules:
  - enable: no
    name: dummy
    options: numdummies=2
```

- modules: list of configs needed for load kernel modules on boot.
- enable: Enable/disable loading of the specific module (yes/no)
- name: kernel module name
- options: kernel module options if needed

monitoring

The following is the monitoring section:

Copyright © Sipwise GmbH - All rights reserved

```
monitoring:
  backend: prometheus
  filesystems:
  - /
  - /ngcp-data
  - /ngcp-fallback
  - /mnt/glusterfs
  interval: 10
  prometheus_server: victoria-metrics
  retention_policy_long_duration: 12
  retention_policy_short_duration: 15
  retrospect_interval: 30
  threshold:
    cpu_idle_min: '0.1'
    disk_used_max: '0.9'
    kamailio_lb_pkgmem_min: 1048576
    kamailio_lb_shmem_min: '1048576'
    kamailio_proxy_pkgmem_min: 1048576
    kamailio_proxy_shmem_min: '1048576'
    load_long_max: '2'
    load_medium_max: '2'
    load_short_max: '3'
    mem_used_max: 0.98
    mta_queue_len_max: '15'
    mysql_replication_delay_max: 60
    sip_responsiveness_max: '15'
    sslcert_timetoexpiry: '30'
    sslcert_whitelist: []
    swap_free_min: 0.02
  timeout: 10
```

- monitoring.backend: The monitoring implementation backend to use. Valid value is: 'prometheus' (default).
- monitoring.filesystems: The filesystem mount points to monitor.
- monitoring.interval: The number of seconds between each data gathering iteration.
- monitoring.prometheus_server: The prometheus server implementation to use. Either 'victoria-metrics' (default) or 'prometheus'.
- monitoring.retention_policy_long_duration: The long term retention policy for metrics in the monitoring database in months.
- monitoring.retention_policy_short_duration: The short term retention policy for metrics in the monitoring database in days.
- monitoring.restrospect_interval: The number of seconds to look into the past, when checking for the last value for a data point.
- monitoring.threshold.*: These settings specify the thresholds that once crossed will make various components on the system (that is *ngcp-status*, *ngcp-collective-check*, *snmpd* or *monit*) emit alarms or warnings.
- monitoring.threshold.cpu_idle_min: Sets the minimum value for CPU usage (0.1 means 10%).

Copyright © Sipwise GmbH - All rights reserved

- monitoring.threshold.disk_used_max: Sets the maximum value for DISK usage (0.9 means 90%).
- monitoring.threshold.kamailio_lb_pkgmem_min: Sets the minimum value for Kamailio lb package memory usage per process.
- monitoring.threshold.kamailio_lb_shmem_min: Sets the minimum value for Kamailio lb shared memory usage.
- monitoring.threshold.kamailio_proxy_pkgmem_min: Sets the minimum value for Kamailio proxy package memory usage per process.
- monitoring.threshold.kamailio_proxy_shmem_min: Sets the minimum value for Kamailio proxy shared memory usage.
- monitoring.threshold.load_long_max/load_long_max/load_short_max: Max values for load (long, short, medium term).
- monitoring.threshold.mem_used_max: Sets the maximum value for memory usage (0.7 means 70%).
- monitoring.threshold.mta_queue_len_max: Sets the maximum value for the MTA queue length.
- monitoring.threshold.mysql_replication_delay_max: Sets the maximum MySQL replication delay in seconds.
- monitoring.threshold.sip_responsiveness_max: Sets the maximum SIP responsiveness time timeout for the SIP options.
- monitoring.threshold.sslcert_timetoexpiry: Sets the number of days before a SSL certificate expiry starts to warn.
- monitoring.threshold.sslcert_whitelist: Sets a list of SSL certificate fingerprints to whitelist from the expiry check.
- monitoring.threshold.swap_free_min: Sets the minimum value for free swap (0.5 means 50%).
- monitoring.timeout: The timeout for backend queries in seconds, after which the query is considered to have failed due to lack of responsiveness.

nginx

The following is the nginx section:

```
nginx:
  status_port: 8081
  xcap_port: 1080
```

- nginx.status_port: Status port used by nginx server
- nginx.xcap_port: XCAP port used by nginx server

ntp

The following is the ntp server section:

Copyright © Sipwise GmbH - All rights reserved

```
ntp:
  servers:
    - 0.debian.pool.ntp.org
    - 1.debian.pool.ntp.org
    - 2.debian.pool.ntp.org
    - 3.debian.pool.ntp.org
```

- ntp.servers: Define your NTP server list.

ossbss

The following is the ossbss section:

```
ossbss:
  apache:
    port: 2443
    proxyluport: 1080
    restapi:
      sslcertfile: '/etc/ngcp-panel/api_ssl/api_ca.crt'
      sslcertkeyfile: '/etc/ngcp-panel/api_ssl/api_ca.key'
    serveradmin: support@sipwise.com
    servername: "\"myserver\""
    ssl_enable: yes
    sslcertfile: '/etc/ngcp-config/shared-files/ssl/myserver.crt'
    sslcertkeyfile: '/etc/ngcp-config/shared-files/ssl/myserver.key'
  frontend: no
  htpasswd:
    -
      pass: '{SHA}w4zj3mxbmynIQ1jsUEjSkN2z2pk='
      user: ngcpsoap
  logging:
    apache:
      acc:
        facility: daemon
        identity: oss
        level: info
      err:
        facility: local7
        level: info
    ossbss:
      facility: local0
      identity: provisioning
      level: DEBUG
    web:
      facility: local0
      level: DEBUG
  provisioning:
    allow_ip_as_domain: 1
    allow_numeric_usernames: 0
    auto_allow_cli: 1
```

Copyright © Sipwise GmbH - All rights reserved

```
    carrier:
      account_distribution_function: roundrobin
      prov_distribution_function: roundrobin
    credit_warnings:
      -
        domain: example.com
        recipients:
          - nobody@example.com
        threshold: 1000
    faxpw_min_char: 0
    log_passwords: 0
    no_logline_truncate: 0
    pw_min_char: 6
    routing:
      ac_regex: '[1-9]\d{0,4}'
      cc_regex: '[1-9]\d{0,3}'
      sn_regex: '[1-9]\d+'
    tmpdir: '/tmp'
```

- ossbss.frontend: Enable disable SOAP interface. Set value to 'fcgi' to enable old SOAP interface.
- ossbss.htpasswd: Sets the username and SHA hashed password for SOAP access. You can generate the password using the following command: htpasswd -nbs myuser mypassword.
- ossbss.provisioning.allow_ip_as_domain: Allow or not allow IP address as SIP domain (0 is not allowed).
- ossbss.provisioning.allow_numeric_usernames: Allow or not allow numeric SIP username (0 is not allowed).
- ossbss.provisioning.faxpw_min_char: Minimum number of characters for fax passwords.
- ossbss.provisioning.pw_min_char: Minimum number of characters for sip passwords.
- ossbss.provisioning.log_password: Enable logging of passwords.
- ossbss.provisioning.routing: Regexp for allowed AC (Area Code), CC (Country Code) and SN (Subscriber Number).

pbx (only with additional Cloud PBX module activated)

The following is the PBX section:

```
pbx:
  enable: no
```

- pbx.enable: Enable Cloud PBX module.

prosody

The following is the prosody section:

Copyright © Sipwise GmbH - All rights reserved

```
prosody:
  ctrl_port: 5582
  log_level: info
```

- prosody.ctrl_port: XMPP server control port.
- prosody.log_level: Prosody loglevel.

pushd

The following is the pushd section:

```
pushd:
  apns:
    enable: yes
    endpoint: api.push.apple.com
    endpoint_port: 0
    extra_instances:
    - certificate: '/etc/ngcp-config/shared-
files/ssl/PushCallkitCert.pem'
      enable: yes
      key: '/etc/ngcp-config/shared-files/ssl/PushCallkitKey.pem'
      type: callkit
    http2_jwt:
      ec_key: '/etc/ngcp-config/shared-files/ssl/AuthKey_ABCDE12345.pem'
      ec_key_id: 'ABCDE12345'
      enable: yes
      issuer: 'VWXYZ67890'
      tls_certificate: ''
      tls_key: ''
      topic: 'com.example.appID'
    legacy:
      certificate: '/etc/ngcp-config/shared-files/ssl/PushChatCert.pem'
      feedback_endpoint: feedback.push.apple.com
      feedback_interval: '3600'
      key: '/etc/ngcp-config/shared-files/ssl/PushChatKey.pem'
    socket_timeout: 0
  domains:
  - apns:
      endpoint: api.push.apple.com
      extra_instances:
      - certificate: '/etc/ngcp-config/shared-files/ssl/PushCallkitCert-
example.com.pem'
        enable: no
        key: '/etc/ngcp-config/shared-files/ssl/PushCallkitKey-
example.com.pem''
        type: callkit
      http2_jwt:
        ec_key: '/etc/ngcp-config/shared-
files/ssl/AuthKey_54321EDCBA.pem'
        ec_key_id: '54321EDCBA'
```

Copyright © Sipwise GmbH - All rights reserved

```
        issuer: '09876ZYXWV'
        tls_certificate: ''
        tls_key: ''
        topic: 'com.example.otherAppID'
      legacy:
        certificate: '/etc/ngcp-config/shared-files/ssl/PushChatCert-
example.com.pem'
        feedback_endpoint: feedback.push.apple.com
        key: '/etc/ngcp-config/shared-files/ssl/PushChatKey-
example.com.pem'
    domain: example.com
    enable: yes
    android:
      key: 'google_api_key_for_example.com_here'
  enable: yes
  android:
    enable: yes
    key: 'google_api_key_here'
    priority:
      call: high
      groupchat: normal
      invite: normal
      message: normal
  muc:
    exclude: []
    force_persistent: 'true'
    owner_on_join: 'true'
  one_device_per_subscriber: no
  port: 45060
  processes: 4
  ssl: yes
  sslcertfile: /etc/ngcp-config/shared-files/ssl/CAsigned.crt
  sslcertkeyfile: /etc/ngcp-config/shared-files/ssl/CAsigned.key
  unique_device_ids: no
```

- pushd.enable: Enable/Disable the Push Notification feature.
- pushd.apns.enable: Enable/Disable Apple push notification.
- pushd.apns.endpoint: API endpoint hostname or address. Should be one of 'api.push.apple.com' or 'api.development.push.apple.com' for the newer HTTP2/JWT based protocol, or one of 'gateway.push.apple.com' or 'gateway.sandbox.push.apple.com' for the legacy protocol.
- pushd.apns.endpoint_port: API endpoint port. Normally 443 or alternatively 2197 for the newer HTTP2/JWT based protocol, or 2195 for the legacy protocol.
- pushd.apns.legacy: Contains all options specific to the legacy APNS protocol. Ignored when HTTP2/JWT is in use.
- pushd.apns.legacy.certificate: Specify the Apple certificate for push notification https requests from Sipwise C5 to an endpoint.
- pushd.apns.legacy.key: Specify the Apple key for push notification https requests from Sipwise C5 to an endpoint.

Copyright © Sipwise GmbH - All rights reserved

- pushd.apns.legacy.feedback_endpoint: Hostname or address of the APNS feedback service. Normally one of 'feedback.push.apple.com' or 'feedback.sandbox.push.apple.com'.
- pushd.apns.legacy.feedback_interval: How often to poll the feedback service, in seconds.
- pushd.apns.extra_instances: If the iOS app supports Callkit push notifications, they can be enabled here and the required separate certificate and key can be specified. Ignored if HTTP2/JWT is enabled.
- pushd.http2_jwt: Contains all options specific to the newer HTTP2/JWT based APNS API protocol.
- pushd.http2_jwt.ec_key: Name of file that contains the elliptic-curve (EC) cryptographic key provided by Apple, in PEM format.
- pushd.http2_jwt.ec_key_id: 10-digit identification string of the EC key in use.
- pushd.http2_jwt.enable: Master switch for the HTTP2/JWT based protocol. Disables the legacy protocol when enabled.
- pushd.http2_jwt.issuer: Issuer string for the JWT token. Normally the 10-digit team ID string for which the EC key was issued.
- pushd.http2_jwt.tls_certificate: Optional client certificate to use for the TLS connection.
- pushd.http2_jwt.tls_key: Optional private key for the client certificate to use for the TLS connection.
- pushd.http2_jwt.topic: Topic string for the JWT token. Normally the bundle ID for the iOS app.
- pushd.android.enable: Enable/Disable Google push notification.
- pushd.android.key: Specify the Google key for push notification https requests from Sipwise C5 to an endpoint.
- pushd.domains: Supports a separate set of push configurations (API keys, certificates, etc) for all subscribers of the given domain.
- pushd.muc.exclude: list of MUC room jids excluded from sending push notifications.
- pushd.muc.force_persistent: Enable/Disable MUC rooms to be persistent. Needed for Sipwise C5 app to work with other clients.
- pushd.muc.owner_on_join: Enable/Disable all MUC participants to be owners of the MUC room. Needed for Sipwise C5 app to work with other clients.
- pushd.ssl: The security protocol Sipwise C5 uses for https requests from the app in the push notification process.
- pushd.sslcertfile: The trusted certificate file purchased from a CA
- pushd.sslcertkeyfile: The key file that purchased from a CA
- pushd.unique_device_ids: Allows a subscriber to register the app and have the push notification enabled on more than one mobile device.

qos

The QoS section allows configuring the ToS (Type of Service) feature:

```
qos:
  tos_rtp: 184
  tos_sip: 184
```

Copyright © Sipwise GmbH - All rights reserved

- qos.tos_rtp: a ToS value for RTP traffic.
- qos.tos_sip: a ToS value for SIP traffic.

TIP The ToS byte includes both DSCP and ECN bits. So, specify the DSCP value multiplied by four (46x4=184) and, optionally, add the required ECN value to it (1, 2 or 3).

Set the rtpengine.control_tos parameter higher than zero to enable ToS.

ngcp-rate-o-mat

The following is the *ngcp-rate-o-mat* section:

```
rateomat:
  enable: yes
  loopinterval: 10
  splitpeakparts: 0
```

- rateomat.enable: Enable/Disable *ngcp-rate-o-mat*
- rateomat.loopinterval: How long we shall sleep before looking for unrated CDRs again.
- rateomat.splitpeakparts: Whether we should split CDRs on peaktime borders.

redis

The following is the redis section:

```
redis:
  database_amount: 16
  port: 6379
  syslog_ident: redis
```

- redis.database_amout: Set the number of databases in redis. The default database is DB 0.
- redis.port: Accept connections on the specified port, default is 6379
- redis.syslog_ident: Specify the syslog identity.

reminder

The following is the reminder section:

```
reminder:
  retries: 2
  retry_time: 60
  sip_fromdomain: voicebox.sipwise.local
  sip_fromuser: reminder
  wait_time: 30
  weekdays: '2, 3, 4, 5, 6, 7'
```


Copyright © Sipwise GmbH - All rights reserved

- reminder.retries: How many times the reminder feature have to try to call you.
- reminder.retry_time: Seconds between retries.
- reminder.wait_time: Seconds to wait for an answer.

rsyslog

The following is the rsyslog section:

```
rsyslog:
  external_logging:
  - address: 192.168.32.1
    enable: no
    loglevel: warning
    port: '514'
    proto: udp
  ngcp_logs_max_size: 2G
  ngcp_logs_preserve_days: '93'
```

- rsyslog.external_logging: List of remote syslog servers.
- rsyslog.external_logging.address: Address of this remote syslog server.
- rsyslog.external_logging.enable: Enable or disable this remote syslog destination.
- rsyslog.external_logging.loglevel: Minimum log level to send to this syslog destination.
- rsyslog.external_logging.port: Port of this remote syslog server.
- rsyslog.external_logging.proto: Protocol (udp or tcp) for this remote syslog server.
- rsyslog.ngcp_logs_max_size: Specify a maximum size for log files before they are rotated away.
- rsyslog.ngcp_logs_preserve_days: Specify how many days to preserve old rotated log files in /var/log/ngcp/old path.

rtpengine

The following is the rtp proxy section:

Copyright © Sipwise GmbH - All rights reserved

```
rtpengine:
  allow_userspace_only: yes
  cdr_logging_facility: ''
  control_tos: 0
  delete_delay: 30
  dtls_passive: no
  enable: yes
  final_timeout: 0
  firewall_iptables_chain: ''
  graphite:
    interval: 600
    prefix: rtpengine.
    server: ''
  log_level: '6'
  maxport: '40000'
  minport: '30000'
  num_threads: 0
  prefer_bind_on_internal: no
  recording:
    enable: no
    mp3_bitrate: '48000'
    log_level: '6'
    nfs_host: 192.168.1.1
    nfs_remote_path: /var/recordings
    output_dir: /var/lib/rtpengine-recording
    output_format: wav
    output_mixed: yes
    output_single: yes
    resample: no
    resample_to: '16000'
    spool_dir: /var/spool/rtpengine
  rtcp_logging_facility: ''
  rtp_timeout: '60'
  rtp_timeout_onhold: '3600'
```

- rtpengine.allow_userspace_only: Enable/Disable the user space failover for rtpengine ('yes' means enable). By default rtpengine works in kernel space.
- rtpengine.cdr_logging_facility: If set, rtpengine will produce a CDR-like syslog line after each call finishes. Must be set to a valid syslog facility string (such as 'daemon' or 'local0').
- rtpengine.control_tos: If higher than 0, the control messages port uses the configured ToS (Type of Service) bits. See the QoS section below for details.
- rtpengine.delete_delay: After a call finishes, rtpengine will wait this many seconds before cleaning up resources. Useful for possible late branched calls.
- rtpengine.dtls_passive: If enabled, rtpengine will always advertise itself as a passive role in DTLS setup. Useful in WebRTC scenarios if used behind NAT.
- rtpengine.final_timeout: If set, any calls lasting longer than this many seconds will be terminated, no matter the circumstances.
- rtpengine.firewall_iptables_chain: If set, rtpengine will create an iptables rule for each individual

Copyright © Sipwise GmbH - All rights reserved

media port opened in this chain.

- rtpengine.graphite.interval: Interval in seconds between sending updates to the Graphite server.
- rtpengine.graphite.prefix: Graphite keys will be prefixed with this string. Must include a separator character (such as a trailing dot) if one should be used.
- rtpengine.graphite.server: Graphite server to send periodic statistics updates to. Disabled if set to an empty string. Must be in format 'IP:port' or 'hostname:port'.
- rtpengine.log_level: Verbosity of log messages. The default '6' logs everything except debug messages. Increase to 7 to log everything, or decrease to make logging more quiet.
- rtpengine.maxport: Maximum port used by rtpengine for RTP traffic.
- rtpengine.minport: Minimum port used by rtpengine for RTP traffic.
- rtpengine.num_threads: Number of worker threads to use. If set to 0, the number of CPU cores will be used.
- rtpengine.recording.enable: Enable support for call recording.
- rtpengine.recording.mp3_bitrate: If saving audio as MP3, bitrate of the output file.
- rtpengine.recording.log_level: Same as log_level above, but for the recording daemon.
- rtpengine.recording.nfs_host: Mount an NFS share from this host for storage.
- rtpengine.recording.nfs_remote_path: Remote path of the NFS share to mount.
- rtpengine.recording.output_dir: Local mount point for the NFS share.
- rtpengine.recording.output_format: Either 'wav' for PCM output or 'mp3'.
- rtpengine.recording.output_mixed: Create output audio files with all contributing audio streams mixed together.
- rtpengine.recording.output_single: Create separate audio files for each contributing audio stream.
- rtpengine.recording.resample: Resample all audio to a fixed bitrate ('yes' or 'no').
- rtpengine.recording.resample_to: If resampling is enabled, resample to this sample rate.
- rtpengine.recording.spool_dir: Local directory for temporary metadata file storage.
- rtpengine.rtcp_logging_facility: If set, rtpengine will write the contents of all received RTCP packets to syslog. Must be set to a valid syslog facility string (such as 'daemon' or 'local0').
- rtpengine.rtp_timeout: Consider a call dead if no RTP is received for this long (60 seconds).
- rtpengine.rtp_timeout_onhold: Maximum limit in seconds for an onhold (1h).

security

The following is the security section. Usage of the firewall subsection is described in Firewalling:

Copyright © Sipwise GmbH - All rights reserved

