



The sip:provider CE Handbook mr4.4.2

Sipwise GmbH
<support@sipwise.com>

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 1 |
| 1.1 | About this Document | 1 |
| 1.2 | Getting Help | 1 |
| 1.2.1 | Community Support | 1 |
| 1.2.2 | Commercial Support | 1 |
| 1.3 | What is the sip:provider CE? | 1 |
| 1.4 | What is inside the sip:provider CE? | 2 |
| 1.5 | Who should use the sip:provider CE? | 2 |
| 2 | Platform Architecture | 3 |
| 2.1 | SIP Signaling and Media Relay | 3 |
| 2.1.1 | SIP and Media Elements | 4 |
| | SIP Load-Balancer | 4 |
| | SIP Proxy/Registrar | 5 |
| | SIP Back-to-Back User-Agent (B2BUA) | 5 |
| | SIP App-Server | 6 |
| | Media Relay | 6 |
| 2.1.2 | Basic Call Flows | 8 |
| | General Call Setup | 8 |
| | Endpoint Registration | 9 |
| | Basic Call | 12 |
| | Session Keep-Alive | 13 |
| | Voicebox Calls | 14 |
| 3 | Upgrading from previous versions | 16 |
| 3.1 | Upgrade from previous versions to mr4.4.2 | 16 |
| 4 | Initial Installation | 17 |

| | |
|--|-----------|
| 4.1 Prerequisites | 17 |
| 4.2 Using the NGCP install CD (recommended) | 17 |
| 4.3 Using the NGCP installer | 18 |
| 4.3.1 Installing the Operating System | 18 |
| Using special Debian setups | 18 |
| 4.3.2 Installing the sip:provider CE | 19 |
| 4.4 Using a pre-installed virtual machine | 20 |
| 4.4.1 Vagrant box for VirtualBox | 20 |
| 4.4.2 VirtualBox image | 22 |
| 4.4.3 VMware image | 23 |
| 4.4.4 Amazon EC2 image | 23 |
| 5 Initial System Configuration | 28 |
| 5.1 Network Configuration | 28 |
| 5.2 Apply Configuration Changes | 29 |
| 5.3 Start Securing Your Server | 29 |
| 5.4 Configuring the Email Server | 30 |
| 5.5 Advanced Network Configuration | 30 |
| 5.5.1 Audiocodes devices workaround | 30 |
| 5.6 What's next? | 31 |
| 6 Administrative Configuration | 32 |
| 6.1 Creating a Customer | 32 |
| 6.2 Creating a Subscriber | 37 |
| 6.3 Domain Preferences | 42 |
| 6.4 Subscriber Preferences | 44 |
| 6.5 Creating Peerings | 45 |
| 6.5.1 Creating Peering Groups | 45 |
| 6.5.2 Creating Peering Servers | 47 |
| 6.5.3 Authenticating and Registering against Peering Servers | 54 |

| | |
|---|-----------|
| Proxy-Authentication for outbound calls | 54 |
| Registering at a Peering Server | 56 |
| 6.6 Configuring Rewrite Rule Sets | 56 |
| 6.6.1 Inbound Rewrite Rules for Caller | 59 |
| 6.6.2 Inbound Rewrite Rules for Callee | 61 |
| 6.6.3 Outbound Rewrite Rules for Caller | 62 |
| 6.6.4 Outbound Rewrite Rules for Callee | 63 |
| 6.6.5 Emergency Number Handling | 63 |
| 6.6.6 Assigning Rewrite Rule Sets to Domains and Subscribers | 64 |
| 6.6.7 Creating Dialplans for Peering Servers | 65 |
| 7 Advanced Subscriber Configuration | 66 |
| 7.1 Access Control for SIP Calls | 66 |
| 7.1.1 Block Lists | 66 |
| Block Modes | 67 |
| Block Lists | 67 |
| Block Anonymous Numbers | 68 |
| 7.1.2 NCOS Levels | 68 |
| Creating NCOS Levels | 69 |
| Creating Rules per NCOS Level | 70 |
| Assigning NCOS Levels to Subscribers/Domains | 72 |
| Assigning NCOS Level for Forwarded Calls to Subscribers/Domains | 73 |
| 7.1.3 IP Address Restriction | 73 |
| 7.2 Call Forwarding and Call Hunting | 74 |
| 7.2.1 Setting a simple Call Forward | 74 |
| 7.2.2 Advanced Call Hunting | 75 |
| Configuring Destination Sets | 75 |
| Configuring Time Sets | 77 |
| 7.3 Header Manipulation | 79 |

| | | |
|-------|---|----|
| 7.3.1 | Header Filtering | 79 |
| 7.3.2 | Codec Filtering | 79 |
| 7.3.3 | Enable History and Diversion Headers | 80 |
| 7.4 | SIP Trunking with SIPconnect | 80 |
| 7.4.1 | User provisioning | 80 |
| 7.4.2 | Inbound calls routing | 80 |
| 7.4.3 | Number manipulations | 80 |
| | Rewrite rules | 81 |
| | User parameter | 82 |
| | Forwarding number | 82 |
| | Allowed CLIs | 83 |
| 7.4.4 | Registration | 83 |
| | Trusted sources | 83 |
| 7.5 | Limiting Subscriber Preferences via Subscriber Profiles | 84 |
| 7.5.1 | Subscriber Profile Sets | 84 |
| 7.6 | Creating Trusted Subscribers | 85 |
| 7.7 | Voicemail System | 86 |
| 7.7.1 | Accessing the IVR Menu | 86 |
| | Mapping numbers and codes to IVR access | 86 |
| | External IVR access | 86 |
| 7.7.2 | IVR Menu Structure | 86 |
| 7.7.3 | Type Of Messages | 88 |
| | Unavailable Message | 88 |
| | Busy Message | 88 |
| | Temporary Greeting | 88 |
| 7.7.4 | Folders | 88 |
| | The Default Folder List | 89 |
| 7.8 | XMPP Instant Messaging | 89 |

| | | |
|----------|--|------------|
| 7.9 | Configuring Subscriber IVR Language | 89 |
| 7.10 | Sound Sets | 90 |
| 7.10.1 | Configuring Early Reject Sound Sets | 91 |
| 7.11 | Conference System | 95 |
| 7.11.1 | Configuring Call Forward to Conference | 95 |
| 7.11.2 | Configuring Conference Sound Sets | 96 |
| 7.11.3 | Entering the Conference with a PIN | 97 |
| 7.12 | Malicious Call Identification (MCID) | 97 |
| 7.12.1 | Setup | 97 |
| 7.12.2 | Usage | 98 |
| 7.12.3 | Advanced configuration | 98 |
| 7.13 | Handling WebRTC Clients | 98 |
| 7.14 | SIP loop detection | 99 |
| 8 | Customer Self-Care Interfaces | 100 |
| 8.1 | The Customer Self-Care Web Interface | 100 |
| 8.1.1 | Login Procedure | 100 |
| 8.1.2 | Site Customization | 100 |
| 8.2 | The Vertical Service Code Interface | 100 |
| 8.2.1 | Vertical Service Codes for PBX customers | 101 |
| 8.2.2 | Configuration of Vertical Service Codes | 102 |
| 8.3 | The Voicemail Interface | 102 |
| 9 | Billing Configuration | 103 |
| 9.1 | Billing Data Import | 103 |
| 9.1.1 | Creating Billing Profiles | 103 |
| 9.1.2 | Creating Billing Fees | 105 |
| 9.1.3 | Creating Off-Peak Times | 107 |
| 9.1.4 | Fraud Detection and Locking | 109 |
| 9.2 | Billing Data Export | 110 |

| | | |
|-----------|---|------------|
| 9.2.1 | File Name Format | 110 |
| 9.2.2 | File Format | 111 |
| | File Header Format | 111 |
| | File Body Format for Call Detail Records (CDR) | 111 |
| | File Body Format for Event Detail Records (EDR) | 115 |
| | File Trailer Format | 117 |
| 9.2.3 | File Transfer | 118 |
| 10 | Invoices and invoice templates | 119 |
| 10.1 | Invoices management | 119 |
| 10.2 | Invoice templates | 121 |
| 10.2.1 | Invoice Templates management | 121 |
| 10.2.2 | Invoice Template content | 122 |
| | Layers | 123 |
| | Edit SVG XML source | 125 |
| | Change logo image | 127 |
| 10.2.3 | Save and preview invoice template content. | 128 |
| 10.3 | Invoices generation | 130 |
| 11 | Email templates | 133 |
| 11.1 | Email events | 133 |
| 11.2 | Initial template values and template variables | 133 |
| 11.3 | Password reset email template | 133 |
| 11.4 | New subscriber notification email template | 134 |
| 11.5 | Invoice email template | 134 |
| 11.6 | Email templates management | 136 |
| 12 | Local Number Porting | 139 |
| 12.1 | Local LNP Database | 139 |
| 12.1.1 | LNP Carriers | 139 |

| | |
|--|------------|
| 12.1.2 LNP Numbers | 139 |
| 12.1.3 Enabling local LNP support | 140 |
| 12.1.4 LNP Routing Procedure | 140 |
| Calls to non-authoritative Carriers | 140 |
| 12.1.5 Transit Calls using LNP | 141 |
| 12.1.6 CSV Format | 141 |
| 13 Provisioning interfaces | 143 |
| 13.1 REST API | 143 |
| 13.1.1 API Workflows | 143 |
| Managing Customers and Subscribers | 143 |
| 13.2 SOAP and XMLRPC API | 148 |
| 14 Configuration Framework | 150 |
| 14.1 Configuration templates | 150 |
| 14.1.1 .tt2 and .customtt.tt2 files | 150 |
| 14.1.2 .prebuild and .postbuild files | 151 |
| 14.1.3 .services files | 152 |
| 14.2 config.yml, constants.yml and network.yml files | 153 |
| 14.3 ngcpcfg and its command line options | 153 |
| 14.3.1 apply | 153 |
| 14.3.2 build | 153 |
| 14.3.3 commit | 153 |
| 14.3.4 decrypt | 154 |
| 14.3.5 diff | 154 |
| 14.3.6 encrypt | 154 |
| 14.3.7 help | 154 |
| 14.3.8 initialise | 154 |
| 14.3.9 pull | 154 |
| 14.3.10push | 154 |

| | | |
|------------------------------|---|------------|
| 14.3.1 | services | 154 |
| 14.3.2 | status | 155 |
| 15 | Network Configuration | 156 |
| 15.1 | General Structure | 156 |
| 15.2 | Available Host Options | 156 |
| 16 | Advanced Network Configuration | 158 |
| 16.1 | Extra SIP Sockets | 158 |
| 16.2 | Extra SIP and RTP Sockets | 158 |
| 17 | Security and Maintenance | 160 |
| 17.1 | Sipwise SSH access to sip:provider CE | 160 |
| 17.2 | Firewalling | 160 |
| 17.3 | Password management | 161 |
| 17.4 | SSL certificates | 162 |
| 17.5 | Securing your sip:provider CE against SIP attacks | 163 |
| 17.5.1 | Denial of Service | 163 |
| 17.5.2 | Bruteforcing SIP credentials | 163 |
| 17.6 | Backup and recovery | 164 |
| 17.6.1 | Backup | 164 |
| What data to back up | | 164 |
| The built-in backup solution | | 165 |
| 17.6.2 | Recovery | 165 |
| 17.7 | Reset database | 165 |
| 17.8 | System requirements and performance | 166 |
| 17.9 | Troubleshooting | 168 |
| 17.9.1 | Collecting call information from logs | 170 |
| 17.9.2 | Collecting SIP traces | 171 |
| A | NGCP configs overview | 172 |

| | | |
|--------|---|-----|
| A.1 | config.yml overview | 172 |
| A.1.1 | asterisk | 172 |
| A.1.2 | autoprov | 173 |
| A.1.3 | backuptools | 174 |
| A.1.4 | cdrexport | 174 |
| A.1.5 | checktools | 175 |
| A.1.6 | cleanuptools | 177 |
| A.1.7 | database | 177 |
| A.1.8 | faxserver | 177 |
| A.1.9 | general | 178 |
| A.1.10 | heartbeat | 178 |
| A.1.11 | intercept | 179 |
| A.1.12 | kamailio | 179 |
| A.1.13 | mediator | 183 |
| A.1.14 | nginx | 183 |
| A.1.15 | ntp | 184 |
| A.1.16 | ossbss | 184 |
| A.1.17 | pbx (only with additional cloud PBX module installed) | 185 |
| A.1.18 | prosody | 186 |
| A.1.19 | pushd | 186 |
| A.1.20 | qos | 187 |
| A.1.21 | rate-o-mat | 187 |
| A.1.22 | redis | 187 |
| A.1.23 | reminder | 188 |
| A.1.24 | rsyslog | 188 |
| A.1.25 | rtpproxy | 189 |
| A.1.26 | security | 189 |
| A.1.27 | sems | 189 |

| | |
|----------------------------|-----|
| A.1.28 snmpagent | 191 |
| A.1.29 sshd | 191 |
| A.1.30 voisniff | 191 |
| A.1.31 www_admin | 192 |

1 Introduction

1.1 About this Document

This document describes the architecture and the operational steps to install, operate and modify the Sipwise sip:provider CE.

In the 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 like rewrite rules, call blockings, call forwards 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 provided APIs.

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

1.2 Getting Help

1.2.1 Community Support

We have set up the [spce-user](#) mailing list, where questions are answered on a best-effort basis and discussions can be started with other community users.

1.2.2 Commercial Support

If you need professional help setting up and maintaining the sip:provider CE, send an email to support@sipwise.com.

Sipwise also provides training and commercial support for the platform. Additionally, we offer a migration path to the sip:provider PRO appliance, which is the commercial, carrier-grade version of the sip:provider CE. If the user base grows on the CE, this will allow operators to migrate seamlessly to a highly available and scalable platform with defined service level agreements, phone support and on-call duty. Please visit www.sipwise.com for more information on commercial offerings.

1.3 What is the sip:provider CE?

The sip:provider CE is a SIP based Open Source Class5 VoIP soft-switch platform providing rich telephony services. It offers a wide range of features to end users (call forwards, voicemail, conferencing, call blocking, click-to-dial, call-lists showing near-realtime accounting information etc.), which can be configured by them using the customer-self-care web interface. For operators, it offers a fully web-based administrative panel, allowing them to configure users, peerings, billing profiles etc., as well as viewing real-time statistics of the system. For tight integration into existing infrastructures, it provides a powerful REST API.

The sip:provider CE can be installed in a few steps within a couple of minutes and requires no knowledge about configuration files of specific software components.

1.4 What is inside the sip:provider CE?

Opposed to other free VoIP software, the sip:provider CE is not a single application, but a whole software platform, the Sipwise NGCP (Sipwise Next Generation Communication Platform), which is based on Debian GNU/Linux.

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

After downloading and starting the installer, it will fetch and install all the required Debian packages from the relevant Debian repositories. The installed applications are managed by the NGCP Configuration Framework, which makes it possible to change system parameters in a single place, so administrators don't need to have any knowledge of the dozens of different configuration files of the different packages. This provides a very easy and bullet-proof way of operating, changing and tweaking the otherwise quite complex system.

Once configured, integrated web interfaces are provided for both end users and administrators to use the sip:provider CE. By using the provided provisioning and billing APIs, it can be integrated tightly into existing OSS/BSS infrastructures to optimize work-flows.

1.5 Who should use the sip:provider CE?

The sip:provider CE is specifically tailored to companies and engineers trying to start or experiment with a fully-featured SIP based VoIP service without having to go through the steep learning curve of SIP signalling, integrating the different building blocks to make them work together in a reasonable way and implementing the missing components to build a business on top of that.

In the past, creating a business-ready VoIP service included installation and configuration of SIP software like Asterisk, OpenSER, Kamailio etc., which can get quite difficult when it comes to implementing advanced features. It required to implement different web interfaces, billing engines and connectors to existing OSS/BSS infrastructure. These things are now obsolete due to the CE, which covers all these requirements.

2 Platform Architecture

The sip:provider CE platform is one single node running all necessary components of the system. The components are outlined in the following figure:

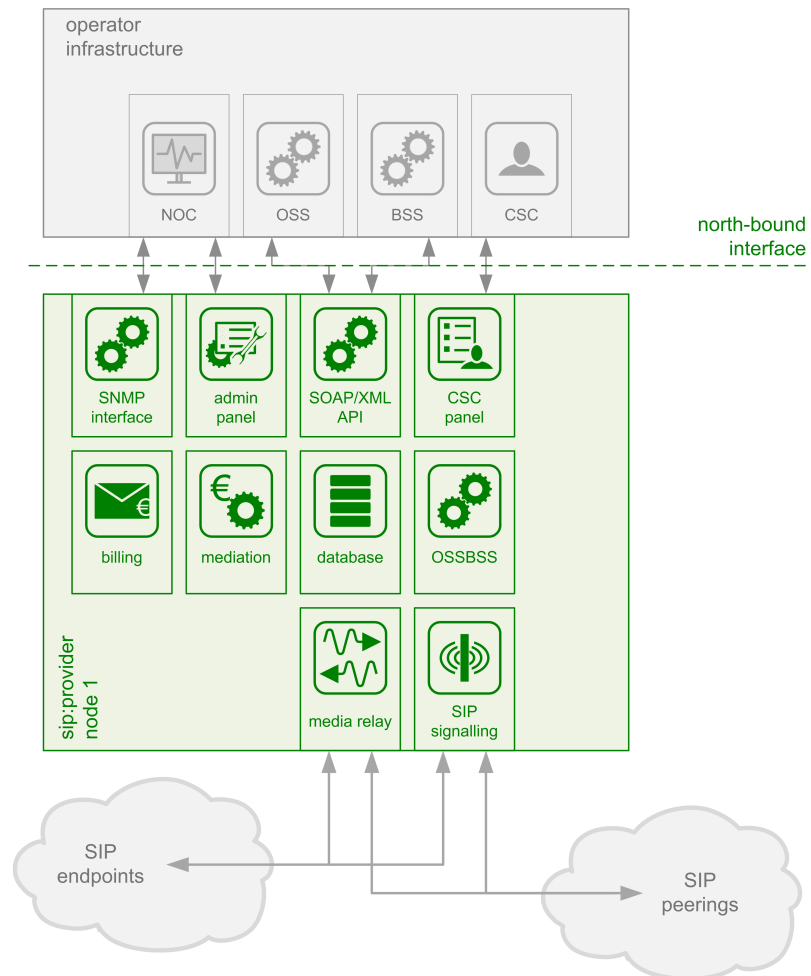


Figure 1: Architecture Overview

The main building blocks of the sip:provider CE are:

- SIP Signaling and Media Relay
- Provisioning
- Mediation and Billing

2.1 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.

2.1.1 SIP and Media Elements

The components involved in SIP and Media on the sip:provider CE are shown in the following figure:

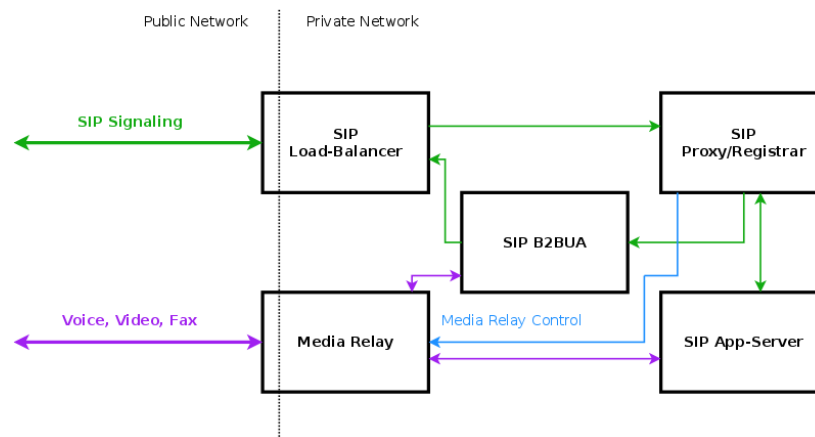


Figure 2: SIP and Media Relay Components

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 the sip:provider CE, 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 in the commercial version, when scaling out the system beyond just 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 is used by the OSSBSS system to control the daemon.

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`.

Tip

The SIP load-balancer can be managed via the commands `/etc/init.d/kamailio-lb start`, `/etc/init.d/kamailio-lb stop` and `/etc/init.d/kamailio-lb restart`. Its status can be queried by executing `/etc/init.d/kamailio-lb status`. Also `ngcp-kamctl lb` and `ngcp-sercmd lb` are provided for querying kamailio functions, for example: `ngcp-sercmd lb htable.dump ipban`.

SIP Proxy/Registrar

The SIP proxy/registrar (or short *proxy*) is the work-horse of the sip:provider CE. 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. 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 the sip:provider CE, 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 `/etc/init.d/kamailio-proxy start`, `/etc/init.d/kamailio-proxy stop` and `/etc/init.d/kamailio-proxy restart`. Its status can be queried by executing `/etc/init.d/kamailio-proxy status`. Also `ngcp-kamctl proxy` and `ngcp-sercmd proxy` are provided for querying kamailio functions, for example: `ngcp-kamctl proxy ul show`.

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

The SIP B2BUA (also called SBC within the system) decouples the first call-leg (calling party to sip:provider CE) from the second call-leg (sip:provider CE to the called party).

The software part used for this element is SEMS.

This element is typically optional in SIP systems, but it is always used for SIP calls (INVITE) that don't have the sip:provider CE 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 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 the sip:provider CE) 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 the sip:provider CE. 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 5040 for control messages from the cli tool and on TCP port 8090 for XMLRPC connections from the OSSBSS to control the daemon.

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

Tip

The SIP B2BUA can be controlled via the commands `/etc/init.d/ngcp-sems start`, `/etc/init.d/ngcp-sems stop` and `/etc/init.d/ngcp-sems restart`. Its status can be queried by executing `/etc/init.d/ngcp-sems status`

SIP App-Server

The SIP App-Server is an Asterisk instance used for voice applications like Voicemail and Reminder Calls. 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 1.4 to make Asterisk aware of the sip:provider CE 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 20000.

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 `/etc/init.d/asterisk start`, `/etc/init.d/asterisk stop` and `/etc/init.d/asterisk restart`. Its status can be queried by executing `/etc/init.d/asterisk status`

Media Relay

The Media Relay (also called *rtengine*) 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 40000. When the media streams are negotiated, *rtengine* 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 odd port numbers for the media data, and one pair on the next even port numbers for meta data, 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.

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 `rtpproxy`. The QoS/ToS value can be changed via the key `qos.tos_rtp`.

Tip

The Media Relay can be controlled via the commands `/etc/init.d/ngcp-rtpengine-daemon start`, `/etc/init.d/ngcp-rtpengine-daemon stop` and `/etc/init.d/ngcp-rtpengine-daemon restart`. Its status can be queried by executing `/etc/init.d/ngcp-rtpengine-daemon status`

2.1.2 Basic Call Flows

General Call Setup

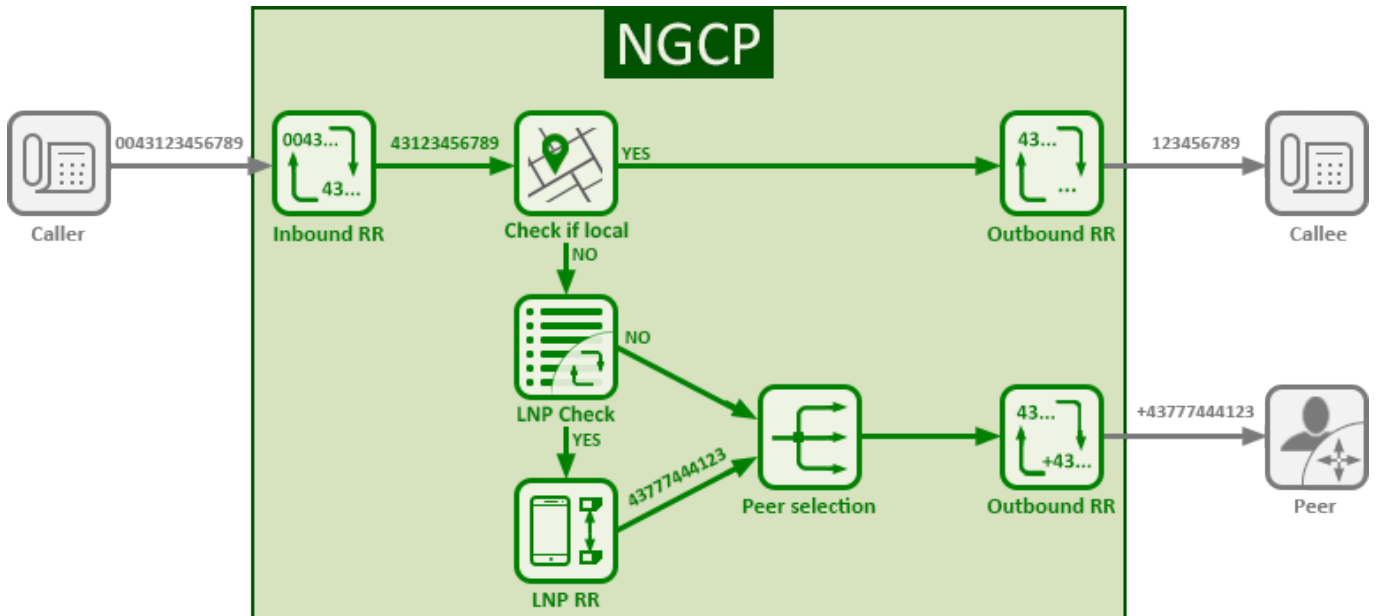


Figure 3: General Call Setup

NGCP 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, NGCP performs the digest authentication.
- When the subscriber is authorized to make the call, NGCP 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, NGCP 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 please refer to [Local Number Porting](#) Section 12 chapter.
 - If it was ported, NGCP applies the LNP Rewrite Rules to the called number.
- Based on the priorities of peering groups and peering rules (see [?simpara] for details), NGCP 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 NGCP will send the call to it.
- Applies the Outbound Rewrite Rules for the caller and the callee assigned to a peering server when sending the call to it.

Endpoint Registration

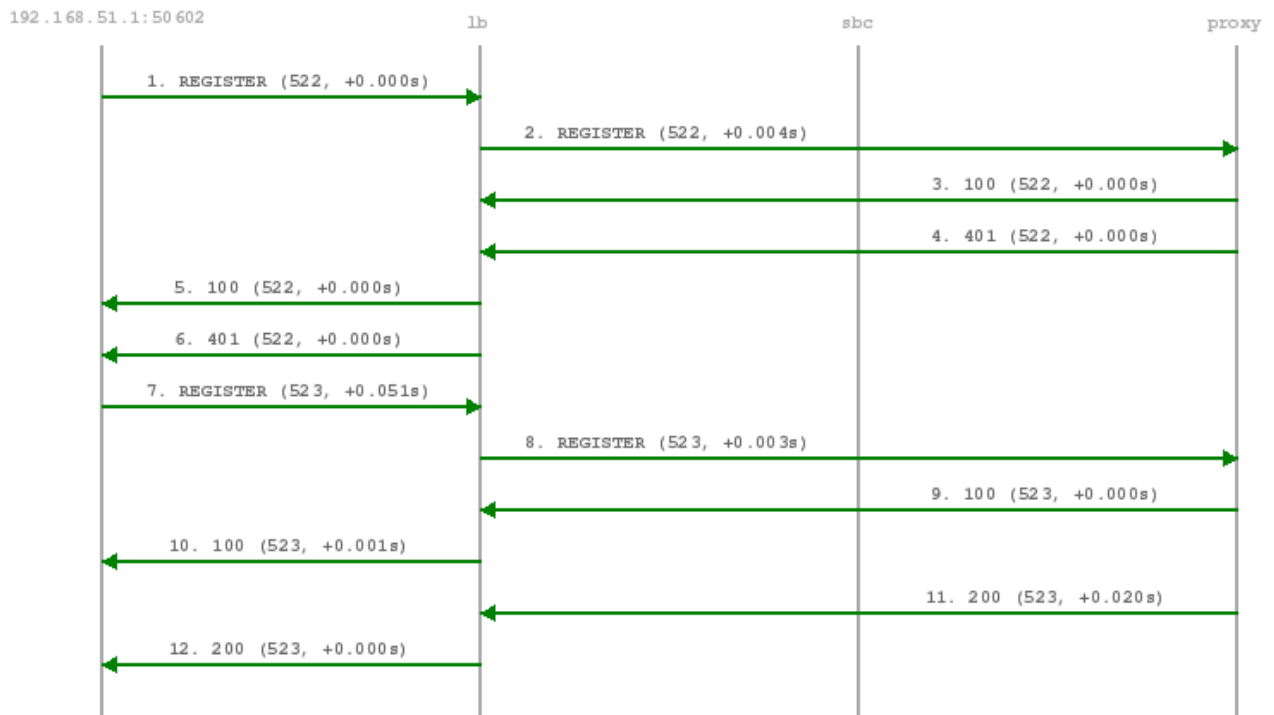
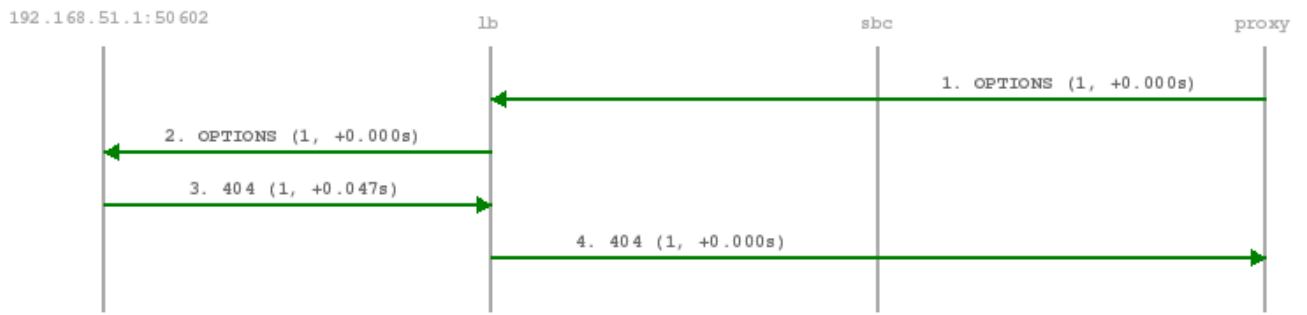


Figure 4: Registration Call-Flow

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.

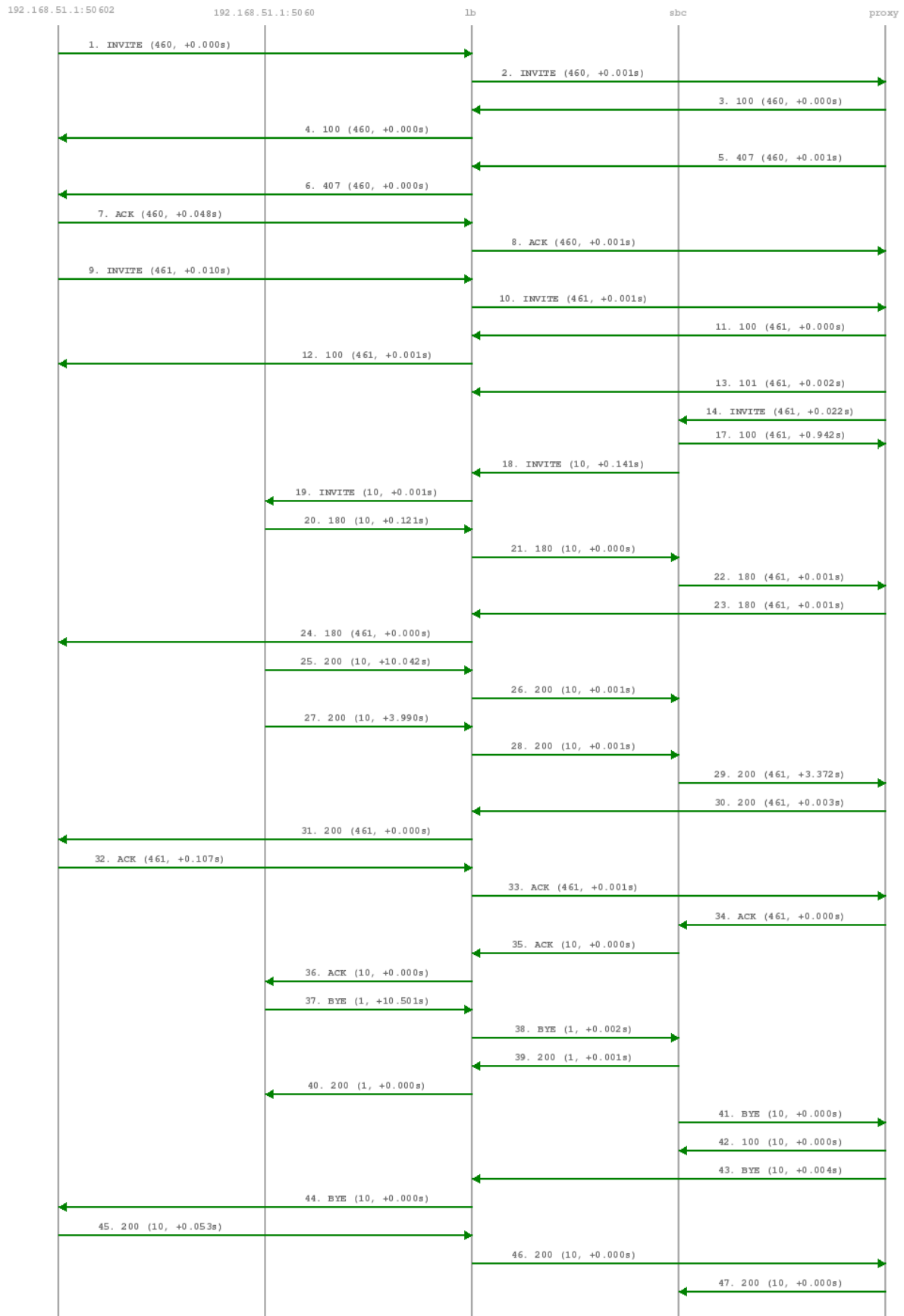
The SIP proxy writes the content of the Contact header (e.g. sip:me@1.2.3.4:1234;transport=UDP) into its location table (in case of NAT the content is changed by the SIP load-balancer to the IP/port from where the request was received), so it knows where to reach a subscriber in case of an inbound call 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 OPTION 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, the sip:provider PRO 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



The calling party sends an INVITE (e.g. `sip:someuser@example.org`) via the SIP load-balancer to the SIP 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 via the SIP load-balancer 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 200 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 200, the SIP proxy writes a start record for the accounting process. The 200 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 *mediator* service in a regular interval and are converted into a Call Detail Record (CDR), which will be rated by the *rate-o-mat* process and can be billed to the calling party.

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 the UPDATE method during call-setup, then the SIP B2BUA will use an UPDATE 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.

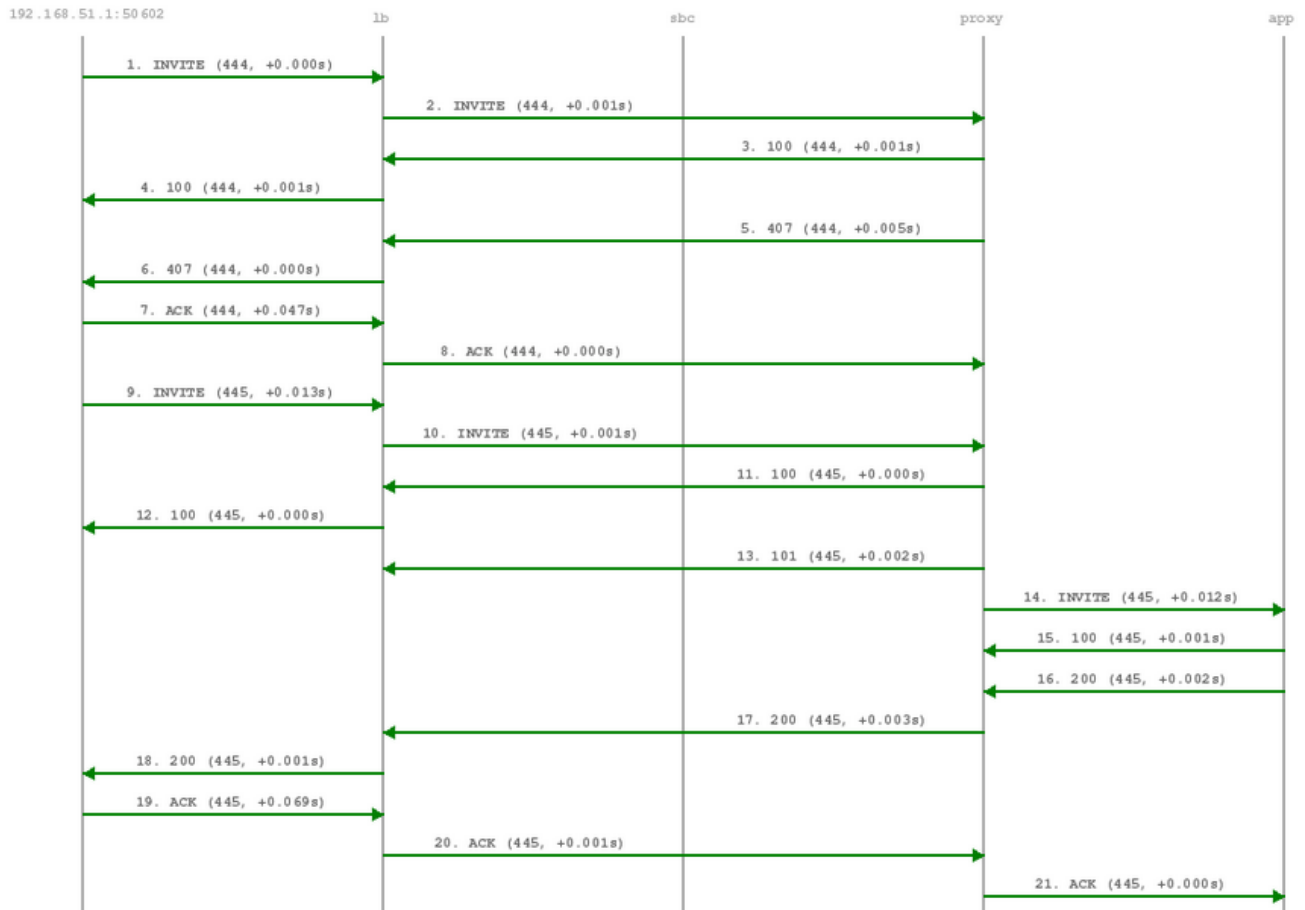
Tip

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 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



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.

**Important**

The sip:provider CE 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.

3 Upgrading from previous versions

3.1 Upgrade from previous versions to mr4.4.2

The sip:provider CE system upgrade to mr4.4.2 will be performed in a couple of tasks:

- Upgrade NGCP software packages
- Upgrade NGCP configuration templates
- Upgrade NGCP DB schema
- Upgrade the base system within Debian (v8) to the latest package versions

For upgrading the sip:provider CE to release mr4.4.2, execute the following commands:

```
NGCP_CURRENT_VERSION=$(cat /etc/ngcp_version)
sed -i "s/$NGCP_CURRENT_VERSION/mr4.4.2/" /etc/apt/sources.list.d/sipwise.list
apt-get update
apt-get install ngcp-upgrade-ce
```

Run the upgrade script as *root* like this:

```
ngcp-upgrade
```

Note

sip:provider CE can be upgraded to mr4.4.2 from previous release or previous build only. The script ngcp-upgrade will find all the possible destination releases for the upgrade and allow to choose the proper one.

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).

Once up again, double-check your config file `/etc/ngcp-config/config.yml` (sections will be rearranged now and will contain more parameters) and your domain/subscriber/peer configuration and test the setup. You can find a backup of some important configuration files of your existing installation under `/var/backup/ngcp-mr4.4.2-*` (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 `/var/backup/ngcp-mr4.4.2-*/upgrade.log`.

Note

sip:provider CE mr4.4.2 requires at least 2GB of RAM available as the minimum requirements of the installation section, otherwise certain features won't work and you will run into arbitrary issues.

4 Initial Installation

4.1 Prerequisites

For an initial installation of the sip:provider CE, it is mandatory that your production environment meets the following criteria:

HARDWARE REQUIREMENTS

- Recommended: Dual-core, x86_64 compatible, 3GHz, 4GB RAM, 128GB HDD
- Minimum: Single-core, x86_64 compatible, 1GHz, 2GB RAM, 16GB HDD

SUPPORTED OPERATING SYSTEMS

- Debian Jessie (v8) 64-bit

INTERNET CONNECTION

- Hardware needs connection to the Internet



Important

Only **Debian Jessie (v8) 64-bit** is currently supported as a host system for the sip:provider CE.



Important

It is **HIGHLY** recommended that you use a **dedicated server** (either a physical or a virtual one) for sip:provider CE, because the installation process will wipe out existing MySQL databases and modify several system configurations.

4.2 Using the NGCP install CD (recommended)

The custom Sipwise NGCP install CD provides the ability to easily install sip:provider CE, including automatic partitioning and installation of the underlying Debian system. You can install the latest available or Long Term Support (LTS) sip:provider CE release using sip:provider CE [install CD image](#) (checksums: [sha1](#), [md5](#)).



Important

The NGCP install CD automatically takes care of partitioning, any present data will be overwritten! While the installer prompts for the disk that should be used for installation before its actual execution, it's strongly recommended to boot the ISO in an environment with empty disks or disks that you don't plan to use for anything else than the newly installed NGCP system.

To configure network options please choose the according boot menu entries with either `DHCP` or `static NW` (static network configuration) in its name. Press the `<tab>` key on the menu entry you want to boot, then adjust the `ip=...` and `dns=...` boot options as needed.

Tip

When DHCP is available in your infrastructure then you shouldn't have to configure anything, just choose `DHCP` and press enter. If network configuration still doesn't work as needed a console based network configuration system will assist you in setting up your network configuration.

Also, you can use sip:provider CE **install CD** to boot the Grml (Debian based live system) rescue system, check RAM using a memory testing tool or install plain Debian squeeze/wheezy/jessie system for manual installation using NGCP installer.

4.3 Using the NGCP installer

4.3.1 Installing the Operating System

You need to install Debian Jessie (v8) 64-bit on the server. A **basic** installation without any additional task selection (like *Desktop System*, *Web Server* etc.) is sufficient.

Tip

Sipwise recommends using the latest **Netinstall ISO** as installation medium.

Important

If you use other kinds of installation media (e.g. provided by your hosting provider), prepare for some issues that might come up during installation. For example, you might be forced to manually resolve package dependencies in order to install the sip:provider CE. Therefore, it is **HIGHLY RECOMMENDED** to use a clean Debian installation to simplify the installation process.

Note

If you installed your system using the Debian CDs/DVDs (so neither using the NGCP install CD nor **the Debian Netinstall ISO**) `apt-get` might prompt to insert disk to proceed during NGCP installation. The prompt won't be visible for you and installation hangs. Please disable the `cdrom` entries in `/etc/apt/sources.list` and enable a Debian mirror (e.g. <http://http.debian.net/debian>) instead.

Using special Debian setups

If you plan to install the sip:provider CE on Virtual Hosting Providers like *Dreamhost* with their provided Debian installer, you might need to manually prepare the system for the NGCP installation, otherwise the installer will fail installing certain package versions required to function properly.

Using Dreamhost Virtual Private Server

A Dreamhost virtual server uses apt-pinning and installs specific versions of MySQL and apache, so you need to clean this up beforehand.

Note

Apache is not used by default since mr3.6.1, still better to remove pinned Apache version.

```
apt-get remove --purge mysql-common ndn-apache22
mv /etc/apt/preferences /etc/apt/preferences.bak
apt-get update
apt-get dist-upgrade
```

**Warning**

Be aware that this step will break your web-based system administration provided by Dreamhost. Only do it if you are certain that you won't need it.

4.3.2 Installing the sip:provider CE

The sip:provider CE is based on the *Sipwise NGCP*, so download and install the latest *Sipwise NGCP* installer package:

```
PKG=ngcp-installer-latest.deb
wget http://deb.sipwise.com/spce/${PKG}
dpkg -i ${PKG}
```

Run the installer as root user:

```
ngcp-installer
```

Note

You can find the previous versions of *Sipwise NGCP* installer package [here](#).

The installer will ask you to confirm that you want to start the installation. Read the given information **carefully**, and if you agree, proceed with *y*.

The installation process will take several minutes, depending on your network connection and server performance. If everything goes well, the installer will (depending on the language you use), show something like this:

```
Installation finished. Thanks for choosing NGCP sip:provider Community Edition.
```

During the installation, you can watch the background processing by executing the following command on a separate console:

```
tail -f /tmp/ngcp-installer.log
```

4.4 Using a pre-installed virtual machine

For quick test deployments, pre-installed virtualization images are provided. These images are intended to be used for quick test, not recommended for production use.

4.4.1 Vagrant box for VirtualBox

Vagrant is an open-source software for creating and configuring virtual development environments. Sipwise provides a so called Vagrant base box for your service, to easily get direct access to your own sip:provider CE Virtual Machine without any hassles.

Note

The following software must be installed to use Vagrant boxes:

- **VirtualBox (v.5.0.8+ is recommended, while v.4.3.16+ should work too)**
- **Vagrant v.1.8.1+**

Get your copy of sip:provider CE by running:

```
vagrant init spce-mr4.4.2 http://deb.sipwise.com/spce/images/sip_provider_CE_mr4.4.2 ↔
  _vagrant.box
vagrant up
```

As soon as the machine is up and ready you should have your local copy of sip:provider CE with the following benefits:

- all the software and database are automatically updated to the latest available version
- the system is configured to use your LAN IP address (received over DHCP)
- basic SIP credentials to make SIP-2-SIP calls out of the box are available

Use the following command to access the terminal:

```
vagrant ssh
```

or login to Administrator web-interface at <https://127.0.0.1:1443/login/admin> (with default user *administrator* and password *administrator*).

There are two ways to access VM resources, through NAT or Bridge interface:

Note

a.b.c.d is IP address of VM machine received from DHCP; x.y.z.p is IP address of your host machine

Table 1: Vagrant based VirtualBox VM interfaces:

| Description | Host-only address | LAN address | Notes |
|----------------------------------|---|---|--|
| SSH | ssh://127.0.0.1:2222 | ssh://a.b.c.d:22 or ssh://x.y.z.p:2222 | Also available via "vagrant ssh" |
| Administrator interface | https://127.0.0.1:1443/login/admin | https://a.b.c.d:1443/login/admin or https://x.y.z.p:1443/login/admin | |
| New Customer self care interface | https://127.0.0.1:1443 | https://a.b.c.d:1443 or https://x.y.z.p:1443 | new self-care interface based on powerful ngcp-panel framework |
| Old Customer self care interface | https://127.0.0.1:22443 | https://a.b.c.d:443 or https://x.y.z.p:22443 | will be removed in upcoming releases |
| Provisioning interfaces | https://127.0.0.1:2443 | https://a.b.c.d:2443 or https://x.y.z.p:2443 | |
| SIP interface | not available | sip://a.b.c.d:5060 | Both TCP and UDP are available. |

Note

VM ports smaller than 1024 mapped to ports 22<vm_port> through NAT, otherwise root on host machine requires to map them. It means SSH port 22 mapped to port 2222, WEB port 443 → 22443.

VM IP address (a.b.c.d), as well as SIP credentials will be printed to terminal during "vagrant up" stage, e.g.:

```
[20_add_sip_account] Adding SIP credentials...
[20_add_sip_account] - removing domain 192.168.1.103 with subscribers
[20_add_sip_account] - adding domain 192.168.1.103
[20_add_sip_account] - adding subscriber 43991002@192.168.1.103 (pass: 43991002)
[20_add_sip_account] - adding subscriber 43991003@192.168.1.103 (pass: 43991003)
[20_add_sip_account] - adding subscriber 43991004@192.168.1.103 (pass: 43991004)
[20_add_sip_account] - adding subscriber 43991005@192.168.1.103 (pass: 43991005)
[20_add_sip_account] - adding subscriber 43991006@192.168.1.103 (pass: 43991006)
[20_add_sip_account] - adding subscriber 43991007@192.168.1.103 (pass: 43991007)
[20_add_sip_account] - adding subscriber 43991008@192.168.1.103 (pass: 43991008)
[20_add_sip_account] - adding subscriber 43991009@192.168.1.103 (pass: 43991009)
[20_add_sip_account] You can USE your VM right NOW: https://192.168.1.103:1443/login/admin
```

To turn off your sip:provider CE virtual machine, just type:

```
vagrant halt
```

To completely remove sip:provider CE virtual machine, use:


```
vagrant destroy
vagrant box remove spce-mr4.4.2
```

Further documentation for Vagrant is available [at the official Vagrant website](#).

Vagrant usage tips:

- Default SSH login is *root* and password is *sipwise*. SSH connection details can be displayed via:

```
vagrant ssh-config
```

- You can download a Vagrant box for VirtualBox from [here](#) manually (checksums: [sha1](#), [md5](#)).

4.4.2 VirtualBox image

You can download a VirtualBox image from [here](#) (checksums: [sha1](#), [md5](#)). Once you have downloaded the file you can import it to VirtualBox via its import utility.

The format of the image is *ova*. If you have VirtualBox 3.x running, which is not compatible with *ova* format, you need to extract the file with any *tar* compatible software and import the *ovf* file which is inside the archive.

On Linux, you can do it like this:

```
tar xvf sip_provider_CE_mr4.4.2_virtualbox.ova
```

On Windows, right-click on the ova file, choose *Open with* and select *WinZIP* or *WinRAR* or any other application able to extract *tar* archives. Extract the files to any place and import the resulting *ovf* file in VirtualBox.

Considerations when using this virtual machine:

- You will need a 64bit guest capable VirtualBox setup.
- The root password is *sipwise*
- You should use *bridge mode* networking (adjust your bridging interface in the virtual machine configuration) to avoid having the sip:provider CE behind NAT.
- You'll need to adjust your timezone and keyboard layout.
- The network configuration is set to DHCP. You'll need to change it to the appropriate static configuration.
- As the virtual image is a static file, it won't contain the most updated versions of our software. Please upgrade the system via *apt* as soon as you boot it for the first time.

4.4.3 VMware image

You can download a VMware image from [here](#) (checksums: [sha1](#), [md5](#)). Once you have downloaded the file just extract the *zip* file and copy its content to your virtual machines folder.

Considerations when using this virtual machine:

- You will need a 64bit guest capable vmware setup.
- The root password is *sipwise*
- You'll need to adjust your timezone and keyboard layout.
- The network configuration is set to DHCP. You'll need to change it to the appropriate static configuration.
- As the virtual image is a static file, it won't contain the most updated versions of our software. Please upgrade the system via `apt` as soon as you boot it for the first time.

4.4.4 Amazon EC2 image

Sipwise provides AMI (Amazon Machine Images) images in all Amazon EC2 regions for the latest and LTS sip:provider CE releases. Please find the appropriate AMI ID for your region in [release announcement](#).

Note

The following documentation will use Amazon region *eu-west-1* with AMI ID *ami-8bef6cfc* as an example. Please find the appropriate AMI ID for your region in [the latest release announcement](#).

As a next step please visit <https://console.aws.amazon.com/ec2/v2/home?region=eu-west-1> with your EC2 account.

Choose "Launch Instance":

Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

[Launch Instance](#)

Note: Your instances will launch in the EU West (Ireland) region

Figure 5: Launch Amazon EC2 Instance for your region

Select "Community AMIs" option, enter "ami-8bef6cfc" inside the search field and press "Select" button:



Figure 6: Choose sip:provider CE image (different for each region)

Select the Instance Type you want to use for running sip:provider CE (recommended: >=2GB RAM):

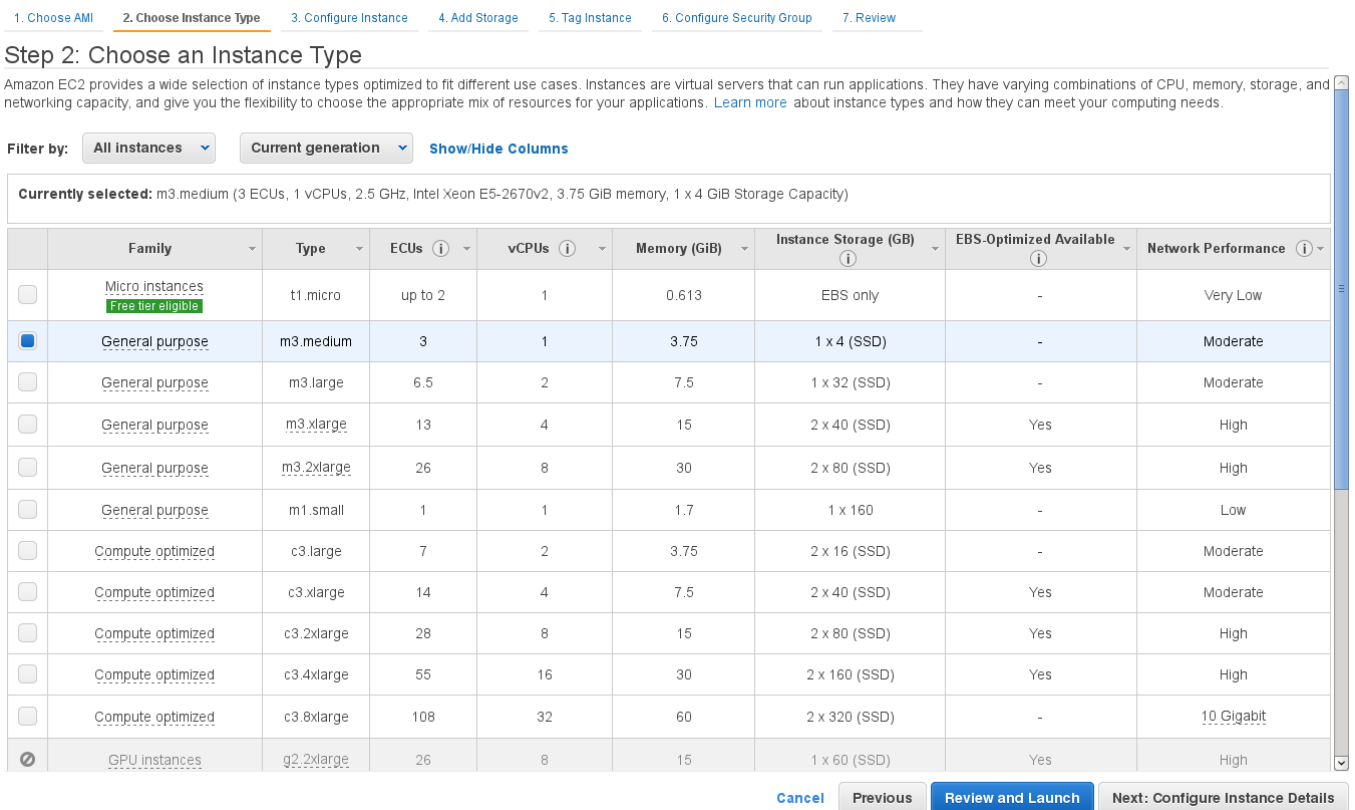


Figure 7: Choose Amazon EC2 instance

Tip

Do not forget to tune necessary sip:provider CE performance parameters depending on Amazon EC2 instance type and performance you are looking for. Sipwise image is tuned for minimum performance to fit Micro instances. Feel free to read more about sip:provider CE performance tuning in Section 17.8.

Run through next configuration options

- Configure Instance: optional (no special configuration required from sip:provider CE)
- Add Storage: choose >=8GB disk size (no further special configuration required from sip:provider CE)
- Tag Instance: optional (no special configuration required from sip:provider CE)
- Configure Security Group: create a new security group (SSH on port 22, HTTPS on port 443, TCP on ports 1443, 2443, 1080 and 5060 as well as UDP on port 5060 are suggested)

Note

Please feel free to restrict the *Source* options in your Security Group to your own (range of) IP addresses.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Tag Instance **6. Configure Security Group** 7. Review

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: Create a **new** security group
 Select an **existing** security group

Security group name:

Description:

| Type ⁱ | Protocol ⁱ | Port Range ⁱ | Source ⁱ |
|-------------------|-----------------------|-------------------------|---------------------|
| SSH | TCP | 22 | Anywhere 0.0.0.0/0 |
| HTTPS | TCP | 443 | Anywhere 0.0.0.0/0 |
| Custom TCP Rule | TCP | 1443 | Anywhere 0.0.0.0/0 |
| Custom TCP Rule | TCP | 2443 | Anywhere 0.0.0.0/0 |
| Custom TCP Rule | TCP | 1080 | Anywhere 0.0.0.0/0 |
| Custom TCP Rule | TCP | 5060 | Anywhere 0.0.0.0/0 |
| Custom UDP Rule | UDP | 5060 | Anywhere 0.0.0.0/0 |

Warning
 Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Figure 8: Configure Security Group

Finally Review instance launch and press "Launch" button:

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Tag Instance 6. Configure Security Group 7. Review

Step 7: Review Instance Launch

AMI Details [Edit AMI](#)

ngcp-ce-mr3.3.1 - ami-37b87340
 Official sip:provider CE AMI for release mr3.3.1.3 [2014-06-23_20:51]
 Root Device Type: ebs Virtualization type: paravirtual

Instance Type [Edit instance type](#)

| Instance Type | ECUs | vCPUs | Memory (GiB) | Instance Storage (GB) | EBS-Optimized Available | Network Performance |
|---------------|------|-------|--------------|-----------------------|-------------------------|---------------------|
| m3.medium | 3 | 1 | 3.75 | 1 x 4 | - | Moderate |

Security Groups [Edit security groups](#)

Security group name ngcp-ce-ec2-demo
Description Settings for sip:provider CE

| Type | Protocol | Port Range | Source |
|-----------------|----------|------------|-----------|
| SSH | TCP | 22 | 0.0.0.0/0 |
| HTTPS | TCP | 443 | 0.0.0.0/0 |
| Custom TCP Rule | TCP | 1443 | 0.0.0.0/0 |
| Custom TCP Rule | TCP | 2443 | 0.0.0.0/0 |
| Custom TCP Rule | TCP | 1080 | 0.0.0.0/0 |
| Custom TCP Rule | TCP | 5060 | 0.0.0.0/0 |
| Custom UDP Rule | UDP | 5060 | 0.0.0.0/0 |

Instance Details [Edit instance details](#)

Storage [Edit storage](#)

[Cancel](#) [Previous](#) [Launch](#)

Figure 9: Launch Amazon EC2 instance with sip:provider CE

Choose an existing key pair which you want to use for logging in, or create a new one if you don't have one.