```
security:
  firewall:
    enable: no
    logging:
      days_kept: '7'
      enable: yes
      file: /var/log/firewall.log
      tag: NGCPFW
    nat_rules4: ~
    nat_rules6: ~
    policies:
      forward: DROP
      input: DROP
      output: ACCEPT
    rules4: ~
    rules6: ~
```

- security.firewall.enable: Enable/disable iptables configuration and rule generation for IPv4 and IPv6 (default: no)
- security.firewall.logging.days_kept: Number of days logfiles are kept on the system before being deleted (log files are rotated daily, default: 7)
- security.firewall.logging.enable: Enables/disables logging of all packets dropped by Sipwise C5 firewall (default: yes)
- security.firewall.logging.file: File firewall log messages go to (default: /var/log/firewall.log)
- security.firewall.logging.tag: String prepended to all log messages (internally DROP is added to any tag indicating the action triggering the message, default: NGCPFW)
- security.firewall.nat_rules4: Optional list of IPv4 firewall rules added to table nat using iptables-persistent syntax (default: undef)
- security.firewall.nat_rules6: Optional list of IPv6 firewall rules added to table nat using iptables-persistent syntax (default: undef)
- security.firewall.policies.forward: Default policy for iptables FORWARD chain (default: DROP)
- security.firewall.policies.input: Default policy for iptables INPUT chain (default: DROP)
- security.firewall.policies.output: Default policy for iptables OUTPUT chain (default: ACCEPT)
- security.firewall.rules4: Optional list of IPv4 firewall rules added to table filter using iptables-persistent syntax (default: undef)
- security.firewall.rules6: Optional list of IPv6 firewall rules added to table filter using iptables-persistent syntax (default: undef)

sems

The following is the SEMS section:

Copyright © Sipwise GmbH - All rights reserved

```
sems:
  bindport: 5080
  conference:
    enable: yes
    max_participants: 10
  debug: no
  highport: 50000
  lowport: 40001
  media_processor_threads: 10
  prepaid:
    enable: yes
  sbc:
    calltimer_enable: yes
    calltimer_max: 3600
    outbound_timeout: 6000
    profile:
    - custom_header: []
      name: ngcp
    - custom_header: []
      name: ngcp_cf
    sdp_filter:
      codecs: PCMA,PCMU,telephone-event
      enable: yes
      mode: whitelist
    session_timer:
      enable: yes
      max_timer: 7200
      min_timer: 90
      session_expires: 300
  session_processor_threads: 10
  vsc:
    block_override_code: 80
    cfb_code: 90
    cfna_code: 93
    cft_code: 92
    cfu_code: 72
    clir_code: 31
    directed_pickup_code: 99
    enable: yes
    park_code: 97
    reminder_code: 55
    speedial_code: 50
    unpark_code: 98
    voicemail_number: 2000
  xmlrpcport: 8090
```

- b2b.conference.enable: Enable/Disable conference feature.
- b2b.conference.max_participants: Sets the number of concurrent participant.
- b2b.highport: Maximum ports used by sems for RTP traffic.

Copyright © Sipwise GmbH - All rights reserved

- b2b.debug: Enable/Disable debug mode.
- b2b.lowport: Minimum ports used by sems for RTP traffic.
- b2b.prepaid.enable: Enable/Disable prepaid feature.
- b2b.sbc.calltimer_max: Set the default maximum call duration. Note that this value can be overwritten in subscriber/customer/domain preferences setting max_call_duration parameter. Attention: in case of call transfer done by the callee, with max_call_duration set, the timer will be restarted from 0 for the new transferred call.
- b2b.sbc.outbound_timeout: Set INVITE transaction timeout if the destination is not responding with provisional response message.
- b2b.sbc.profile.name: Profile's name where to add the custom headers in 'header_list' config parameter. Supported values: ngcp and ngcp_cf.
- b2b.sbc.profile.custom_header: List of the custom headers that has to be whitelisted (default) by sems sbc in the corresponding profile.
- b2b.sbc.session_timer.enable: If set to "no" all session timer headers are stripped off without considering the session timer related configuration done via the web interface. If set to "yes" the system uses the subscriber/peer configurations values set on the web interface. If set to "transparent" no validation is performed on Session Timer headers, they are ignored by SEMS and therefore negotiated end-to-end.
- b2b.vsc.*: Define here the VSC codes.

sms

This section provides configuration of **S**hort **M**essage **S**ervice on the NGCP. Description of the SMS module is provided earlier in this handbook here.

In the below example you can see the default values of the configuration parameters.

Copyright © Sipwise GmbH - All rights reserved

```
sms:
  core:
    admin_port: '13000'
    smsbox_port: '13001'
  enable: no
  loglevel: '0'
  sendsms:
    max_parts_per_message: '5'
    port: '13002'
  smsc:
    dest_addr_npi: '1'
    dest_addr_ton: '1'
    enquire_link_interval: '58'
    host: 1.2.3.4
    id: default_smsc
    max_pending_submits: '10'
    no_dlr: yes
    password: password
    port: '2775'
    source_addr_npi: '1'
    source_addr_ton: '1'
    system_type: ''
    throughput: '5'
    transceiver_mode: '1'
    username: username
```

- sms.core.admin_port: Port number of admin interface of SMS core module (running on LB nodes).
- sms.core.smsbox_port: Port number used for internal communication between *bearerbox* module on LB nodes and *smsbox* module on PRX nodes. This is a listening port of the *bearerbox* module (running on LB nodes).
- sms.enable: Set to `yes` if you want to enable SMS module.
- sms.loglevel: Log level of SMS module; the default '0' will result in writing only the most important information into the log file.
- sms.sendsms.max_parts_per_message: If the SM needs to be sent as concatenated SM, this parameter sets the max. number of parts for a single (logical) message.
- sms.sendsms.port: Port number of *smsbox* module (running on PRX nodes).
- sms.smsc. : Parameters of the connection to an SMSC

  dest_addr_npi: Telephony numbering plan indicator for the SM destination, as defined by standards (e.g. '1' stands for E.164)

  dest_addr_ton: Type of number for the SM destination, as defined by standards (e.g. '1' stands for "international" format)

  enquire_link_interval: Interval of SMSC link status check in seconds

  host: IP address of the SMSC

  id: An arbitrary string for identification of the SMSC; may be used in log files and for routing SMs.

  max_pending_submits: The maximum number of outstanding (i.e. not acknowledged) SMPP

Copyright © Sipwise GmbH - All rights reserved

operations between Sipwise C5 and SMSC. As a guideline it is recommended that no more than 10 (default) SMPP messages are outstanding at any time.

no_dlr: Do not request delivery report; when sending an SM and this parameter is set to `yes`, Sipwise C5 will not request DR for the message(s). May be required for some particular SMSCs, in order to avoid "Incorrect status report request parameter usage" error messages from the SMSC.

password: This is the password used for authentication on the SMSC.

port: Port number of the SMSC where Sipwise C5 will connect to.

source_addr_npi: Telephony numbering plan indicator for the SM source, as defined by standards (e.g. '1' stands for E.164)

source_addr_ton: Type of number for the SM source, as defined by standards (e.g. '1' stands for "international" format)

system_type: Defines the SMSC client category in which Sipwise C5 belongs to; defaults to "VMA" (Voice Mail Alert) when no value is given. (No need to set any value)

throughput: The max. number of messages per second that Sipwise C5 will send towards the SMSC. (Value type: float)

transceiver_mode: If set to `1` (yes / true), Sipwise C5 will attempt to use a TRANSCEIVER mode connection to the SMSC. It uses the standard transmit port of the SMSC for receiving SMs too.

username: This is the username used for authentication on the SMSC.

snmpd

The following is the snmpd section:

```
snmpd:
  agentx_timeout: 15
  communities:
    - name: public
      sources:
      - localhost
  trap_communities:
    - name: public
      targets:
      - localhost
  traps:
    if:
      link: yes
    ucd:
      disk: yes
      exec: yes
      load: yes
      process: yes
      swap: yes
```

- snmpd.agentx_timeout: Sets the Agent X connection timeout, when communicating with a sub-agent (such as `ngcp-snmp-agent`).

Copyright © Sipwise GmbH - All rights reserved

- snmpd.communities.*: Sets the SNMP community and sources. Entries (i.e. the `sources`) for a community (like *public* in the example) are in a list of hashes format, each line starting with "-" and followed by the name and a list of source addresses.
- snmpd.trap_communities.*: Sets the SNMP TRAP community and destination for traps sent by NGCP. Format is the same as for `snmpd.communities`, but instead of `sources` it uses `targets`. If `snmptrapd` is enabled, it will be automatically configured to listen for any community specified as sending traps to a *localhost* target.
- snmpd.traps.if.*: Enables/disables the emission of SNMP IF MIB traps.
- snmpd.traps.ucd.*: Enables/disables the emission of SNMP UCD MIB traps.

snmptrapd

The following is the snmptrapd section:

```
snmptrapd:
  enable: no
```

- snmptrapd.enable: Enable the snmptrap daemon, to log any trap sent by the *snmpd* daemon into `/var/log/ngcp/snmp-trap.log`.

snmpagent

The following is the SNMP Agent section:

```
snmpagent:
  debug: no
  retrospect_interval: 30
  traps:
    origin: mgmt
    events:
      collective_check: yes
      database: yes
      ha_switchover: yes
      peering: yes
      process: yes
    alarms:
      database: yes
      ha_switchover: yes
      peering: yes
      process: yes
  update_interval: '30'
```

- debug: Enables/disables debug output (that will be logged in `/var/log/ngcp/snmp-agent.log`)
- retrospect_interval: Sets the interval the agent will use when looking into past fetched data.
- traps.origin: Sets the trap emission origin mode. The values can be one of 'legacy', 'mgmt' or 'distributed'.
- traps.events.*: Enables/disables emission of SNMP SIPWISE-NGCP-MONITOR-MIB event traps

Copyright © Sipwise GmbH - All rights reserved

(based on state changes).

- traps.alarms.*: Enables/disables emission of SNMP SIPWISE-NGCP-ALARMS-MIB alarm traps (based on severities and clearances).
- update_interval: Sets the interval in seconds used to update the fetched data.

sshd

The following is the sshd section:

```
sshd:
  listen_addresses:
    - 0.0.0.0
```

- sshd: specify interface where SSHD should run on. By default sshd listens on all IPs found in network.yml with type 'ssh_ext'. Unfortunately sshd can be limited to IPs only and not to interfaces. The current option makes it possible to specify allowed IPs (or all IPs with 0.0.0.0).

sudo

The following is in the sudo section:

```
sudo:
  logging: no
  max_log_sessions: 0
```

- logging: enable/disable the I/O logging feature of sudo. See man page of 'sudoreplay(8)'.
- max_log_sessions: when I/O logging is enabled, specifies how many log sessions per individual user sudo should keep before it starts overwriting old ones. The default '0' means no limit.

voisniff

The following is the voice sniffer section:

 Copyright © Sipwise GmbH - All rights reserved

```
voisniff:
  admin_panel: yes
  daemon:
    custom_bpf: ''
    filter:
      exclude:
      - active: '0'
        case_insensitive: '1'
        pattern: '\ncseq: *\d+ +(register|notify|options)'
      include: []
      sip_ports:
      - 5060
      - 5062
    interfaces:
      extra: []
      types:
      - sip_int
      - sip_ext
      - rtp_ext
    li_x1x2x3:
      call_id:
        del_patterns:
        - _pbx\-1(?:_[0-9]{1,10})?$
        - _b2b\-1(?:_[0-9]{1,10})?$
        - _xfer\-1(?:_[0-9]{1,10})?$
      captagent:
        cin_max: '3000'
        cin_min: '0'
        x2:
          threads: 20
      client_certificate: ''
      enable: no
      fix_checksums: no
      fragmented: no
      interface:
        excludes: []
      local_name: sipwise
      x1:
        port: '18090'
      x23:
        protocol: sipwise
    mysql_dump:
      enable: yes
      max_query_len: 67108864
      num_threads: '4'
    rtp_filter: yes
    start: yes
    threads_per_interface: '2'
  partitions:
    increment: '700000'
    keep: '10'
```

Copyright © Sipwise GmbH - All rights reserved

Parameters commonly used for call statistics retrievable on the web interface and for lawful interception:

- voisniff.daemon.filter.exclude and voisniff.daemon.filter.include: Additional filter to determine packets that need to be excluded from / included in capturing.
- voisniff.daemon.start: Change to yes if you want *ngcp-voisniff* start at boot. Default is no.
- voisniff.daemon.threads_per_interface: Controls how many threads per enabled sniffing interface should be launched.

Parameters used only for call statistics:

- voisniff.admin_panel: Enable/Disable call statistics on Admin interface. Default: no.
- voisniff.daemon.mysql_dump.enable: Needs to be switched to yes to enable call statistics.

The parameters relevant to Lawful Interception are described in Configuration of LI Service

ngcp-witnessd

The following is the ngcp-witnessd tool section:

```
witnessd:
  debug: no
  interval: ~
  gather:
    asr_ner_statistics: yes
    ha_node_force: no
    ha_node_state: yes
    kamailio_concurrent_calls: yes
    kamailio_dialog_active: yes
    kamailio_dialog_early: yes
    kamailio_dialog_incoming: yes
    kamailio_dialog_local: yes
    kamailio_dialog_outgoing: yes
    kamailio_dialog_relay: yes
    kamailio_shmem: yes
    kamailio_usrloc_regdevices: yes
    kamailio_usrloc_regusers: yes
    peering_groups: yes
    mta_queue_len: yes
    mysql_global_status: yes
    mysql_slave_status: yes
    oss_provisioned_subscribers: yes
    sip_responsiveness: yes
    sip_stats_num_packets: yes
    sip_stats_num_packets_perday: yes
    sip_stats_partition_size: yes
```

- witnessd.interval: The number of seconds between each data gathering iteration, when the value is undefined, the code will fallback to use monitoring.interval.

Copyright © Sipwise GmbH - All rights reserved

- witnessd.gather.asr_ner_statistics: Enable ASR/NER statistics data.
- witnessd.gather.ha_node_force: Enable data gathering, even if the High-Availability node status is not active.
- witnessd.gather.ha_node_status: Enable High-Availability node status data.
- witnessd.gather.kamailio_*: Enable Kamailio statistics data.
- witnessd.gather.mta_queue_len: Enable MTA (exim4) queue length data.
- witnessd.gather.mysql_global_status: Enable global MySQL data.
- witnessd.gather.mysql_slave_status: Enable slave (replication) MySQL data.
- witnessd.gather.oss_provisioned_subscribers: Enable OSS provisioned subscribers count data.
- witnessd.gather.sip_*: Enable SIP statistics data.

www_admin

The following is the WEB Admin interface (www_admin) section:

```
www_admin:
  apache:
    autoprov_port: 1444
  callingcard_features: 0
  callthru_features: 0
  conference_features: 1
  contactmail: adjust@example.org
  fastcgi_workers: 2
  fax_features: 1
  fees_csv:
    element_order:
      - source
      - destination
      - direction
      - zone
      - zone_detail
      - onpeak_init_rate
      - onpeak_init_interval
      - onpeak_follow_rate
      - onpeak_follow_interval
      - offpeak_init_rate
      - offpeak_init_interval
      - offpeak_follow_rate
      - offpeak_follow_interval
      - use_free_time
  http_admin:
    autoprov_port: 1444
    port: 1443
    serveradmin: support@sipwise.com
    servername: "\"myserver\""
    ssl_enable: yes
    sslcertfile: '/etc/ngcp-config/shared-files/ssl/myserver.crt'
    sslcertkeyfile: '/etc/ngcp-config/shared-files/ssl/myserver.key'
```

Copyright © Sipwise GmbH - All rights reserved

```
  http_csc:
    autoprov_bootstrap_port: 1445
    autoprov_port: 1444
    port: 443
    serveradmin: support@sipwise.com
    servername: "\"myserver\""
    ssl_enable: yes
    sslcertfile: '/etc/ngcp-config/shared-files/ssl/myserver.crt'
    sslcertkeyfile: '/etc/ngcp-config/shared-files/ssl/myserver.key'
  logging:
    apache:
      acc:
        facility: daemon
        identity: oss
        level: info
      err:
        facility: local7
        level: info
  security:
    password_allow_recovery: 0
    password_max_length: 40
    password_min_length: 6
    password_musthave_digit: 0
    password_musthave_lowercase: 1
    password_musthave_specialchar: 0
    password_musthave_uppercase: 0
    password_sip_autogenerate: 0
    password_sip_expose_subadmin: 1
    password_web_autogenerate: 0
    password_web_expose_subadmin: 1
  speed_dial_vsc_presets:
    vsc:
      - '*0'
      - '*1'
      - '*2'
      - '*3'
      - '*4'
      - '*5'
      - '*6'
      - '*7'
      - '*8'
      - '*9'
```

- www_admin.http_admin.*: Define the Administration interface and certificates.
- www_admin.http_csc.*: Define the Customers interface and certificates.
- www_admin.contactmail: Email to show in the GUI's Error page.

constants.yml Overview

`/etc/ngcp-config/constants.yml` is one of the main configuration files that contains important

Copyright © Sipwise GmbH - All rights reserved

(static) configuration parameters, like Sipwise C5 system-user data.

CAUTION	Sipwise C5 platform administrator should not change content of constants.yml file unless absolutely necessary. Please contact Sipwise Support before changing any of the parameters within the constants.yml file!

network.yml Overview

`/etc/ngcp-config/network.yml` is one of the main configuration files that contains network-related configuration parameters, like IP addresses and roles of the node(s) in Sipwise C5 system.

The next example shows a part of the network.yml configuration file. Explanation of all the configuration parameters is provided in Network Configuration section of the handbook.

Sample host configuration for Sipwise C5

A PRO would look like:

```
sp1:
  dbnode: '1'
  eth0:
    dns_nameservers:
      - 192.168.51.30
      - 192.168.51.31
    gateway: 192.168.22.1
    hwaddr: 06:1e:bc:e2:ec:fb
    ip: 10.0.2.15
    netmask: 255.255.255.0
    shared_ip: ~
    shared_v6ip: ~
    type:
      - web_ext
      - ssh_ext
      - web_int
  eth1:
    hwaddr: 6e:7f:3a:f9:db:1f
    ip: 192.168.255.251
    netmask: 255.255.255.248
    shared_ip:
      - 192.168.255.250
    shared_v6ip: ~
    type:
      - ha_int
      - ssh_ext
  eth2:
    ip: 10.15.20.107
    netmask: 255.255.255.0
    shared_ip:
      - 10.15.20.151
    type:
      - ssh_ext
```

Copyright © Sipwise GmbH - All rights reserved

```
        - web_ext
        - web_int
        - sip_ext
        - rtp_ext
        - mon_ext
    interfaces:
      - lo
      - eth0
      - eth1
      - eth2
    lo:
      advertised_ip: []
      cluster_sets:
        - default
      hwaddr: 00:00:00:00:00:00
      ip: 127.0.0.1
      netmask: 255.0.0.0
      shared_ip: []
      shared_v6ip: []
      type:
        - sip_int
        - web_ext
        - web_int
        - aux_ext
        - ssh_ext
        - api_int
      v6ip: '::1'
    peer: sp2
    role:
      - proxy
      - lb
      - mgmt
      - rtp
      - db
    status: 'online'
hosts_common:
  etc_hosts_global_extra_entries:
  - 10.100.1.1 server-1 server-1.internal.example.com
  - 10.100.1.2 server-2 server-2.internal.example.com
```

A CARRIER would look like:

```
  web01a:
    bond0:
      bond_miimon: '100'
      bond_mode: active-backup
      bond_slaves: 'eth0 eth1'
      hwaddr: 00:00:00:00:00:00
      ip: 192.168.1.2
      netmask: 255.255.255.0
      shared_ip:
```


Copyright © Sipwise GmbH - All rights reserved

```
      - 192.168.1.1
    type:
      - boot_int
  eth0:
    hwaddr: 00:00:00:00:00:00
  eth1:
    hwaddr: 00:00:00:00:00:00
  interfaces:
    - vlan11
    - vlan666
    - vlan35
    - vlan100
    - vlan80
    - vlan90
    - vlan15
    - vlan20
    - lo
    - eth0
    - eth1
    - bond0
  lo:
    advertised_ip: []
    hwaddr: 00:00:00:00:00:00
    ip: 127.0.0.1
    netmask: 255.0.0.0
    shared_ip: []
    shared_v6ip: []
    type:
      - ssh_ext
      - api_int
    v6ip: '::1'
  peer: web01b
  role:
    - mgmt
  status: 'online'
  vlan20:
    advertised_ip: []
    hwaddr: 00:00:00:00:00:00
    ip: 172.31.3.75
    netmask: 255.255.255.240
    shared_ip:
      - 172.31.3.74
    type:
      - web_int
    vlan_raw_device: bond0
    post_up:
      - 'route add -host 172.30.172.247 gw 172.31.3.65 dev vlan20'
  vlan100:
    hwaddr: 00:0a:f7:8d:32:ec
    ip: 172.31.3.5
    netmask: 255.255.255.224
    shared_ip:
```

Copyright © Sipwise GmbH - All rights reserved