Select an existing key pair or create a new key pair ✕

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Create a new key pair

Key pair name

sip-provider-ce

Download Key Pair



You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

Cancel

Launch Instances

Figure 10: Choose key pair to access sip:provider CE

You should have a running instance after a few seconds/minutes now (check DNS name/IP address).

The screenshot shows the AWS Management Console interface for EC2 instances. On the left, there is a navigation menu with options like 'EC2 Dashboard', 'Events', 'Tags', 'Reports', 'Limits', 'INSTANCES', 'Instances', 'Spot Requests', and 'Reserved Instances'. The main area displays a table of instances. The table has columns for Name, Instance ID, Instance Type, Availability, Instance State, Status Checks, Alarm Sta, Public DNS, and Public IP. One instance is listed with the following details:

| Name | Instance ID | Instance Type | Availability | Instance State | Status Checks | Alarm Sta | Public DNS | Public IP |
|-----------------|-------------|---------------|--------------|----------------|---------------|-----------|-------------------------|--------------|
| sip-provider-ce | i-ab68c7e8 | m3.medium | eu-west-1b | running | Initializing | None | ec2-54-195-40-219.eu... | 54.195.40... |

Figure 11: Running Amazon EC2 sip:provider CE instance

Logging in via SSH should work now, using the key pair name (being sip-provider-ce.pem as \$keypair in our example) and the DNS name/IP address the system got assigned.

```
ssh -i $keypair.pem admin@$DNS
```

First step should be logging in to the Admin panel (username *administrator*, password *administrator*) and changing the default password: [https://\\$DNS:1443/login/admin](https://$DNS:1443/login/admin) and then follow Section 17 to secure your installation.

Don't forget to add the Advertised IP for kamailio lb instance, since it's required by the Amazon EC2 network infrastructure:

```
ngcp-network --set-interface=eth0 --advertised-ip=<your_public_amazon_ip>
```

Now feel free to use your newly started Amazon EC2 sip:provider CE instance!



Warning

Do not forget to stop unnecessary instance(s) to avoid unexpected costs (see <http://aws.amazon.com/ec2/pricing/>).

5 Initial System Configuration

After the installation went through successfully, you are ready to adapt the system parameters to your needs to make the system work properly.

5.1 Network Configuration

If you have only one network card inside your system, its device name is *eth0*, it's configured and only IPV4 is important to you then there should be nothing to do for you at this stage. If multiple network cards are present, your network card does *not* use *eth0* for its device name or you need IPV6 then the only parameter you need to change at this moment is the listening address for your SIP services.

To do this, you have to specify the interface where your listening address is configured, which you can do with the following command (assuming your public interface is *eth0*):

```
ngcp-network --set-interface=eth0 --ip=auto --netmask=auto
ngcp-network --move-from=lo --move-to=eth0 --type=web_ext --type=sip_ext --type=rtp_ext -- ←
type=ssh_ext --type=web_int
```

If you want to enable IPV6 as well, you have to set the address on the proper interface as well, like this (assuming you have an IPV6 address *fd5a:5cc1:23:4:0:0:0:1f* on interface *eth0*):

```
ngcp-network --set-interface=eth0 --ipv6='FD5A:5CC1:23:4:0:0:0:1F'
```

Tip

Always use a full IPV6 address with 8 octets. Leaving out zero octets (e.g. *FD5A:5CC1:23:4::1F*) is **not allowed**.



Important

You should use the IPV6 address in **upper-case** because LB (kamailio) handles the IPV6 addresses internally in upper-case format.

If you haven't fully configured your network interfaces, do this by adapting also the file `/etc/network/interfaces`:

```
vim /etc/network/interfaces
```

Add or adapt your interface configuration accordingly. For example, if you just want to use the system in your internal network 192.168.0.0/24, it could look something like this:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
    address 1.2.3.4
    netmask 255.255.255.0
    gateway 1.2.3.1
    dns-nameservers 8.8.8.8
    dns-search yourdomain.com
```

```
/etc/init.d/networking restart
```

5.2 Apply Configuration Changes

In order to apply the changes you made to `/etc/ngcp-config/config.yml`, you need to execute the following command to re-generate your configuration files and to automatically restart the services:

```
ngcpcfg apply 'added network interface'
```

Tip

At this point, your system is ready to serve.

5.3 Start Securing Your Server

During installation, the system user `cdrexport` is created. This jailed system account is supposed to be used to export CDR files via sftp/scp. Set a password for this user by executing the following command:

```
passwd cdrexport
```

The installer has set up a MySQL database on your server. You need to set a password for the MySQL root user to protect it from unauthorized access by executing this command:

```
mysqladmin password <your mysql root password>
```


For the *Administrative Web Panel* located at <https://<your-server-ip>:1443/login/admin>, a default user *administrator* with password *administrator* has been created. Connect to the panel (accept the SSL certificate for now) using those credentials and change the password of this user by going to *Settings*→*Administrators* and click the *Edit* when hovering over the row.

5.4 Configuring the Email Server

The NGCP installer will install *mailx* (which has *Exim4* as MTA as a default dependency) on the system, however the MTA is not configured by the installer. If you want to use the *Voicemail-to-Email* feature of the Voicebox, you need to configure your MTA properly. If you are fine to use the default MTA *Exim4*, execute the following command:

```
dpkg-reconfigure exim4-config
```

Depending on your mail setup in your environment (whether to use a smarthost or not), configure Exim accordingly. In the most simple setup, apply the following options when prompted for it:

- **General type of mail configuration:** `internet site;mail is sent and received directly using SMTP`
- **System mail name:** the FQDN of your server, e.g. `ce.yourdomain.com`
- **IP-addresses to listen on for incoming SMTP connections:** `127.0.0.1`
- **Other destinations for which mail is accepted:** the FQDN of your server, e.g. `ce.yourdomain.com`
- **Domains to relay mail for:** leave empty
- **Machines to relay mail for:** leave empty
- **Keep number of DNS-queries minimal (Dial-on-Demand)?** `No`
- **Delivery method for local mail:** `mbox format in /var/mail/`
- **Split configuration into small files?** `No`



Important

You are free to install and configure any other MTA (e.g. postfix) on the system, if you are more comfortable with that.

5.5 Advanced Network Configuration

You have a typical test 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.

5.5.1 Audiocodes devices workaround

As reported by many users, Audiocodes devices suffer from a problem where they replace `127.0.0.1` address in Record-Route headers (added by the sip:provider CE's internal components) with its own IP address. The problem has been reported

to Audiocodes but as of end 2012 the fixed firmware is not available yet so supposedly the whole range of Audiocodes devices, including but not limited to the MP202, MP252 CPEs as well as Audiocodes media gateways, is malfunctioning. As a workaround, you may change the internal IP address from 127.0.0.1 to some dummy network interface. Please execute the following command (in this example 192.168.2.2 is a new internal IP address):

```
ifconfig dummy0 192.168.2.2 netmask 255.255.255.0
```

Adapt your `/etc/network/interfaces` file accordingly:

```
auto dummy0
iface dummy0 inet static
address 192.168.2.2
netmask 255.255.255.0
```

Update the network configuration in the sip:provider CE:

```
ngcp-network --set-interface=dummy0 --ip=auto --netmask=auto
ngcp-network --move-from=lo --move-to=dummy0 --type=sip_int --type=web_int
```

Refer to [the Network Configuration chapter](#) for more details about the `ngcp-network` tool.

Apply configuration:

```
ngcpcfg apply 'audiocodes devs workaround'
```

5.6 What's next?

To test and use your installation, you need to follow these steps now:

1. Create a SIP domain
2. Create some SIP subscribers
3. Register SIP endpoints to the system
4. Make local calls and test subscriber features
5. Establish a SIP peering to make PSTN calls

Please read the next chapter for instructions on how to do this.

6 Administrative Configuration

To be able to configure your first test clients, you will need a Customer, a SIP domain and some subscribers in this domain. Throughout this steps, let's assume you're running the NGCP on the IP address *1.2.3.4*, and you want this IP to be used as SIP domain. This means that your subscribers will have an URI like *user1@1.2.3.4*.

Tip

You can of course set up a DNS name for your IP address (e.g. letting *sip.yourdomain.com* point to *1.2.3.4*) and use this DNS name throughout the next steps, but we'll keep it simple and stick directly with the IP as a SIP domain for now.



Warning

Once you started adding subscribers to a SIP domain, and later decide to change the domain, e.g. from *1.2.3.4* to *sip.yourdomain.com*, you'll need to recreate all your subscribers in this new domain. It's currently not possible to easily change the domain part of a subscriber.

Go to the *Administrative Web Panel (Admin Panel)* running on *https://<ip>:1443/login/admin* and follow the steps below. The default user on the system is *administrator* with the password *administrator*, if you haven't changed it already in [*?simpara*].

6.1 Creating a Customer

A Customer is a special type of contract on the system acting as billing container for SIP subscribers. You can create as many SIP subscribers within a Customer as you want.

To create a Customer, got to *Settings*→*Customers*.

The screenshot shows the sip:wise NGCP Dashboard interface. At the top right, it indicates the user is logged in as 'administrator' with a 'Logout' link. The dashboard title is 'sip:wise NGCP Dashboard'. A 'Settings' dropdown menu is open, listing various configuration options: Administrators, Resellers, Customers (highlighted in orange), Domains, Subscribers, Billing, Peerings, Rewrite Rule Sets, NCOS Levels, Sound Sets, and Security Bans. Below the menu, the dashboard is divided into four main sections: System Status, Resellers, Billing, and a fourth section partially visible. The System Status section shows 'All services running' with green status indicators for Applications, System, and Hardware. The Resellers section shows 9 Resellers, 0 Domains, 0 Customers, and 0 Subscribers. The Billing section shows 6 Billing Profiles, 0.00 Peering Costs, 0.00 Reseller Revenue, and 0.00 Customer Revenue. Each section has a 'Configure' button at the bottom.

Click on *Create Customer*.

The screenshot shows the sip:wise NGCP Dashboard interface. At the top right, it indicates the user is logged in as 'administrator' with a 'Logout' link. The dashboard title is 'sip:wise NGCP Dashboard' and includes a home icon and a 'Settings' dropdown menu. The main section is titled 'Customers'. Below this, there are two buttons: 'Back' and 'Create Customer', with the latter highlighted by a red box. A search input field is located to the right of the buttons. Below the search field is a table with columns: '#', 'External #', 'Reseller', 'Contact Email', 'Billing Profile', and 'Status'. The table is currently empty, displaying the message 'No data available in table'. Below the table, it shows 'Showing 0 to 0 of 0 entries' and navigation controls. The footer contains the copyright notice: '© 2013 Sipwise GmbH, all rights reserved.'

Each *Customer* needs a *Contact*. We can either reuse the default one, but for a clean setup, we create a new *Contact* for each *Customer* to be able to identify the *Customer*. Click on *Create Contact* to create a new *Contact*.

The screenshot shows a web application interface for creating a contract. The main window is titled "Create Contract" and contains two sections: "Contact" and "Billing Profile".

Contact Section:

| # | Reseller | First Name | Last Name | Email |
|---|----------|--------------------|-------------------|--|
| 1 | default | Contact first name | Contact last name | default-customer@default.invalid.contact |

Showing 1 to 1 of 1 entries

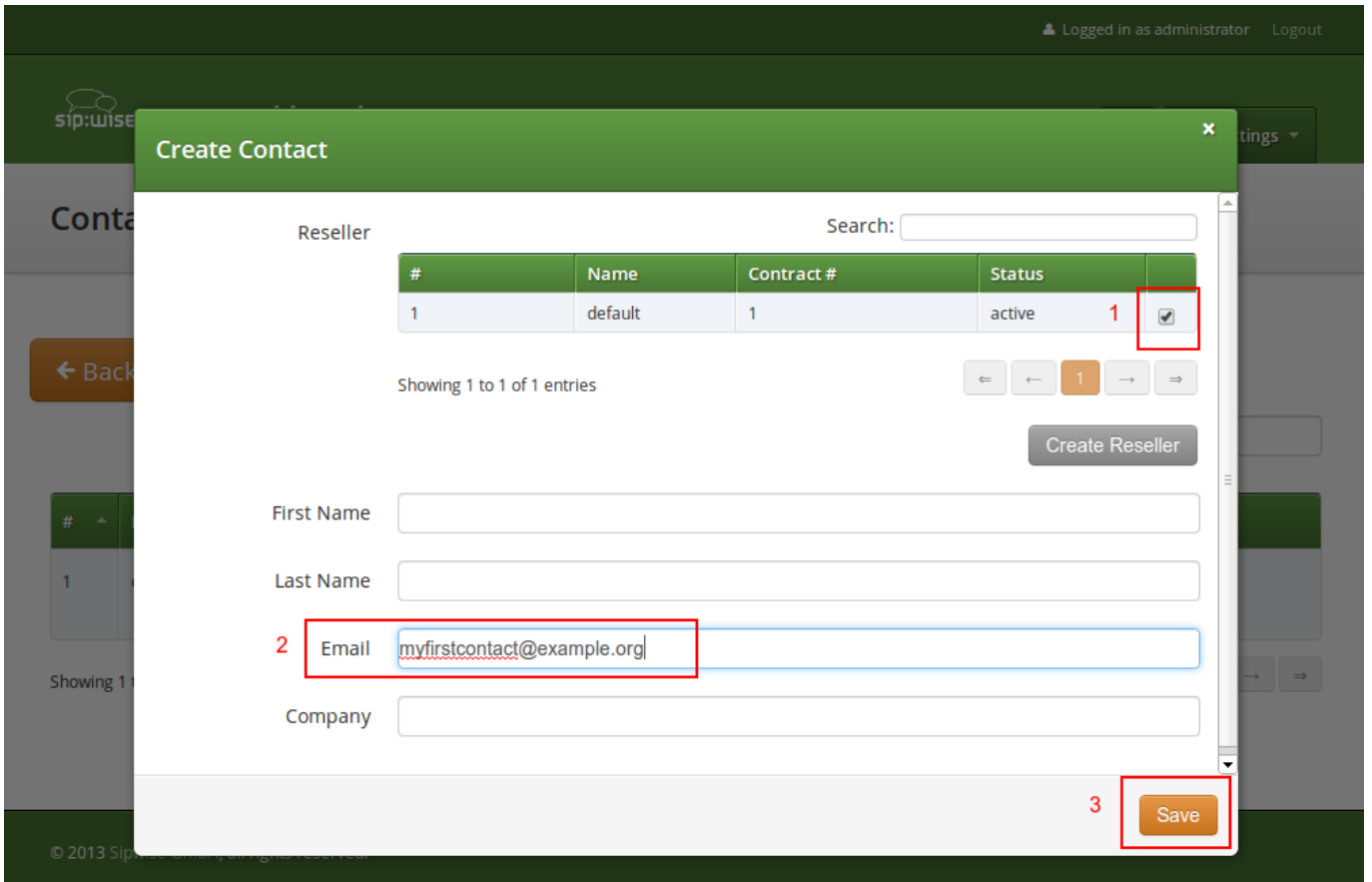
Billing Profile Section:

| # | Reseller | Profile |
|---|----------|-------------------------|
| 1 | default | Default Billing Profile |

Showing 1 to 1 of 1 entries

A red box highlights the "Create Contact" button in the Contact section. At the bottom right of the dialog is a "Save" button.

We assign the Contact to the default *Reseller*. You can create a new one if you want, but for a simple setup the default *Reseller* is sufficient. Select the *Reseller* and enter the contact details (at least an *Email* is required), then press *Save*.



You will be redirected back to the *Contract* form. The newly created *Contact* is selected by default now, so you only have to select a *Billing Profile*. Again you can create a new one on the fly, but we will go with the default profile for now. Select it and press *Save*.

You will now see your first *Customer* in the list. Hover over the customer and click *Details* to view the details.

The screenshot shows the sip:wise NGCP Dashboard interface. At the top right, it indicates the user is logged in as 'administrator' with a 'Logout' link. The dashboard title is 'sip:wise NGCP Dashboard' and includes a home icon and a 'Settings' dropdown menu. The main heading is 'Customers'. Below this are two buttons: 'Back' and 'Create Customer'. A green notification bar states 'Contract successfully created'. A search input field is present. The main content is a table with the following data:

| # | External # | Reseller | Contact Email | Billing Profile | Status | |
|----|------------|----------|----------------------------|-------------------------|--------|--|
| 20 | | default | myfirstcontact@example.org | Default Billing Profile | active | Edit Terminate Details |

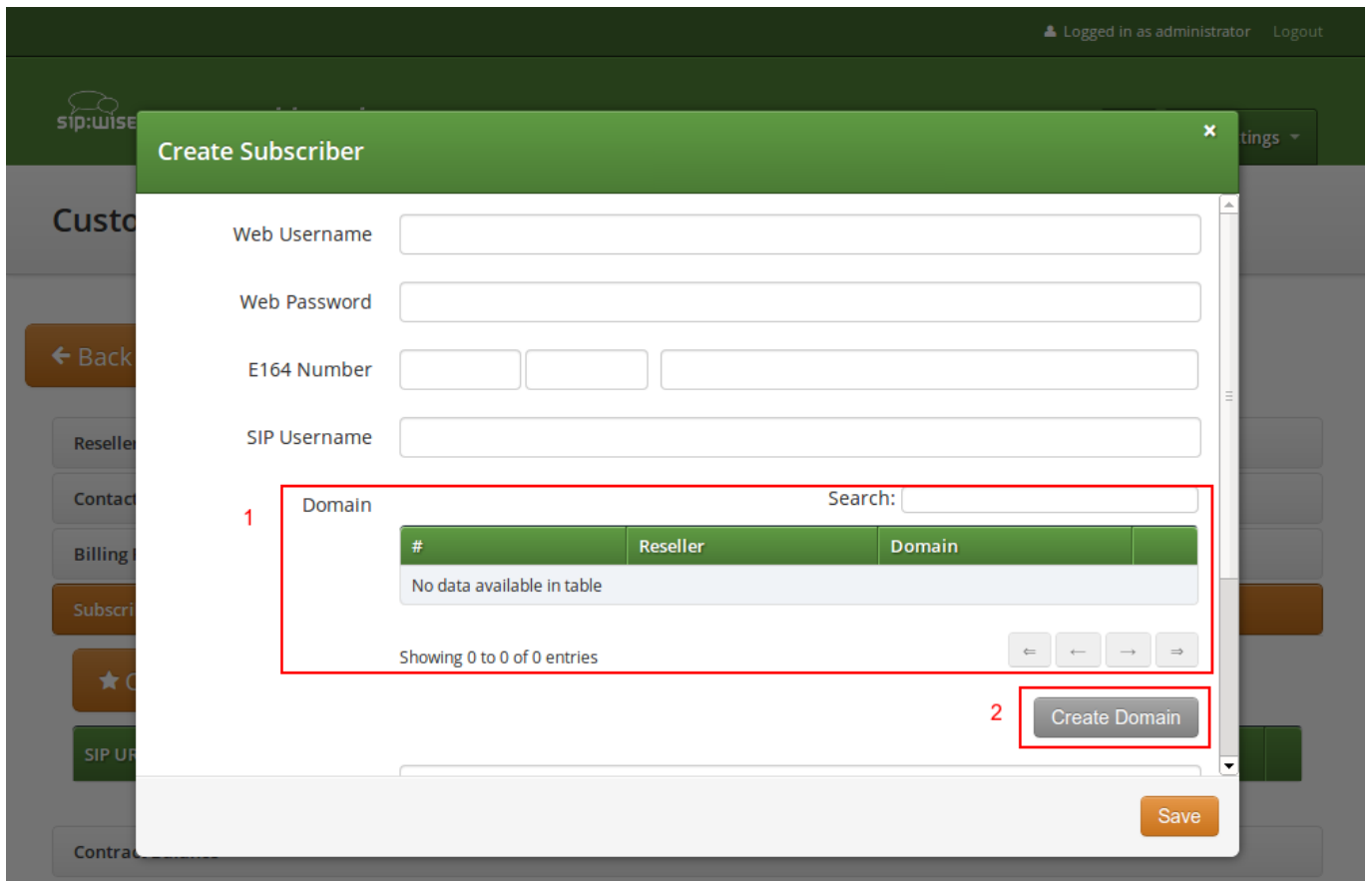
Below the table, it says 'Showing 1 to 1 of 1 entries' and includes pagination controls with a '1' in an orange box.

6.2 Creating a Subscriber

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

The screenshot shows the sip:wise NGCP Dashboard interface. At the top right, it indicates the user is logged in as 'administrator' with a 'Logout' link. The dashboard title is 'sip:wise NGCP Dashboard' with a home icon and a 'Settings' dropdown menu. The main section is titled 'Customer Details'. Below this, there are two buttons: 'Back' and 'Edit'. A list of menu items includes 'Reseller', 'Contact Details', 'Billing Profiles', 'Subscribers', and 'Contract Balance'. The 'Subscribers' item is highlighted in orange and marked with a red box and the number '1'. Below the 'Subscribers' item is a button labeled '★ Create Subscriber', which is also highlighted with a red box and the number '2'. Below the menu items is a table with columns: 'SIP URI', 'Primary Number', and 'Registered Devices'. At the bottom, there is a 'Contract Balance' section.

As you can see, we don't have any *SIP Domains* yet, so click on *Create Domain* to create one.



Select the *Reseller* (make sure to use the same reseller where your *Customer* is created in) and enter your domain name, then press *Save*.

The screenshot shows the 'Create Domain' dialog box. At the top, it says 'Reseller' and has a search field. Below is a table with the following data:

| # | Name | Contract # | Status |
|---|---------|------------|--------|
| 1 | default | 1 | active |

Below the table, it says 'Showing 1 to 1 of 1 entries'. There are navigation buttons and a 'Create Reseller' button. Below the table, there is a 'SIP Domain' input field with the value '1.2.3.4'. At the bottom right, there is a 'Save' button.

Your *Domain* will be preselected now, so fill out the rest of the form:

- **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 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.
- **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 a made-up 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 phantasy 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 Section 6.6.

Important

NGCP allows single subscriber to have multiple E.164 numbers to be used as aliases for receiving incoming calls. Also NGCP supports "implicit" extensions, e.g. if a subscriber has number 012345, but somebody calls 012345100, then it first tries to send the call to number 012345100 (even though the user is registered as myusername), and only after 404 it falls back to the user-part for which the user is registered.

- **SIP Username:** The user part of the SIP URI for your subscriber.
- **SIP Domain:** The domain 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 *active*.
- **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*.

The screenshot shows the 'Create Subscriber' form in the NGCP Dashboard. The form includes the following elements:

- Web Password:** An empty text input field.
- E164 Number:** A form with three input boxes containing '43', '99', and '1001'.
- SIP Username:** A text input field containing 'testuser1'.
- Domain:** A section with a search bar and a table.

| # | Reseller | Domain | |
|---|----------|---------|-------------------------------------|
| 6 | default | 1.2.3.4 | <input checked="" type="checkbox"/> |
- SIP Password:** A text input field containing 'mysecretpassword'.
- Save:** A button at the bottom right of the form.

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

Tip

At this point, you're able to register your subscribers to the NGCP and place calls between these subscribers.

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

6.3 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 *Settings*→*Domains* and click on the *Preferences* button of the domain you want to configure.

The screenshot shows the NGCP Dashboard interface. At the top, it says "Logged in as administrator" and "Logout". The main header is "sip:wise NGCP Dashboard" with a "Settings" dropdown menu. The main content area is titled "Domains". There are two buttons: "Back" and "Create Domain". A search bar is present. Below is a table with one entry:

| # | Reseller | Domain | |
|---|----------|---------|--|
| 6 | default | 1.2.3.4 | <input type="button" value="Delete"/> <input type="button" value="Preferences"/> |

Below the table, it says "Showing 1 to 1 of 1 entries" and there are pagination controls. At the bottom, there is a copyright notice: "© 2013 Sipwise GmbH, all rights reserved."

The most important settings are in the group *Number Manipulations*, where you can configure where from a SIP message to take numbers from for incoming messages, where in the SIP messages to put which numbers for outgoing SIP messages, and 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 Section 6.6, then assign it to the domain by editing the *rewrite_rule_set* preference.

Domain "1.2.3.4" - Preferences







← Back

Call Blockings

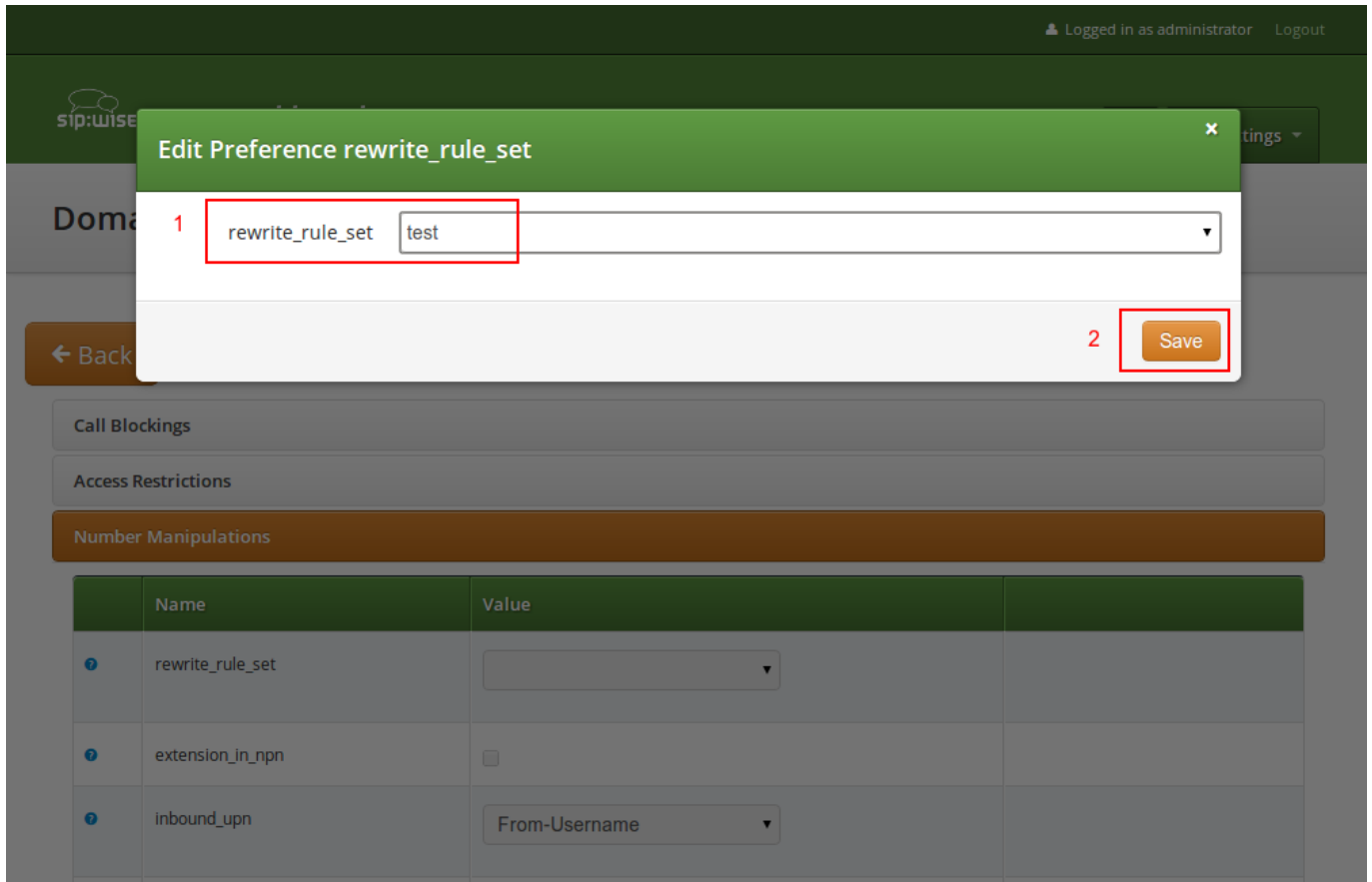
Access Restrictions

1

Number Manipulations

| | Name | Value | |
|---|-----------------------|---|--|
|  | rewrite_rule_set | <input type="text" value=""/> | 2  Edit |
|  | extension_in_npn | <input type="checkbox"/> | |
|  | inbound_upn | <input type="text" value="From-Username"/> | |
|  | outbound_from_user | <input type="text" value="User-Provided-Number"/> | |
|  | outbound_from_display | <input type="text" value="None"/> | |

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 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.

6.4 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 *Settings*→*Subscribers* 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 also.
- 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 *oss-bss→provisioning→auto_allow_cli* is set to **1** in */etc/ngcp-config/config.yml*.

Note

Subscribers preference *cli* will be synchronized with subscribers primary number and aliases if *oss-bss→provisioning→auto_sync_cli* is set to **yes** in */etc/ngcp-config/config.yml*.

6.5 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 *Settings→Peerings*. 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.

6.5.1 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.

The screenshot shows the 'Create SIP Peering Groups' modal dialog in the Sipwise web interface. The dialog is titled 'Create SIP Peering Groups' and has a green header. It contains a table for selecting a contract, a search bar, and form fields for Name, Priority, and Description. A 'Create Contract' button is highlighted with a red box.

Contract

Search:

| # | Status | Billing Profile |
|----------------------------|--------|-----------------|
| No data available in table | | |

Showing 0 to 0 of 0 entries

Name

Priority

Description

© 2013 Sipwise GmbH, all rights reserved.

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

The screenshot shows the 'Create Contract' modal window. It contains two tables for selection. The 'Contact' table has the following data:

| # | Reseller | First Name | Last Name | Email | |
|----|----------|--------------------|-------------------|--|---|
| 1 | default | Contact first name | Contact last name | default-customer@default.invalid.contact | 1.1 <input checked="" type="checkbox"/> |
| 17 | default | | | myfirstcontact@example.org | <input type="checkbox"/> |

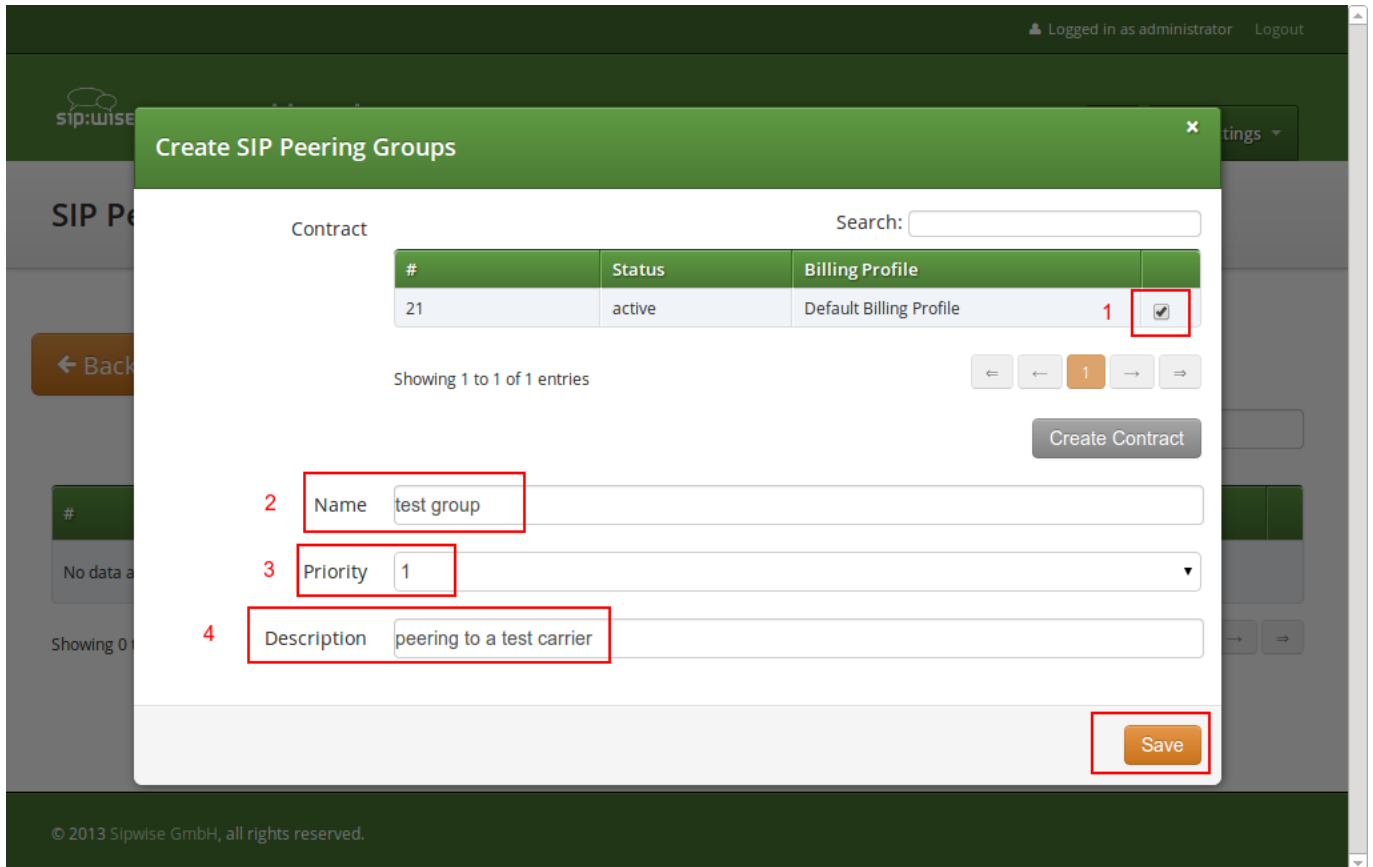
The 'Billing Profile' table has the following data:

| # | Reseller | Profile | |
|---|----------|-------------------------|---------------------------------------|
| 1 | default | Default Billing Profile | 2 <input checked="" type="checkbox"/> |

Navigation and action elements include: 'Showing 1 to 2 of 2 entries' for contacts, 'Showing 1 to 1 of 1 entries' for billing profiles, a 'Create Contact' button (labeled 'or 1.2'), and a 'Save' button (labeled '3').

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



The *Priority* option defines which *Peering Group* to favor if two peering groups have peering rules matching an outbound call. *Peering Rules* are described below.

Then click *Save* to create the group.

6.5.2 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.

The screenshot shows a web interface for managing peering servers and rules. At the top, there is a section titled "Peering Servers" with two buttons: "← Back" and "★ Create Peering Server". The "Create Peering Server" button is highlighted with a red box and a red number "1". Below this is a search input field. A table with columns: #, Name, IP Address, Hostname, Port, Protocol, Weight, and Via Route Set is shown, containing the message "No data available in table". Below the table, it says "Showing 0 to 0 of 0 entries" and has four navigation buttons: ←, ←, →, ⇒.

Below the "Peering Servers" section is a section titled "Peering Rules" with a button "★ Create Peering Rule". It also has a search input field. A table with columns: #, Callee Prefix, Callee Pattern, Caller Pattern, and Description is shown, containing the message "No data available in table". Below the table, it says "Showing 0 to 0 of 0 entries" and has four navigation buttons: ←, ←, →, ⇒.

At the bottom of the interface, there is a green footer bar with the text "© 2013 Sipwise GmbH, all rights reserved."

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

The screenshot shows the 'Create Peering Servers' dialog in the NGCP Dashboard. The dialog is a white box with a green header and a close button. It contains several input fields: 'Name' (test-gw-1), 'IP Address' (2.3.4.5), 'Port' (5060), 'Protocol' (UDP), and 'Weight' (1). There are also empty fields for 'Hostname' and 'Via Route' (set to 'None'). A 'Save' button is at the bottom right. Red boxes highlight the input fields and the Save button, with numbers 1 through 6 indicating the sequence of steps.

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 domain part in the Request URI. Some peers may require you to set a particular hostname instead of the IP address there, which can be done by filling in this field. The IP address must always be given though, and the request will always be sent to the IP address, no matter what you put into the *hostname* field.

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 `kamailio→lb→external_sbc` like this:

```
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.



Important

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

Peering Servers

← Back
★ Create Peering Server

Peering server successfully created

Search:

| # ^ | Name | IP Address | Hostname | Port | Protocol | Weight | Via Route Set |
|-----|-----------|------------|----------|------|----------|--------|---------------|
| 3 | test-gw-1 | 2.3.4.5 | | 5060 | 1 | 1 | |

Showing 1 to 1 of 1 entries ← → 1

Peering Rules

★ Create Peering Rule ¹

Search:

| # ^ | Callee Prefix | Callee Pattern | Caller Pattern | Description |
|----------------------------|---------------|----------------|----------------|-------------|
| No data available in table | | | | |

Showing 0 to 0 of 0 entries ← →

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

The screenshot shows the 'Create Peering Rules' modal in the sip:wise NGCP Dashboard. The modal contains the following fields:

- Callee prefix
- Callee pattern
- Caller pattern
- Description: Default Rule (highlighted with a red box and a red '1')

A 'Save' button is located at the bottom right of the modal, highlighted with a red box and a red '2'.

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.

Peering Servers

[← Back](#)
[★ Create Peering Server](#)
Search:

| # ^ | Name | IP Address | Hostname | Port | Protocol | Weight | Via Route Set |
|-----|-----------|------------|----------|------|----------|--------|---------------|
| 3 | test-gw-1 | 2.3.4.5 | | 5060 | 1 | 1 | |

Showing 1 to 1 of 1 entries

← ← 1 → ⇒

Peering Rules

[★ Create Peering Rule](#)

Peering rule successfully created

Search:

| # ^ | Callee Prefix | Callee Pattern | Caller Pattern | Description |
|-----|---------------|----------------|----------------|--------------|
| 1 | | | | Default Rule |

Showing 1 to 1 of 1 entries

← ← 1 → ⇒

The selection of peering groups for outgoing calls is done in the following order:

- 1. whether caller or callee pattern matched.
- 2. length of the callee prefix.
- 3. priority of the peering group.
- 4. weight of the peering servers in the selected peering group.

After one or more peering group(s) is matched for an outbound call, all servers in this group are tried, according to their weight (higher weight has more precedence). Weight of peering servers just give you a probability that the peer will be choose first. In order to know this probability, knowing the peering weights, it's possible to use the following script:

```
#!/usr/bin/php
<?php

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

if ($argc < 2) {
    echo "Usage: lcr_weight_test.php <list of weights (integers 1-254)>\n";
```

```
    exit;
}

$iters = 10000;

$rand = array();
for ($i = 1; $i <= $iters; $i++) {
    $elem = array();
    for ($j = 1; $j < $argc; $j++) {
        $elem["$j"] = $argv[$j] * (rand() >> 8);
    }
    $rand[] = $elem;
}

$sorted = array();
foreach ($rand as $r) {
    asort($r);
    $sorted[] = $r;
}

$count = array();
for ($j = 1; $j < $argc; $j++) {
    $count["$j"] = 0;
}

foreach ($sorted as $r) {
    end($r);
    $count[key($r)]++;
}

for ($j = 1; $j < $argc; $j++) {
    echo "Peer with weight " . $argv[$j] . " has probability " . $count["$j"]/$iters . "\n";
}
?>
```

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

```
# lcr_weight_test.php 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 used as a fallback, one after the other until the call succeeds. If no more servers are left in the current peering group, the next group which matches the peering rules is going to be used.

6.5.3 Authenticating and Registering against Peering Servers

Proxy-Authentication for outbound calls

If a peering server requires the SPCE 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.

Peering Servers

[← Back](#)
[★ Create Peering Server](#)

 Search:

| # | ^ | Name | IP Address | Hostname | Port | Protocol | Weight | |
|---|---|-----------|------------|----------|------|----------|--------|---|
| 1 | | test-gw-1 | 2.3.4.5 | | 5060 | 1 | 1 | Edit Delete Preferences |

Showing 1 to 1 of 1 entries

Peering Rules

[★ Create Peering Rule](#)

 Search:

| # | ^ | Callee Prefix | Callee Pattern | Callee Pattern | Description | |
|---|---|---------------|----------------|----------------|--------------|--|
| 2 | | | | | Default Rule | |

Showing 1 to 1 of 1 entries

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

- **peer_auth_user:** <username for peer auth>
- **peer_auth_pass:** <password for peer auth>
- **peer_auth_realm:** <domain for peer auth>

← Back


Preference peer_auth_realm successfully updated.

Access Restrictions

Number Manipulations

NAT and Media Flow Control

Remote Authentication

| | Name | Value | |
|---|-------------------------|--------------------------|--|
|  | peer_auth_user 1 | peeruser1 | |
|  | peer_auth_pass 2 | peerpass1 | |
|  | peer_auth_realm 3 | testpeering.com | |
|  | peer_auth_register | <input type="checkbox"/> | |
|  | find_subscriber_by_uuid | <input type="checkbox"/> | |

Session Timers

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

You will notice that these three preferences are also shown in the *Subscriber Preferences* for each subscriber. There you can override the authentication details for all peer host if needed, e.g. if every user authenticates with his own separate credentials at your peering provider.

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 NGCP 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

Unfortunately, the credentials configured above are not yet automatically used to register the SPCE at your peer hosts. There is however an easy manual way to do so, until this is addressed.

Configure your peering servers with the corresponding credentials in `/etc/ngcp-config/templates/etc/ngcp-sems/etc/reg_agent.conf.tt2`, then execute `ngcpcfg apply 'added upstream credentials'`.

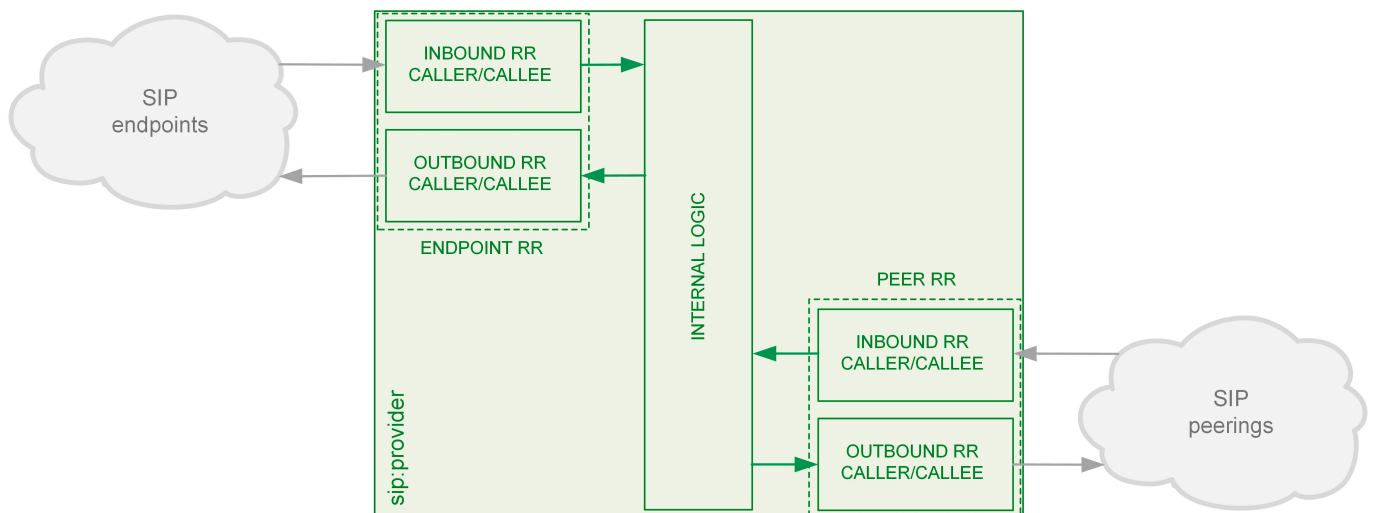


Important

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

6.6 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:

- 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 how to use them are also provided in the next chapters:

- `${caller_cc}` : This is the value taken from the subscriber's preference CC value under Number Manipulation
- `${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

To create a new Rewrite Rule Set, go to *Settings*→*Rewrite 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.

The screenshot shows the 'Rewrite Rule Sets' page in the sip:wise NGCP Dashboard. At the top right, it indicates 'Logged in as administrator' and a 'Logout' link. The dashboard header includes the 'sip:wise NGCP Dashboard' logo and a 'Settings' menu. The main heading is 'Rewrite Rule Sets'. Below this, there are two buttons: 'Back' and 'Create Rewrite Rule Set', with the latter highlighted by a red box. A search bar is located to the right of the buttons. Below the search bar is a table with the following data:

| # | Reseller | Name | Description |
|---|----------|------------|----------------|
| 1 | default | defaultdom | Default Domain |

Below the table, it says 'Showing 1 to 1 of 1 entries' and there are pagination controls showing '1'.

At the bottom of the dashboard, there is a footer: '© 2013 Sipwise GmbH, all rights reserved.'

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

Logged in as administrator Logout

Create Rewrite Rule Sets

Reseller Search:

| # | Name | Contract # | Status | |
|---|---------|------------|--------|---------------------------------------|
| 1 | default | 1 | active | 1 <input checked="" type="checkbox"/> |

Showing 1 to 1 of 1 entries

Create Reseller

Name 2

Description 3

4

© 2013 Sipwise GmbH, all rights reserved.

Press the *Save* button to create the set.

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

Logged in as administrator Logout

sip:wise NGCP Dashboard

Home Settings

Rewrite Rule Sets

Back Create Rewrite Rule Set

Rewrite rule set successfully created

Search:

| # | Reseller | Name | Description | |
|---|----------|-----------------|----------------------|---|
| 1 | default | defaultdom | Default Domain | |
| 2 | default | domain-dialplan | Dialplan for Domains | Edit Delete Rules |

Showing 1 to 2 of 2 entries

← 1 →

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.

6.6.1 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

- 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

The screenshot shows the 'Create Rule' dialog box in the Sipwise interface. The dialog is a white box with a green header and a light gray footer. It contains several input fields and dropdown menus, each with a red box and a number indicating a specific field:

- 1. Match pattern: `^([00|+)([1-9][0-9]+)$`
- 2. Replacement Pattern: `\2`
- 3. Description: International to E.164
- 4. Direction: Inbound
- 5. Field: Caller
- 6. Save button

The background shows a blurred interface with the 'sip:wise' logo and 'Rewr...' text.

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.



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 just 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.

Rewrite Rules for domain-dialplan

← Back

★ Create Rewrite Rule

Rewrite rule successfully created

Inbound Rewrite Rules for Caller

| | Match Pattern | Replacement Pattern | Description | |
|---|---------------|-------------------------|------------------------------|------------------------|
| 1 | ⬆️ ⬇️ | ^(00 \+)([1-9][0-9]+)\$ | \2 | International to E.164 |
| | ⬆️ ⬇️ 2 | ^0([1-9][0-9]+)\$ | \${caller_cc}\1 | National to E.164 |
| | ⬆️ ⬇️ | ^([1-9][0-9]+)\$ | \${caller_cc}\${caller_ac}\1 | Local to E.164 |

Inbound Rewrite Rules for Callee

Outbound Rewrite Rules for Caller

Outbound Rewrite Rules for Callee

6.6.2 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

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.

6.6.3 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

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).

6.6.4 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

6.6.5 Emergency Number Handling

Configuring Emergency Numbers is also done via Rewrite Rules.

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.

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

6.6.6 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 *Settings*→*Domains* and click *Preferences* on the domain you want the set to assign to. Click on *Edit* and select the Rewrite Rule Set created before.

The screenshot shows the 'sip:wise NGCP Dashboard' for the domain 'demo.sipwise.com' - Preferences. The interface includes a 'Back' button, a 'Call Blockings' section, an 'Access Restrictions' section with a count of 1, and a highlighted 'Number Manipulations' section. Below this is a table with columns 'Name' and 'Value'. The first row is 'rewrite_rule_set' with a value of 'defaultdom' and an 'Edit' button. The other rows are 'extension_in_npn', 'inbound_upn', and 'outbound_from_user'.

| Name | Value |
|--------------------|--------------------------|
| rewrite_rule_set | defaultdom |
| extension_in_npn | <input type="checkbox"/> |
| inbound_upn | From-Username |
| outbound_from_user | User-Provided-Number |

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, then click the *Preferences* button.

6.6.7 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 Section 6.6 (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.

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.

7 Advanced Subscriber Configuration

The sip:provider CE provides a large amount of subscriber features in order to offer compelling VoIP services to end customers, and also to cover as many deployment scenarios as possible. In this chapter, we will go over the features and describe their behavior and their use cases.

7.1 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 outbound direction and which numbers or patterns are allowed to call a subscriber in inbound direction. The other is *NCOS Levels*, where the admin predefines rules for outbound calls, which are grouped in certain levels. The user can then just choose the level, or the admin can restrict a user to a certain level. Also sip:provider CE offers some options to restrict the IP addresses that subscriber is allowed to use the service from. The following chapters will discuss these features in detail.

7.1.1 Block Lists

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

| Trusted Sources | |
|--------------------|--------------------------|
| Call Blockings | |
| Name | Value |
| block_in_mode | <input type="checkbox"/> |
| block_in_list | |
| block_in_clir | <input type="checkbox"/> |
| block_out_mode | <input type="checkbox"/> |
| block_out_list | |
| adm_block_in_mode | <input type="checkbox"/> |
| adm_block_in_list | |
| adm_block_in_clir | <input type="checkbox"/> |
| adm_block_out_mode | <input type="checkbox"/> |
| adm_block_out_list | |
| ncos | <input type="text"/> |

Block Lists are separated into *Administrative Block Lists* (*adm_block_**) and *Subscriber Block Lists* (*block_**). They both have

the same behavior, 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.

Block Modes

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

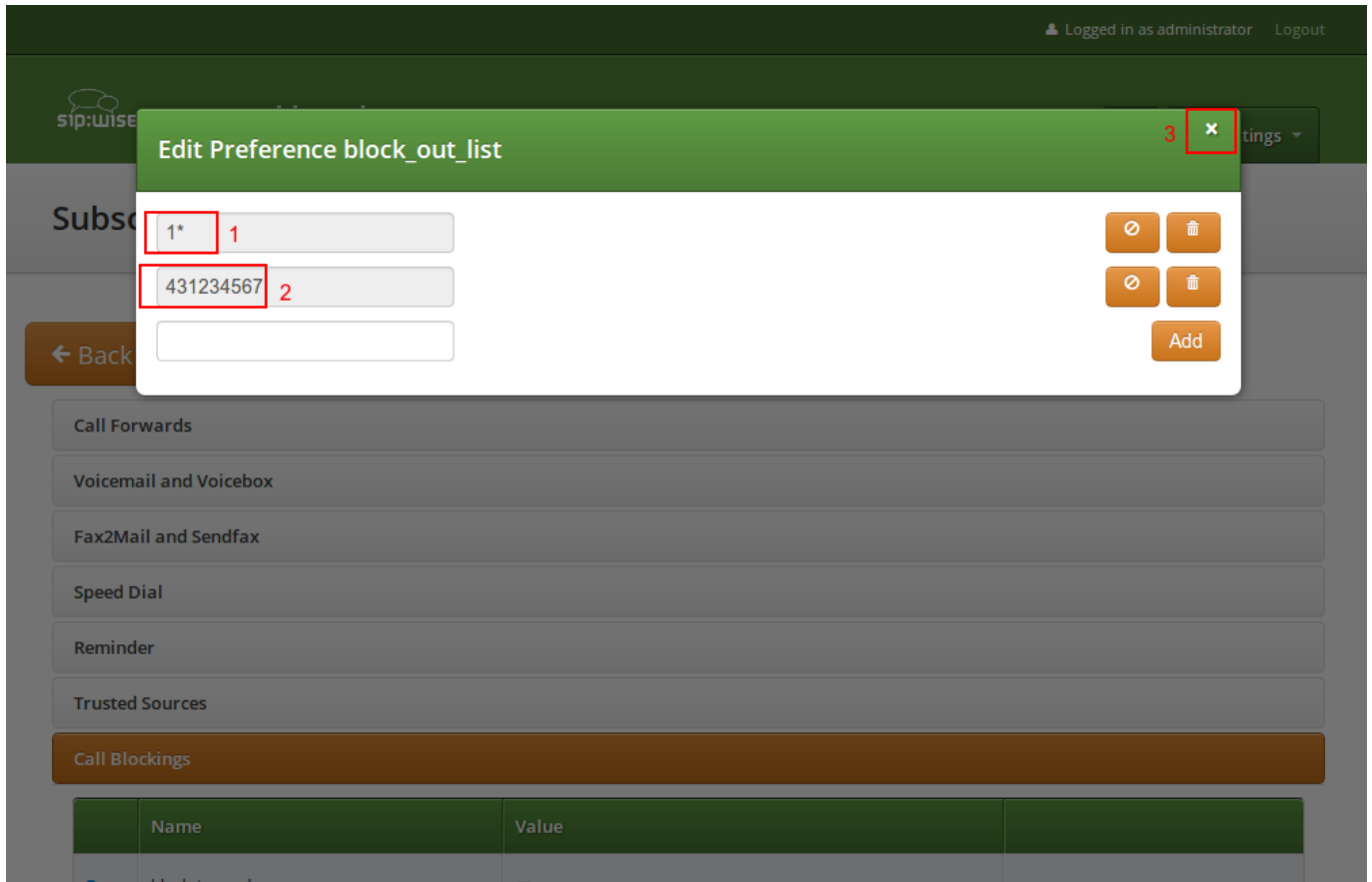
- The *blacklist* mode (option is not checked) tells the system to **allow anything except the entries in the list**. This mode is used if you want to just block certain numbers and allow all the rest.
- The *whitelist* mode indicates to **reject anything except the entries in the list**. This is used 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 From-URI or a header *Privacy: id* is set). This flag is independent from the Block Mode.

7.1.2 NCOS Levels

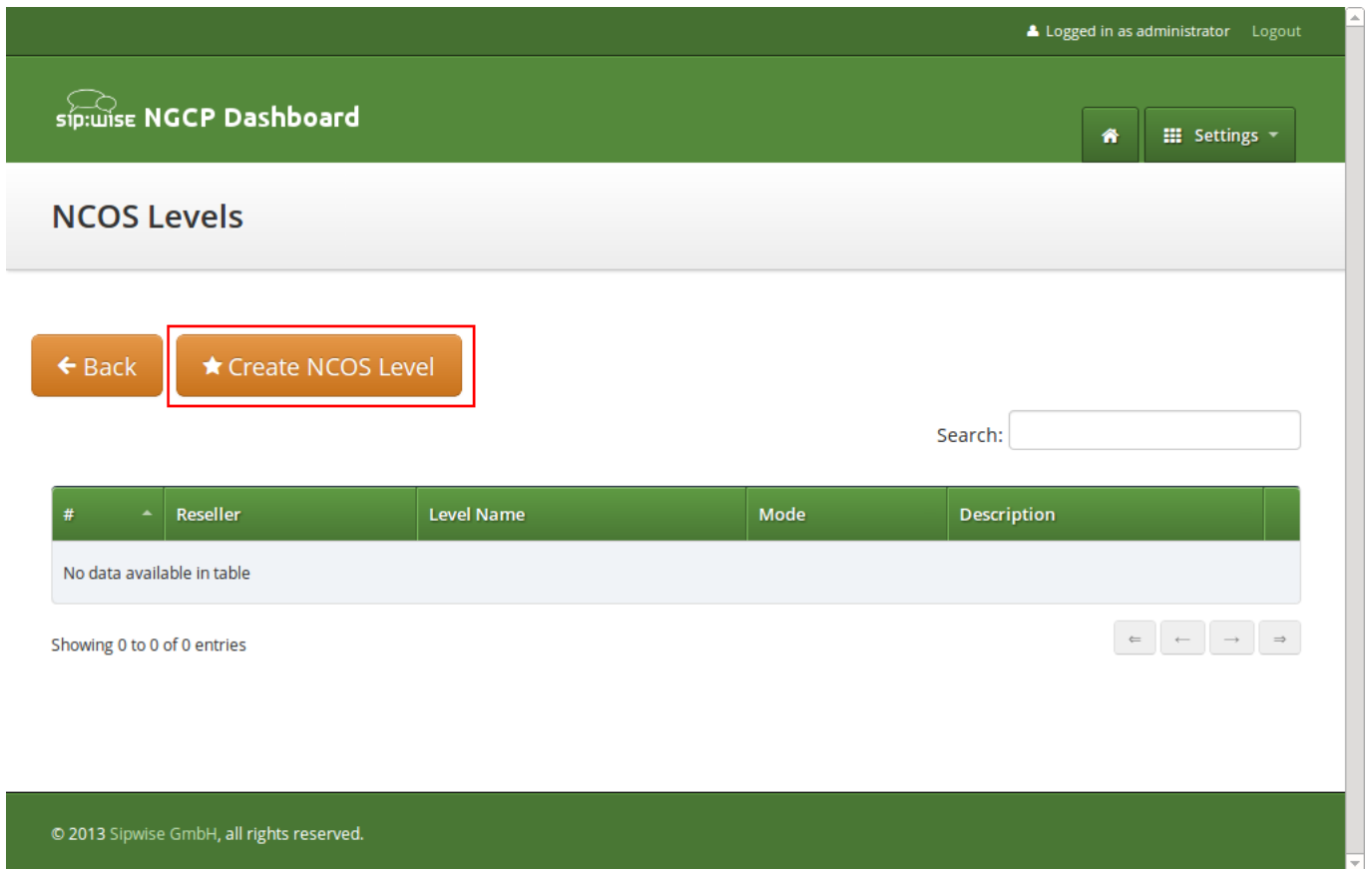
NCOS Levels provide predefined lists of allowed or denied destinations for outbound calls of local subscribers. Compared to *Block Lists*, they are much easier to manage, because they are defined on a global scope, and the individual levels can then be assigned to each subscriber. Again there is the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

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

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

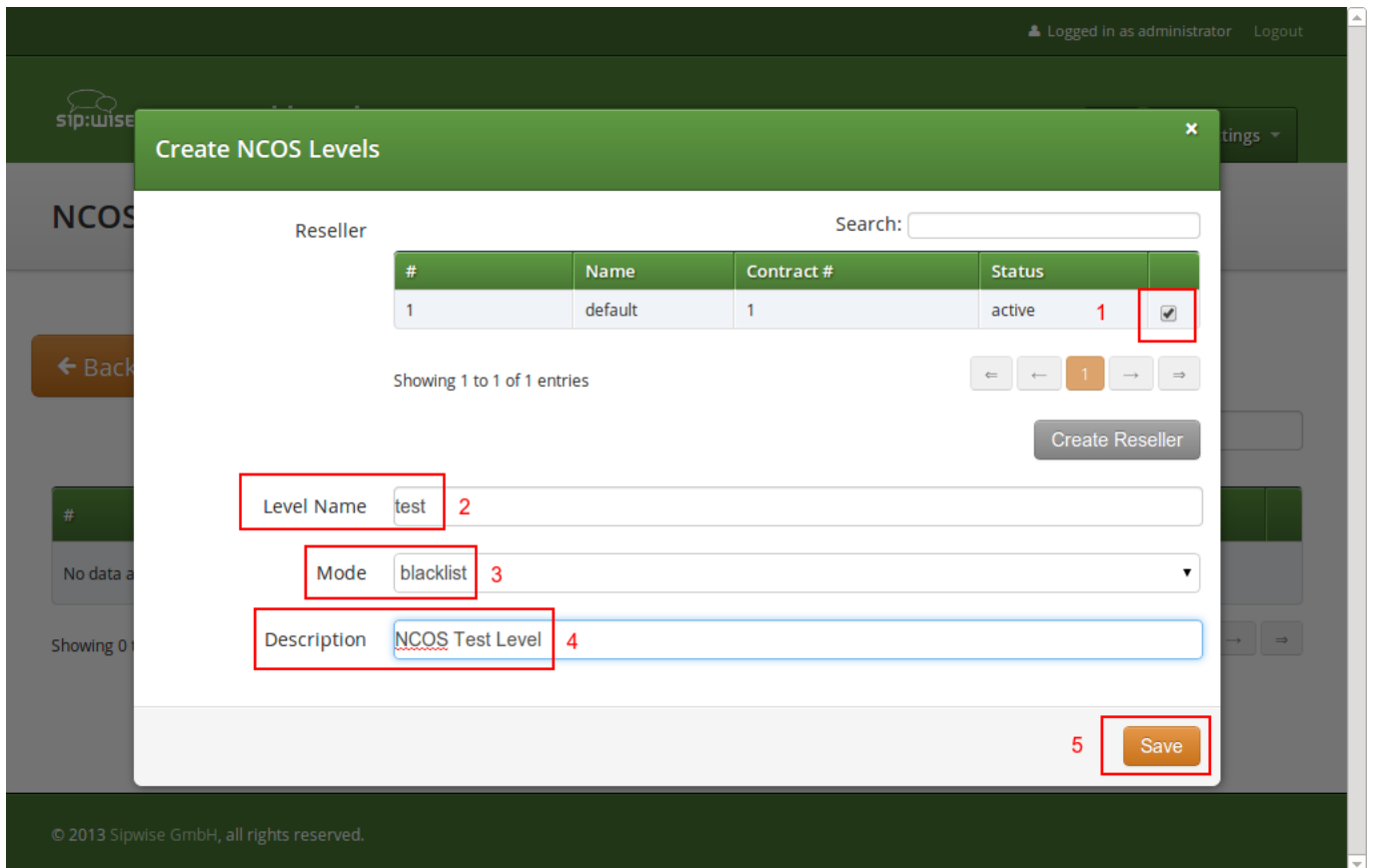
Creating NCOS Levels

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



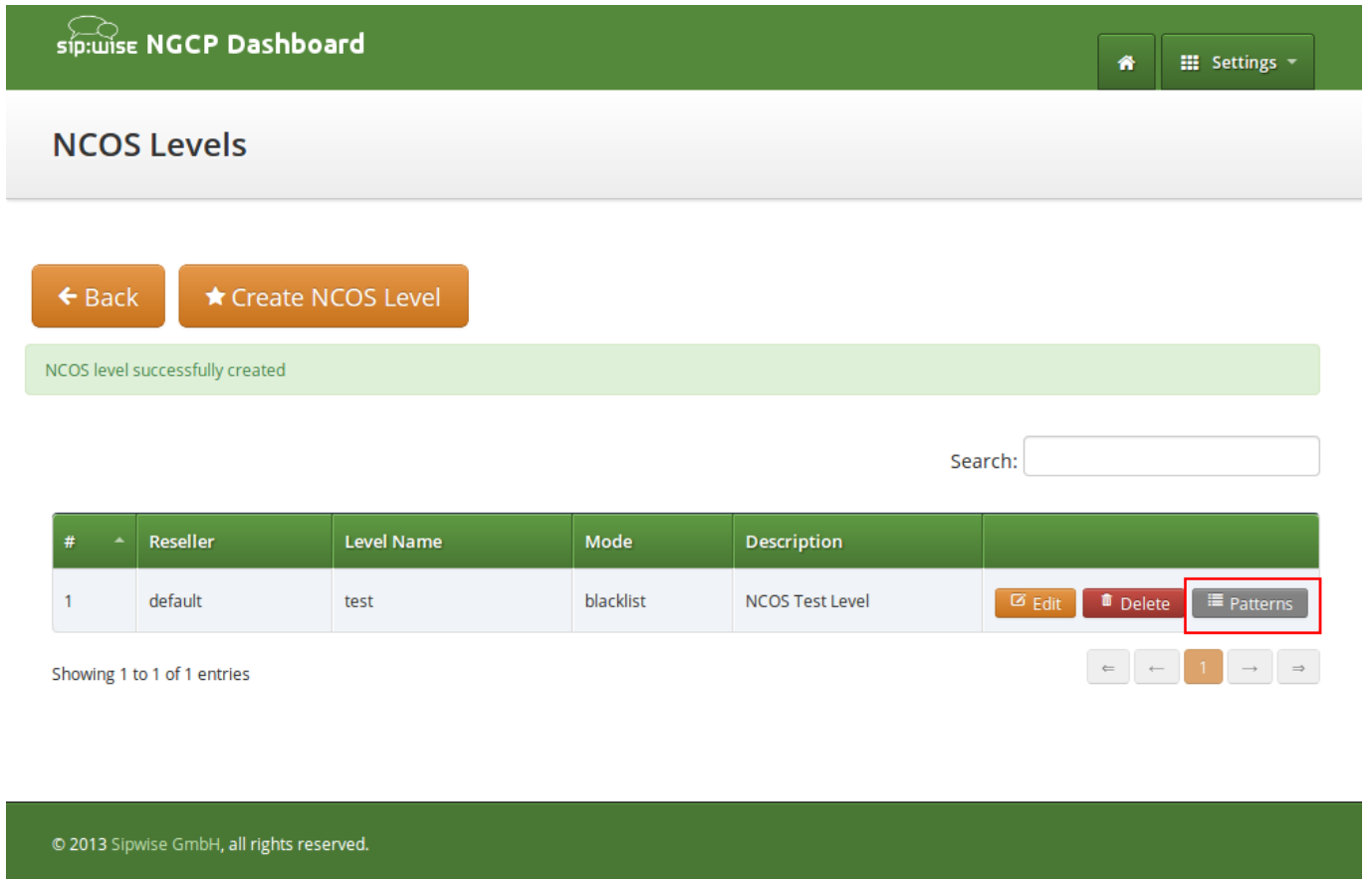
The screenshot shows the sip:wise NGCP Dashboard interface. At the top right, it indicates the user is logged in as an administrator. The main header displays the sip:wise logo and the text 'NGCP Dashboard'. Below this, the page title is 'NCOS Levels'. There are two buttons: a 'Back' button and a 'Create NCOS Level' button, which is highlighted with a red rectangular box. To the right of these buttons is a search input field labeled 'Search:'. Below the buttons and search field is a table with the following columns: '#', 'Reseller', 'Level Name', 'Mode', and 'Description'. The table currently contains no data, with the message 'No data available in table' displayed. Below the table, it shows 'Showing 0 to 0 of 0 entries' and navigation arrows. At the bottom of the dashboard, there is a footer with the text '© 2013 Sipwise GmbH, all rights reserved.'

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



Creating Rules per NCOS Level

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



sip:wise NGCP Dashboard

Home Settings

NCOS Levels

← Back ★ Create NCOS Level

NCOS level successfully created

Search:

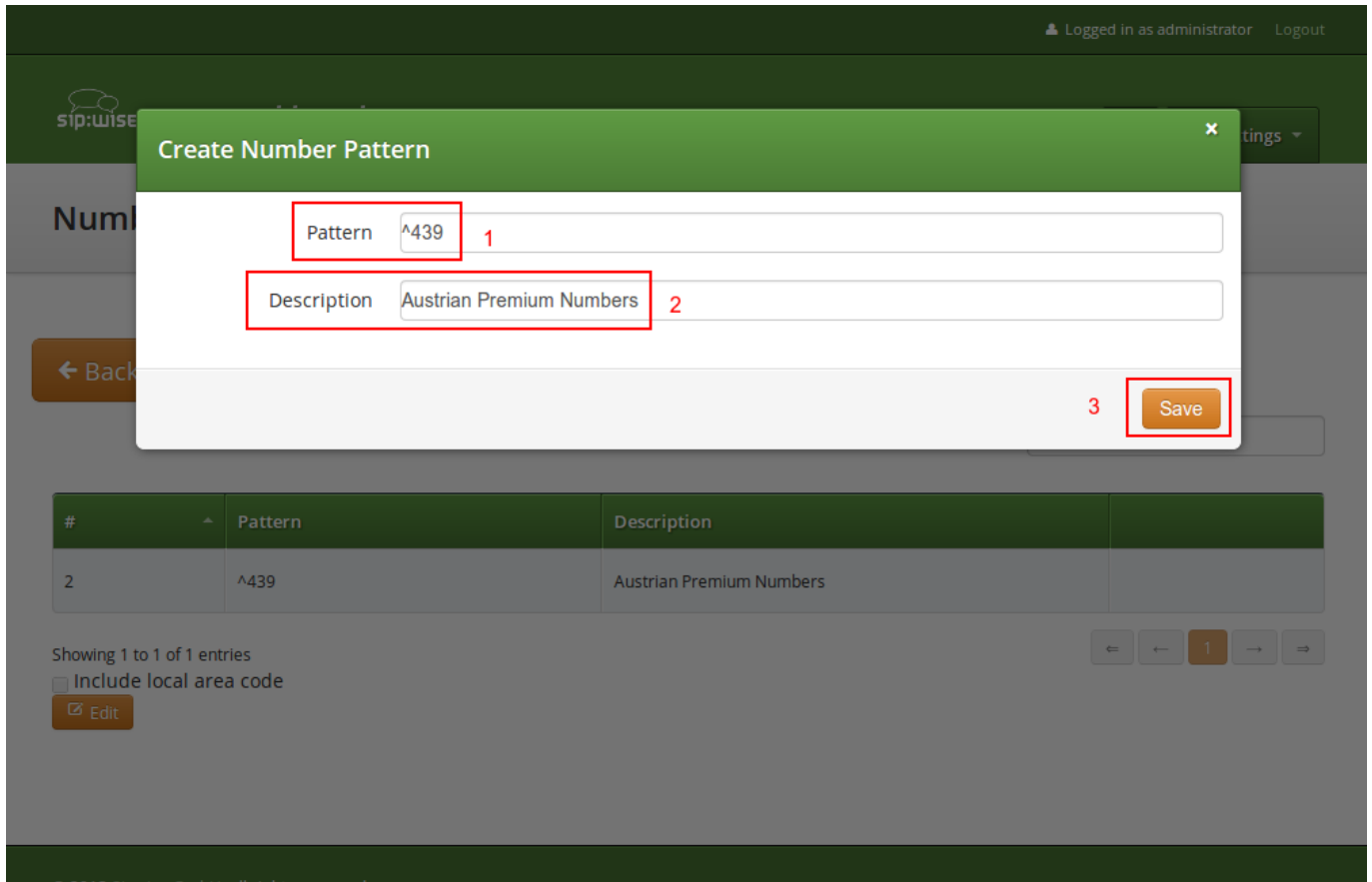
| # | Reseller | Level Name | Mode | Description | |
|---|----------|------------|-----------|-----------------|--|
| 1 | default | test | blacklist | NCOS Test Level | Edit Delete Patterns |

Showing 1 to 1 of 1 entries

← 1 →

© 2013 Sipwise GmbH, all rights reserved.

In the *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.



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.

The option *include local area code in list* for a blacklist means that calls within the area code of the subscribers are denied, and for whitelist that they are allowed, respectively. For example if a subscriber has country-code 43 and area-code 1, then selecting this checkbox would result in an implicit entry 431 .

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 *Settings*→*Subscribers*, 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 *admncos* setting in the *Call Blockings*__ section.

| Call Blockings | | | |
|----------------|--------------------|--------------------------|---------------------------------------|
| | Name | Value | |
| 1 | block_in_mode | <input type="checkbox"/> | |
| | block_in_list | | |
| | block_in_clir | <input type="checkbox"/> | |
| | block_out_mode | <input type="checkbox"/> | |
| | block_out_list | 1* 431234567 | |
| | adm_block_in_mode | <input type="checkbox"/> | |
| | adm_block_in_list | | |
| | adm_block_in_clir | <input type="checkbox"/> | |
| | adm_block_out_mode | <input type="checkbox"/> | |
| | adm_block_out_list | | |
| | ncos 2 | <input type="text"/> | 3 <input type="button" value="Edit"/> |

You can assign the NCOS level to all subscribers within a particular domain. To do so, navigate to *Settings*→*Domains*, 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 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 the sip:provider CE 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: just 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.

7.1.3 IP Address Restriction

The sip:provider CE provides subscriber preference *allowed_ips* to restrict the IP addresses that subscriber 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, the sip:provider CE 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 *Settings*→*Subscribers*, search for the subscriber you want to edit, press *Details* and then *Preferences* and press *Edit* for the *allowed_ips* preference in the *Access Restrictions* section.

The screenshot shows a web interface for 'Call Blockings'. A red box highlights the 'Access Restrictions' tab. Below it, a table with a green header is shown. The table has three columns: 'Name', 'Value', and an empty column. The 'allowed_ips' row is highlighted in red, and a red box around the 'Edit' button in the third column of that row is labeled '3'. A red box around the 'allowed_ips' text in the first column is labeled '2'. A red box around the first cell of the table is labeled '1'.

| | Name | Value | |
|---|--------------------------------|--------------------------|--------|
| 1 | lock | | |
| | concurrent_max | | |
| | concurrent_max_out | | |
| | allowed_clis | | |
| | reject_emergency | <input type="checkbox"/> | |
| | concurrent_max_per_account | | |
| | concurrent_max_out_per_account | | |
| | allowed_ips 2 | | 3 Edit |
| | man_allowed_ips | | |
| | ignore_allowed_ips | <input type="checkbox"/> | |
| | allow_out_foreign_domain | <input type="checkbox"/> | |

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.

7.2 Call Forwarding and Call Hunting

The sip:provider CE 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.

Call Forwards and Call Hunting Groups can either be executed unconditionally or based on a *Time Set Definition*, so you can define deflections based on time period definitions (e.g. Monday to Friday 8am to 4pm etc).

7.2.1 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*).

Logged in as administrator Logout

sip:wise

Edit Call Forward Unconditional

Destination

- Voicemail
- Conference
- URI/Number

1

2 URI/Number 4312345

for (seconds) 300

3

Advanced View Save

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.

7.2.2 Advanced Call Hunting

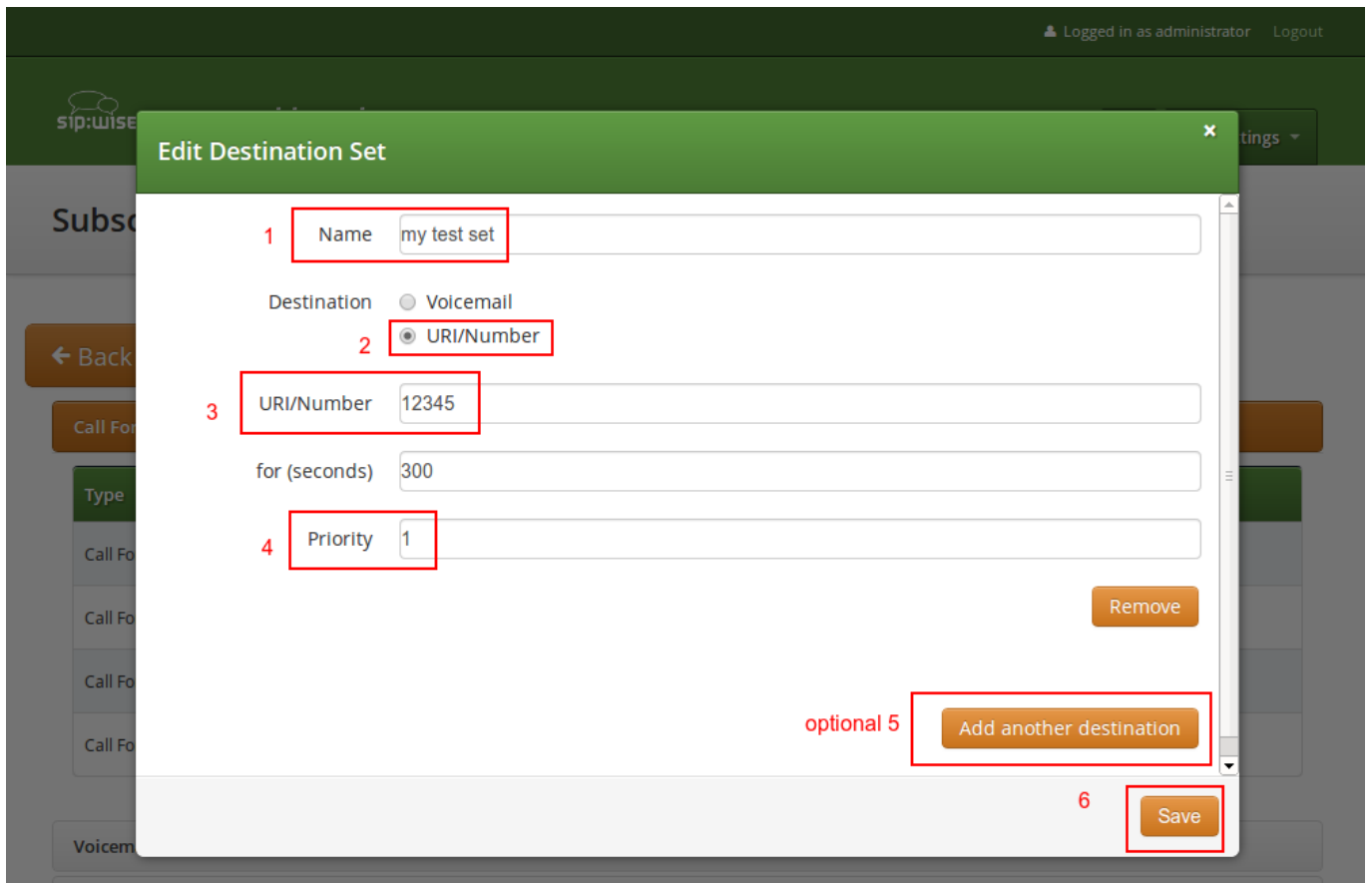
If you want multiple destinations to be executed one after the other, you need to change into the *Advanced View* when editing your call forward. There, you can select multiple *Destination Set/Time Set* pairs to be executed.

A *Destination Set* is a list of destinations to be executed one after another.

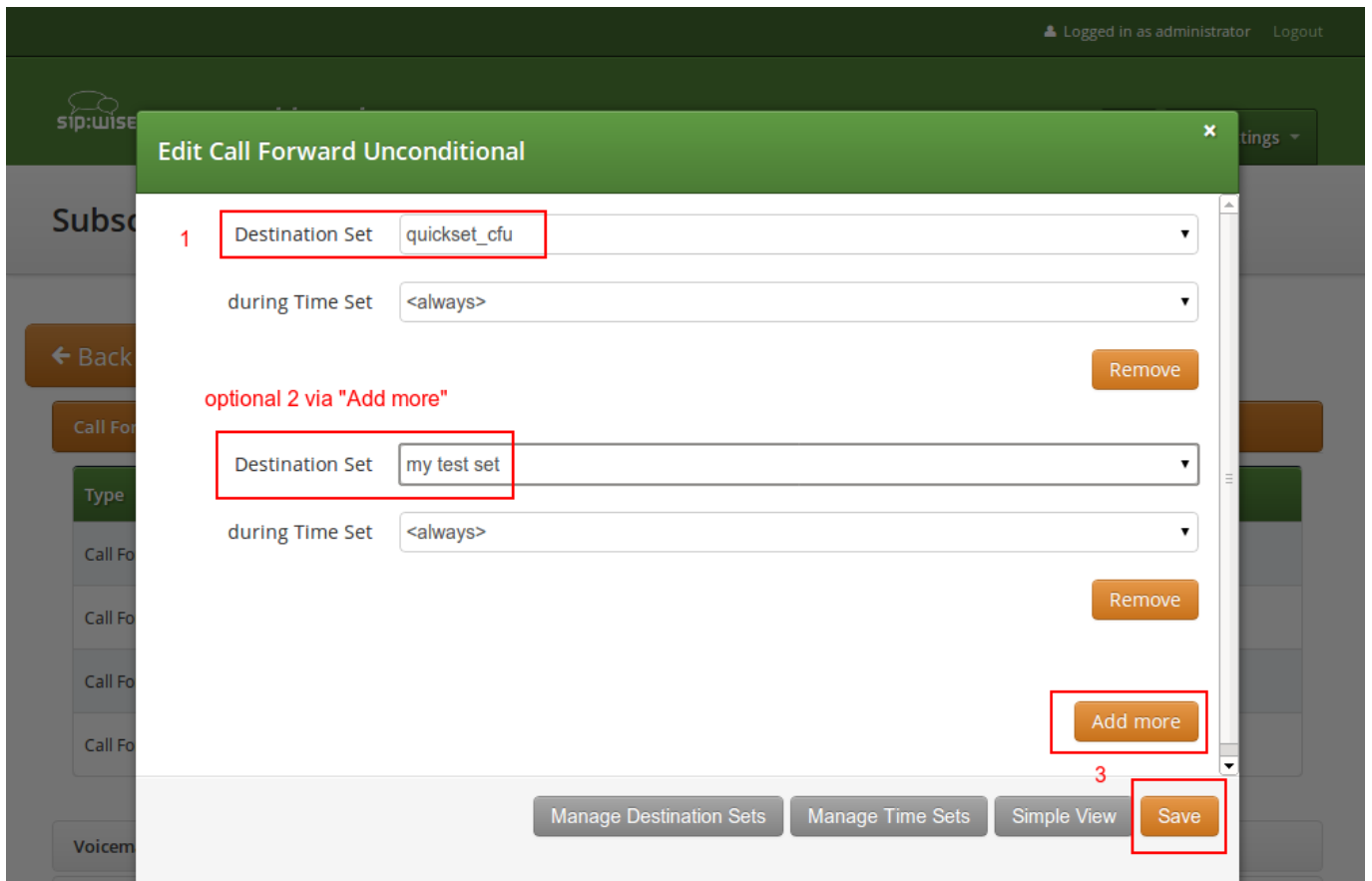
A *Time Set* is a time definition when to execute this *Destination Set*.

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.



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.

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 just 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.

7.3 Header Manipulation

7.3.1 Header Filtering

Adding additional SIP headers to the initial INVITEs relayed to the callee (second leg) is possible by modifying the following template file: `/etc/ngcp-config/templates/etc/ngcp-sems/etc/ngcp.sbcprofile.conf.customtt.tt2`. The following section can be changed:

```
header_filter=whitelist
header_list=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
```

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 just need to add them at the end of the `header_list=` list. After that, as usual, you need to apply 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.

7.3.2 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 just 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, as usual, you need to run `ngcpcfg apply Enable CODEC filtering`.

7.3.3 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)

7.4 SIP Trunking with SIPconnect

7.4.1 User provisioning

For the purpose of external SIP-PBX interconnect with sip:provider CE 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 *Settings*→*Subscribers* 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.

7.4.2 Inbound calls routing

Enable preference *Number Manipulations*→*e164_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).

7.4.3 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 Section 6.6. 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 the sip:provider CE 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 [?]) 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`
- 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 of course 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.

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): NGCP 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.

7.4.4 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 as described in Section 16 this poses additional maintenance overhead on the operator. The administrator should create a static registration for the SIP-PBX: go to Subscribers, *Details*→*Registered Devices*→*Create Permanent Registration* and put address of the SIP-PBX in the following format: sip:username@ipaddress 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 sip:provider CE will take care of routing the aliases to the AOR with *e164_to_ruri* preference.

Trusted sources

You can do IP-based authentication for subscribers using the Trusted Source mechanism in your subscriber's preferences (*Details*→*Preferences*→*Trusted Sources*) if the SIP-PBX can't authenticate.

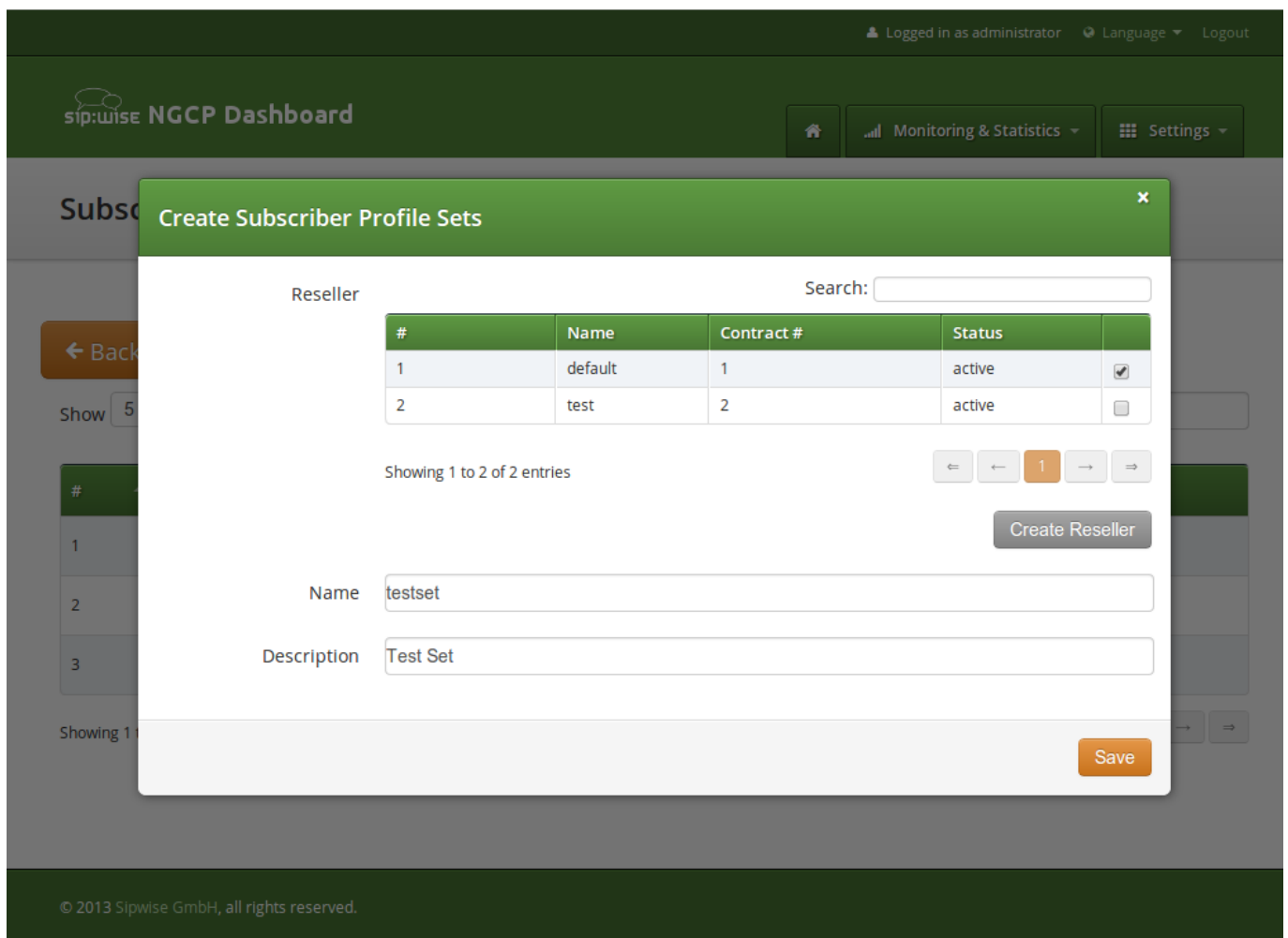
7.5 Limiting Subscriber Preferences via Subscriber Profiles

The preferences a subscriber can provision by himself via the CSC can be limited via profiles within profile sets assigned to subscribers.

7.5.1 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 *Settings*→*Subscriber Profiles*. To create a new Profile Set, click *Create Subscriber Profile Set*.



The screenshot shows the 'Create Subscriber Profile Sets' modal window in the sip:wise NGCP Dashboard. The modal is titled 'Create Subscriber Profile Sets' and has a search bar for 'Reseller'. Below the search bar is a table with the following data:

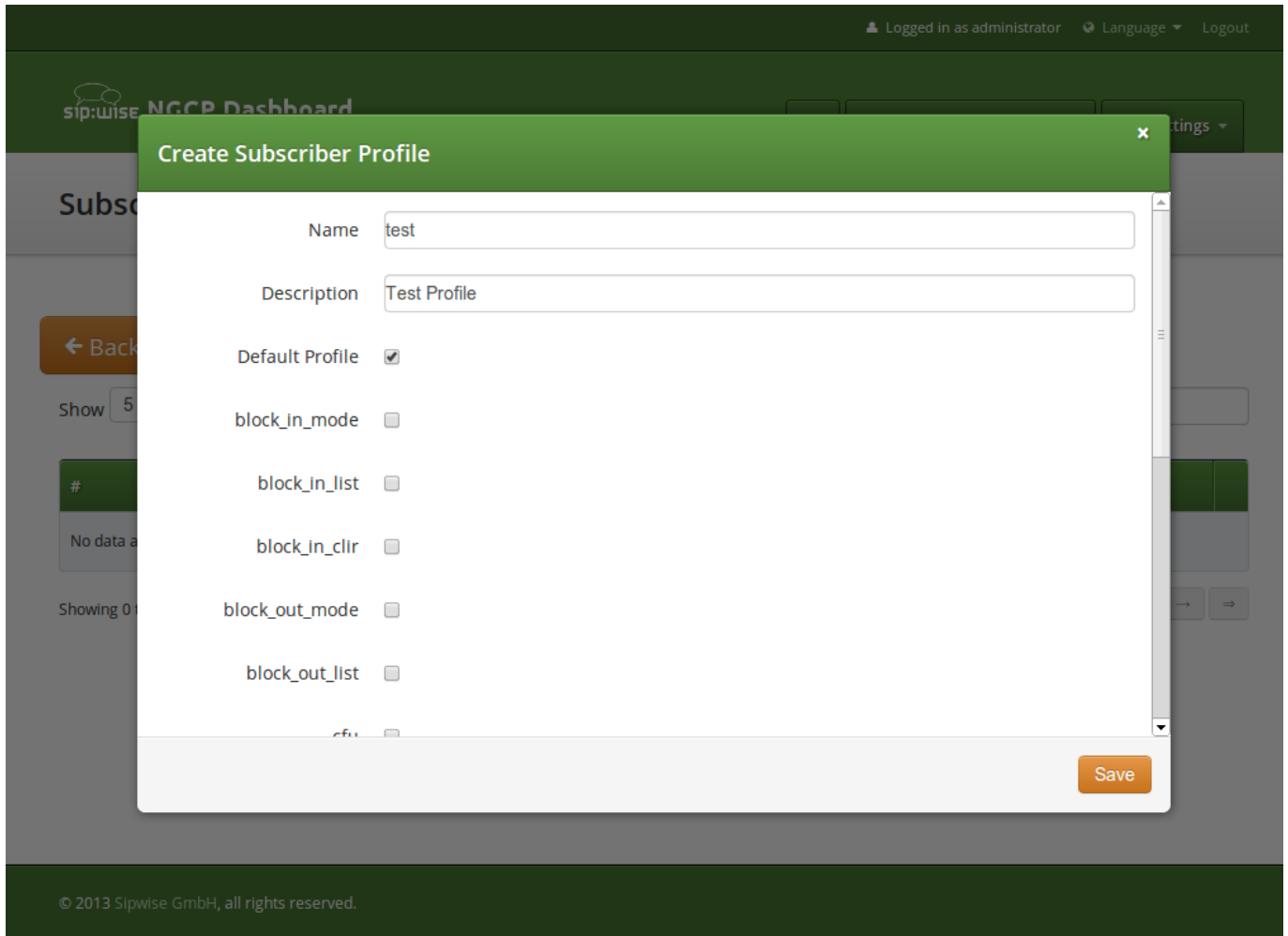
| # | Name | Contract # | Status | |
|---|---------|------------|--------|-------------------------------------|
| 1 | default | 1 | active | <input checked="" type="checkbox"/> |
| 2 | test | 2 | active | <input type="checkbox"/> |

Below the table, it says 'Showing 1 to 2 of 2 entries' and has navigation buttons. There is a 'Create Reseller' button. Below the table are form fields for 'Name' (containing 'testset') and 'Description' (containing 'Test Set'). At the bottom right of the modal is a 'Save' button.

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.



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.

7.6 Creating Trusted Subscribers

In some cases, when you have a device that cannot authenticate itself towards sip:provider CE, you may need to create Trusted Subscriber. Trusted Subscribers use IP-based authentication and they have a Permanent SIP Registration URI in order to receive messages from sip:provider CE. In order to create a Trusted Subscriber you just need to create a normal subscriber, then Create a Permanent Registration via (*Subscribers*→*Details*→*Registered Devices*→*Create Permanent Registration*) and also you need to add the devices IP as Trusted Source in your subscriber's preferences (*Details*→*Preferences*→*Trusted Sources*). In this way, all messages coming from your device IP will be trusted (and authenticate just via the source IP), on the other side all the SIP messages to your devices will be sent to the SIP URI specified in the Permanent Registration.

7.7 Voicemail System

7.7.1 Accessing the IVR Menu

For a subscriber to manage his voicebox via IVR, there are two 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 subscriber, causing the system to prompt for a mailbox and a PIN.

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 don't support calling alphanumeric numbers to dial `voicebox`, you can map any arbitrary number to the voicebox URIs using rewrite rules.

To do so, you can provision a match pattern like `^(00|\+)12345$` with a replace pattern `voicebox` or `voiceboxpass` to map a number to either password-less or password-based IVR access.

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.

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.

7.7.2 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
 - * 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

7.7.3 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."

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.

7.7.4 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

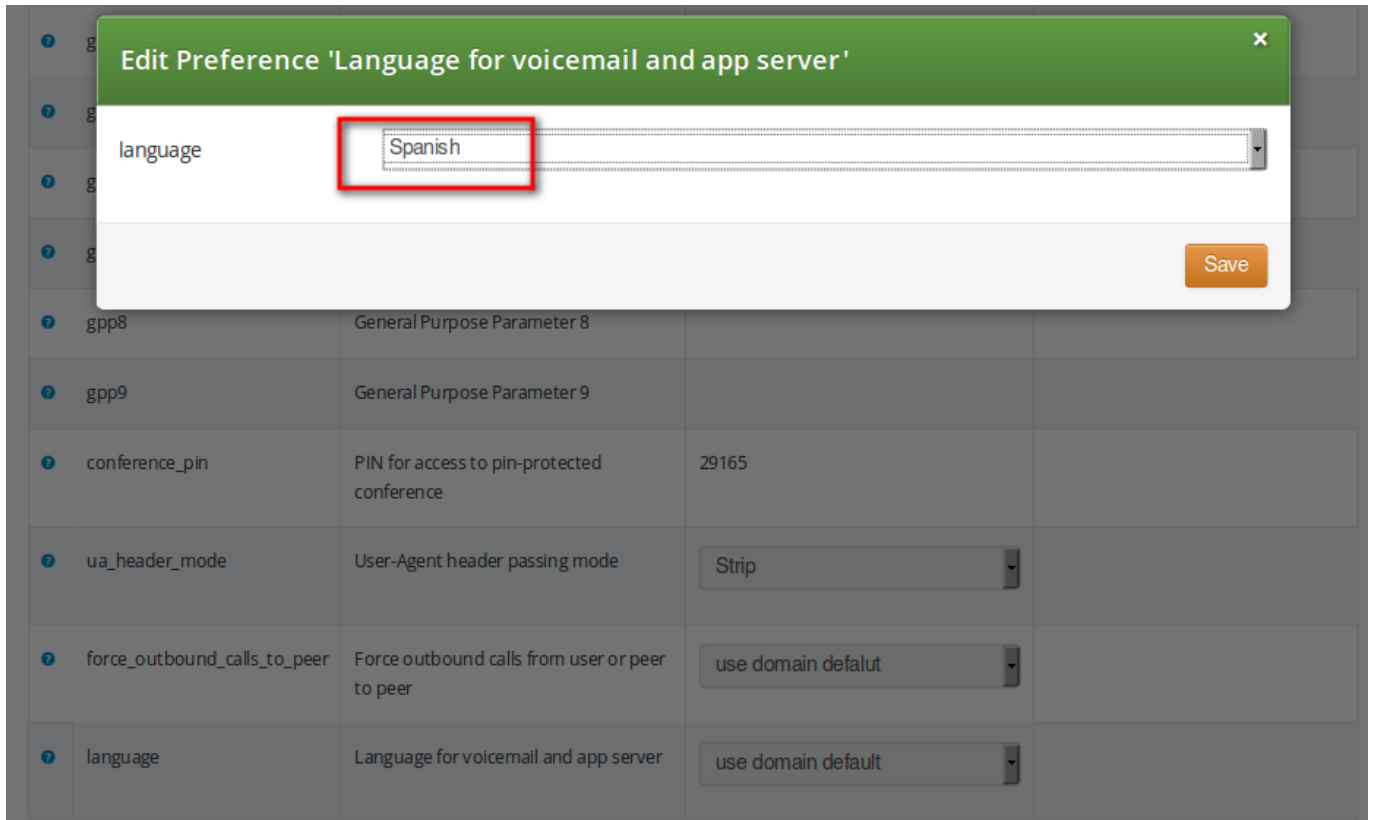
When a caller leaves a message for you, the system will put the message into the "New Messages" folder. If you listen to the message, but do not delete the message or save the message to a different folder, it will automatically move the message to the "Old Messages" folder. When you first log into your 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.

7.8 XMPP Instant Messaging

Instant Messaging (IM) based on XMPP comes with sip:provider CE out of the box. sip:provider CE 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`.

7.9 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*.



The sip:provider CE 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 NGCP Panel and uploaded by the administrator in his language of choice.

7.10 Sound Sets

The sip:provider CE 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* 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). Sound Sets can be defined in *Settings*→*Sound Sets*. To create a new Sound Set, click *Create Sound Set*. Then click the *Files* button.

Logged in as administrator Language Logout

sip:wise NGCP Dashboard Monitoring & Statistics Settings

Sound Sets

[← Back](#) [★ Create Sound Set](#)

Sound set successfully created

Show entries Search:

| # | Reseller | Customer | Name | Description | |
|---|----------|----------|---------------------|-----------------------------------|---|
| 1 | default | | Conference | | Edit Delete Files |
| 2 | default | | Early media rejects | Failed call attempt announcements | |

Showing 1 to 2 of 2 entries

Note

You may use 8 or 16 bit mono WAV audio files for all of the voice prompts.

7.10.1 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 just once. Click Save to upload the file.

| early_rejects | | | |
|--------------------------|----------|------|---|
| Name | Filename | Loop | |
| block_in | | ■ |  |
| block_out | | ■ | |
| block_ncos | | ■ | |
| block_override_pin_wrong | | ■ | |
| locked_in | | ■ | |
| locked_out | | ■ | |
| max_calls_in | | ■ | |
| max_calls_out | | ■ | |
| max_calls_peer | | ■ | |
| unauth_caller_ip | | ■ | |

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 the sip:provider CE will fallback to sending the error response code back to the user.

Table 2: Early Reject Sound Sets

| 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 (<i>adm_block_in_list</i> , <i>block_in_list</i> 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 (<i>adm_block_out_list</i> , <i>block_out_list</i> subscriber preferences) | Your call to the number you are trying to reach is blocked. |
| 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 <i>ncos</i> and <i>adm_ncos</i> preferences) | Your call to the number you are trying to reach is not permitted. |

Table 2: (continued)

| Handle | Description | Message played |
|--|--|--|
| 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 <i>block_out_override_pin</i> and <i>adm_block_out_override_pin</i> preferences) | The PIN code you have entered is not correct. |
| 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 <i>concurrent_max</i> limit by sum of incoming and outgoing calls or whose customer has exceeded the <i>concurrent_max_per_account</i> limit by sum of incoming and outgoing 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 <i>concurrent_max</i> (total limit) or <i>concurrent_max_out</i> (limit on number of outbound calls) or whose customer has exceeded the <i>concurrent_max_per_account</i> or <i>concurrent_max_out_per_account</i> limit |
| All outgoing lines are currently in use. Please try again later. | 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 <i>concurrent_max</i> preference of peering server) |
| The network you are trying to reach is currently busy. Please try again later. | unauth_caller_ip | This is what the calling party hears when it tries to make a call from unauthorized IP address or network (<i>allowed_ips</i> , <i>man_allowed_ips</i> preferences) |

Table 2: (continued)

| Handle | Description | Message played |
|---|------------------------|---|
| You are not allowed to place calls from your current network location. | 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. | 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. | 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. | 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_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_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. | 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. | 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. | voicebox_unavailable | Announcement played on call to voicebox if the voicemail server is not configured (system operation is impaired) |
| The voicemail of the number you are trying to reach is currently not available. Please try again later. | emergency_unsupported | Announcement played when emergency destination is dialed but the emergency calls are administratively prohibited for this user or domain (<i>reject_emergency</i> preference is enabled) |
| You are not allowed to place emergency calls from this line. Please use a different phone. | no_credit | Announcement played when prepaid account has insufficient balance to make a call to this destination |

7.11 Conference System

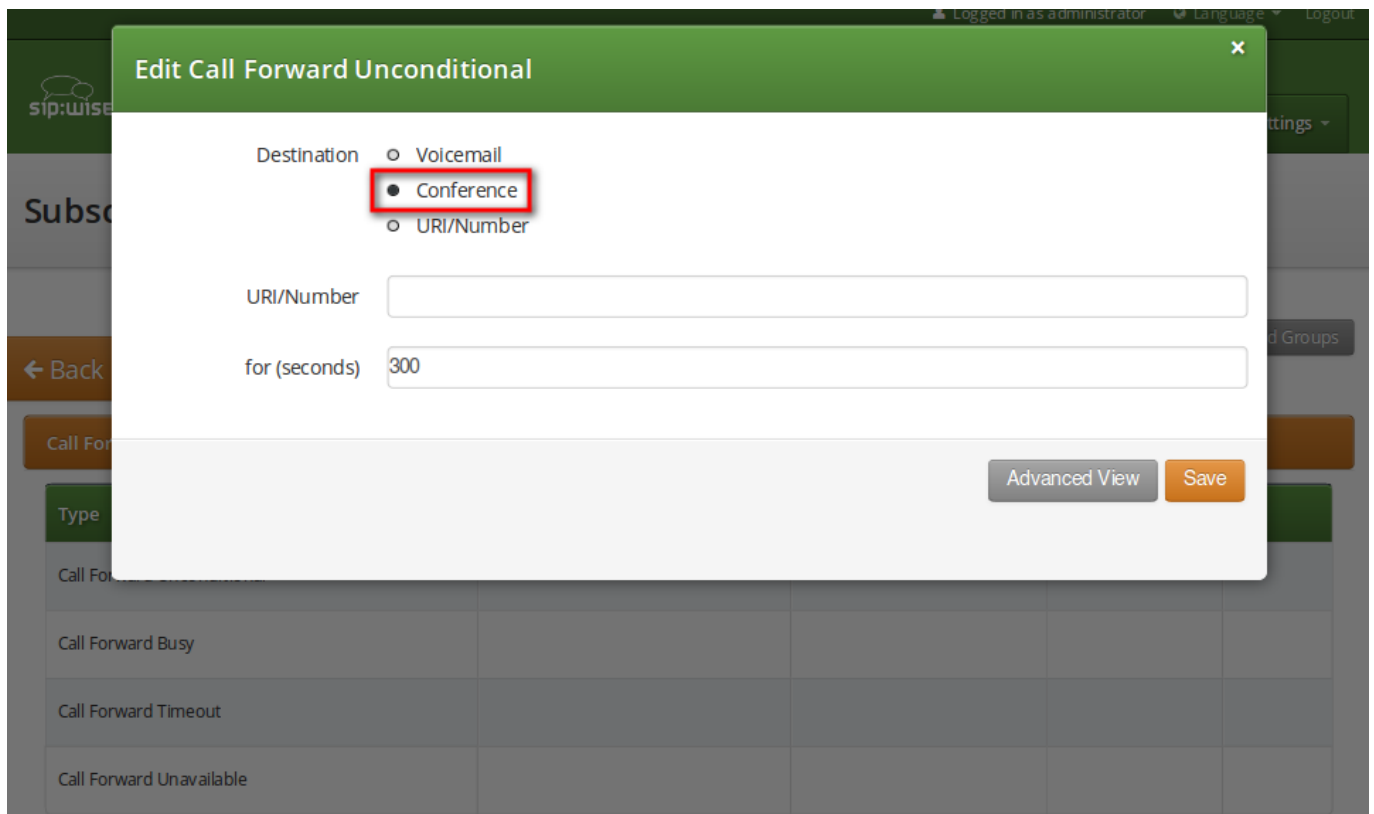
The sip:provider CE provides the simple pin-protected conferencing service built using the SEMS DSM scripting language. Hence it is open for all kinds of modifications and extensions.

Template files for the sems conference scripts stored in `/etc/ngcp-config/templates/etc/ngcp-sems/`:

- IVR script: `/etc/ngcp-config/templates/etc/ngcp-sems/dsm/confpin.dsm.tt2`
- Config: `/etc/ngcp-config/templates/etc/ngcp-sems/dsm/confpin.conf.tt2`

7.11.1 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*).

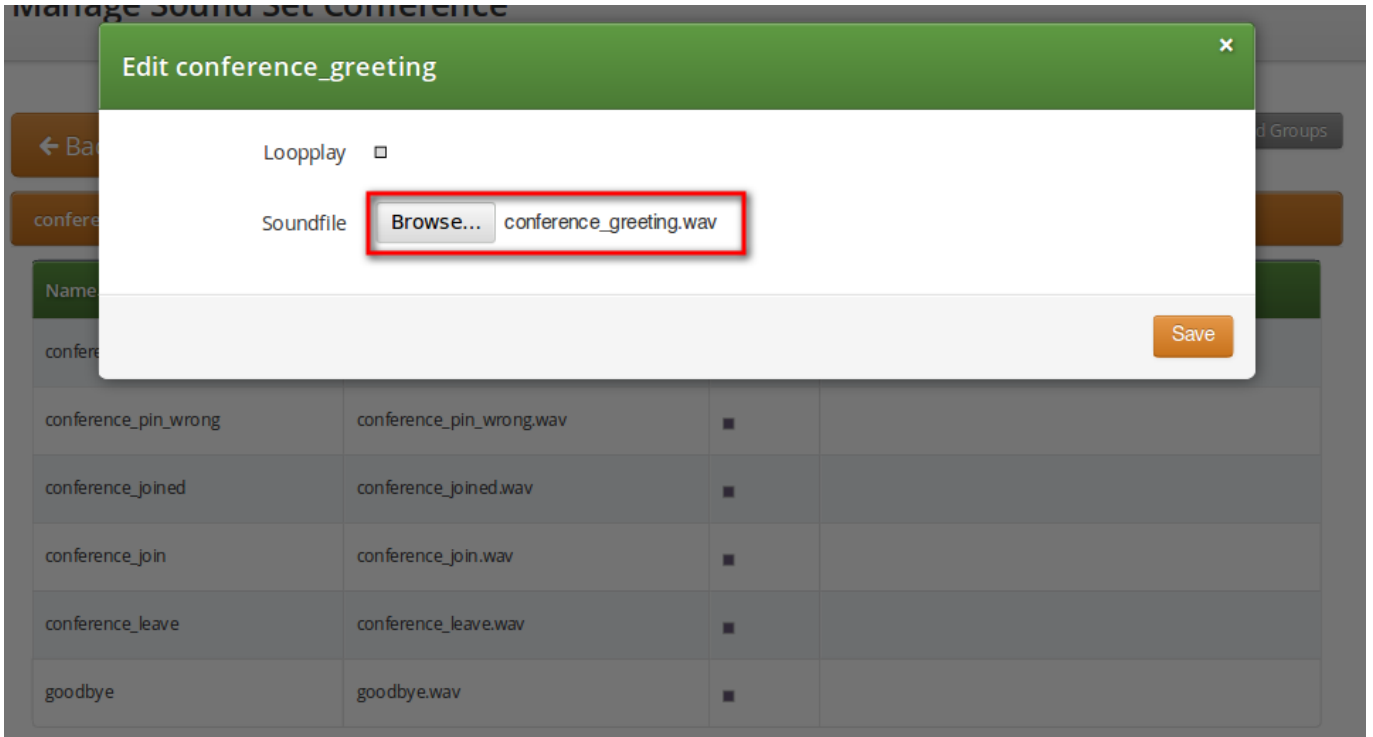


The screenshot shows a modal dialog titled "Edit Call Forward Unconditional". The "Destination" section has three radio button options: "Voicemail", "Conference" (which is selected and highlighted with a red box), and "URI/Number". Below this, there is an empty text input field for "URI/Number" and another text input field for "for (seconds)" containing the value "300". At the bottom right of the dialog are two buttons: "Advanced View" and "Save".

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.

7.11.2 Configuring Conference Sound Sets

Sound Sets can be defined in *Settings*→*Sound Sets*. To create a new Sound Set, click *Create Sound Set*. Then click the *Files* button.



Upload the following files:

Table 3: Conference Sound Sets

| Handle | Message played |
|----------------------|--|
| conference_greeting | Welcome to the conferencing service. 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_join | A person has joined the conference. |
| conference_leave | A person has left the conference. |
| goodbye | Goodbye. |

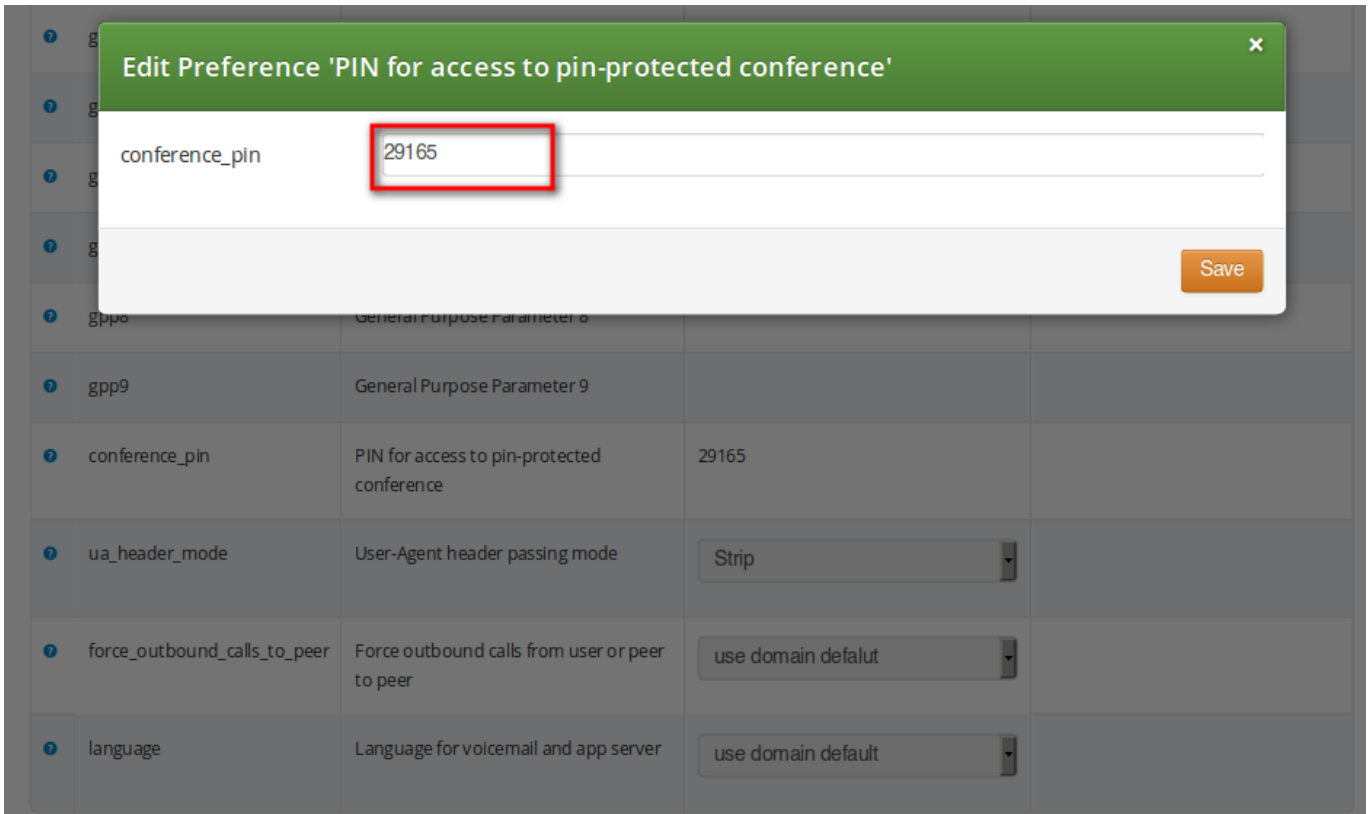
Note

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).

7.11.3 Entering the Conference with a PIN

It is mandatory to configure the PIN code for entrance to the conference on the same subscriber which has the Call Forwarding active. Responsible for this is the `conference_pin` preference in the Internals section of subscriber preferences.



When calling the conference IVR you are requested to enter this PIN. Upon the successful entry of the PIN the caller hears the announcement that he is going to be placed into a conference and at the same time this is announced to all participants in the conference.

7.12 Malicious Call Identification (MCID)

MCID feature allows customers to report unwanted calls to the platform operator.

7.12.1 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 subscriber UUID in the redis DB (the amount of stored recent calls will not exceed the amount of provisioned subscribers).

Next step is to 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 a fileset *malicious_call_identification* → for that purpose.

Once the *Sound Set* is created the Subscriber's Preferences *Malicious Call Identification* must be enabled under *Subscriber* →

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 *Caller Inbound* rewrite rule `^(*123)$ → malicious_call`

Finally you run `ngcpconfig apply Enabling MCID` to recreate the templates and automatically restart depended services.

7.12.2 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.

To check reported malicious calls as the platform operator open *Settings* → *Malicious 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.

7.12.3 Advanced configuration

By default the expiration time for the most recent call per subscriber is 3600 seconds (1 hour). If you wish to prolong or shorten the expiration time open `constants.yml` and set there `recentcalls:expire:3600` to a new value, and issue `ngcpconfig apply Enabling MCID` afterwards.

7.13 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. Sip Provider 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 just a normal subscriber which has just a different configuration in his Preferences. You need to change the following preferences under *Subscribers* → *Details* → *Preferences* → *NAT 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 Sip Provider to translate between Plain RTP and RTP/SAVPF when you have calls between normal SIP clients and WebRTC clients.

7.14 SIP loop detection

In order to detect a SIP loop (incoming call as a response for a call request) sip:provider CE 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_d` loops within `kamailio.proxy.loop_detection.expire` seconds. Higher occurrence of loops will be reported with a SIP 482 "Loop Detected" error message

8 Customer Self-Care Interfaces

There are two ways for end users to maintain their subscriber settings: via the *Customer Self-Care Web Interface* and via *Vertical Service Codes* using their SIP phones.

8.1 The Customer Self-Care Web Interface

The NGCP provides a web panel for end users (CSC panel) to maintain their subscriber accounts, which is running on `https://<ce-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.

8.1.1 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 Section 6.2. 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 just 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.

8.1.2 Site Customization

As an operator (as well as a Reseller), you can change the branding logo of the CSC panel by modifying the CSS via web interface. For changing the branding log you just need to access the web interface as administrator and move to *Reseller_menu*. Once there click on *Details* button for "default" reseller. Then on *Branding* → *Edit Branding*. Now you can upload your logo and copy/paste the CSS code line in the `CSS__` field. The logo will be visible into the Customer Self Care interface.

Also Reseller can customize their web page (CSC and Admin interface) by uploading their logo and change the CSS. To do that, just access the Admin interface with the Reseller web credentials and then access the *Panel Branding* menu. From them you can upload the logo as explained before. The logo will appear in the CSC web page related to that reseller as well as to the Admin page of the reseller.

You can also enable/disable specific languages a user can choose from in the CSC panel. Currently, English (`en`), German (`de`), Spanish (`es`) and Russian (`ru`) are supported and English is activated by default. You can change the default language provided by CSC by changing the parameter *force_language* in `config.yml`.

8.2 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 Section 6.6.

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.
- **50** - set *Speed Dial Slot*, e.g. set slot 1 to 431000 by dialing $*50*101000$, which then can be used by dialing *1.
- **55** - set *One-Shot Reminder Call* e.g. to 08:30 by dialing $*55*0830$.
- **31** - set *Calling Line Identification Restriction* for one call, e.g. to call 431000 anonymously dial $*31*01000$.
- **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.

8.2.1 Vertical Service Codes for PBX customers

Subscribers under the same PBX customer can enjoy some PBX-specific features by means of special VSCs.

NGCP 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.

8.2.2 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 `sems→vsc`. 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 the NGCP via SIP like normal telephone calls.

8.3 The Voicemail Interface

NGCP 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: simply 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 `sems→vsc`. 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:

- 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 Section 6.6.) 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.

9 Billing Configuration

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

9.1 Billing Data Import

Service billing on the NGCP is based on billing profiles, which may be assigned to VoIP accounts 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.)

9.1.1 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 *Settings*→*Billing* and click on *Create Billing Profile*.

Logged in as administrator Logout

Create Billing Profiles

Reseller Search:

| # | Name | Contract # | Status | |
|---|---------|------------|--------|---------------------------------------|
| 1 | default | 1 | active | 1 <input checked="" type="checkbox"/> |

Showing 1 to 1 of 1 entries

← 1 →

Create Reseller

Handle 2

Name 3

Prepaid

Interval charge

4

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 VoIP account 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.
- **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.

9.1.2 Creating Billing Fees

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

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_admin→fees_csv→element_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  
"^01.+$", "^02145.+$", out, "AT", "ZONE Test", 0.06250, 1, 0.06250, 1, 0.01755, 1, 0.01733, 1, 0
```

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

Zone

Search:

| # | Zone | Zone Detail | Action |
|---|------|-------------|-------------------------------------|
| 2 | test | test zone | <input checked="" type="checkbox"/> |

Showing 1 to 1 of 1 entries

1

Source

3 Destination

4 Direction

Onpeak init rate

created by "Create Zone" button below

In both cases, the following information may be specified independently for every destination:

- **Zone:** A zone for a group of destinations. May be used to group destinations for simplified display, e.g. on invoices. (e.g. foreign zone 1)
- **Source:** The source pattern. This is a POSIX regular expression matching the complete source URI (e.g. `^.*@sip\.example\.org$` or `^someone@sip\.sipwise\.com$` or just `.` to match everything). If you leave this field empty, the default pattern `.` matching everything will be set implicitly. Internally, this pattern will be matched against the `<source_cli>`@`<source_domain>` fields of the CDR.
- **Destination:** The destination pattern. This is a POSIX regular expression matching the complete destination URI (e.g. `someone@sip\.example\.org` or `^43`). This field must be set.
- **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 {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, both for the file upload and the input via the web interface.