```
        - 172.31.3.4
      type:
        - ha_int
        - web_int
        - ssh_ext
      vlan_raw_device: bond0
    vlan11:
      dns_nameservers:
        - 172.31.3.244
        - 192.168.56.11
        - 192.168.57.11
      gateway: 172.31.3.33
      hwaddr: 00:00:00:00:00:00
      ip: 172.31.3.37
      netmask: 255.255.255.224
      shared_ip:
        - 172.31.3.36
      shared_v6ip: []
      type:
        - mon_ext
        - ssh_ext
      vlan_raw_device: bond0
    vlan15:
      hwaddr: 00:00:00:00:00:00
      ip: 192.168.181.201
      netmask: 255.255.255.0
      post_up:
        - 'route add -net 172.25.240.0/24  gw 192.168.181.1 dev vlan15'
        - 'route add -net 192.168.6.0/24 gw 192.168.181.1 dev vlan15'
      shared_ip:
        - 192.168.181.200
      type:
        - ssh_ext
        - web_int
        - mon_ext
      vlan_raw_device: bond0
    vlan35:
      hwaddr: 00:00:00:00:00:00
      ip: 172.31.3.101
      netmask: 255.255.255.240
      shared_ip:
        - 172.31.3.100
      type:
        - sip_int
      vlan_raw_device: bond0
    vlan666:
      hwaddr: 00:00:00:00:00:00
      ip: 46.5.10.37
      netmask: 255.255.255.240
      shared_ip:
        - 46.5.10.36
      type:
```


Copyright © Sipwise GmbH - All rights reserved

```
        - web_ext
      vlan_raw_device: bond0
    vlan80:
      hwaddr: 00:00:00:00:00:00
      ip: 172.31.3.237
      netmask: 255.255.255.248
      shared_ip:
        - 172.31.3.236
      type:
        - phone_ext
        - web_ext
      vlan_raw_device: bond0
      post_up:
        - 'ip route add default via 172.31.3.233 dev vlan80 table
phones_ext'
        - 'ip rule add from 172.31.3.236 lookup phones_ext prio 1000'
    vlan90:
      hwaddr: 00:00:00:00:00:00
      ip: 46.5.10.53
      netmask: 255.255.255.248
      post_up:
        - 'route add -host 77.244.249.93 gw 46.5.10.49 dev vlan90'
      shared_ip:
        - 46.5.10.52
      type:
        - repos_ext
      vlan_raw_device: bond0
hosts_common:
  etc_hosts_global_extra_entries:
  - 10.100.1.1 server-1 server-1.internal.example.com
  - 10.100.1.2 server-2 server-2.internal.example.com
```

NOTE The option 'hosts_common' is optional and it allows administrator to provide extra entries in /etc/hosts.

The administrator can create new entries in network.yml to specify extra entries for the file /etc/hosts. One entry is global and two per-host, one of which is local only for the host, and the other overrides global for this host. These entries will be appended without further processing.

The example of adding new entries using 'ngcpcfg set':

```
ngcpcfg set --diff /etc/ngcp-config/network.yml \
  hosts_common.etc_hosts_global_extra_entries='["10.100.1.1 server-1
server-1.internal.example.com","10.100.1.2 server-2 server-
2.internal.example.com"]'
```

Global is useful if the entries are to be added to all hosts. These probably make more sense in most set-ups.

Per-host local is useful if the entries are only to be added to some node, but not needed or convenient

Copyright © Sipwise GmbH - All rights reserved

to add to all of them.

Per-host override of the global config is useful if the global entries are to be added to a potentially large number of nodes and to be excluded only in a few of them, to not have to do the contrary (duplicate entries in many hosts except a couple).

Example of modifications to network.yml:

```
hosts_common:
  etc_hosts_global_extra_entries:
  - 10.100.1.1 server-1 server-1.internal.example.com
  - 10.100.1.2 server-2 server-2.internal.example.com
hosts:
  db01b:
    etc_hosts_local_extra_entries:
    - 127.0.1.1 local-alias-1.db01b
    - 127.0.2.1 local-alias-2.db01b
    - 172.30.52.180 db01b.example.com
  ...
  web01a:
    etc_hosts_local_extra_entries:
    - 127.0.1.1 local-alias-1.web01a
    - 127.0.2.1 local-alias-2.web01a
    - 172.30.52.168 web01a.example.com
    etc_hosts_global_extra_entries:
    - 10.100.1.1 server-1 server-1.internal.example.com
  ...
```

With this, the output in /etc/hosts for *db01b* will be:

```
  # local extra entries for host 'db01b'
  127.0.1.1 local-alias-1.db01b
  127.0.2.1 local-alias-2.db01b
  172.30.52.180 db01b.example.com

  # global extra entries
  10.100.1.1 server-1 server-1.internal.example.com
  10.100.2.1 server-2 server-2.internal.example.com
```

the content in /etc/hosts on *web01a* will be:

```
  # local extra entries for host 'web01a'
  127.0.1.1 local-alias-1.web01a
  127.0.2.1 local-alias-2.web01a
  172.30.52.168 web01a.example.com

  # global extra entries overridden for host 'web01a'
  10.100.1.1 server-1 server-1.internal.example.com
```


Copyright © Sipwise GmbH - All rights reserved

Appendix D: MariaDB encryption

Overview

MariaDB encryption support (officially called as "Data-at-Rest") enables innodb files, tables and binlogs data encryption so that if copied over the data is not usable without the master key. All the data accessed or modified by clients is encrypted/decrypted on the fly and transparent for the users. The feature comes with a price of 3% to 5% MariaDB performance loss (depending on the hardware, and CPU in particular).

Configuration

There are new options in constants.yml

```
mysql:
  encryption:
    enable: yes
    encrypt_binlog: yes
    key:
1;a356c82422a9031f2e472047ad8220eeea257d611849fbdc9f75b49933f75241
    threads: 1
```

NOTE: all changes in the configuration section will cause the MariaDB server to restart when ngcpcfg templates are applied.

- mysql.encryption.enable: Switch encryption on/off. Values: 'yes','no', Default: 'yes'. When enabled, all tables are being encrypted, it takes from a few seconds to several minutes for MariaDB to encrypt all the data (depending on the overall size) and the encryption procedure is performed in the background, while all the data continues to be fully accessible. Also all new tables are created encrypted by default and it is not possible to disable encryption for specific tables as the encryption is 'forced'.
- mysql.encryption.encrypt_binlog: Encrypt binlogs. Values: 'yes','no', Default: 'yes'. While it is preferred to have this option enabled by default, for scenarios where binlog files need to be parsed, this option can be turned off. It is also possible to use mysqlbinlog with --read-from-remote-server option to read encrypted binlogs.
- mysql.encryption.key: Encryption key. The value is randomly generated during the cfg-schema upgrade when the option is added into constants.yml. The key is located in /etc/mysql/keyfile and normally MUST NOT be changed. Changing or losing the key permanently will render all the MariaDB tablespaces data (databases/tables) unusable.
- mysql.encryption.threads: Amount of encryption threads. Default: 1 How many MariaDB encryption threads should be running, this value depends on how many tables are created/removed or the encryption keys are rotated.

What is not encrypted

- slow-queries log
- mysqld.err log
- general queries log, if enabled

Copyright © Sipwise GmbH - All rights reserved

Data restoration remarks

- When restoring data from an sql backup from another platform it is safe to do that as the currently used 'encryption_key' (inside my.cnf) is not affected this way.
- When copying constants.yml file from another platform and the encryption is enabled, the current 'mysql.encryption.key' (inside constants.yml) must be restored in constants.yml to the same one the MariaDB server is originally started with or it will fail to start otherwise after ngcpcfg apply.

Copyright © Sipwise GmbH - All rights reserved

Appendix E: Disk partitioning

This chapter documents possible disk partitioning on Sipwise C5 available after installation the Sipwise C5. It should be helpful to understand the overall disk partitioning schema.

Supported IO drives

At the moment the following drives are supported: HDD, SSD, and NVMe. We recommend installing NVMe type SSD storage for the best performance. Otherwise, install SATA SSDs for an average performance as SATA hard disks are a good option only for test/development purpose.

The exact model and size depend on the type of the system and the load. We recommend running the initial performance test on the selected hardware before going into production.

Hardware vs. software RAID

Depending on the specific hardware specification, Sipwise will configure either RAID 1 for CARRIER or RAID 10 for PRO (HW-RAID, usually for installations with HDDs), or software RAID (SW-RAID, generally for installations with SSDs).

The default disk partitions

The Sipwise C5 supports the modern concept of installing several releases side by side. The ability to switch between the releases simplifies software upgrades and enables rollbacks. You can find all the benefits here here.

The new partitioning logic is simple. The 'code' of services (e.g., kamailio, MariaDB) is separated from the 'data' (e.g., databases, CDR files) generated and processed by the 'code', and is located in a different partition of the disk. Additionally, there are two partitions for 'code' with different services versions. This way, the version of the code can be switched very quickly, by rebooting the system. The 'data' partition will be the same for both versions of the 'code', and it will always be mounted and ready to be used before the services start.

New partition layout:

Copyright © Sipwise GmbH - All rights reserved

```
  NAME                MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
  sda                   8:0    0    xG  0 disk              # Your disk
with size X Gb
  |-sda1                8:1    0    1M  0 part              # BIOS legacy
boot
  |-sda2                8:2    0  486M  0 part  /boot/efi   # UEFI boot
  `-sda3                8:3    0    yG  0 part              # LVM
partition
    `-md0               9:0    0    yG  0 raid1             # SW RAID (if
requested)
      |-ngcp-root     253:0    0   10G  0 lvm   /           # 'code'
partition
      |-ngcp-fallback 253:1    0   10G  0 lvm               # 'fallback'
partition
      `-ngcp-data     253:3    0    zG  0 lvm   /ngcp-data # 'data'
partition
                                                           # unassigned
space
```

- 1st partition: 1M BIOS boot, for BIOS/GPT (legacy) boot

  this allows fallback to grub-pc package (This partition must have its GUID set to 21686148-6449-6E6F-744E-656564454649 To switch to grub-pc, boot from a rescue/live CD, set to bios_grub with parted, then install grub to disk, so it properly embeds core.img)

- 2nd partition: ~500MB EFI System, for UEFI/GPT boot

  used as /boot/efi, if EFI support is available

- 3rd partition: LVM that is divided into:

  /dev/mapper/ngcp-root with 10GB (rootfs target)
  /dev/mapper/ngcp-fallback with 10GB (for rollback/install/upgrade)

  10% or >=500MB (whichever is bigger) of the remaining space is unassigned to allow LVM snapshots during maintenance

  /dev/mapper/ngcp-data is the **/ngcp-data** partition with the rest of the disk space for the whole platform 'data' (e.g., databases, CDR files, logs, etc.)

NOTE The installer can only boot from GPT and does not support ms-dos partitions anymore. The legacy 'BIOS' systems can also boot from GPT, while (U)EFI systems can only boot from GPT (and not from BIOS/legacy boot).

UEFI

UEFI installation is supported. The dedicated UEFI partition has been created on the disk during the installation (being the second partition in the list).

Swap partition vs. file

Copyright © Sipwise GmbH - All rights reserved

IMPORTANT

The Sipwise C5 performance heavily depends on the IO operations, hence if Swap is used (either the Swap file and/or the Swap partition), the performance might deteriorate. We highly recommend increasing RAM if the platform uses Swap during normal operation.

The Swap partition is no longer in use. The Sipwise C5 has been migrated to the Swap file on the 'data' partition. It gives the following benefits:

- more space is now available for the 'root', 'rollback' and 'data' partitions.
- the Swap file size can be easily changed on the fly (if necessary).
- the Swap file can be migrated to a new location easily: create a new Swap file with the necessary size and location using the 'mkswap' command and activate the new Swap file with 'swapon'. Add the new location to /etc/fstab. Now, you can deactivate the old swapfile with 'swapoff' and remove it to release the disk space.
- The main reason for the Swap partition usage, used to be 'data fragmentation' on hard disk drives (HDDs) and old types of filesystems. For modern SSD drives, the fragmentation issue is irrelevant and the 'ext4' filesystem does not require manual defragmentation either. The free space on fast SSDs is more important nowadays, as it allows storing more 'data'.

Copyright © Sipwise GmbH - All rights reserved

Appendix F: Faxserver Configuration

For an overview of Faxserver architecture and features, please see the Faxserver chapter.

Faxserver Components

Enabling Faxserver

In order to configure functions of Sipwise C5 Faxserver one needs to update the main NGCP configuration file /etc/ngcp-config/config.yml with the correct fax options:

```
faxserver:
  enable: yes
  fail_attempts: '3'
  fail_retry_secs: '60'
  keep_failed_fax: yes
  keep_failed_fax_days: '60'
  keep_received_fax: yes
  keep_received_fax_days: '60'
  keep_sent_fax: yes
  keep_sent_fax_days: '60'
  mail_from: 'Sipwise C5 FaxServer <voipfax@ngcp.sipwise.local>'
```

Parameters are:

- enable: must be yes to enable Faxserver
- fail_...: the number and timeout of fax sending retrials
- keep_...: fax retention definitions: enabling and length in days
- mail_from: the *From* header in the e-mail that is sent by Fax2Mail feature when a fax is received

IMPORTANT

Ensure that in network.yml the *api_int* interface is assigned to the appropriate network interface or on a CARRIER to a VLAN of the node with the *mgmt* role. Usually, this is the same network interface or on a CARRIER the VLAN where the *ha_int* interface is assigned to. The api_int interface must be removed from all other nodes.

Fax Templates Configuration

One needs to update /etc/ngcp-config/templates/etc/ngcp-faxserver/faxserver.conf.tt2 if he wants to use custom content in the fax and e-mail templates that are used by Faxserver to generate the actual fax or e-mail. This may be done under the "User templates" section in the file.

Applying new Faxserver configuration

Once the above mentioned configuration files have been modified the new settings must be applied:

Copyright © Sipwise GmbH - All rights reserved

```
ngcpcfg apply 'Configured fax server'
ngcpcfg push all
```

Fax Services Configuration per Subscriber

Fax services must be explicitly activated for subscribers before they can send or receive faxes. This activation and the custom settings may be set on Sipwise C5 Web panel in the following way (as an administrator):

- Go to *Subscribers* and find the subscriber that you want to modify settings for
- Click on *Preferences* button
- Select *FaxFeatures*

In both sections *Fax2Mail and SendFax* and *Mail2Fax* there is a field: Active. This must be changed from no to yes if the particular fax service must be activated.

When fax services have been activated the user sees a summary of settings in *FaxFeatures* section on his Preferences page:

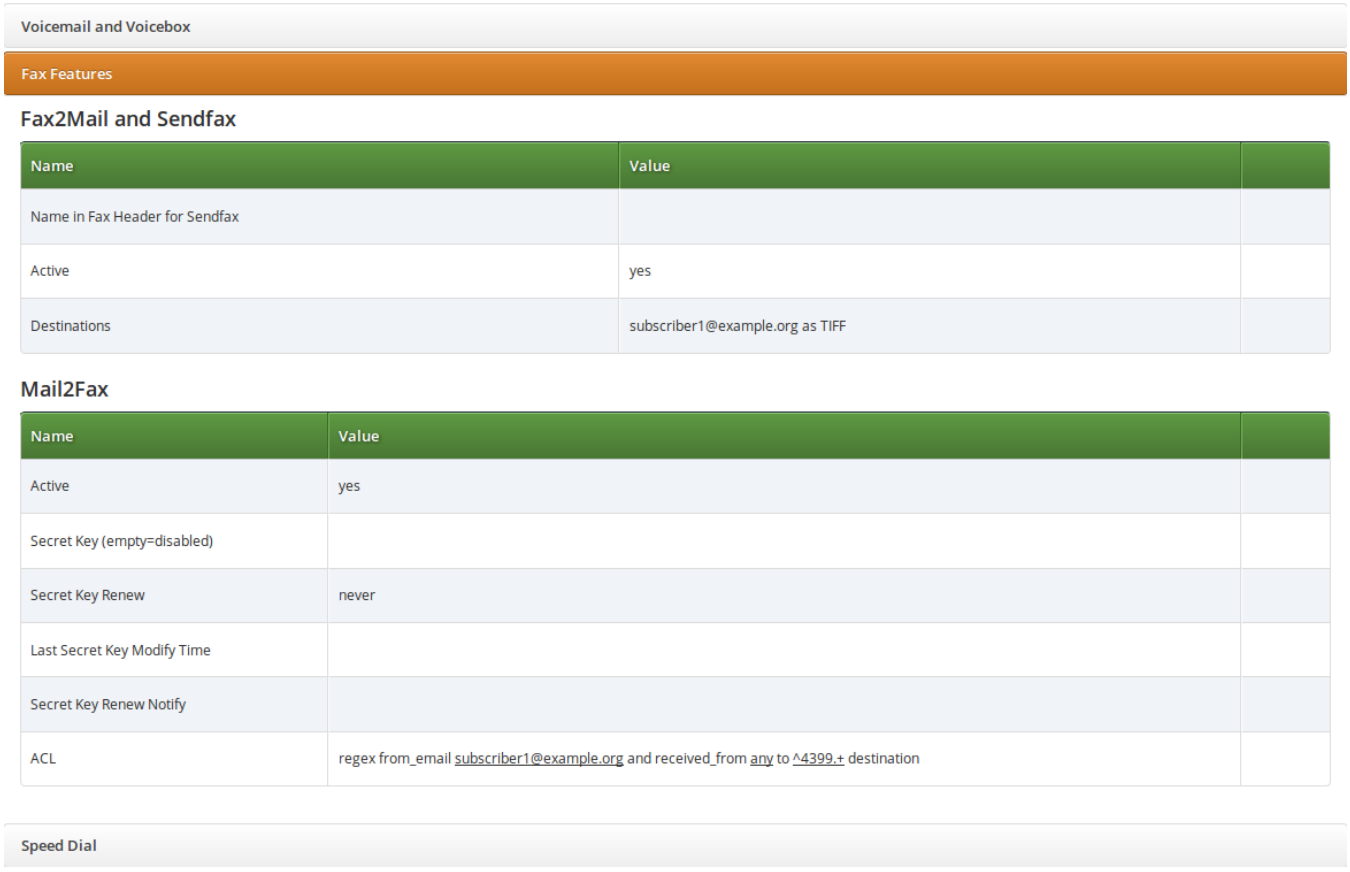


Figure 212. Fax Settings

Details of Fax2Mail, SendFax and Mail2Fax settings are described in subsequent paragraphs.

Copyright © Sipwise GmbH - All rights reserved

Fax2Mail and SendFax Settings

- Name in Fax Header for SendFax: optional field that contains the subscribers name on faxes sent from the Web panel directly
- Destinations: e-mail addresses and selections of notification items that define about which event and where an e-mail is sent; this is a list of such definitions

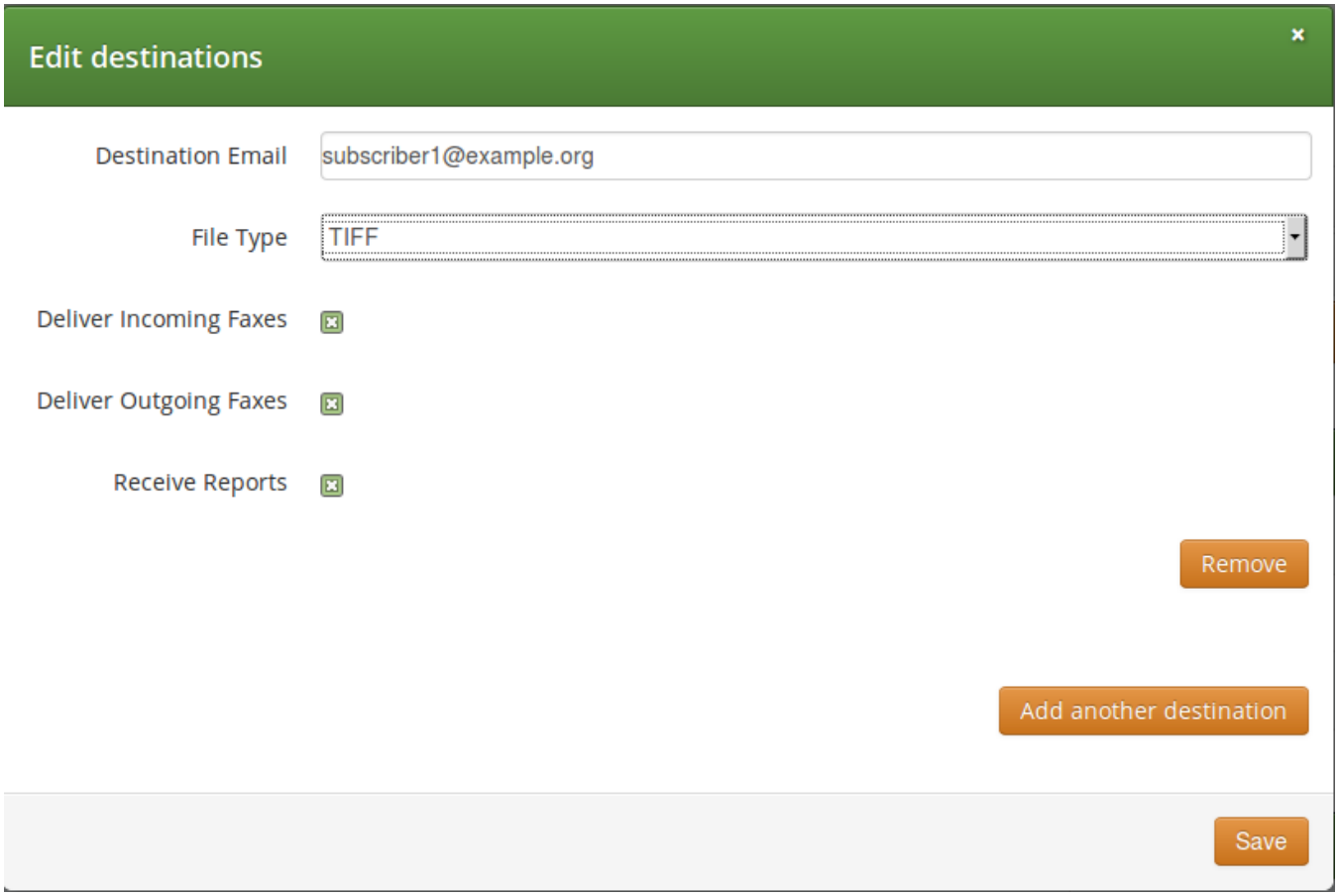


Figure 213. Fax2Mail Destination

The parameters for a destination are as follows:

- Destination Email: the e-mail address where the notification must be sent
- File Type: file format of faxes attached to e-mails
- Deliver Incoming Faxes: select this in order to receive incoming faxes in e-mail
- Deliver Outgoing Faxes: select this in order to receive a report about sent faxes
- Receive Reports: select this in order to receive reports about success / failure of fax transmissions

Mail2Fax Settings

A subscriber can restrict access to his Mail2Fax service with some methods, those can also be combined:

- using a *secret key* that is only known to him, and is inserted in every mail that he sends to Sipwise C5 to be forwarded as fax

Copyright © Sipwise GmbH - All rights reserved

- using an *access control list (ACL)* that determines from which endpoint and for which destination a mail-to-fax is accepted by Sipwise C5 platform
- Secret Key: the secret key used to validate the sender of an e-mail; not used if left empty
- Secret Key Renew: secret key renewal period; Sipwise C5 platform will enforce renewal of the secret key when the defined time has elapsed
- Last Secret Key Modify Time: information about the last secret key modification time
- Secret Key Renew Notify: an e-mail address where the notification about secret key modification is sent
- ACL: access control list, see the details below; this is a list of access control rules

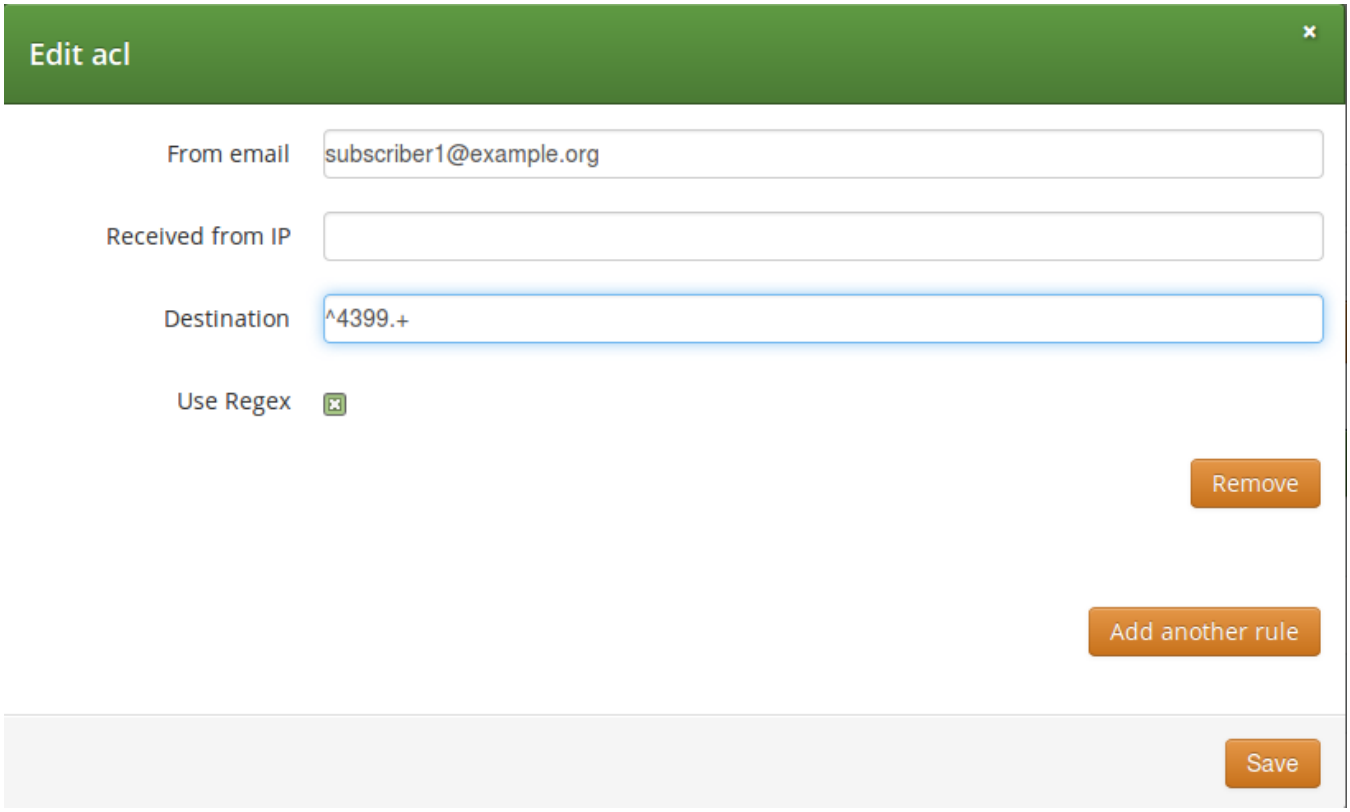


Figure 214. Mail2Fax Access Control List

The parameters for access control rules:

- From email: this sender is allowed to use Mail2Fax service. It can be either an email address or a regular expression (if 'Use Regex' checkbox is ticked). Note: Only the first `To` email header will be considered to obtain the destination subscriber.
- Received from IP: this IP address or host name must be present in any of the `Received` e-mail headers. It can also be a regular expression if 'Use Regex' checkbox is ticked.
- Destination: either a complete phone number in E.164 format, or a regular expression ("Use Regex" checkbox must be ticked) that may define a range of numbers. Examples: "4313334445" as a single number; "^4399.+" as a regular expression: all destinations starting with "4399".

CAUTION When neither *Secret Key*, nor *ACL* is defined then Mail2Fax service will deny accepting any e-mail for sending faxes!

Copyright © Sipwise GmbH - All rights reserved

Sending Fax from Web Panel

A subscriber can log in to his *Customer Self Care* website and send faxes directly from there. In order to do this, one needs to do the following:

- Go to *Settings* *Web Fax* page

  TIP The list of received faxes is also available here.

- Press *Send Fax* button to start entering data, such as recipient and content for the fax being sent:

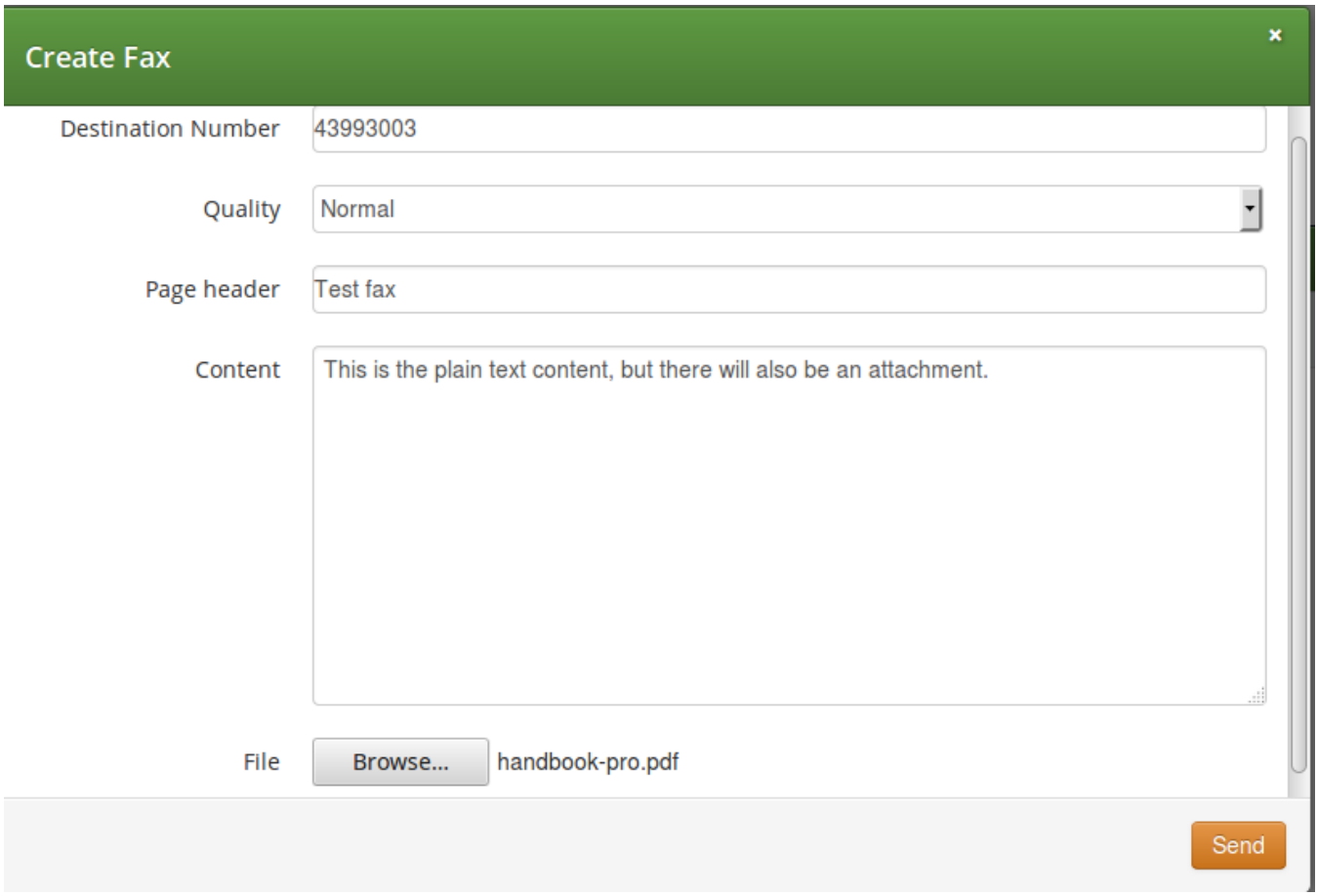


Figure 215. Sending Fax from Web Panel

Both plain text message and attached files can be sent in the fax. First page(s) will contain the plain text message and the content of attached files will follow that.

Faxserver Mail2Fax Configuration

Using Sipwise C5 Faxserver's Mail2Fax service requires the configuration of Sipwise C5's local mail server that is *Exim*. It has to be configured in a way that it can receive mails from outside of the server, because *Exim* by default listens only on the local interfaces for incoming mails.

Exim Configuration

The Sipwise C5 platform administrator must reconfigure *Exim* in order to enable receiving e-mails for fax sending:

Copyright © Sipwise GmbH - All rights reserved

```
sudoedit /etc/ngcp-config/config.yml  # edit section 'email:' according
to your needs
sudo ngcpcfg apply 'adjust exim4 / MTA configuration'
```

PLEASE NOTE: When entering configuration data the following points must be kept in mind:

- operation mode has to be set to "mail sent by smarthost; no local mail"
- "mail2fax.example.org" must be added to accepted domains, where "example.org" is the domain name of Sipwise C5 platform operator

DNS Configuration

It is necessary to add a subdomain starting as mail2fax. to the list of domain names. That is where the faxes will be sent by users to trigger Mail2Fax service.

TIP Alternatively, edit /etc/ngcp-config/templates/etc/exim4/conf.d/router/999_mail2fax.tt2 file and adjust it to your personal preferences. Although this is not recommended and should only be done by Sipwise support engineers.

Sending Fax Using E-mail Clients

When sending an e-mail that should be converted to a fax, there are some points to keep in mind so that Faxserver properly processes the e-mail.

- ***To* header**:

  must contain the subscriber's number who is sending the fax, as the username part of the mail address

  must contain the specific domain starting with mail2fax.
- ***Subject* header**: must contain the fax destination number
- ***Body*** should consist of plain text data
- Adding **attachments** is possible, but only plain text and PDF formats are supported

Secret Key

In order to use the "secret key" access control feature, it should be either put in the first row of the e-mail body followed by an empty line, or included as a plain text attachment. Once it has been validated, it will be removed from the email.

IMPORTANT Either add the secret key to the body, or attach it. Never do both as only one will be recognized and removed, leaving the other one to be sent as part of the fax.

Mail Example

Provided there is a subscriber on Sipwise C5 platform with the 43130111 number, the destination fax is 43130222 and the secret key is "MySecretKey":

Copyright © Sipwise GmbH - All rights reserved

```
From: User Name <username@example.org>
To: 43130111@mail2fax.example.org
Subject: 43130222
- - - - - - - - - - - - - - - - - -
MySecretKey

This is a test fax.

Cheers
```

Managing Faxes via the REST API

It is possible to send and receive faxes and configure fax settings using the built-in REST API interface.

In subsequent sections you can find examples of using the API for sending, receiving faxes and changing fax settings.

Configuring Fax Settings

Retrieving Fax Settings

The following example retrieves the fax settings for the subscriber with ID 3.

```
Method: GET
Content-Type: application/hal+json

https://127.0.0.1:1443/api/faxserversettings/3
```

The output format is as follows (only the relevant output data is shown):

```
  "active" : true,
   "destinations" : [
      {
         "destination" : "user@company.com",
         "filetype" : "PDF14",
         "incoming" : true,
         "outgoing" : true,
         "status" : true
      }
   ],
   "name" : null,
   "password" : null
```

Updating Fax Settings

The following example updates a specific parameter. Namely, it deactivates the fax feature for the subscriber with ID 3.

Copyright © Sipwise GmbH - All rights reserved

```
Method: PATCH
Content-Type: application/json-patch+json

https://127.0.0.1:1443/api/faxserversettings/3

--data-binary '[ { "op" : "replace", "path" : "/active", "value" : 0 }
]'
```

Sending a Fax

The following request sends a PDF file located at /tmp/test_fax.pdf as fax to 431110002 from the subscriber with ID 3.

```
Method: POST
Content-Type: multipart/form-data

https://127.0.0.1:1443/api/faxes/

--form 'json={"destination" : "431110002", "subscriber_id" : 3}' --form
'faxfile=@/tmp/test_fax.pdf'
```

Receiving a Fax

All received faxes are stored on the server and can be retrieved on demand. You can retrieve a stored fax by following these steps:

1. Firstly, obtain the internal ID of the fax:

   ```
   Method: GET
   Content-Type: application/json

   https://127.0.0.1:1443/api/faxes/3
   ```

   This request returns the list of stored faxes for the subscriber with ID 3. One of the available faxes is returned like this:

Copyright © Sipwise GmbH - All rights reserved

```
   "callee" : "431110002",
   "caller" : "431110001",
   "direction" : "out",
   "duration" : "0",
   "filename" : "d9799276-b7d9-454f-98c3-714edf7e3072.tif",
   "id" : 5,
   "pages" : "1",
   "quality" : "8031x7700",
   "reason" : "Normal Clearing / SIP 200 OK [1/3]",
   "signal_rate" : "14400",
   "status" : "SUCCESS",
   "subscriber_id" : 1,
   "time" : "2016-07-30 09:49:59"
```

2. Now, to retrieve the fax with ID 5, use the following request:

```
Method: GET
Content-Type: application/hal+json

https://127.0.0.1:1443/api/faxerecordings/5
```

By default, the fax is in the TIFF format. It is also possible to request it in a different format. To retrieve the same fax in PDF14, use the following request:

```
https://127.0.0.1:1443/api/faxerecordings/5?format=pdf14
```

Configuring Mail2Fax Settings

The configuration of Mail2Fax settings via the REST API is similar to the fax settings configuration.

Retrieving Mail2Fax Configuration

To get the Mail2Fax configuration for the subscriber with ID 3, use the following request:

```
Method: GET
Content-Type: application/hal+json

https://127.0.0.1:1443/api/mailtofaxsettings/3
```

The output format is as follows (only the relevant output data is shown):

Copyright © Sipwise GmbH - All rights reserved

```
"acl" : [],
"active" : false,
"secret_key" : "secretkeypassword",
"secret_key_renew" : "daily",
"secret_renew_notify" : [
   {
      "destination" : "user1@company.com"
   }
]
```

Updating Mail2Fax Configuration

The following set of requests changes the Mail2Fax configuration with new secret key settings.

- Secret key value:

```
Method: PATCH
Content-Type: application/json-patch+json

https://127.0.0.1:1443/api/faxserversettings/3

--data-binary '[ { "op" : "replace", "path" : "/secret_key", "value" :
"newsecretkeypassword" } ]'
```

- Secret key renewal interval:

```
Method: PATCH
Content-Type: application/json-patch+json

--data-binary '[ { "op" : "replace", "path" : "/secret_key_renew",
"value" : "monthly" } ]'
```

- List of email addresses that receive the automatic secret key update notifications:

```
Method: PATCH
Content-Type: application/json-patch+json

--data-binary '[ { "op" : "replace", "path" : "/secret_renew_notify",
"value" : [ { "destination": "user2@company.com" }, { "destination":
"user3@company.com" } ] } ]'
```

Using Advanced Faxserver and Mail2Fax Settings via the REST API

On Sipwise C5 REST API documentation web page you can find the complete list of available Faxserver and Mail2Fax configuration parameters: https://<ngcp_ip_address>:1443/api

Copyright © Sipwise GmbH - All rights reserved

IMPORTANT	The information on the web page is relevant for your platform version and may change in next releases.

After visiting the API documentation main page, you can find the following entries related to Faxserver operations:

- Faxes (https://<ngcp_ip_address>:1443/api/#faxes)
- FaxRecordings (https://<ngcp_ip_address>:1443/api/#faxrecordings)
- FaxserverSettings (https://<ngcp_ip_address>:1443/api/#faxserversettings)

Troubleshooting

The following log file may be used to check Faxserver functionality: /var/log/ngcp/faxserver.log

Session ID (SID)

Faxserver stores basic information about each processed fax in a session file. The most important element within this set of data is the *Session ID* (SID) that uniquely identifies a fax throughout its lifetime.

Session ID is a long hexadecimal string (a kind of UUID) that can be read from the above mentioned Faxserver logfile, and which itself is used also as the filename in files that belong to a specific sent / received fax. An example:

```
root@sp1:~# cat /ngcp-data/spool/faxserver/failed/1e480167-5de6-4cc2-
948b-de58d1a0bb8c.err

created: 2016-09-06 04:41:32
caller: 111111111
callee: 222222222
file: 1e480167-5de6-4cc2-948b-de58d1a0bb8c.tif
sid: 1e480167-5de6-4cc2-948b-de58d1a0bb8c
dir: out
attempts: 0
fail_attempts: 3
fail_retry_secs: 60
quality: normal
status: FAILED
error: Internal error
modified: 2016-09-06 17:41:30

root@sp1:~#
```

The data element sid is the session ID. Other important elements are:

- caller and callee: these are probably searched for when trying to figure out what happened to a specific fax transmission, if you don't know the SID
- dir: direction of fax transmission: 'in'coming or 'out'going or 'mtf' for mail-to-fax
- status: shows success or failure

Copyright © Sipwise GmbH - All rights reserved

- error: the error cause in case of failed faxes

Fax Storage Location

Faxserver stores all of its processed faxes at the path: /ngcp-data/spool/faxserver/... Within that directory the most relevant subdirectories are failed and completed that store the SID file and the fax itself in TIFF format of those faxes that failed or were successful, respectively.

Copyright © Sipwise GmbH - All rights reserved

Appendix G: Storage Node

Overview

Storage is an optional feature that enables one or more pairs of 'storage' nodes that are especially configured for intense DB writing and storing data.

On the 'storage' nodes MariaDB comes with dedicated configuration tuned for extensive writing. Currently 'storage' nodes can be used to redirect and store 'voisniff' traffic from the PRO nodes or the CARRIER 'db' nodes.

From the running processes perspective on the 'storage' nodes there is only MariaDB running (as well as usual mandatory processes like 'rsyslog', 'systemd-timesyncd', 'ssh' and alike).

Deployment

'storage' nodes are not part of the default deployment and must be deployed manually using the NGCP deployment procedure.

Nodes must be deployed with type **stor**.

Default and internal names are:

- stor01 (vip)
- stor01a
- stor01b

Multiple 'storage' pairs are supported ('stor02', 'stor03').

Configuration

Once deployed you can move 'stor_int' type in the network.yml to the interface where the traffic is expected on. It is a good practice to have a dedicated interface for that so that it is separated with SIP/RTP/HA traffic.

Copyright © Sipwise GmbH - All rights reserved

Appendix H: NGCP Internals

This chapter documents internals of Sipwise C5 that should not be usually needed, but might be helpful to understand the overall system.

Pending reboot marker

The Sipwise C5 has the ability to mark a pending reboot for any server, using the file */run/reboot-required*. As soon as the file exists, several components will report about a pending reboot to the end-user. The following components report about a pending reboot right now: *ngcp-status*, *ngcpcfg status*, *motd*, *ngcp-upgrade*. Also, ngcp-upgrade will NOT allow proceeding with an upgrade if it notices a pending reboot. It might affect *rtpengine* dkms module building if there is a pending reboot requested by a newly installed kernel, etc.

Redis id constants

The list of current Sipwise C5 Redis DB IDs:

Service	central (role 'db')	pair	Release	Ticket	Description
sems	-	0	mr3.7.1+	-	HA switchover
rtpengine	-	1	mr3.7.1+	-	HA switchover
proxy	2	-	mr3.7.1+	-	Counter of hunting groups
proxy	3	-	mr3.7.1+	-	Concurrent dialog counters
proxy	-	4	mr3.7.1+	-	List of keys of the central counters
prosody	5	-	mr3.7.1+	-	XMPP cluster
sems	7	-	mr4.1.1+	MT#12707	Sems malicious_call app
captagent	-	8	mr4.1.1+ - mr7.1	MT#15427	Old captagent internal data (unused)
monitoring	9	-	mr4.3+ - mr5.5	MT#31	Old SNMP agent monitoring data (unused)
proxy	10	-	mr4.3+	MT#16079	SIP Loop detection
ngcp-panel sessions	-	19	mr6.3+	TT#35523	Panel login sessions
proxy usrloc	20	-	mr6.2+	TT#32971	SIP registrations

Copyright © Sipwise GmbH - All rights reserved

Service	central (role 'db')	pair	Release	Ticket	Description
proxy acc	-	21	mr6.2+	TT#32971	Accounting records
proxy auth	-	22	mr6.2+	TT#32971	Subscriber data
proxy dialog	-	23	mr6.2+	TT#34100	Dialog data
lb topos	-	24	mr6.5+	TT#40617	Topos data
proxy b2b	25	-	mr8.0+	TT#64404	B2B in use by each subscriber
proxy 181 HIH	26	-	mr8.5+	TT#89301	HIH stored for 181 messages
lb debug_uri	-	27	mr9.1+	TT#93950	Send SIP messages to inactive node
websocket	-	30	mr7.1+	TT#49703	Internal data
websocket monitors	-	31	mr7.1+	TT#49703	Monitors
websocket subscriptions	-	32	mr7.1+	TT#49703	Subscriptions
proxy push	33	-	mr10.0+	TT#131255	Suspended transactions

Monitoring database metrics

Prometheus monitoring metrics

The *Prometheus* monitoring database contains time series metrics of several monitoring sources.

The 'ngcp_' namespace is provided by *ngcp-witnessd*, and here are some of its current matrics subnamespaces:

ngcp_node_	Cluster node information.
ngcp_tls_certs_	TLS certificate information.
ngcp_monit_	Monit supervised processes information.
ngcp_mail_	MTA information.
ngcp_mysql_	MySQL database information.
ngcp_kamailio_	Kamailio statistics information.
ngcp_peer_	Peering information.
ngcp_sip_	SIP statistics information.
ngcp_cdr_	CDR statistics information.

The 'ngcp_node_' namespace consists of the following metrics:

Copyright © Sipwise GmbH - All rights reserved

ngcp_node_active	Cluster node HA state (boolean: 1/0).
ngcp_node_ha_proc_state	Cluster node GCS/CRM process state (boolean: stopped/running).
ngcp_node_ha_host_state	Cluster node host state (boolean: up/down).
ngcp_node_ha_node_state	Cluster node HA state (ngcp-check-active -q).

The 'ngcp_tls_certs_' namespace consists of the following metrics:

ngcp_tls_certs_expires_on	TLS certificate expiration information; the value is the expiration date as seconds since the epoch, with labels: **filename** of the certificate or bundle; **fingerprint** of the certificate, relevant on bundles).

The 'ngcp_monit_' namespace consists of the following metrics:

ngcp_monit_proc_info	Information about the process, the value is always 1, with labels: **name**, **pid**, **ppid**, **proc_status**, **monit_status**, **service_cur_status** is the current systemd service status, **service_exp_status** is the expected systemd service status.
ngcp_monit_proc_children	The number of children.
ngcp_monit_proc_uptime	The process uptime in seconds.
ngcp_monit_proc_cpu_ratio	The CPU usage in for this process.
ngcp_monit_proc_cpu_total_ratio	The CPU usage in for the process group.
ngcp_monit_proc_memory_bytes	The memory in bytes for this process.
ngcp_monit_proc_memory_total_bytes	The memory in bytes for the process group.
ngcp_monit_proc_port_response_seconds	The monitored port response in seconds.
ngcp_monit_proc_sock_response_seconds	The monitored socket response in seconds.
ngcp_monit_proc_data_collected	The timestamp when the data was collected.

The 'ngcp_mysql_' namespace consists of the following metrics:

ngcp_mysql_queries_per_second_average	Average of queries per second.
ngcp_mysql_global_status_innodb_buffer_pool_pages_total	Total number of InnoDB buffer pool pages.
ngcp_mysql_global_status_innodb_buffer_pool_pages_free	Number of free InnoDB buffer pool pages.
ngcp_mysql_global_status_innodb_buffer_pool_pages_dirty	Number of dirty InnoDB buffer pool pages.
ngcp_mysql_global_status_commands_total	Total number of commands used.
ngcp_mysql_slave_status_slave_io_running	Whether the IO slave is running.
ngcp_mysql_slave_status_slave_sql_running	Whether the SQL slave is running.

Copyright © Sipwise GmbH - All rights reserved

ngcp_mysql_slave_status_seconds_behind_master	Seconds behind master replication.
ngcp_mysql_slave_status_last_io_errno	Last IO error description.
ngcp_mysql_slave_status_last_sql_errno	Last SQL error description.

The 'ngcp_peer_' namespace consists of the following metrics:

ngcp_peer_group_info	Peer group information; the value is the group ID, and contains group **name**, **priority** and **description** labels.
ngcp_peer_host_info	Peer host information; the value is the host ID, and contains host **name**, **ip**, and **group_id** labels.
ngcp_peer_host_status	Peer host status, where the value is -1=unknown on standby node, 0=unknown, 1=admin-down, 2=admin-up, 3=pending, 4=down, 5=up; contains the host **id** label.
ngcp_peer_host_cc_inout	Peer host concurrent call (in/out) counter; contains the host **id** label.
ngcp_peer_host_cc_out	Peer host concurrent call (in) counter; contains the host **id** label.

The 'ngcp_sip_' namespaces consists of the following metrics:

ngcp_sip_answer_seizure_ratio	Current ASR (Answer Seizure Ratio).
ngcp_sip_concurrent_calls	Number of concurrent calls.
ngcp_sip_dialog_active	Number of calls currently in active dialog stage.
ngcp_sip_dialog_early	Number of calls currently in early media dialog stage.
ngcp_sip_dialog_incoming	Number of calls currently in incoming dialog stage.
ngcp_sip_dialog_local	Number of calls currently in local dialog stage.
ngcp_sip_dialog_outgoing	Number of calls currently in outgoing dialog stage.
ngcp_sip_dialog_relay	Number of calls currently in relay dialog stage.
ngcp_sip_network_efficiency_ratio	Current NER (Network Efficiency Ratio).
ngcp_sip_packets_total	Total number of SIP packets in storage.
ngcp_sip_packets_total_perday	Total number of SIP packets since 00:00 today.
ngcp_sip_partition_bytes	Size of packets partition.
ngcp_sip_provisioned_subscribers	Provisioned subscribers.
ngcp_sip_registered_devices	Registered devices.
ngcp_sip_registered_subscribers	Registered subscribers.
ngcp_sip_responsiveness_seconds	SIP server responsiveness in seconds.

Copyright © Sipwise GmbH - All rights reserved

The 'ngcp_cdr_' namespaces consists of the following metrics:

ngcp_cdr_total	Total number of generated CDRs.
ngcp_cdr_rated_total	Total number of rated CDRs.

The 'node_' namespace is provided by the *prometheus-node-exporter*.

The 'namedprocess_' namespace is provided by the *prometheus-process-exporter*.

The 'redis_' namespace is provided by the *prometheus-redis-exporter*.

The 'mysql_' and 'mysqld_' namespaces are provided by the *prometheus-mysqld-exporter*.

The 'nginx_' and 'nginxexporter_' namespaces are provided by the *prometheus-nginx-exporter*.

The 'haproxy_' namespace is provided by the *prometheus-haproxy-exporter*.

The 'rtpengine_' namespace is provided by *rtpengine* itself.

The 'go_', 'process_' and 'promhttp_' namespaces are provided by exporters based on the Prometheus Golang libraries and export introspecting metrics for the exporters themselves.

The 'flag_', 'up_', 'promscrape_', 'scrape_', 'vm_', 'vminsert_' and 'vmselect_' namespaces are provided by *victoria-metrics*.

NGCP Preferences

Tables

Currently available tables for preferences are

provisioning.voip_preferences	contains all available preferences, do not contain user data.
provisioning.voip_preference_group	contains preference group names, so the preferences can be put into groups.
provisioning.voip_preferences_enum	contains enum values for preferences, do not contain user data.

The following tables contain user data and depend on voip_preferences and optionally on voip_preferences_enum:

provisioning.voip_dev_preferences	PBX device model preferences
provisioning.voip_devprof_preferences	PBX device profile preferences
provisioning.voip_dom_preferences	domain preferences, replicated by triggers to kamailio.dom_preferences
provisioning.voip_contract_preferences	customer preferences, replicated by triggers to kamailio.contract_preferences
provisioning.voip_peer_preferences	peering server preferences, replicated by triggers to kamailio.peer_preferences

Copyright © Sipwise GmbH - All rights reserved

provisioning.voip_prof_preferences	subscriber profile preferences
provisioning.voip_reseller_preferences	reseller preferences
provisioning.voip_usr_preferences	subscriber preferences, replicated by triggers to kamailio.usr_preferences

Columns

Columns for table 'provisioning.voip_preferences'

id	primary key, used in user tables as the foreign key
voip_preference_groups_id	preference group id
attribute	preference name
label	tooltip that can be used as a mouseover tooltip on the UI
type	0 - 'string', 1 - 'integer'/'boolean'
max_occur	how many preferences with the name are allowed 0: list, 1: only one
usr_pref	defines if the preference can be used in subscribers
prof_pref	defines if the preference can be used in subscriber profiles
dom_pref	defines if the preference can be used in domains
peer_pref	defines if the preference can be used in peering servers
contract_pref	defines if the preference can be used in customers
contract_location_pref	defines if the preference can be used in customer locations
dev_pref	defines if the preference can be used in PBX device models
devprof_pref	defines if the preference can be used in PBX device profiles
fielddev_pref	defines if the preference can be used in PBX devices that are assigned to a subscriber
modify_timestamp	preference last modification time
internal	preference if for internal use only and not shown in the UI/API
expose_to_customer	unused, as now there are dedicated customer preferences table
data_type	data type 'enum', 'boolean', 'int', 'string'
read_only	ready only flag
description	long description of the preference

Copyright © Sipwise GmbH - All rights reserved

dynamic	set to 1 if it is a custom preference that is created by a user (usually for a PBX device model that requires specific preferences) but can be used for all preferences when needed
reseller_pref	defines if the preference can be used in resellers

Enum

All tables are in database "provisioning".

So called "enum preferences" allow a fixed set of possible values, an enumeration, for preferences. Following the differences between other preferences are described.

Setting the attribute "data_type" of table "voip_preferences" to "enum" marks a preferences as an enum. The list of possible options is stored in table "voip_preferences_enum".

Columns for table 'provisioning.voip_preferences_enum' are:

id	primary key
preference_id	Reference to table voip_preferences.
label	A label to be displayed in frontends.
value	Value that will be written to voip_[usr
dom	peer]*preferences.value if it is NOT NULL. Will not be written if it IS NULL. This can be used to implement a "default value" for a preference that is visible in frontends as such (will be listed first if nothing is actually selected), but will not be written to voip*[usr
dom	peer]_preferences.value. Usually forcing a domain or peer default. Should also be named unambiguous (e.g. *"use domain default"*). (Note: Therefore will also not be written to any kamailio table.)
usr_pref	Flag if this is to be used for usr preferences.
dom_pref	Flag if this is to be used for dom preferences.
peer_pref	Flag if this is to be used for peer preferences.
default_val	Flag indicating if this should be used as a default value when creating new entities or introducing new enum preferences (both done via triggers). (Note: For this to work, value must also be set.)

Relevant triggers:

enum_update	Propagates changes of voip_preferences_enum.value to voip_[usr	dom	peer]_preferences.value

Copyright © Sipwise GmbH - All rights reserved

enum_set_default	Will create entries for default values when adding a new enum preference. The default value is the tuple from voip_preferences_enum WHERE default_val=1 AND value NOT NULL.	voip_dom_crepl_trig	The trigger will set possible default values (same condition as for enum_set_default) when creating new domains.
voip_phost_crepl_trig	The trigger will set possible default values (same condition as for enum_set_default) when creating new peers.	voip_sub_crepl_trig	The trigger will set possible default values (same condition as for enum_set_default) when creating new subscribers.

Find a usage example in a section in *db-schema/db_scripts/diff/9086.up*.

Copyright © Sipwise GmbH - All rights reserved

Appendix I: Kamailio pv_headers module

This chapter documents the kamailio "pv_headers" module.

Module overview

This module enables storing all headers in XAVP to freely modify them in the kamailio logic and only apply them once when it's time for the packet to be routed outside. The main goal of the module is to offload the intermediate header processing into the XAVP dynamic container as well as provide with high level methods and pseudovariables to simplify SIP message header modifications.

In few words:

- as soon as a SIP message enters the proxy, kamailio reads all the headers (using the function "pv_collect_headers()") and stores them in an XAVP called "headers".
- starting from this point all the header changes are directly performed on the "headers" XAVP. For example the From header is available at '$xavp(headers[0]From[0])'.
- right before the SIP message leaves the proxy, kamailio writes back all the headers changes (using the function "pv_apply_headers()").

RURI and the headers listed in the module parameter "skip_headers" are left untouched and not saved in the XAVP. Therefore they should be handled in the usual way.

Template changes

As described before in the upgrade procedures, the module is enabled by default in kamailio proxy and all the templates have been already updated to use this new logic. Before proceeding with the upgrade, it is essential that the customtt/patchtt files you have in place are updated to this new format.

Here some few examples of what has been changed in the proxy templates:

- variables $fu, $fU, $fd, $fn, $ft have been substituted by $x_fu, $x_fU, $x_fd, $x_fn, $x_ft
- variables $tu, $tU, $td, $tn, $tt have been substituted by $x_tu, $x_tU, $x_td, $x_tn, $x_tt
- variables $rr, $rs have been substituted by $x_rr, $x_rs
- variables $ua have been substituted by $x_hdr(User-Agent)
- variables $ai have been substituted by $x_hdr(P-Asserted-Identity)
- variables $pU, $pd have been substituted by $x_hdr(P-Preferred-Identity)
- variables $re have been substituted by $x_hdr(Remote-Party-ID)
- variables $di have been substituted by $x_hdr(Diversion)
- variables $ct have been substituted by $x_hdr(Contact)
- $hdr("name") has been substituted by $x_hdr("name")
- is_present_hf("name") has been sustituted by $x_hdr(name)!= $null
- remove_hf("name") has been substituted by pv_remove_header("name") function or $(x_hdr(name)[*]) = $null
- append_hf("name: value\r\n") has been substituted by pv_append_header("name", "value") / pv_modify_header("name", "value") functions or $(x_hdr(name)[*]) = value

Copyright © Sipwise GmbH - All rights reserved

- t_check_status(code) has been substituted by $T_reply_code == code
- save("location") has been updated in save("location", "0x00", "$x_tu")
- sd_lookup("speed_dial") has been updated in sd_lookup("speed_dial", $x_fu)
- added pv_collect_headers() and pv_reset_headers() functions in the dedicated ROUTE_COLLECT_HDR route
- added pv_apply_headers() function in the dedicated ROUTE_APPLY_HDR route
- added pv_reset_headers() function in the following routing sections

Module documentation

Parameters

xavp_name (string)

Name of the XAVP there the collected headers are stored.

Default: headers

```
modparam("pv_headers", "xavp_name", "headers")

Result:
  $xavp(headers[0]=>From)
  $xavp(headers[0]=>To)
  $xavp(headers[0]=>Call-ID)
  ....
```

skip_headers (string)

A comma separated headers list that must be excluded from processing (they are skipped when pv_apply_headers() changes the sip message headers). If the parameter is not set then the "Default" list is used. If the parameter is set to an empty string then all the sip message headers are processed.

Default: Record-Route,Via,Route,Content-Length,Max-Forwards

split_headers (string)

A comma separated headers list that must be split into multi headers if their value is a comma separated list. If the parameter is not set then the "Default" is used. If the parameter is set to an empty string then no headers are split.

Default: None

Copyright © Sipwise GmbH - All rights reserved

```
modparam("pv_headers", "split_headers", "Diversion")

    Result:
        Received Diversion header:
            Diversion:
<user1@test.local>,<user2@test.local>,<user3@test.local>
        After split:
            Diversion: <user1@test.local>
            Diversion: <user2@test.local>
            Diversion: <user3@test.local>
        Becomes handy if used together with pv_modify_header() or
pv_remove_header()
        to change or remove value 2 for instance.
```

Functions

pv_collect_headers()

This function collects all headers from the message into the XAVP. It should be used preferably just when the sip message is reveived by kamailio.

Returns:

- 1 - on success
- -1 - if there were errors

pv_apply_headers()

This function applies the current XAVP headers state to the real headers and should be called only once per branch when the message is about to leave kamailio.

The following rules apply:

- all headers in the XAVP except for ones provided in the "skip_headers" parameter and From/To are recreated in the sip message.
- From/To headers are processed by the uac module if it is loaded.
- From/To headers are not changed in the reply messages.
- headers with NULL value are removed if exist in the sip message.
- the initial order of the sip headers is preserved.

Usage:

Copyright © Sipwise GmbH - All rights reserved