Important

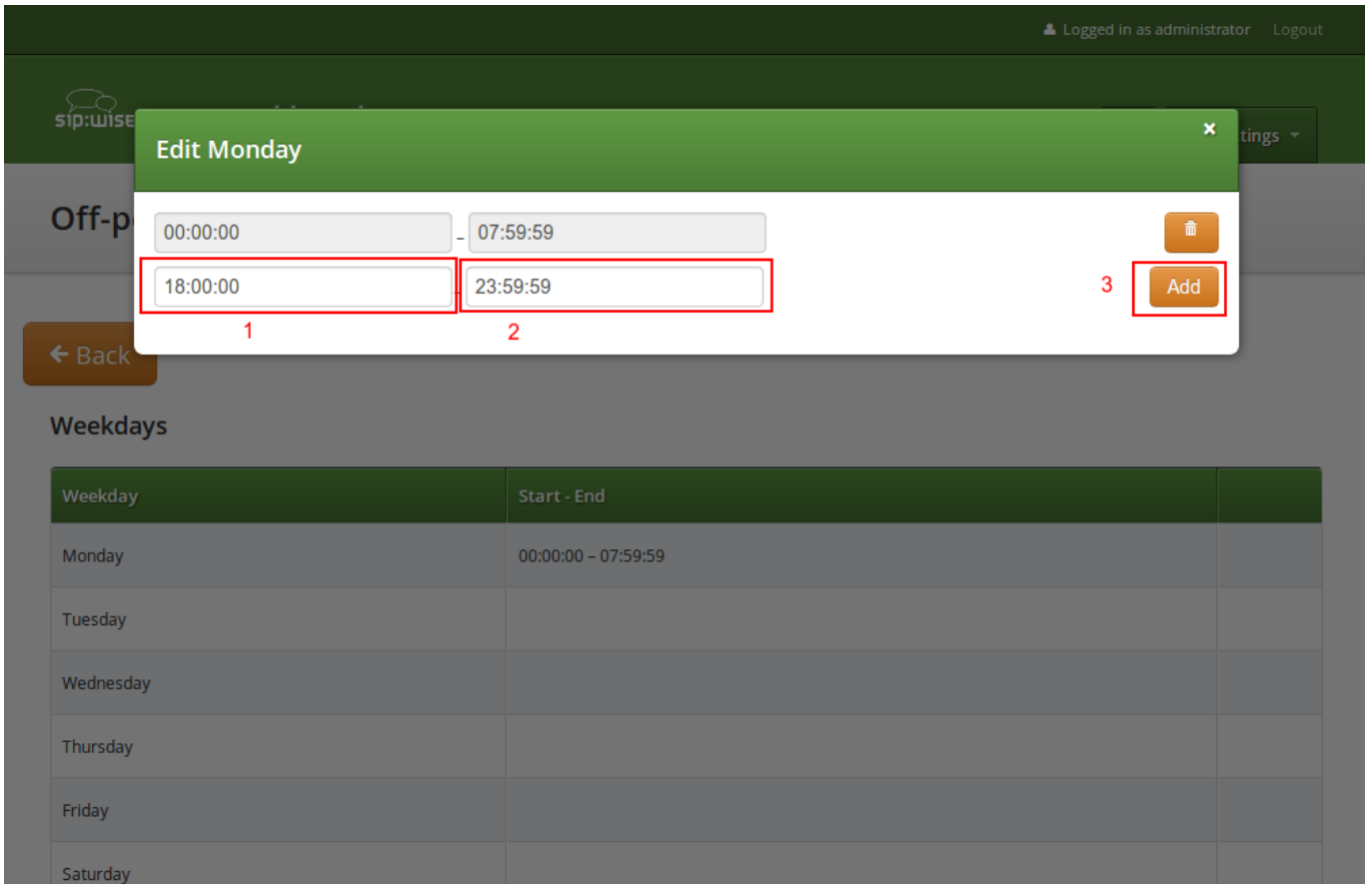
There are several internal services (vsc, conference, voicebox) which will need a specific destination entry with a domain-based destination. If you don't want to charge the same (or nothing) for those services, add a fee for destination `\.local$` there. If you want to charge different amounts for those services, break it down into separate fee entries for `@vsc\.local$`, `@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.
- **Use free time:** Specifies whether free time minutes may be used when calling this destination. May be specified in the file upload as 0, n[o], f[alse] and 1, y[es], t[rue] respectively.

9.1.3 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 *Settings*→*Billing* and press the *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, just 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.

1 Start Date/Time 2013-12-24 00:00:00

2 End Date/Time 2013-12-24 23:59:59

3 Save

| Weekday | Start - End |
|-----------|--|
| Monday | 00:00:00 - 07:59:59 18:00:00 - 23:59:59 |
| Tuesday | |
| Wednesday | |
| Thursday | |
| Friday | |

9.1.4 Fraud Detection and Locking

The NGCP 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 Section 9.1.1, 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.

Monthly fraud limit check runs once a day, shortly after midnight local time and daily fraud limit check runs every 30min. A background script (managed by cron daemon) automatically checks all accounts which are linked to a billing profile enabled for fraud detection, and selects those which have caused a higher cost than the *fraud monthly limit* configured in the billing profile, within the currently active billing interval (e.g. in the current month), or a higher cost than the *fraud daily limit* configured in the billing profile, within the calendar day. It then proceeds to perform at least one of the following actions on those accounts:

- If **fraud lock** is set to anything other than *none*, it will lock the account accordingly (e.g. if **fraud lock** is set to *outgoing*, the account will be locked for all outgoing calls).
- If anything is listed in **fraud notify**, an email will be sent to the email addresses configured. 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 for the customer.

**Important**

You can override these settings 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 when the next billing interval starts. This has to be done manually through the administration panel or through the provisioning interface.

**Important**

If fraud detection is configured to only send an email and not lock the affected accounts, it will continue to do so for over-limit accounts every day. The accounts must either be locked in order to stop the emails (only currently active accounts are considered when the script looks for over-limit accounts) or some other action to resolve the conflict must be taken, such as disabling fraud detection for those accounts.

9.2 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.

9.2.1 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 4: 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
```

9.2.2 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 5: 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 maximum of 5000 lines. Each line holds one call detail record in CSV format and is constructed by the following fields, all of them enclosed in single quotes:

Table 6: CDR export file body line format

| Body Element | Length | Type | Description |
|--------------|--------|------|------------------|
| <id> | 1-10 | uint | Internal CDR id. |

Table 6: (continued)

| Body Element | Length | Type | Description |
|----------------------------|--------|-----------|---|
| <update_time> | 19 | timestamp | Timestamp of last modification. |
| <source_user_id> | 36 | string | Internal UUID of calling party subscriber. |
| <source_provider_id> | 1-255 | string | Internal ID of calling party provider. |
| <source_ext_subscriber_id> | 0-255 | string | External ID of calling party subscriber. |
| <source_subscriber_id> | 1-10 | uint | Internal ID of calling party subscriber. |
| <source_ext_account_id> | 0-255 | string | External ID of calling party VoIP account. |
| <source_account_id> | 1-10 | uint | Internal ID of calling party VoIP account. |
| <source_user> | 1-255 | string | SIP username of calling party. |
| <source_domain> | 1-255 | string | SIP domain of calling party. |
| <source_cli> | 1-64 | 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> | 1 / 36 | string | Internal UUID of called party subscriber or 0 if callee is not local. |
| <destination_provider_id> | 1-255 | string | Internal ID of called party provider. |
| <dest_ext_subscriber_id> | 0-255 | string | External ID of called party subscriber. |
| <dest_subscriber_id> | 1-10 | uint | Internal ID of called party subscriber. |
| <dest_ext_account_id> | 0-255 | string | External ID of called party VoIP account. |
| <destination_account_id> | 1-10 | uint | Internal ID of called party VoIP account. |
| <destination_user> | 1-255 | string | Final SIP username of called party. |
| <destination_domain> | 1-255 | string | Final SIP domain of called party. |
| <destination_user_in> | 1-255 | string | Incoming SIP username of called party. |
| <destination_domain_in> | 1-255 | string | Incoming SIP domain of called party. |
| <dialed_digits> | 1-255 | string | The user-part of the SIP Request URI as received by the soft-switch. |
| <peer_auth_user> | 0-255 | string | User to authenticate towards peer. |
| <peer_auth_realm> | 0-255 | string | Realm to authenticate towards peer. |
| <call_type> | 3-4 | string | The type of the call - one of: call: normal call cfu: call forward unconditional cft: call forward timeout cfb: call forward busy cfna: call forward no answer |

Table 6: (continued)

| Body Element | Length | Type | Description |
|-------------------------------|--------|-----------------|--|
| <call_status> | 2-7 | string | The final call status - one of: ok: successful call busy: callee busy noanswer: no answer from callee cancel: cancel from caller offline callee offline timeout: no reply from callee other: unspecified, see <call_code> for details |
| <call_code> | 3 | uint | The final SIP status code. |
| <init_time> | 23 | timestamp | Timestamp of call initiation (invite received from caller). Seconds include fractional part (3 decimals). |
| <start_time> | 23 | timestamp | Timestamp of call establishment (final response received from callee). Seconds include fractional part (3 decimals). |
| <duration> | 4-11 | fixed precision | Length of call (beginning at start_time) in seconds with 3 decimals. |
| <call_id> | 1-255 | string | The SIP call-id. |
| <rating_status> | 2-7 | string | The internal rating status - 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 | timestamp | Timestamp of rating or empty if not rated. |
| <source_carrier_cost> | 4-11 | fixed precision | The originating carrier cost or empty if not rated. In cent with two decimals. Only available in system exports, not for resellers. |
| <source_customer_cost> | 4-11 | fixed precision | The originating customer cost or empty if not rated. In cent with two decimals. |
| <source_carrier_zone> | 0-127 | string | The originating carrier billing zone or empty if not rated. Only available in system exports, not for resellers. |
| <source_customer_zone> | 0-127 | string | The originating customer billing zone or empty if not rated. |
| <source_carrier_destination> | 0-127 | string | The originating carrier billing destination or empty if not rated. Only available in system exports, not for resellers. |
| <source_customer_destination> | 0-127 | string | The originating customer billing destination or empty if not rated. |

Table 6: (continued)

| Body Element | Length | Type | Description |
|------------------------------------|--------|-----------------|--|
| <source_carrier_free_time> | 1-10 | uint | The number of originating free time seconds used on carrier side or empty if not rated. Only available in system exports, not for resellers. |
| <source_customer_free_time> | 1-10 | uint | The number of originating free time seconds used from the customer's account balance or empty if not rated. |
| <destination_carrier_cost> | 4-11 | fixed precision | The termination carrier cost or empty if not rated. In cent with two decimals. Only available in system exports, not for resellers. |
| <destination_customer_cost> | 4-11 | fixed precision | The termination customer cost or empty if not rated. In cent with two decimals. |
| <destination_carrier_zone> | 0-127 | string | The termination carrier billing zone or empty if not rated. Only available in system exports, not for resellers. |
| <destination_customer_zone> | 0-127 | string | The termination customer billing zone or empty if not rated. |
| <destination_carrier_destination> | 0-127 | string | The termination carrier billing destination or empty if not rated. Only available in system exports, not for resellers. |
| <destination_customer_destination> | 0-127 | string | The termination customer billing destination or empty if not rated. |
| <destination_carrier_free_time> | 1-10 | uint | The number of termination free time seconds used on carrier side or empty if not rated. Only available in system exports, not for resellers. |
| <destination_customer_free_time> | 1-10 | uint | The number of termination free time seconds used from the customer's account balance or empty if not rated. |
| <source_reseller_cost> | 4-11 | fixed precision | The originating reseller cost or empty if not rated. In cent with two decimals. Only available in system exports, not for resellers. |
| <source_reseller_zone> | 0-127 | string | The originating reseller billing zone or empty if not rated. Only available in system exports, not for resellers. |
| <source_reseller_destination> | 0-127 | string | The originating reseller billing destination or empty if not rated. Only available in system exports, not for resellers. |
| <source_reseller_free_time> | 1-10 | uint | The number of originating free time seconds used from the reseller's account balance or empty if not rated. Only available in system exports, not for resellers. |
| <destination_reseller_cost> | 4-11 | fixed precision | The termination reseller cost or empty if not rated. In cent with two decimals. Only available in system exports, not for resellers. |

Table 6: (continued)

| Body Element | Length | Type | Description |
|------------------------------------|--------|--------|--|
| <destination_reseller_zone> | 0-127 | string | The termination reseller billing zone or empty if not rated. Only available in system exports, not for resellers. |
| <destination_reseller_destination> | 0-127 | string | The termination reseller billing destination or empty if not rated. Only available in system exports, not for resellers. |
| <destination_reseller_free_time> | 1-10 | uint | The number of termination free time seconds used from the reseller's account balance or empty if not rated. 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 a rated CDR is (line breaks added for clarity):

```
'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. Reseller CDR CSV files don't contain the fields for *carrier* or *reseller* ratings, neither in *source* nor *destination* direction. Thus, the reseller CSV files have 16 fewer fields.

File Body Format for Event Detail Records (EDR)

The body of a EDR consists of a minimum of zero and a maximum of 5000 lines. Each line holds one call detail record in CSV format and is constructed by the following fields, all of them enclosed in single quotes:

Table 7: EDR export file body line format

| Body Element | Length | Type | Description |
|--------------|--------|------|------------------|
| <event_id> | 1-10 | uint | Internal EDR id. |

Table 7: (continued)

| Body Element | Length | Type | Description |
|--------------------------|--------|--------|---|
| <event_type> | 1-255 | 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 group. end_huntgroup: A subscriber has been deprovisioned as group. start_ivr: A subscriber has a new call-forward to auto-attendant set. end_ivr: A subscriber has removed a call-forward to auto-attendant. |
| <customer_external_id> | 0-255 | string | The external customer ID as provisioned for the subscriber. |
| <contact_company> | 0-255 | string | The company name of the customer's contact. |
| <subscriber_external_id> | 0-255 | string | The external subscriber ID as provisioned for the subscriber. |
| <subscriber_number> | 0-255 | string | The voip number of the subscriber with the highest ID (DID or primary number). |
| <old_status> | 0-255 | string | The old status of the event. Depending on the event_type: start_profile: Empty. end_profile: The profile id of the profile which got removed from the subscriber. update_profile: The old profile id which got updated. start_huntgroup: Empty. end_huntgroup: The profile id of the group which got deprovisioned. start_ivr: Empty. end_ivr: Empty. |

Table 7: (continued)

| Body Element | Length | Type | Description |
|-------------------|--------|--------|--|
| <new_status> | 0-255 | string | The new status of the event. Depending on the event_type: start_profile: The profile id which got assigned to the subscriber. end_profile: Empty. update_profile: The new profile id which got updated. start_huntgroup: The current profile id assigned to the group subscriber. end_huntgroup: The current profile id assigned to the group subscriber. start_ivr: Empty. end_ivr: Empty. |
| <timestamp> | 0-255 | string | The time when the event occurred. |
| <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):

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

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. An example bash script to validate the integrity of the file is given below:

```
#!/bin/sh

error() { echo $@; exit 1; }
test -n "$1" || error "Usage: $0 <cdr-file>"
test -f "$1" || error "File '$1' not found"
```

```
TMPFILE="/tmp/${basename "$1"}.${$.}"
```



```
MD5="$(sed -rn '$ s/^[a-z0-9]{32}).*/\1/i p' "$1") $TMPFILE"
sed '$d' "$1" > "$TMPFILE"
echo "$MD5" | md5sum -c -
rm -f "$TMPFILE"
```

Given the script is located in `cdr-md5.sh` and the CDR-file is `sipwise_001_20071110123000_0000000004.cdr`, the output of the integrity check for an intact CDR file would be:

```
$ ./cdr-md5.sh sipwise_001_20071110123000_0000000004.cdr
/tmp/sipwise_001_20071110123000_0000000004.cdr: OK
```

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

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

9.2.3 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 easy 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_keys2` below the home directory of the `cdrexport` user. 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 commandline utilities.

10 Invoices and invoice templates

IMPORTANT: Invoice generation is deprecated since mr4.0+. Current invoice generation will damage billing records.

The sip:provider CE allows to generate and send customer invoices for each billing period based on Calls Detailed Records (CDR). Generated invoices can be sent to customers emails using [invoice generation script](#) Section 10.3.

Invoices present billing information from the reseller point of view. Recipients of the invoices are customers. Invoices include information related to the calls made by subscribers associated with the customer.

By default invoice contains information about billing plan fixed fee, calls zones fees and calls detailed information.

Content and vision of the invoices are customizable by [invoice templates](#) Section 10.2.

Note

The sip:provider CE generates invoices in pdf format.

10.1 Invoices management

Invoices can be requested for generation, searched, downloaded and deleted in the invoices interface.

The screenshot displays the 'Invoices' management interface in the sipwise NGCP Dashboard. At the top, there is a navigation bar with 'Logged in as administrator', 'Language', and 'Logout'. Below this, the 'Invoices' section is titled. A 'Create Invoice' button is highlighted in orange. The main area shows a table of invoices with the following data:

| # | Customer # | Customer Email | Serial | |
|---|------------|--------------------------|------------------|---|
| 1 | 9 | ipeshinskaya@sipwise.com | INV2014070000001 | Download Delete |
| 2 | 9 | ipeshinskaya@sipwise.com | INV2014080000002 | |
| 4 | 143 | ipeshinskaya@sipwise.com | INV2014080000004 | |

Below the table, it indicates 'Showing 1 to 3 of 3 entries' and includes pagination controls.

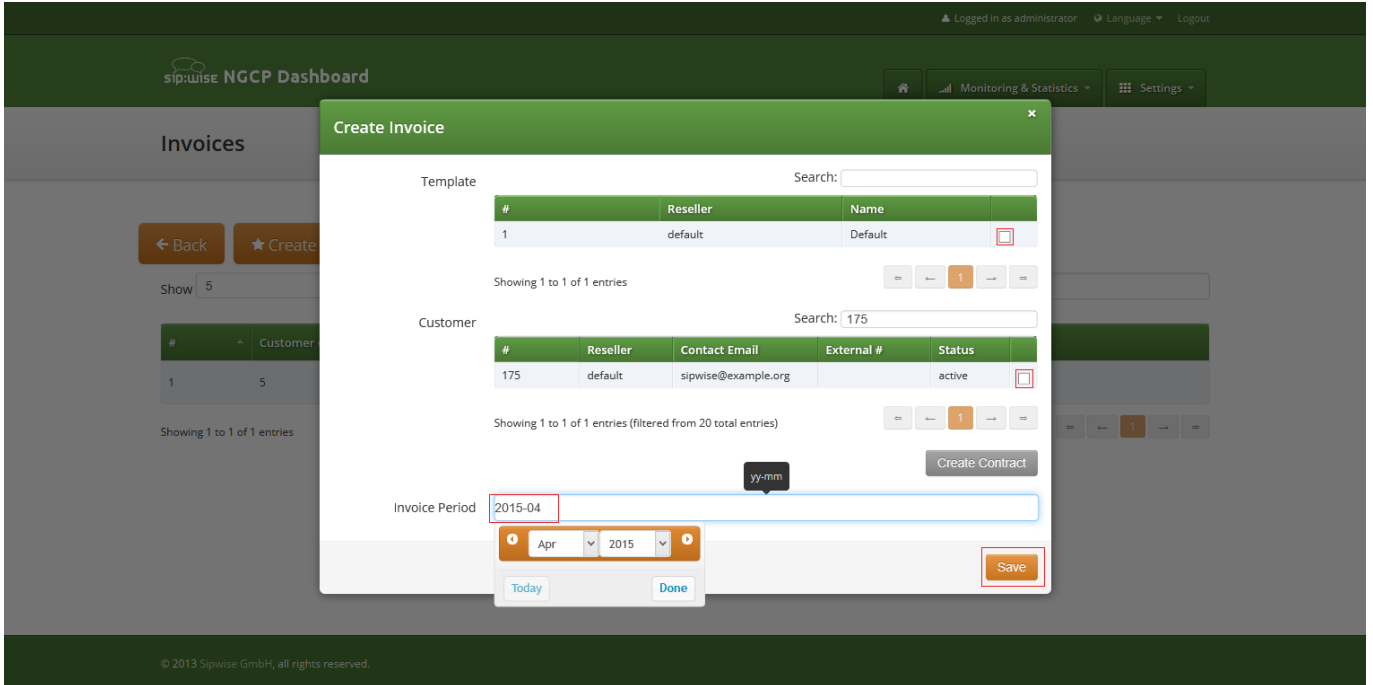
© 2013 Sipwise GmbH, all rights reserved.

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:

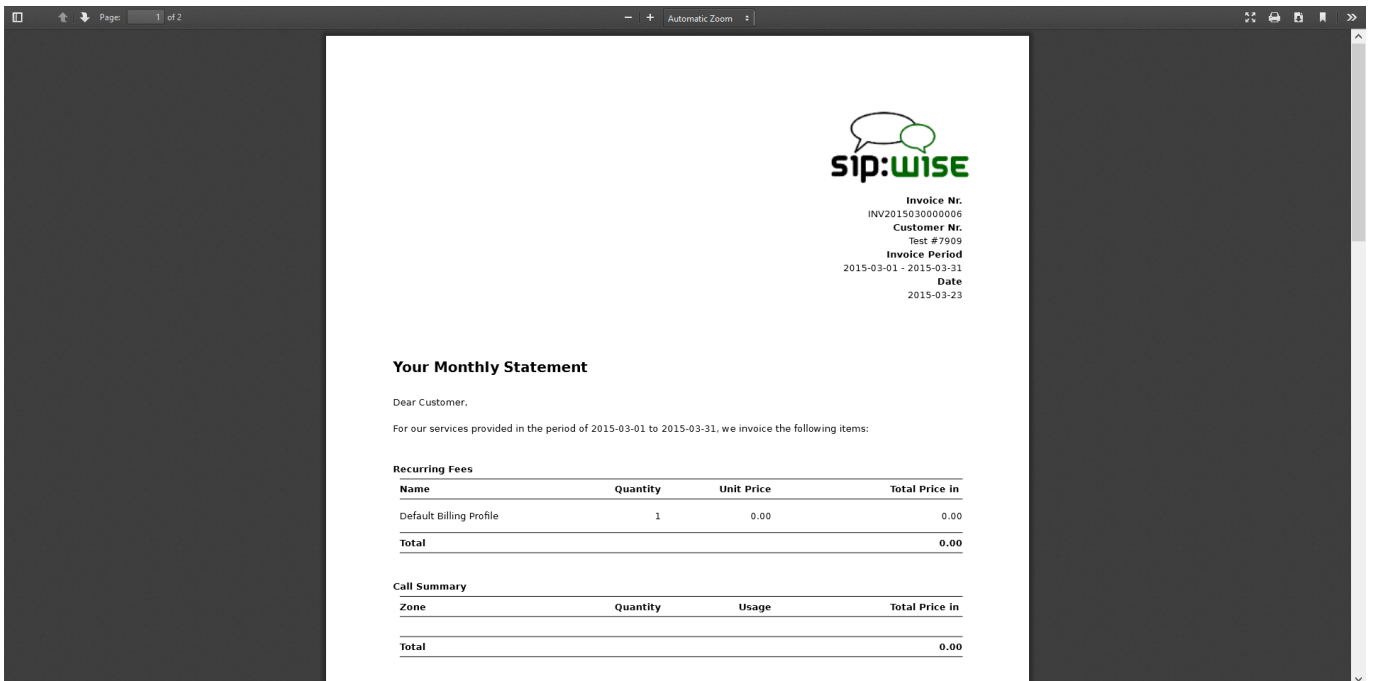
- **Template:** any of existent invoice template can be selected for the invoice generation.
- **Customer:** owner of the billing account, recipient of the invoice.

- **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.



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.

10.2 Invoice templates

Invoice template defines structure and look of the generated invoices. The sip:provider CE 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.

10.2.1 Invoice Templates management

Invoice templates can be searched, created, edited and deleted in the invoice templates management interface.

Logged in as administrator | Language | Logout

sip:wise NGCP Dashboard | Monitoring & Statistics | Settings

Invoice Templates

← Back | ★ Create Invoice Template

Search:

| # | Reseller | Name | Type | |
|---|----------|---------|------|---|
| 1 | default | Default | svg | Edit Meta Edit Content Delete |

Showing 1 to 1 of 1 entries

© 2013 Sipwise GmbH, all rights reserved.

Invoice template creation is separated on two steps:

- Register new invoice template meta information.
- Edit content (template itself) of the invoice template.