```
if (pv_apply_headers())
{
    "success"
}
else
{
    "errors"
}
```

pv_reset_headers()

This function resets the current XAVP headers list and enables pv_collect_headers() and pv_apply_headers() to be called again in the same branch.

Usage:

```
if (pv_reset_headers())
{
    "success"
}
else
{
    "errors"
}
```

pv_check_header(hname)

This function checks if the header already exists in the XAVP. It can be freely called from anywhere, but only after pv_collect_headers().

Usage:

```
if (pv_check_header(hname))
{
    "exists"
}
else
{
    "does not exist"
}
```

pv_append_header(hname, hvalue)

This function appends a new header into the XAVP. It can be freely called from anywhere, but only after pv_collect_headers(). Please note that subsequent "pv_append_header" calls will result in multiple headers. If the provided "hvalue" is $null then the header is added into the XAVP but it is not going to be added into the message.

Usage:

 Copyright © Sipwise GmbH - All rights reserved

```
if (pv_append_header(hname, hvalue))
{
    "appended"
}
else
{
    "errors"
}
```

pv_modify_header(hname, hvalue)

This function modifies an existing header in the XAVP. It can be freely called from anywhere, but only after pv_collect_headers(). Please note that if the header does not exist it will be explicitly appended. If there are multiple headers with the same name only the first one will be affected. If the provided header value is $null then the header is modified in the XAVP then it is removed from the sip message when pv_apply_headers() is called.

Usage:

```
if (pv_modify_header(hname, hvalue))
{
    "modified"
}
else
{
    "errors"
}
```

pv_modify_header(hname, idx, hvalue)

This function works similar to pv_modify_header(hname, hvalue) but should be used when there are multiple headers with the same name one of them to be modified. Index order is top to bottom.

Usage:

```
if (pv_modify_header(hname, idx, hvalue))
{
    "modified"
}
else
{
    "errors"
}
```

pv_remove_header(hname)

This function removes an existing header from the XAVP. It can be freely called from anywhere, but only after pv_collect_headers(). If there are multiple headers with the same name all of them are

Copyright © Sipwise GmbH - All rights reserved

removed. It returns -1 if the header does not exist.

Usage:

```
if (pv_remove_header(hname, hvalue))
{
    "removed"
}
else
{
    "does not exist or errors"
}
```

pv_remove_header(hname, idx, hvalue)

This function works similar to pv_remove_header(hname, hvalue) but should be used when there are multiple headers with the same name one of them to be removed. Index order is top to bottom.

Usage:

```
if (pv_remove_header(hname, idx, hvalue))
{
    "removed"
}
else
{
    "does not exist or errors"
}
```

Pseudovariables

$x_hdr

This pseudovariable is used to append/modify/remove headers by their name and can be used instead of the pv_append_header(), pv_modify_header(), pv_remove_header() functions.

Usage:

- append header "X-Header" with value "example". NOTE: It always appends a header, even there is already one with the same name

```
$x_hdr(X-Header) = "example";
```

- modify header "X-Header" with index 0. Returns an error if there is no such index

```
$(x_hdr(X-Header)[0]) = "example";
```


Copyright © Sipwise GmbH - All rights reserved

- remove all occurrences of header "X-Header" and append one with value "example"

```
$(x_hdr(X-Header)[*]) = "example";
```

- remove header "X-Header" with index 2 (if there are multiple headers). Returns an error if there is no such index

```
$(x_hdr(X-Header)[2]) = $null;
```

- remove all occurrences of the header. Does not produce an error if there is no such header

```
$(x_hdr(X-Header)[*]) = $null;
```

- retrieve a value of header "X-Header" with index 0, otherwise $null

```
$var(test) = $x_hdr(X-Header);
```

- retrieve a value of header "X-Header" with index 0 otherwise $null

```
$var(test) = $x_hdr(X-Header)[*]);
```

- retrieve a value of header "X-Header" with index 2 otherwise $null

```
$var(test) = $(x_hdr(X-Header)[2]);
```

$x_fu, $x_tu

These pseudovariables are used to modify/retrieve the "From" and "To" headers.

Usage:

- modify the header

```
$x_fu = "User1 <440001@example.local>";
```

- retrieve a value of the header

```
$var(test) = $x_fu;
```

- $x_tu usage is the same

$x_fU, $x_tU

Copyright © Sipwise GmbH - All rights reserved

These pseudovariables are used to modify/retrieve the username part of the "From" and "To" headers.

Usage:

- modify the username part

```
$x_fU = "440001";
```

- retrieve the username part

```
$var(test) = $x_fU;
```

- $x_tU usage is the same

$x_fd, $x_td

These pseudovariables are used to modify/retrieve the domain part of the "From" and "To" headers.

Usage:

- modify the domain part

```
$x_fd = "example.local";
```

- retrieve the domain part

```
$var(test) = $x_fd;
```

- $x_td usage is the same

$x_fn, $x_tn

These pseudovariables are used to modify/retrieve the display part of the "From" and "To" headers.

Usage:

- modify the username part

```
$x_fn = "User1";
```

- retrieve the domain part

```
$var(test) = $x_fn;
```

- $x_tn usage is the same

Copyright © Sipwise GmbH - All rights reserved

$x_ft, $x_tt

These pseudovariables are used to retrieve the tag part of the "From" and "To" headers.

Usage:

- retrieve the tag part

```
$var(test) = $x_ft;
```

- $x_tt usage is the same

$x_rs, $x_rr

These pseudovariables are used to modify/retrieve or change "status" and "code" of the SIP reply NOTE: Only messages with reply status > 300 can be changed as well as reply status 1xx and 2xx cannot be set

Usage:

- modify the reply status

```
$x_rs = 486
```

- retrieve the reply status

```
$var(test) = $x_rs;
```

- modify the reply reason

```
$x_rr = "Custom Reason"
```

- retrieve the reply reason

```
$var(test) = $x_rr;
```

Copyright © Sipwise GmbH - All rights reserved

Appendix J: Extra Configuration Scenarios

AudioCodes devices workaround

Old AudioCodes devices suffer from a problem where they replace 127.0.0.1 address in Record-Route headers (added by Sipwise C5's internal components) with the device's IP address. Supposedly, the whole range of AudioCodes devices with a firmware version below 6.8.X are affected. As a workaround, you may enable the topos feature to stop sending Record-Route headers out. To achieve this, execute the following commands:

```
ngcpcfg set /etc/ngcp-config/config.yml
kamailio.lb.security.topos.enable=yes
ngcpcfg apply 'enable topos for audiocodes devs workaround'
```

"Debug Proxy" for troubleshooting

IMPORTANT This functionality only makes sense on Sipwise C5 CARRIER appliance environment that has an inactive proxy node available.

In order to troubleshoot/debug/capture a scenario, the "debug proxy" allows defining a list of "debug subscribers" that will be matched against From/To headers for every SIP message on a specific lb node. If matched that SIP message will be delivered to the assigned proxy node. In summary, any call to/from that subscriber will be easily traced since no calls are delivered by default to an inactive "debug proxy" node. Also, you can enable extra debug levels on the "debug proxy" node which will NOT affect production traffic on the platform.

IMPORTANT The subscribers have to be specified in the same format as they are received on Kamailio LB (before all the rewrite rules applied).

IMPORTANT In the following examples both proxy node 'prx99a' and 'prx99b' must be set to **inactive** nodes 'hosts.prx99a.status=inactive' and 'hosts.prx99b.status=inactive' in network.yml!

To enable the feature for INACTIVE 'prx99' proxy pair, execute:

```
ngcpcfg set /etc/ngcp-config/network.yml hosts.prx99a.status=inactive #
for the safety
ngcpcfg set /etc/ngcp-config/network.yml hosts.prx99b.status=inactive #
for the safety
ngcpcfg set /etc/ngcp-config/config.yml kamailio.lb.debug_uri.enable=yes
ngcpcfg apply 'Enable debug proxy on node prx99'
ngcpcfg push-parallel all
```

And make sure 'prx99' active node has this new config applied.

There's a command-line tool available to manage the subscriber list. An example of use:

Copyright © Sipwise GmbH - All rights reserved

```
ngcp-debug-subscriber add lb01 +4310001000@example.org
sipuser@example.org prx99
ngcp-debug-subscriber delete lb01 +4310001000@example.org
ngcp-debug-subscriber list lb01
```

Be aware that the list of debug subscribers belongs to just one lb pair, the info is kept in REDIS 'local' database, it is necessary to survive LB restarts and/or HA switchovers. To skip saving in REDIS 'local' database, specifying the option '--no-store' (in this case the information will stay in memory only and will be void on Kamailio LB restart):

```
ngcp-debug-subscriber add --no-store lb01 +10001042@example.org prx99
```

See more available options in general and per-action help messages:

```
ngcp-debug-subscriber --help
ngcp-debug-subscriber add --help
```

WARNING It is recommended to keep the amount of "debug subscribers" as small as possible (for performance reasons).

The "debug subscribers" is kept in a kamailio htable. htable index size value can be changed if necessary in config.yml using the option 'kamailio.lb.debug_uri.htable_idx_size'. This is **not** the maximum size. From Kamailio htable documentation:

```
size - number to control how many slots (buckets) to create for the hash
table.
A larger value means more slots with a higher probability for fewer
collisions.
The actual number of slots (or buckets) created for the table is 2^size.
The possible range for this value is from 2 to 31, smaller or larger
values
will be increased to 3 (8 slots) or decreased to 14 (16384 slots).
Note that each slot can store more than one item, when there are
collisions of
hash ids computed for keys. The items in the same slot are stored in a
linked list.
In other words, the size is not setting a limit of how many items can be
stored in a hash table, as long as there is enough free shared memory,
new items can be added.
```

The default value 'kamailio.lb.debug_uri.htable_idx_size=4' is enough for all the use cases in production.

Copyright © Sipwise GmbH - All rights reserved

Appendix K: NGCP CLI helpers and tools

Main NGCP tools

Sipwise C5 provides a list of various scripts, helpers, and tools to successfully maintain the system from the POSIX console. You can access those scripts using ssh or login into the Unix terminal. All Sipwise C5 scripts start with the prefix 'ngcp-'.

NOTE Currently services and daemon executables namespaced too with 'ngcp-' can be found within PATH, but are not intended for general execution, and might eventually be moved out under /usr/libexec/. These are listed in the table "Internal NGCP component" below.

A must-have list to learn for everyday usage:

- ngcpcfg
- ngcp-approx-cache
- ngcp-loglevel
- ngcp-check-active
- ngcp-collective-check
- ngcp-make-active, ngcp-make-standby
- ngcp-service
- ngcp-status

Public NGCP tools

Name	Description
Configuration tools:	
ngcpcfg	main NGCP configuration tool
ngcp-initial-configuration	configure 'bare' node as NGCP (cluster) member
ngcp-insert-pbx-devices	handle PBX hardware phone firmware/configurations
ngcp-network	command line interface to ngcp-config network configuration settings
ngcp-support-access	enable/disable Sipwise Support team access to NGCP
ngcp-toggle-performance-config	switch NGCP performance modes: low/high
Maintenance tools:	
ngcp-approx-cache	update cached APT metadata in Approx
ngcp-approx-snapshots	manage (create/delete/export/import) snapshots of Approx
ngcp-customtt-diff-helper	prepare local customtt/patchtt files for Sipwise analyses

Copyright © Sipwise GmbH - All rights reserved

Name	Description
ngcp-loglevel	show/set debug level per component in run-time
ngcp-ppa	handle Sipwise PPA APT repositories for NGCP customization
ngcp-prepare-upgrade	prepare NGCP to upgrade to the new release/build
ngcp-reset-db	reset MariaDB database. NOTE: think twice before calling it!
ngcp-sync-db	sync MariaDB instances and configure replication between them
ngcp-update	wrapper for the latest hotfixes installations inside the same release
ngcp-update-cfg-schema	update ngcp-config YML files. Safe to re-execute
ngcp-update-db-schema	upgrade MariaDB. Safe to re-execute
ngcp-upgrade	Sipwise NGCP platform upgrade framework to upgrade on new release/build. Execute with caution!
ngcp-upgrade-pre-checks	check possible issues/misconfigurations before the upgrade
Operational tools:	
ngcp-active-calls	show/drop active calls matching search criteria
ngcp-cdr-md5	validate the integrity of exported CDR files
ngcp-check-active	show HA node status (active/standby)
ngcp-check-redis-dialogs	script to check stalled Kamailio dialogs in Redis DB
ngcp-clish	CISCO-like CLI on a UNIX system. Provides various cluster helpers
ngcp-collective-check	main self-monitoring tool
ngcp-debug-subscriber	forward particular subscriber to "debug proxy" node
ngcp-disk-usage	disk usage statistic tool (symlink to ncdu)
ngcp-kamcmd	Kamailio console debug tool
ngcp-kamctl	Kamailio command-line interface
ngcp-location-cleanup	Delete expired location entries from Redis
ngcp-log-flow	create a call flow of a single Call-ID taking NGCP logs as input
ngcp-logs	search a string in NGCP logs. Useful to retrieve Kamailio and SEMS logs given a call-id
ngcp-lookup-obfuscated	show unobfuscated plain text string from its obfuscated version (GDPR)

Copyright © Sipwise GmbH - All rights reserved

Name	Description
ngcp-make-active	make HA node active (revoke standby HA state from the peer)
ngcp-make-standby	make HA node standby (revoke active HA state from the peer)
ngcp-mariadb-replication-check	check the status of MariaDB replications
ngcp-memory-usage	report processes with the biggest RAM/SWAP usage
ngcp-mysql-compare-dbs	fast compare MariaDB schemas between two hosts
ngcp-mysql-central	connect to 'central' MariaDB instance
ngcp-mysql-local	connect to 'local' MariaDB instance
ngcp-mysql-pair	connect to 'pair' MariaDB instance
ngcp-parallel-scp	parallel copying of local file(s) to cluster nodes
ngcp-parallel-ssh	parallel execution of commands on NGCP cluster nodes
ngcp-redis-helper	easier various data requests from Redis DB
ngcp-service	manage NGCP system services. See 'man ngcp-service' for more details
ngcp-status	show general overall NGCP status. It is a wrapper for many NGCP tools
ngcp-support-upload	upload files to the ticket on support.sipwise.com
ngcp-sync-db-wrapper	wrapper to run ngcp-sync-db easier
ngcp-system-tests	main self-check tool. Safe to execute in production
ngcp-usr-location	show the correct VoIP registrations (from Redis DB). See 'ngcp-usr-location --help'
REST API tools:	
ngcp-api-admins	manage NGCP Administrators
ngcp-api-delete	send arbitrary DELETE request to NGCP REST API
ngcp-api-get	send arbitrary GET request to NGCP REST API
ngcp-api-patch	send arbitrary PATCH request to NGCP REST API
ngcp-api-ping	fast check for NGCP REST API availability
ngcp-api-post	send arbitrary POST request to NGCP REST API
ngcp-api-put	send arbitrary PUT request to NGCP REST API
ngcp-create-customer	create NGCP Customer
ngcp-create-domain	create NGCP Domain
ngcp-create-subscriber	create NGCP Subscriber

Copyright © Sipwise GmbH - All rights reserved

Name	Description
ngcp-delete-domain	delete an NGCP Domain. Execute with caution (subscribers are deleted with a domain)!
ngcp-get-customer	get customer info by the customer ID
ngcp-terminate-customer	terminate an NGCP Customer
ngcp-terminate-subscriber	terminate an NGCP Subscriber
Deprecated tools (will be removed soon):	
ngcp-delete-subscriber	backward compatibility symlink to ngcp-terminate-subscriber
ngcp-delete-voip-account	backward compatibility symlink to ngcp-terminate-customer
ngcp-get-voip-account	backward compatibility symlink to ngcp-get-customer
ngcp-mysql-replication-check	backward compatibility symlink to ngcp-mariadb-replication-check
ngcp-voicemail-table-cleanup	backward compatibility symlink to ngcp-cleanup-voicemail-table

Internal NGCP tools

TIP they can be used, but they might be changed without the notification.

Name	Description
ngcp-bcrypt-webpassword	migration encrypt subscribers' WEB passwords using bcrypt
ngcp-check-rev-applied	check which DB/config revisions have been executed
ngcp-check-sip-option	monitoring tool that sends an OPTIONS request to a SIP server
ngcp-chroot-shell	a suid root program that will take care of securely opening a shell inside a jail
ngcp-create-testusers	developers generate a batch of customers/subscribers
ngcp-dlgcnt-check	check Kamailio dialogs counters in Redis DB
ngcp-dlgcnt-clean	remove Kamailio dialogs from Redis DB
ngcp-dlglist-clean	remove Kamailio queue dialogs from Redis DB
ngcp-ha-clear-failcounts	clean HA failure count for a resource
ngcp-ha-crm-reload	assist with HA peers configuration
ngcp-ha-host-state	keep HA peers in sync (used by ngcp-config)
ngcp-ha-proc-state	keep HA peers in sync (used by ngcp-config)

Copyright © Sipwise GmbH - All rights reserved

Name	Description
ngcp-ha-show-failcounts	print HA failure count for a resource
ngcp-ha-crm	print current CRM in use
ngcp-inventory	generate hardware related report for Sipwise inventory/warranty
ngcp-io-scheduler	systemd helper to set proper IO scheduler on the system boot (HDD related only)
ngcp-kamailio-shm-usage	developers generate Kamailio shared memory usage report
ngcp-location-migrate	temporary migrate from Kamailio locations from MariaDB to Redis DB
ngcp-location-sync	temporary sync Kamailio locations after migration from MariaDB to Redis
ngcp-memdbg-csv	developers generate Kamailio modules memory usage
ngcp-migrate-ha-crm	temporary migrate between Heartbeat and pacemaker/corosync
ngcp-network-validator	dynamically validates the network.yml file (used by ngcp-config)
ngcp-nodename	print the current NGCP HA node name
ngcp-panel-create-keys	generate encryption keys for ngcp-panel
ngcp-peerprobe-status	internal NGCP monitoring tool
ngcp-prepare-translations	developers tool for NGCP localization files. Available on 'trunk' systems only
ngcp-pxe	helper for generation iPXE configuration (used by ngcp-config)
ngcp-screen-check	check if the current session is running inside a screen/tmux
ngcp-ssh	NGCP wrapper for SSH. Used inside various NGCP scripts to access neighbors in the cluster
ngcp-sync-constants	sync MariaDB credentials with /etc/ngcp-config/constants.yml (used by ngcp-config)
ngcp-sync-grants	sync MariaDB grants with /etc/mysql/grants.yml (used by ngcp-config)
ngcp-type	reports back NGCP type: spce/sppro/carrier
ngcp-upgrade-redis-usrloc	move Kamailio locations from MariaDB to Redis in mr7.5
ngcp-virt-identify	check hardware/virtual installation type. See 'man ngcp-virt-identify' for more details

Copyright © Sipwise GmbH - All rights reserved

Internal NGCP component

WARNING do NOT execute them directly. Use ngcp-service to start/stop service.

Name	Description
ngcp-backup	main executable binary file for backup component
ngcp-cdr-exporter	main executable binary file for CDR exporter component
ngcp-cleanup-acc	script responsible for cleaning up accounting database is by cron
ngcp-cleanup-cdr-files	script responsible for cleaning up exported CDR files by cron
ngcp-cleanup-sems	script cleans up SEMS calling card tokens in Redis by cron
ngcp-cleanup-voicemail-table	script cleans up voicemail records on a monthly basis by cron
ngcp-credit-warning	main executable binary file for check for contract balances above credit warning thresholds
ngcp-emergency-mode	main executable binary file for emergency mode component
ngcp-event-exporter	main executable binary file for events exporter component
ngcp-fauditd	main executable binary file for file audit daemon
ngcp-fax	main executable binary file for FAX client component
ngcp-faxserver	main executable binary file for FAX server component
ngcp-fraud-notifier	sends fraud notifications for customers that exceed the threshold
ngcp-installer	main executable binary file for installer component
ngcp-int-cdr-exporter	main executable binary file for intermediate CDRs exporter component
ngcp-licensed	main executable binary file for licensed component
ngcp-lnpd	main executable binary file for lnpd component
ngcp-logfs	main executable binary file for logfs component
ngcp-mediator	main executable binary file for mediator component
ngcp-provisioning-template	create subscribers with detailed settings according to provisioning templates
ngcp-pushd-fcgi	main executable binary file for pushd component

Copyright © Sipwise GmbH - All rights reserved

Name	Description
ngcp-rate-o-mat	main executable binary file for rate-o-mat component
ngcp-reminder	main executable binary file for reminder component
ngcp-rtpengine-iptables-setup	systemd related helper for rtpengine
ngcp-rtpengine-recording-nfs-setup	systemd related helper for rtpengine-recording
ngcp-sems	main executable binary file for B2BUA component
ngcp-snmp-agent	main executable binary file for SNMP agent. It provides access to NGCP SNMP OIDs
ngcp-vmnotify	an Asterisk VoiceMail compatible MWI notification script
ngcp-vmsmsnotify	same as ngcp-vmnotify, but for SMS notifications
ngcp-voisniff	main executable binary file for voisniff component
ngcp-websocket	main executable binary file for websocket component
ngcp-witnessd	main executable binary file for witnessd component

Copyright © Sipwise GmbH - All rights reserved

Appendix L: Generation of 181 Call Is Being Forwarded

[[181_call_is_being_forwarded]]

Overview

SIP message '181 - Call Is Being Forwarded' is an optional provisional response that can be sent towards the caller user to inform it of an ongoing call forward. The message can also contain History-Info headers necessary to the caller to identify which is the new destination of the call.

How to enable it

By default Sipwise C5 doesn't generate any 181 message. To enable this feature two steps are required:

- set config.yml option 'kamailio.proxy.cf_send_181.enable' to 'yes'
- set config.yml option 'b2b.sbc.reset_tag_on_fork' to 'yes'

The first preference will actually activate the 181 message generation. The second one, instead, allows SEMS to forward back to caller all the provisional messages even if they contain a different To-Tag. Without this second preference set, the 181 message is sent towards the caller but all the following 18x messages are blocked by SEMS.

Additionally to add the History-Info headers to the 181 message:

- the option 'outbound_history_info' has to be set in subscriber/domain preferences

How it works

When a call forward is triggered in Sipwise C5, Kamailio Proxy stores in the Redis database, which is selected by 'kamailio.proxy.cf_send_181.redis_db' preference in config.yml, the History-Info headers that are added to the outgoing INVITE message. Kamailio LB receives the outgoing INVITE and it generates a '182 - Connecting' message back towards the caller. The 182 message is received by Kamailio Proxy that converts it in a '181 - Call Is Being Forwarded' message and adds the History-Info headers previously stored in the Redis DB, if any. The message is then sent back to the caller.

Copyright © Sipwise GmbH - All rights reserved

Appendix M: Handling WebRTC Clients

WebRTC is an open project providing browsers and mobile applications with Real-Time Communications (RTC) capabilities. Configuring your platform to offer WebRTC is quite easy and straightforward. This allows you to have a SIP-WebRTC bridge in place and make audio/video call towards normal SIP users from WebRTC clients and vice versa. Sipwise C5 listens, by default, on the following WebSockets and WebSocket Secure: ws://your-ip:5060/ws, wss://your-ip:5061/ws and wss://your-ip:1443/wss/sip/.

The WebRTC subscriber is a normal subscriber which has just a different configuration in his Preferences. You need to change the following preferences under *SubscribersDetailsPreferencesNAT and Media Flow Control*:

- **use_rtpproxy**: Always with rtpproxy as additional ICE candidate
- **transport_protocol**: RTP/SAVPF (encrypted SRTP with RTCP feedback)

The transport_protocol setting may change, depending on your WebRTC client/browser configuration. Supported protocols are the following:

- Transparent (Pass through using the client's transport protocol)
- RTP/AVP (Plain RTP)
- RTP/SAVP (encrypted SRTP)
- RTP/AVPF (RTP with RTCP feedback)
- RTP/SAVPF (encrypted SRTP with RTCP feedback)
- UDP/TLS/RTP/SAVP (Encrypted SRTP using DTLS)
- UDP/TLS/RTP/SAVPF (Encrypted SRTP using DTLS with RTCP feedback)

WARNING	The below configuration is enough to handle a WebRTC client/browser. As mentioned, you may need to tune a little bit your transport_protocol configuration, depending on your client/browser settings.

In order to have a bridge between normal SIP clients (using plain RTP for example) and WebRTC client, the normal SIP clients' preferences have to have the following configuration:

transport_protocol: RTP/AVP (Plain RTP)

This will teach Sipwise C5 to translate between Plain RTP and RTP/SAVPF when you have calls between normal SIP clients and WebRTC clients.

Copyright © Sipwise GmbH - All rights reserved

Appendix N: Batch Provisioning Extras

Built-in Template

NGCP comes with a built-in template with basic settings. It is shown by default when creating a new template via Admin Panel, which can be edited for advanced configurations. It is also possible to define a list of templates in the */etc/ngcp-config/config.yml* file.

NOTE

By default, the built-in template will set the subscriber's SIP URI as `[CC][AC][SN]@[DOMAIN]`, where `[DOMAIN]` is defined within the template as a static value (in the `subscriber.domain` attribute). An alpha-numeric string will be generated automatically for the SIP password. Additionally, the `contract_contact.reseller` and `contract.billing_profile` attributes (among others) may need to be adjusted to particular provisioning needs.

Below, are detailed:

a. The built-in template available in Admin Panel.

b. Configurations for the built-in template in *config.yml* file.

for both JavaScript and Perl languages, respectively.

(a.1) Built-in Admin Panel Template (JavaScript):

Copyright © Sipwise GmbH - All rights reserved

```
fields:
  - name: first_name
    label: "First Name:"
    type: Text
    required: 1
  - name: last_name
    label: "Last Name:"
    type: Text
    required: 1
  - name: cc
    label: "Country Code:"
    type: Text
    required: 1
  - name: ac
    label: "Area Code:"
    type: Text
    required: 1
  - name: sn
    label: "Subscriber Number:"
    type: Text
    required: 1
  - name: sip_username
    type: calculated
    value_code: "function() { return
row.cc.concat(row.ac).concat(row.sn); }"
  - name: purge
    label: "Terminate subscriber, if exists:"
    type: Boolean
contract_contact:
  identifier: "firstname, lastname, status"
  reseller: default
  firstname_code: "function() { return row.first_name; }"
  lastname_code: "function() { return row.last_name; }"
  status: "active"
contract:
  product: "Basic SIP Account"
  billing_profile: "Default Billing Profile"
  identifier: contact_id
  contact_id_code: "function() { return contract_contact.id; }"
subscriber:
  domain: "example.org"
  primary_number:
    cc_code: "function() { return row.cc; }"
    ac_code: "function() { return row.ac; }"
    sn_code: "function() { return row.sn; }"
    username_code: "function() { return row.sip_username; }"
    password_code: "function() { return row.sip_password; }"
subscriber_preferences:
  gpp0: "test"
```


Copyright © Sipwise GmbH - All rights reserved

(a.2) Built-in Admin Panel Template (Perl):

```
fields:
  - name: first_name
    label: "First Name:"
    type: Text
    required: 1
  - name: last_name
    label: "Last Name:"
    type: Text
    required: 1
  - name: cc
    label: "Country Code:"
    type: Text
    required: 1
  - name: ac
    label: "Area Code:"
    type: Text
    required: 1
  - name: sn
    label: "Subscriber Number:"
    type: Text
    required: 1
  - name: sip_username
    type: calculated
    value_code: "sub { return $row{cc}.$row{ac}.$row{sn}; }"
  - name: purge
    label: "Terminate subscriber, if exists:"
    type: Boolean
contract_contact:
  identifier: "firstname, lastname, status"
  reseller: default
  firstname: "sub { return $row{first_name}; }"
  lastname: "sub { return $row{last_name}; }"
  status: "active"
contract:
  product: "Basic SIP Account"
  billing_profile: "Default Billing Profile"
  identifier: contact_id
  contact_id_code: "sub { return $contract_contact{id}; }"
subscriber:
  domain: "example.org"
  primary_number:
    cc_code: "sub { return $row{cc}; }"
    ac_code: "sub { return $row{ac}; }"
    sn_code: "sub { return $row{sn}; }"
    username_code: "sub { return $row{sip_username}; }"
    password_code: "sub { return $row{sip_password}; }"
subscriber_preferences:
  gpp0: "test"
```

Copyright © Sipwise GmbH - All rights reserved

(b.1) Config.yml Template Configuration (JavaScript):

A template can be defined at system level by using the `www_admin.provisioning_templates` property in the *config.yml* file. This template will also be displayed on Admin Panel.

```
www_admin:
  batch_provisioning_features: 1
  provisioning_templates:
    "My First Provisioning Template":
      description: "Create a contract including contact with firstname
and lastname for a single subscriber."
      lang: js
      fields:
        - name: first_name
          label: "First Name:"
          type: Text
          required: 1
        - name: last_name
          label: "Last Name:"
          type: Text
          required: 1
        - name: cc
          label: "Country Code:"
          type: Text
          required: 1
        - name: ac
          label: "Area Code:"
          type: Text
          required: 1
        - name: sn
          label: "Subscriber Number:"
          type: Text
          required: 1
        - name: sip_username
          type: calculated
          value_code: "function() { return
row.cc.concat(row.ac).concat(row.sn); }"
        - name: purge
          label: "Terminate subscriber, if exists:"
          type: Boolean
      contract_contact:
        identifier: "firstname, lastname, status"
        reseller: default
        firstname_code: "function() { return row.first_name; }"
        lastname_code: "function() { return row.last_name; }"
        status: "active"
      contract:
        product: "Basic SIP Account"
        billing_profile: "Default Billing Profile"
        identifier: contact_id
        contact_id_code: "function() { return contract_contact.id; }"
      subscriber:
```


Copyright © Sipwise GmbH - All rights reserved

```
        domain: "example.org"
        primary_number:
          cc_code: "function() { return row.cc; }"
          ac_code: "function() { return row.ac; }"
          sn_code: "function() { return row.sn; }"
        username_code: "function() { return row.sip_username; }"
        password_code: "function() { return row.sip_password; }"
      subscriber_preferences:
        gpp0: "test"
```

Copyright © Sipwise GmbH - All rights reserved

(b.2) Config.yml Template Configuration (Perl):

A template can be defined at system level by using the `www_admin.provisioning_templates` property in the *config.yml* file. This template will also be displayed on Admin Panel.

```
www_admin:
  batch_provisioning_features: 1
  provisioning_templates:
    "My First Provisioning Template":
      description: "Create a contract including contact with firstname
and lastname for a single subscriber."
      fields:
        - name: first_name
          label: "First Name:"
          type: Text
          required: 1
        - name: last_name
          label: "Last Name:"
          type: Text
          required: 1
        - name: cc
          label: "Country Code:"
          type: Text
          required: 1
        - name: ac
          label: "Area Code:"
          type: Text
          required: 1
        - name: sn
          label: "Subscriber Number:"
          type: Text
          required: 1
        - name: sip_username
          type: calculated
          value_code: "sub { return $row{cc}.$row{ac}.$row{sn}; }"
        - name: purge
          label: "Terminate subscriber, if exists:"
          type: Boolean
      contract_contact:
        identifier: "firstname, lastname, status"
        reseller: default
        firstname: "sub { return $row{first_name}; }"
        lastname: "sub { return $row{last_name}; }"
        status: "active"
      contract:
        product: "Basic SIP Account"
        billing_profile: "Default Billing Profile"
        identifier: contact_id
        contact_id_code: "sub { return $contract_contact{id}; }"
      subscriber:
        domain: "example.org"
        primary_number:
```


Copyright © Sipwise GmbH - All rights reserved

```
        cc_code: "sub { return $row{cc}; }"
        ac_code: "sub { return $row{ac}; }"
        sn_code: "sub { return $row{sn}; }"
      username_code: "sub { return $row{sip_username}; }"
      password_code: "sub { return $row{sip_password}; }"
    subscriber_preferences:
      gpp0: "test"
```

Call Forwards Template Example

The following example considers the definition of call forward mappings inside the template. Let us assume that subscribers will be set with the following configuration:

Table 48. Call Forward Mappings Example.

Type	Answer Timeout	Timeset	Sources	To (B-Numbers)	New Destinations	Enabled
Call Forward Busy		always	all sources	any number	123456	Yes
Call Forward Timeout	15s	always	all sources	any number	654321	Yes
Call Forward Unavailable		always	all sources	any number	VoiceMail	Yes

Then, the following `cf_mappings` section can be appended to the batch provisioning template:

Copyright © Sipwise GmbH - All rights reserved

```
cf_mappings:
  cfu: []
  cfb:
    - enabled: 1
      destinationset:
        name: "Phone2"
        destinations:
          - destination: "123456"
            priority: 1
            timeout: 300
  cft:
    - enabled: 1
      destinationset:
        name: "Phone3"
        destinations:
          - destination: "654321"
            priority: 1
            timeout: 300
  cft_ringtimeout: 15
  cfna:
    - enabled: 1
      destinationset:
        name: "VoiceMail"
        destinations:
          - destination: "voicebox"
            priority: 1
            timeout: 300
  cfs: []
  cfr: []
  cfo: []
```


Copyright © Sipwise GmbH - All rights reserved

Appendix O: CSTA

CSTA RequestSystemStatus messages exchange

An example of request:

```
<?xml version="1.0" encoding="UTF-8"?>
<RequestSystemStatus xmlns= http://www.ecma-
international.org/standards/ecma-323/csta/ed6/>
```

An example of answer:

```
<?xml version="1.0" encoding="UTF-8"?>
<RequestSystemStatusResponse xmlns="http://www.ecma-
international.org/standards/ecma-323/csta/ed6">
    <systemStatus>normal</systemStatus>
</RequestSystemStatusResponse>
```

CSTA MonitorStart messages exchange

An example of request:

```
<?xml version="1.0" encoding="UTF-8"?>
<MonitorStart xmlns="http://www.ecma-international.org/standards/ecma-
323/csta/ed6">
    <monitorObject>
        <deviceObject>sip:1000@sipwise.com</deviceObject>
    </monitorObject>
</MonitorStart>
```

An example of answer:

```
<?xml version="1.0" encoding="UTF-8"?>
<MonitorStartResponse xmlns="http://www.ecma-
international.org/standards/ecma-323/csta/ed6">
    <monitorCrossRefID>defr3_1</monitorCrossRefID>
</MonitorStartResponse>
```

Note, that we have an important field here 'monitorCrossRefID', which can be used later on to stop this monitoring session.

CSTA Make Call successful session, internal call

Copyright © Sipwise GmbH - All rights reserved

Activity	A	B	Comments
A Make Call service to a valid device or user is invoked on behalf of device A	MakeCall - callingDevice A - calledDirectoryNumber B - autoOriginate Prompt		
Acknowledgement	MakeCallResult - initiatedCall		
Indication that the service has been initiated from this device	ServiceInitiatedEvent - initiatedConnection - initiatedDevice A - localConnectionState initiated - cause makeCall		The generation of this event is switch specific. The MakeCall cause indicates thatthe device A is being prompted (via ringing, for example) to go off-hook.
Device A goes off hook and is connected in the call	OriginatedEvent - originatedConnection - callingDevice A - calledDevice B - localConnectionState connected - cause newCall		
Device B begins to ring and A receives ringing tone	DeliveredEvent - connection - alertingDevice B - callingDevice A - calledDevice B - lastRedirectionDevice NR - localConnectionState connected - cause newCall	DeliveredEvent - connection - alertingDevice B - callingDevice A - calledDevice B - lastRedirectionDevice NR - localConnectionState alerting - cause newCall	
Device B answers the call by manually going off-hook	EstablishedEvent - establishedConnection - answeringDevice B - callingDevice A - calledDevice B - lastRedirectionDevice NR - localConnectionState connected - cause newCall	EstablishedEvent - establishedConnection - answeringDevice B - callingDevice A - calledDevice B - lastRedirectionDevice NR - localConnectionState connected - cause newCall	

Figure 216. Successful MakeCall scenario

An example of MakeCall request (sent from A, device to be called B):

Copyright © Sipwise GmbH - All rights reserved

```
Content-Type: application/csta+xml
X-CSTA-Seq-ID: ab234_4

<?xml version="1.0" encoding="UTF-8"?>
<MakeCall xmlns="http://www.ecma-international.org/standards/ecma-
323/csta/ed6>
    <callingDevice>sip:A@company1.sipwise.com</callingDevice>

<calledDirectoryNumber>sip:B@company1.sipwise.com</calledDirectoryNumber
>
    <callingConnectionInfo>
        <mediaSessionInfo>G729</mediaSessionInfo>
    </callingConnectionInfo>
</MakeCall>
```

An example of MakeCall response:

```
Content-Type: application/csta+xml
X-CSTA-Seq-ID: ab234_4

<?xml version="1.0" encoding="UTF-8"?>
<MakeCallResponse xmlns="http://www.ecma-
international.org/standards/ecma-323/csta/ed6">
    <callingDevice>
        <callID>02C3KA01EGDRT37QGEJ1LB5AES0000RG</callID>
        <deviceID>sip:A@company1.sipwise.com</deviceID>
    </callingDevice>
</MakeCallResponse>
```

An example of ServiceInitiatedEvent sent to A:

```
Content-Type: application/csta+xml

<?xml version="1.0" encoding="UTF-8"?>
<ServiceInitiatedEvent xmlns="http://www.ecma-
international.org/standards/ecma-323/csta/ed6">
    <monitorCrossRefID>570</monitorCrossRefID>
    <initiatedConnection>
        <callID>02C3KA01EGDRT37QGEJ1LB5AES0000RG</callID>
        <deviceID> sip:A@company1.sipwise.com</deviceID>
    </initiatedConnection>
    <initiatingDevice>
        <deviceIdentifier>sip:A@company1.sipwise.com</deviceIdentifier>
    </initiatingDevice>
    <localConnectionInfo>initiated</localConnectionInfo>
    <cause>makeCall</cause>
</ServiceInitiatedEvent>
```

Copyright © Sipwise GmbH - All rights reserved

An example of OriginatedEvent sent to A:

```
Content-Type: application/csta+xml

<?xml version="1.0" encoding="UTF-8"?>
<OriginatedEvent xmlns="http://www.ecma-
international.org/standards/ecma-323/csta/ed6">
    <monitorCrossRefID>570</monitorCrossRefID>
    <originatedConnection>
        <callID>02C3KA01EGDRT37QGEJ1LB5AES0000RG</callID>
        <deviceID>sip:A@company1.sipwise.com</deviceID>
    </originatedConnection>
    <callingDevice>
        <deviceIdentifier>sip:A@company1.sipwise.com</deviceIdentifier>
    </callingDevice>
    <calledDevice>
        <deviceIdentifier>sip:B@company1.sipwise.com</deviceIdentifier>
    </calledDevice>
    <localConnectionInfo>connected</localConnectionInfo>
    <cause>newCall</cause>
</OriginatedEvent>
```

An example of DeliveredEvent sent to A:

```
Content-Type: application/csta+xml

<?xml version="1.0" encoding="UTF-8"?>
<DeliveredEvent xmlns="http://www.ecma-international.org/standards/ecma-
323/csta/ed6">
    <monitorCrossRefID>570</monitorCrossRefID>
    <connection>
        <callID>02C3KA01EGDRT37QGEJ1LB5AES0000RG</callID>
        <deviceID>sip:B@company1.sipwise.com</deviceID>
    </connection>
    <alertingDevice>
        <deviceIdentifier>sip:B@company1.sipwise.com</deviceIdentifier>
    </alertingDevice>
    <callingDevice>
        <deviceIdentifier>sip:A@company1.sipwise.com</deviceIdentifier>
    </callingDevice>
    <calledDevice>
        <deviceIdentifier>sip:B@company1.sipwise.com</deviceIdentifier>
    </calledDevice>
    <lastRedirectionDevice>
        <notRequired/>
    </lastRedirectionDevice>
    <localConnectionInfo>connected</localConnectionInfo>
    <cause>newCall</cause>
</DeliveredEvent>
```


Copyright © Sipwise GmbH - All rights reserved

An example of DeliveredEvent sent to B:

```
Content-Type: application/csta+xml

<?xml version="1.0" encoding="UTF-8"?>
<DeliveredEvent xmlns="http://www.ecma-international.org/standards/ecma-
323/csta/ed6">
    <monitorCrossRefID>571</monitorCrossRefID>
    <connection>
        <callID>02C3KA01EGDRT37QGEJ1LB5AES0000RG</callID>
        <deviceID>sip:B@company1.sipwise.com</deviceID>
    </connection>
    <alertingDevice>
        <deviceIdentifier>sip:B@company1.sipwise.com</deviceIdentifier>
    </alertingDevice>
    <callingDevice>
        <deviceIdentifier>sip:A@company1.sipwise.com</deviceIdentifier>
    </callingDevice>
    <calledDevice>
        <deviceIdentifier>sip:B@company1.sipwise.com</deviceIdentifier>
    </calledDevice>
    <lastRedirectionDevice>
        <notRequired/>
    </lastRedirectionDevice>
    <localConnectionInfo>alerting</localConnectionInfo>
    <cause>newCall</cause>
</DeliveredEvent>
```

An example of EstablishedEvent sent to A:

Copyright © Sipwise GmbH - All rights reserved

```
Content-Type: application/csta+xml
<?xml version="1.0" encoding="UTF-8"?>
<EstablishedEvent xmlns="http://www.ecma-
international.org/standards/ecma-323/csta/ed6">
    <monitorCrossRefID>570</monitorCrossRefID>
    <establishedConnection>
        <callID>02C3KA01EGDRT37QGEJ1LB5AES0000RG</callID>
        <deviceID>sip:B@company1.sipwise.com</deviceID>
    </establishedConnection>
    <answeringDevice>
        <deviceIdentifier>sip:B@company1.sipwise.com</deviceIdentifier>
    </answeringDevice>
    <callingDevice>
        <deviceIdentifier>sip:A@company1.sipwise.com</deviceIdentifier>
    </callingDevice>
    <calledDevice>
        <deviceIdentifier>sip:B@company1.sipwise.com</deviceIdentifier>
    </calledDevice>
    <lastRedirectionDevice>
        <notRequired/>
    </lastRedirectionDevice>
    <localConnectionInfo>connected</localConnectionInfo>
    <cause>newCall</cause>
</EstablishedEvent>
```

An example of EstablishedEvent sent to B:

Copyright © Sipwise GmbH - All rights reserved

```
<?xml version="1.0" encoding="UTF-8"?>
<EstablishedEvent xmlns="http://www.ecma-
international.org/standards/ecma-323/csta/ed6">
    <monitorCrossRefID>571</monitorCrossRefID>
    <establishedConnection>
        <callID>02C3KA01EGDRT37QGEJ1LB5AES0000RG</callID>
        <deviceID>sip:1001@company1.sipwise.com</deviceID>
    </establishedConnection>
    <answeringDevice>

<deviceIdentifier>sip:1001@company1.sipwise.com</deviceIdentifier>
    </answeringDevice>
    <callingDevice>

<deviceIdentifier>sip:1000@company1.sipwise.com</deviceIdentifier>
    </callingDevice>
    <calledDevice>

<deviceIdentifier>sip:1001@company1.sipwise.com</deviceIdentifier>
    </calledDevice>
    <lastRedirectionDevice>
        <notRequired/>
    </lastRedirectionDevice>
    <localConnectionInfo>connected</localConnectionInfo>
    <cause>newCall</cause>
</EstablishedEvent>
```

CSTA Make Call failed session, internal call (destination is busy)

Copyright © Sipwise GmbH - All rights reserved

Activity	A	B	Comments
A Make Call service to a valid device or user is invoked on behalf of device A	MakeCall - callingDevice A - calledDirectoryNumber B - autoOriginate Prompt		
Acknowledgement	MakeCallResult - initiatedCall		
Indication that the service has been initiated from this device	ServiceInitiatedEvent - initiatedConnection - initiatedDevice A - localConnectionState initiated - cause makeCall		The generation of this event is switch specific. The MakeCall cause indicates thatthe device A is being prompted (via ringing, for example) to go off-hook.
Device A goes off hook and is connected in the call	OriginatedEvent - originatedConnection - callingDevice A - calledDevice B - localConnectionState connected - cause newCall		
Device B is busy – the call cannot be completed.	FailedEvent - failedConnection - failingDevice B - callingDevice A - calledDevice B - lastRedirectionDevice NR - localConnectionState connected - cause busy	FailedEvent - failedConnection - failingDevice B - callingDevice A - calledDevice B - lastRedirectionDevice NR - localConnectionState failed - cause busy	
Event indicates that connection has been removed from the call	ConnectionClearedEvent - droppedConnection - releasingDevice B - localConnectionState connected - eventCause normalCleaning	ConnectionClearedEvent - droppedConnection - releasingDevice B - localConnectionState null - eventCause normalCleaning	
Since A is the only device in the call, it is cleared as the result of B beigng cleared	ConnectionClearedEvent - droppedConnection - releasingDevice A - localConnectionState null - eventCause normalCleaning		

Figure 217. Failed MakeCall scenario

CSTA Make Call successful session, outbound scenario

Copyright © Sipwise GmbH - All rights reserved

Activity	A	Peering B	Comments
A Make Call service to a valid device outside the CSTA subsystem is invoked on behalf of device A	MakeCallRequest - callingDevice A - calledDirectoryNumber B - autoOriginate Prompt		The Make Call service specifies that device A should be prompted to go off-hook
Acknowledgement	MakeCallResult - initiatedCall		
Indication that the service has been initiated from this device	ServiceInitiatedEvent - initiatedConnection - initiatedDevice A - localConnectionState initiated - cause makeCall		The generation of this event is switch specific. The MakeCall cause indicates that the device A is being prompted (via ringing, for example) to go off-hook.
Device A goes off hook and is connected in the call	OriginatedEvent - originatedConnection - callingDevice A - calledDevice Peering - localConnectionState connected - cause newCall		
The call leaves the CSTA subdomain	NetworkReachedEvent - outboundConnection - networkInterfaceUsed - callingDevice A - calledDevice Peering - lastRedirectionDevice - localConnectionState connected - cause newCall		
Peering B begins to ring and A receives ringing tone	DeliveredEvent - connection - alertingDevice B - callingDevice A - calledDevice Peering - lastRedirectionDevice - localConnectionState connected - cause newCall		
Peering B answers the call by manually going off-hook	EstablishedEvent - establishedConnection - answeringDevice (DN) - callingDevice A - calledDevice Peering - lastRedirectionDevice - localConnectionState connected - cause newCall		DN – 'display name' is optional here

Figure 218. Make Call successful session, outbound scenario

CSTA Hold Call

Copyright © Sipwise GmbH - All rights reserved

Activity	A	Peering B	Comments
A Make Call service to a valid device outside the CSTA subsystem is invoked on behalf of device A	MakeCallRequest - callingDevice A - calledDirectoryNumber B - autoOriginate Prompt		The Make Call service specifies that device A should be prompted to go off-hook
Acknowledgement	MakeCallResult - initiatedCall		
Indication that the service has been initiated from this device	ServiceInitiatedEvent - initiatedConnection - initiatedDevice A - localConnectionState initiated - cause makeCall		The generation of this event is switch specific. The MakeCall cause indicates that the device A is being prompted (via ringing, for example) to go off-hook.
Device A goes off hook and is connected in the call	OriginatedEvent - originatedConnection - callingDevice A - calledDevice Peering - localConnectionState connected - cause newCall		
The call leaves the CSTA subdomain	NetworkReachedEvent - outboundConnection - networkInterfaceUsed - callingDevice A - calledDevice Peering - lastRedirectionDevice - localConnectionState connected - cause newCall		
Peering B begins to ring and A receives ringing tone	DeliveredEvent - connection - alertingDevice B - callingDevice A - calledDevice Peering - lastRedirectionDevice - localConnectionState connected - cause newCall		
Peering B answers the call by manually going off-hook	EstablishedEvent - establishedConnection - answeringDevice (DN) - callingDevice A - calledDevice Peering - lastRedirectionDevice - localConnectionState connected - cause newCall		DN – 'display name' is optional here

Figure 219. Hold Call, successful session

CSTA Retrieve Call

Copyright © Sipwise GmbH - All rights reserved

Activity	A	B
RetrieveCall service invoked on behalf of device A .	RetrieveCallRequest - heldCall	
Acknowledge	RetrieveCallResult	
	RetrievedEvent - retrievedConnection - retrievingDevice A - localConnectionState connected - eventCause normal	RetrievedEvent - retrievedConnection - retrievingDevice A - localConnectionState connected - eventCause normal

Figure 220. Retrieve Call

CSTA Clear Connection

Activity	A	B	Comments
Clear Connection service is invoked on behalf of device A	ClearConnectionRequest - connectionToBeCleared		
Acknowledge	ClearConnectionResult		
Event indicates that connection has been removed from the call	ConnectionClearedEvent - droppedConnection - releasingDevice A - localConnectionState null - eventCause normalCleaning	ConnectionClearedEvent - droppedConnection - releasingDevice A - localConnectionState connected - eventCause normalCleaning	
Since B is the only device in the call, it is cleared as the result of A beigng cleared		ConnectionClearedEvent - droppedConnection - releasingDevice B - localConnectionState null - eventCause normalCleaning	

Figure 221. Successful Clear Connection

Copyright © Sipwise GmbH - All rights reserved

Appendix P: Instances

Here you can find a snippet of the network.yml file that can be used on a standard PRO system to test instances and the active-active setup. TIP: Please read carefully all the notes before to proceed in order to not compromise your current setup.

Setup

The following instances are defined:

- 2 instances of kamailio-lb (A and B)
- 2 instances of kamailio-proxy (C and D)
- 2 instances of sems-b2b (E and F)
- 2 instances of asterisk (G and H)

Preferred connections are done as the following:

- A to C to E to G
- B to D to F to H

The following networks and hosts are defined:

- 192.168.1.0/24 for external communications, in particular:
  - 192.168.1.100 shared ip used for 'sip_ext' and 'rtp_ext' of standard services
  - 192.168.1.101 ip of sp1 used for 'sip_ext' and 'rtp_ext' of standard services
  - 192.168.1.102 ip of sp1 used for 'sip_ext' and 'rtp_ext' of standard services
  - 192.168.1.201 ip of kamailio-lb instance A used for 'sip_ext'
  - 192.168.1.211 ip of kamailio-lb instance B used for 'sip_ext'
- 192.168.255.0/24 for internal communications, in particular:
  - 192.168.255.100 shared ip used for 'sip_int' and 'rtp_int' of standard services
  - 192.168.255.101 shared ip used for 'sip_int' and 'rtp_int' of standard services
  - 192.168.255.102 shared ip used for 'sip_int' and 'rtp_int' of standard services
  - 192.168.255.201 ip of kamailio-lb instance A used for 'sip_int'
  - 192.168.255.202 ip of kamailio-proxy instance C used for 'sip_int'
  - 192.168.255.203 ip of sems-b2b instance E used for 'sip_int' (and rtp from/to sems-b2b)
  - 192.168.255.204 ip of asterisk instance G used for 'sip_int' (and rtp from/to asterisk)
  - 192.168.255.211 ip of kamailio-lb instance B used for 'sip_int'
  - 192.168.255.212 ip of kamailio-proxy instance D used for 'sip_int'
  - 192.168.255.213 ip of sems-b2b instance F used for 'sip_int' (and rtp from/to sems-b2b)
  - 192.168.255.214 ip of asterisk instance H used for 'sip_int' (and rtp from/to asterisk)

Copyright © Sipwise GmbH - All rights reserved

IMPORTANT The IP and MAC addresses of the nodes/instances are invented and they may not correspond to the current setup of your system. Carefully update those values before deploying the configuration on your system.

Remember that the *ngcp-network* tool can be used to easily update nodes configurations. For example to move the 'sip_int' and 'rpt_int' types from the 'lo' to the 'neth1' interface, the following command must be executed

```
ngcp-network --move-from=lo --move-to=neth1 --type=sip_int
--type=rtp_int
```

IMPORTANT Sems-b2b supports the definition of the 'rtp_int' or 'rpt_ext' interfaces for instances. This is necessary when Sems-b2b has to play a media directly to the devices: early-media announcements, queues, music-on-hold, etc. The type of interface to be defined has to match what has been configured by the 'rtpproxy.prefer_bind_on_internal' option of the config.yml file. The suggestion is to define 'rtp_int' interface and then switch 'rtpproxy.prefer_bind_on_internal' to 'yes' in order to avoid the definition of another rtp public interface for each instance.

network.yml example

Here is a snippet of the network.yml file with the instances configuration:

```
hosts:
  sp1:
    dbnode: '1'
    interfaces:
    - lo
    - neth0
    - neth1
    lo:
      advertised_ip: []
      cluster_sets:
      - default
      eee: no
      hwaddr: 00:00:00:00:00:00
      ip: 127.0.0.1
      netmask: 255.0.0.0
      shared_ip: []
      shared_v6ip: []
      type:
      - web_ext
      - web_int
      - aux_ext
      - ssh_ext
      - api_int
      - stor_int
      v6ip: ::1
```

Copyright © Sipwise GmbH - All rights reserved

```
      v6netmask: 128
    neth0:
      dns_nameservers:
      - 1.1.1.1
      - 8.8.8.8
      gateway: 192.168.1.1
      hwaddr: 11:22:33:44:55:10
      ip: 192.168.1.101
      netmask: 255.255.255.0
      shared_ip:
      - 192.168.1.100
      shared_v6ip: ~
      type:
      - ssh_ext
      - sip_ext
      - rtp_ext
      - web_ext
      - web_int
      - mon_ext
    neth1:
      cluster_sets:
      - default
      hwaddr: 11:22:33:44:55:11
      ip: 192.168.255.101
      netmask: 255.255.255.0
      shared_ip:
      - 192.168.255.100
      shared_v6ip: ~
      type:
      - ssh_ext
      - ha_int
      - boot_int
      - sip_int
      - rtp_int
    nodename: sp1
    peer: sp2
    role:
    - proxy
    - lb
    - mgmt
    - rtp
    - db
    - storage
    status: online
    swraiddevices: []
    sysdescr: ''
    syslocation: ''
    sysname: ''
  sp2:
    dbnode: '2'
    interfaces:
    - lo
```


Copyright © Sipwise GmbH - All rights reserved

```
    - neth0
    - neth1
    lo:
      cluster_sets:
      - default
      hwaddr: 00:00:00:00:00:00
      ip: 127.0.0.1
      netmask: 255.0.0.0
      shared_ip: []
      shared_v6ip: []
      type:
      - api_int
      - stor_int
      - web_int
      - web_ext
      - aux_ext
      v6ip: ::1
    neth0:
      dns_nameservers:
      - 1.1.1.1
      - 8.8.8.8
      gateway: 192.168.1.1
      hwaddr: 11:22:33:44:55:20
      ip: 192.168.1.102
      netmask: 255.255.255.0
      shared_ip:
      - 192.168.1.100
      shared_v6ip: ~
      type:
      - ssh_ext
      - sip_ext
      - rtp_ext
      - web_ext
      - web_int
      - mon_ext
    neth1:
      cluster_sets:
      - default
      hwaddr: 11:22:33:44:55:21
      ip: 192.168.255.102
      netmask: 255.255.255.0
      shared_ip:
      - 192.168.255.100
      shared_v6ip: ~
      type:
      - ssh_ext
      - ha_int
      - boot_int
      - sip_int
      - rtp_int
    nodename: sp2
    peer: sp1
```

Copyright © Sipwise GmbH - All rights reserved

```
    role:
    - proxy
    - lb
    - mgmt
    - rtp
    - db
    - storage
    status: online
    swraiddevices: []
    sysdescr: ''
    syslocation: ''
    sysname: ''
instances:
  - name: A
    service: kamailio-lb
    host: sp1
    status: online
    label: lb
    interfaces:
      - name: neth0
        ip: 192.168.1.201
        type:
          - sip_ext
      - name: neth1
        ip: 192.168.255.201
        type:
          - sip_int
    connections:
      - name: proxy
        algorithm: random
        links:
          - type: instance
            name: C
            interfaces:
            - name: neth1
              type: sip_int
    databases:
      nosql:
        - name: sp1
          port: 6379
          type: db_replicated_pair
  - name: B
    service: kamailio-lb
    host: sp2
    label: lb
    status: online
    interfaces:
      - name: neth0
        ip: 192.168.1.211
        type:
          - sip_ext
      - name: neth1
```


Copyright © Sipwise GmbH - All rights reserved

```
        ip: 192.168.255.211
        type:
          - sip_int
    connections:
      - name: proxy
        algorithm: random
        links:
          - type: instance
            name: D
            interfaces:
            - name: neth1
              type: sip_int
    databases:
      nosql:
        - name: sp2
          port: 6379
          type: db_replicated_pair
  - name: C
    service: kamailio-proxy
    host: sp1
    label: proxy
    status: online
    interfaces:
      - name: neth1
        ip: 192.168.255.202
        type:
          - sip_int
    connections:
      - name: b2b
        links:
          - type: instance
            name: E
            interfaces:
            - name: neth1
              type: sip_int
      - name: voicemail
        links:
          - type: instance
            name: G
            interfaces:
            - name: neth1
              type: sip_int
    databases:
      nosql:
        - name: sp1
          port: 6379
          type: db_central
        - name: sp1
          port: 6379
          type: db_replicated_pair
      sql:
        - name: sp1
```

Copyright © Sipwise GmbH - All rights reserved

```
          port: 3306
          type: db_central
        - name: sp1
          port: 3306
          type: db_replicated_pair
        - name: sp1
          port: 3306
          type: db_replicated_central
  - name: D
    service: kamailio-proxy
    host: sp2
    label: proxy
    status: online
    interfaces:
      - name: neth1
        ip: 192.168.255.212
        type:
          - sip_int
    connections:
      - name: b2b
        links:
          - type: instance
            name: F
            interfaces:
            - name: neth1
              type: sip_int
      - name: voicemail
        links:
          - type: instance
            name: H
            interfaces:
            - name: neth1
              type: sip_int
    databases:
      nosql:
        - name: sp2
          port: 6379
          type: db_central
        - name: sp2
          port: 6379
          type: db_replicated_pair
      sql:
        - name: sp2
          port: 3306
          type: db_central
        - name: sp2
          port: 3306
          type: db_replicated_pair
        - name: sp2
          port: 3306
          type: db_replicated_central
  - name: E
```


Copyright © Sipwise GmbH - All rights reserved

```
    service: sems-b2b
    host: sp1
    label: b2b
    status: online
    interfaces:
      - name: neth1
        ip: 192.168.255.203
        type:
          - sip_int
    connections:
      - name: proxy
        links:
          - type: instance
            name: C
            interfaces:
            - name: neth1
              type: sip_int
    databases:
      nosql:
        - name: sp1
          port: 6379
          type: db_central
        - name: sp1
          port: 6379
          type: db_replicated_pair
      sql:
        - name: sp1
          port: 3306
          type: db_central
        - name: sp1
          port: 3306
          type: db_replicated_pair
        - name: sp1
          port: 3306
          type: db_replicated_central
  - name: F
    service: sems-b2b
    host: sp2
    label: b2b
    status: online
    interfaces:
      - name: neth1
        ip: 192.168.255.213
        type:
          - sip_int
    connections:
      - name: proxy
        links:
          - type: instance
            name: D
            interfaces:
            - name: neth1
```

Copyright © Sipwise GmbH - All rights reserved

```
            type: sip_int
    databases:
      nosql:
        - name: sp2
          port: 6379
          type: db_central
        - name: sp2
          port: 6379
          type: db_replicated_pair
      sql:
        - name: sp2
          port: 3306
          type: db_central
        - name: sp2
          port: 3306
          type: db_replicated_pair
        - name: sp2
          port: 3306
          type: db_replicated_central
  - name: G
    service: asterisk
    host: sp1
    label: voicemail
    status: online
    interfaces:
      - name: neth1
        ip: 192.168.255.204
        type:
          - sip_int
    connections:
      - name: proxy
        links:
          - type: instance
            name: C
            interfaces:
            - name: neth1
              type: sip_int
      - name: b2b
        links:
          - type: instance
            name: E
            interfaces:
            - name: neth1
              type: sip_int
  - name: H
    service: asterisk
    host: sp2
    label: voicemail
    status: online
    interfaces:
      - name: neth1
        ip: 192.168.255.214
```


Copyright © Sipwise GmbH - All rights reserved

```
        type:
          - sip_int
    connections:
      - name: proxy
        links:
          - type: instance
            name: D
            interfaces:
            - name: neth1
              type: sip_int
      - name: b2b
        links:
          - type: instance
            name: F
            interfaces:
            - name: neth1
              type: sip_int
```

Copyright © Sipwise GmbH - All rights reserved