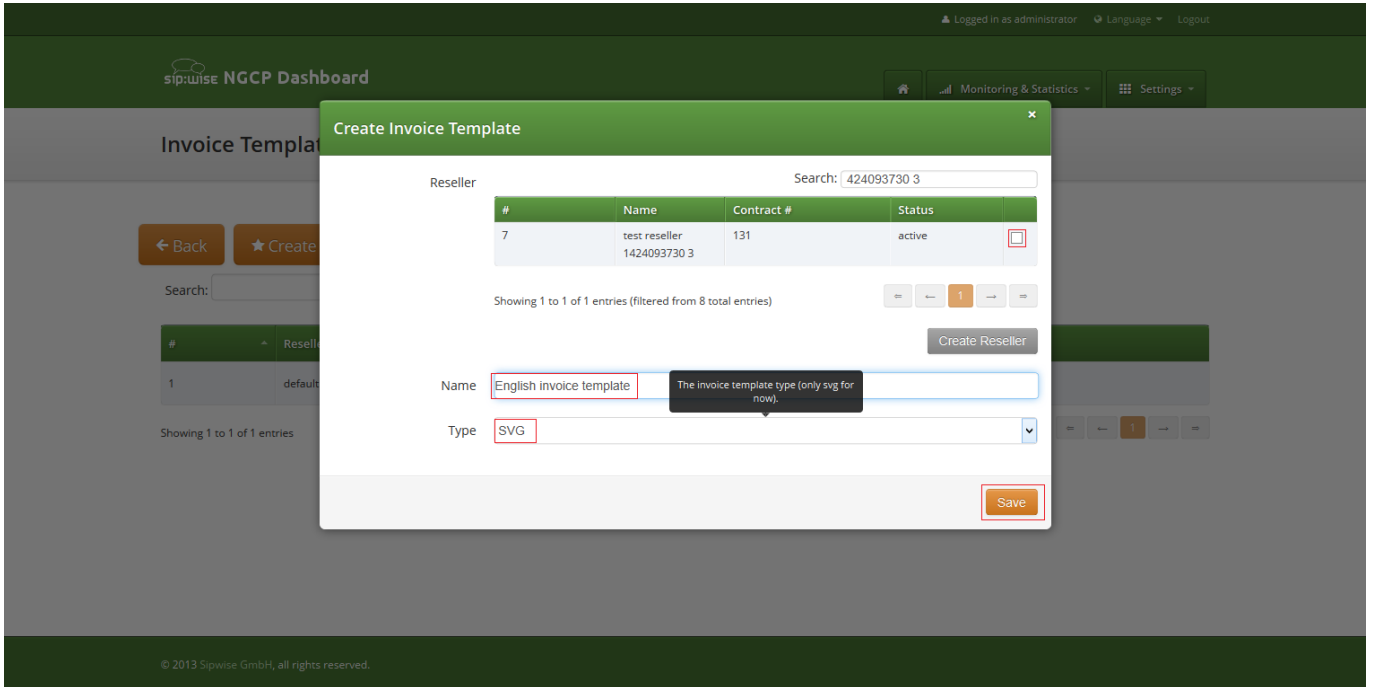
To register new invoice template press "Create Invoice Template" button.

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 sip:provider CE 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.

10.2.2 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 sip:provider CE provides embedded WYSIWYG SVG editor svg-edit 2.6 to customize default template. The sip:provider CE 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:

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](#) Section 10.2.3.

To return to the sip:provider CE 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:

Logged in as administrator | Language | Logout

sipwise NGCP Dashboard

Monitoring & Statistics | Settings

Invoice template TestInvoice

← Back

Save SVG | Preview PDF | Discard Changes | Show Variables

Invoice Nr. [% invoice.serial %]
Customer Nr. [% customer.external_id %]
Invoice Period [% p_start % - [% p_end %]
Date [% date_now(format="%Y-%m-%d") %]

Your Monthly Statement

Dear Customer,

For our services provided in the period of [% p_start %] to [% p_end %], we invoice the following items:

Recurring Fees

| Name | Quantity | Unit Price | Total Price in [% cur %] |
|---------------------|----------|--------------|--------------------------|
| [% billprof.name %] | 1 | [% fixfee %] | [% fixfee %] |
| Total | | | [% fixfee %] |

Call Summary

| Zone | Quantity | Usage | Total Price in [% cur %] |
|--------------|----------|-------|--------------------------|
| | | | |
| Total | | | [% zonefee %] |

Summary

| | in [% cur %] |
|-----------------------------|---------------------|
| Total Summary | [% netfee %] |
| VAT [% customer.vat_rate %] | [% vatfee %] |
| Amount Due | [% allfee %] |

The amount is automatically charged via SEPA within 30 days using Mandate ID MID12345 and Creditor ID CID12345 from your account with IBAN [% rescontact.iban %] and BIC [% rescontact.bic %].

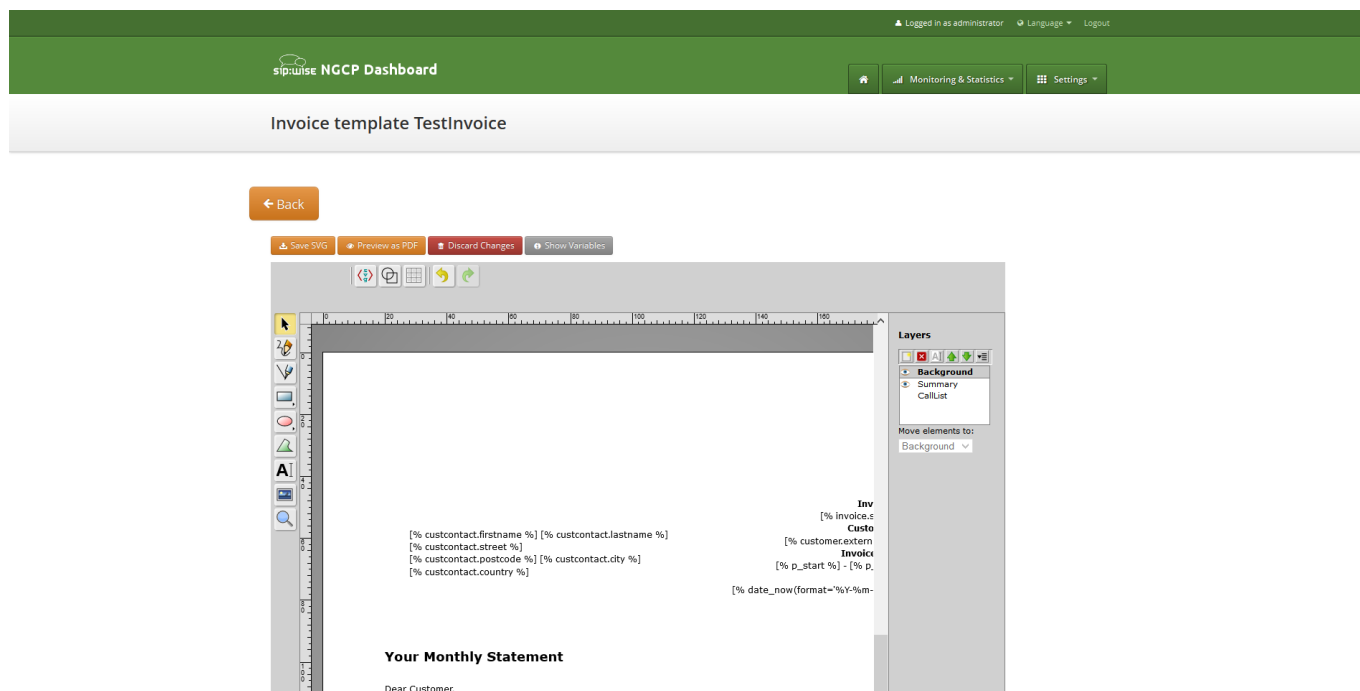
With best regards,
Your [% rescontact.company %] Service Team

[% rescontact.company %] Company Reg.Nr.: [% rescontact.com.regnum %]
[% rescontact.street %] VAT.Nr.: [% rescontact.vatnum %]
[% rescontact.postcode %] [% custcontact.city %] IBAN: [% rescontact.iban %]
[% rescontact.country %] Page [% aux.page %] BIC: [% rescontact.bic %]

Layers

© 2013 Sipwise GmbH, all rights reserved.

Side panel with layers list will be shown.



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:

The screenshot shows the 'sip:wise NGCP Dashboard' interface. At the top, there's a green header with the dashboard name and navigation links for 'Monitoring & Statistics' and 'Settings'. Below the header, the page title is 'Invoice template Rechnung_v1'. A 'Back' button is visible on the left. The main editor area has a toolbar with 'Save SVG', 'Preview as PDF', 'Discard Changes', and 'Show Variables'. The central canvas displays a preview of an invoice template with the following text:

```

[% custcontact.firstname %] [% custcontact.lastname %]
[% custcontact.street %]
[% custcontact.postcode %] [% custcontact.city %]
[% custcontact.country %]

Invoice Nr. [% invoice.serial %]
Customer Nr. [% customer.external_id %]
Invoice Period [% p_start %] - [% p_end %]
Date [% date_now(format="%Y-%m-%d") %]

Your Monthly Statement

Dear Customer,

```

SVG XML source of the invoice template will be shown.

SVG source can be edited in place or just copy-pasted as usual text.

Note

Template keeps sizes and distances in pixels.



Important

When edit svg xml source, please change very carefully and thoughtfully things inside special comment mark-up "`<!--{ }-->`". 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:

Invoice template Default

← Back

2 Save SVG Preview as PDF Discard Changes Show Variables

1 OK Cancel

```

<!--{
  [%
    pagewidth = 210;
    pageheight = 297;
    server_process_units = 'none';
    money_signs = 3;
    PROCESS "invoice/default/invoice_template_aux.tt";
    money_format(amount=(billprof.interval_charge * 100), comma='.'); fixfee = aux.val;
    money_format(amount=(zones.totalcost), comma='.'); zonefee = aux.val;
    money_format(amount=(invoice.amount_net * 100), comma='.'); netfee = aux.val;
    money_format(amount=(invoice.amount_vat * 100), comma='.'); vatfee = aux.val;
    money_format(amount=(invoice.amount_total * 100), comma='.'); allfee = aux.val;
    aux = billprof.currency;
    p_start = date_format(thedate=invoice.period_start, format='%Y-%m-%d');
    p_end = date_format(thedate=invoice.period_end, format='%Y-%m-%d');
  -%]
-->
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" width="210mm" height="297mm" viewBox="0 0 595 842" server-process-units="none">
<!--[ [% MACRO draw_background BLOCK % ] ]-->
<g display="inline" font-size="8" font-family="Verdana" class="page">
<title>Background</title>

```

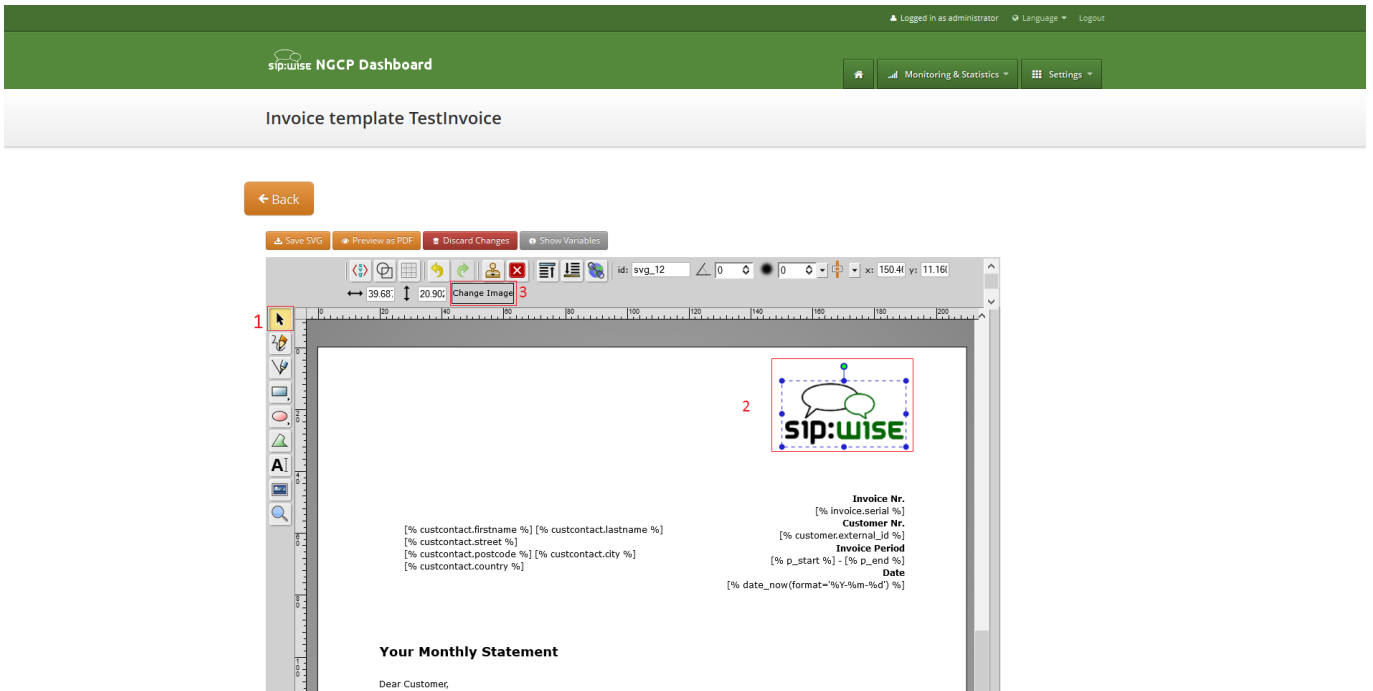
And then [save invoice template changes](#) Section 10.2.3.

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

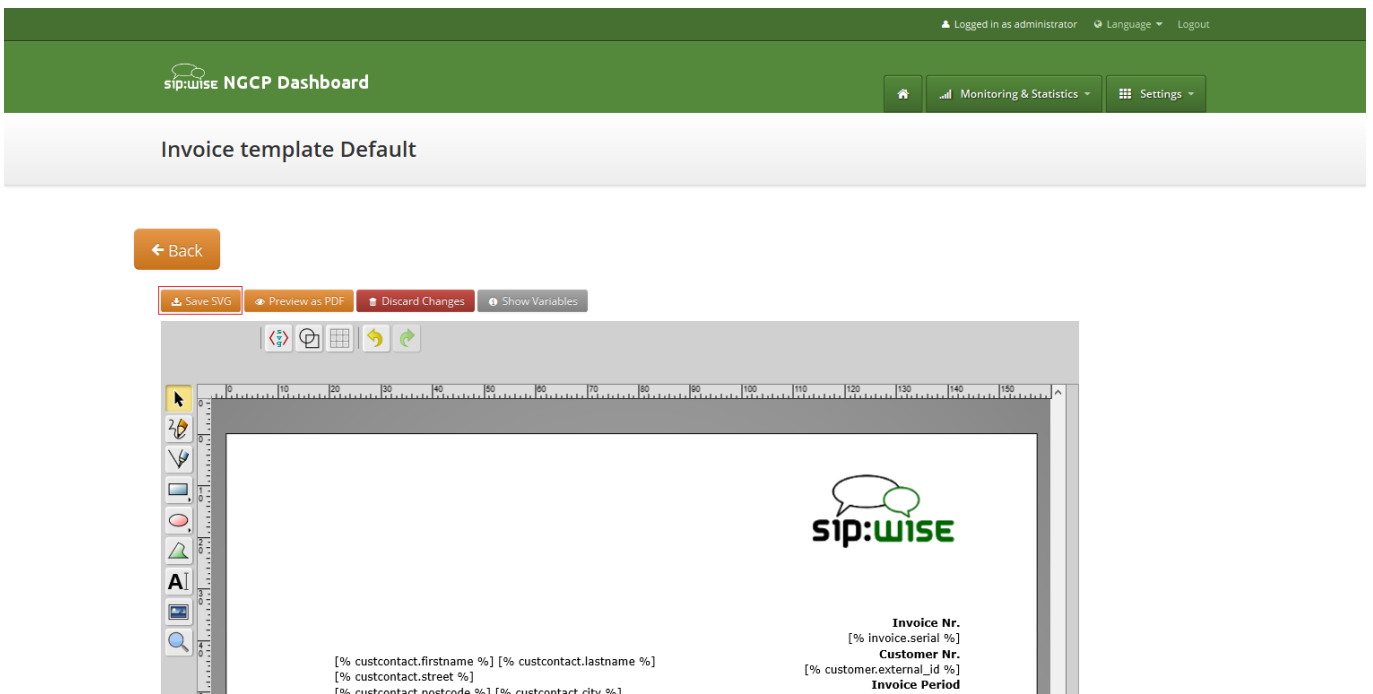
- Make sure that "Select tool" is active.
- Select default logo, clicking on the logo image.
- Press "Change image" button, which should appear on the top toolbar.



After image uploaded [save invoice template changes](#) Section 10.2.3.

10.2.3 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.

Logged in as administrator Language Logout

sip:wise NGCP Dashboard Monitoring & Statistics Settings

Invoice template Default

[← Back](#)

Invoice template successfully saved




Invoice preview will be opened in the new window.

Note

Example fake data will be used for preview generation.

The screenshot shows a browser window with the following content:



Invoice Nr.
1234567
Customer Nr.
Resext1234567890
Invoice Period
2015-04-01 - 2015-04-30
Date
2015-04-05

Customerfirst Customerlast
Customerstreet 12/3
12345 Customercity
Customercountry

Your Monthly Statement

Dear Customer,

For our services provided in the period of 2015-04-01 to 2015-04-30, we invoice the following items:

| Recurring Fees | | | |
|-----------------------|----------|------------|--------------------|
| Name | Quantity | Unit Price | Total Price in EUR |
| Test Billing Profile | 1 | 29.90 | 29.90 |
| Total | | | 29.90 |

10.3 Invoices generation

Except invoices generation on demand using web interface, invoices can be generated automatically for all customers using cron and invoice generator script.

Also invoice generation script is responsible for the sending generated invoices to the customers.

Script is located at: `/usr/share/ngcp-panel/tools/generate_invoices.pl`

In short:

- To generate and immediately send invoices for the previous month:

```
perl /usr/share/ngcp-panel/tools/generate_invoices.pl --send --prevmonth
```

- To generate invoices for the previous month without sending:

```
perl /usr/share/ngcp-panel/tools/generate_invoices.pl --prevmonth
```

- To send already generated invoices for the previous month:

```
perl /usr/share/ngcp-panel/tools/generate_invoices.pl --sendonly --prevmonth
```

- Regenerate invoices for the specified period:

```
perl /usr/share/ngcp-panel/tools/generate_invoices.pl --stime="2015-01-01 ↔  
00:00:00" --etime="2015-01-31 00:00:00" --regenerate
```

Some not obvious options:

- **--allow_terminated** Generates invoices for the terminated contracts too.
- **--force_unrated** Generate invoices despite unrated calls existence in the specified generation period.
- **--no_empty** Skip invoices for the contracts without calls in the specified period and with null permanent fee for the billing profile.

To see all possible script options use `--help` or `--man`:

```
/usr/share/ngcp-panel/tools/generate_invoices.pl --man
```

Script will be run periodically as configured by the cron files. Cron files templates can be found at:

- `/etc/ngcp-config/templates/etc/cron.d/ngcp-invoice-gen.tt2`
- `/etc/ngcp-config/templates/etc/cron.d/ngcp-invoice-gen.services`

After applying your configuration cron file will be located at:

- `/etc/cron.d/ngcp-invoice-gen`

Script uses configuration file located at: `/etc/ngcp-invoice-gen/invoice-gen.conf`

Except common DB connection configuration following specific options can be defined in the config file:

- **RESELLER_ID** *1,2,3,... N*

Comma separated resellers id. Invoice generation will be performed only for the specified resellers.

- **CLIENT_CONTRACT_ID** *1,2,3,... N*

Comma separated customers id. Invoice generation will be performed only for the specified customers.

- **STIME** *YYYY-mm-DD HH:MM:SS*

Usually is not necessary. Script option `--prevmonth` will define correct start and end time for the previous month billing period. Generated invoices will include all calls with call start time more then STIME value and less the ETIME value.

- **ETIME** *YYYY-mm-DD HH:MM:SS*

Usually is not necessary. Script option `--prevmonth` will define correct start and end time for the previous month billing period. Generated invoices will include all calls with call start time more then STIME value and less the ETIME value.

- **SEND** *[0/1]*

Generated invoices will be immediately sent to the customers.

- **RESEND** *[0/1]*

Invoices, already sent to the customers, will be sent again.

- **REGENERATE** *[0/1]*

Already presented invoices files will be generated again. Otherwise they will stay intouched.

- **ALLOW_TERMINATED** *[0/1]*

Generate invoices for the already terminated customers too.

- **ADMIN_EMAIL** *your@email.com*

Purposed for notifications about invoices generation fails. Not in use now.

All generated invoices can be seen in the [invoice management interface](#) Section 10.1.

On request each invoice will be sent to the proper customer as e-mail with the invoice PDF in the attachment. Letter content is defined by the invoice email template.

11 Email templates

11.1 Email events

The sip:provider CE 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.
- 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.

11.2 Initial template values and template variables

Default email templates for each of the email events are inserted on the initial sip:provider CE database creation. Content of the default template is described in the appropriate sections. Default email templates aren't linked to any reseller and can't be changed through sip:provider CE 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 the sip:provider CE Panel code.

11.3 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 | <pre> 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.</pre> |

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.

11.4 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 | <code>subscriber_default_email</code> |
| From | <code>default@sipwise.com</code> |
| Subject | Subscriber created |
| Body | <pre>Dear Customer, A new subscriber [%subscriber%] has been created for you. Your faithful Sipwise system -- This is an automatically generated message. Do not reply.</pre> |

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.

11.5 Invoice email template

| | |
|----------------------|---|
| Template name | <code>invoice_default_email</code> |
| From | <code>default@sipwise.com</code> |
| Subject | Invoice #[%invoice.serial%] from [%invoice.period_start_obj.ymd%] to [%invoice.period_end_obj.ymd%] |

| | |
|-------------|--|
| Body | <p>Dear Customer,</p> <p>Please find your invoice #[%invoice.serial%] for [%invoice. ← period_start_obj.month_name%], [%invoice.period_start_obj.year%] in attachment letter.</p> <p>Your faithful Sipwise system</p> <p>--</p> <p>This is an automatically generated message. Do not reply.</p> |
|-------------|--|

Variables passed to the email template:

- [%**invoice**%]: container variable for the invoice information.

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

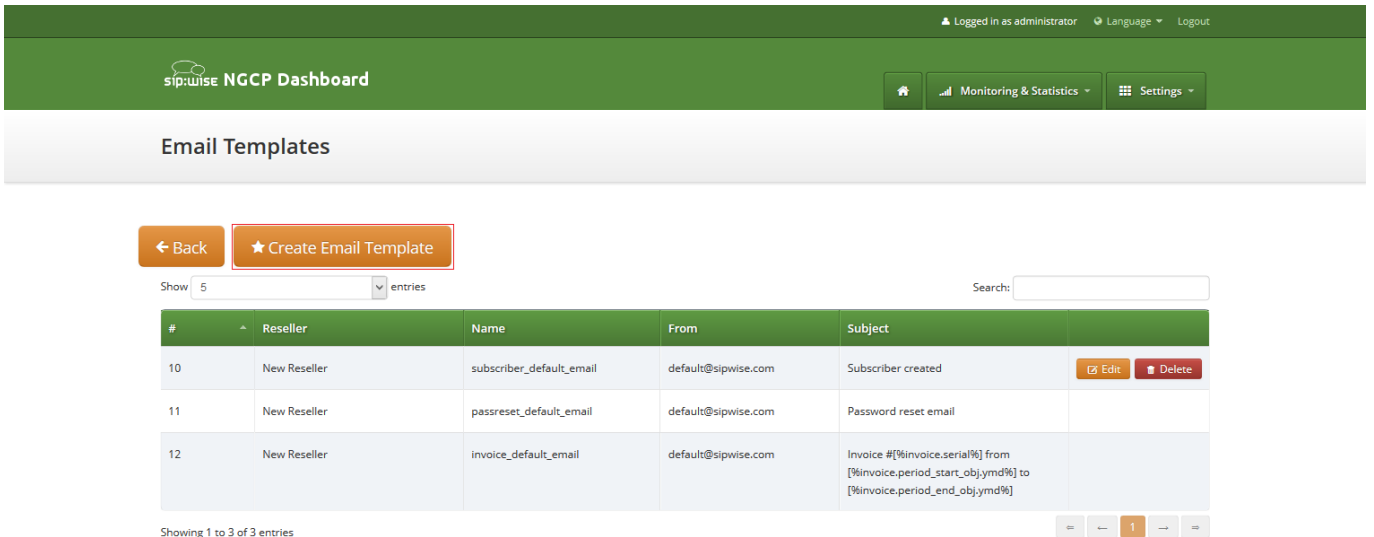
- [%**provider**%]: container variable for the reseller contact. All database contact values will be available.
- [%**client**%]: container variable for the customer contact.

Contact fields example for the "provider". Replace "provider" to client to access proper "customer" contact fields.

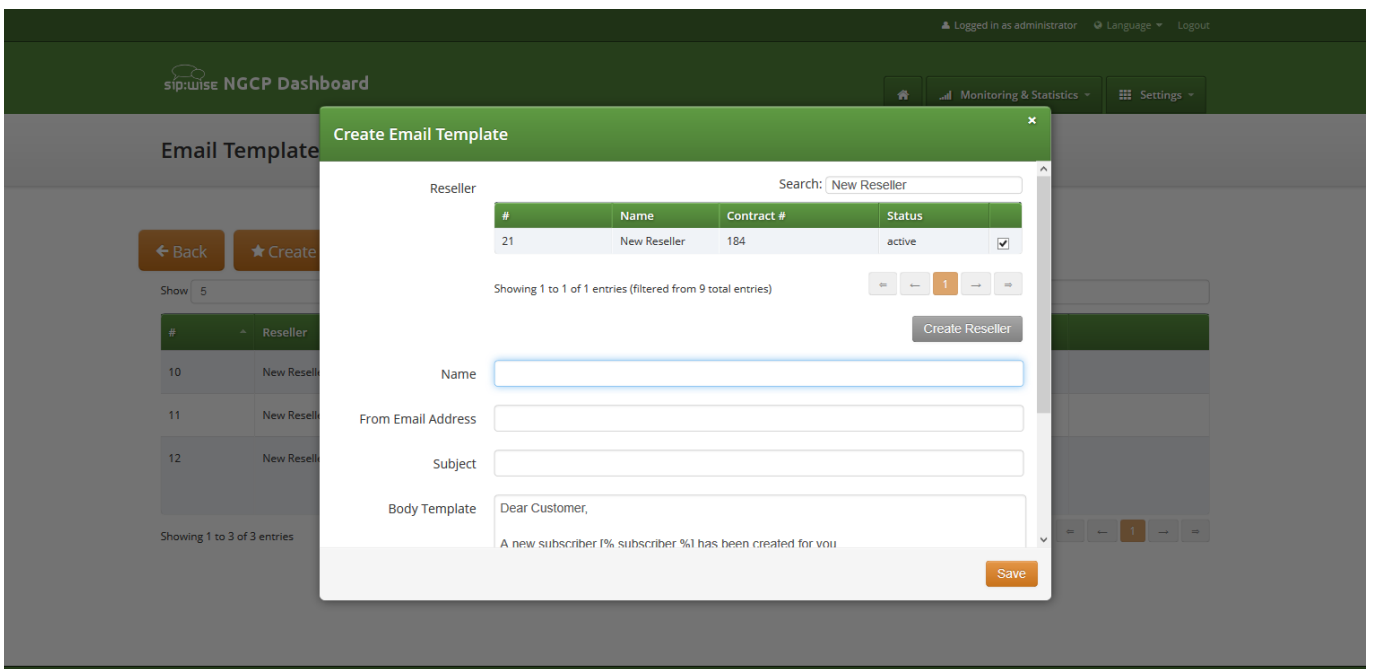
- [%provider.gender%]
- [%provider.firstname%]
- [%provider.lastname%]
- [%provider.comregnum%]
- [%provider.company%]
- [%provider.street%]
- [%provider.postcode%]
- [%provider.city%]
- [%provider.country%]
- [%provider.phonenumber%]
- [%provider.mobilenumber%]
- [%provider.email%]
- [%provider.newsletter%]
- [%provider.faxnumber%]
- [%provider.iban%]
- [%provider.bic%]
- [%provider.vatnum%]
- [%provider.bankname%]
- [%provider.gpp0 - provider.gpp9%]

11.6 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.



To create 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 the sip:provider CE on the [appropriate event](#) Section 11.1 :
 - 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 the sip:provider CE.
- **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.

12 Local Number Porting

The sip:provider CE comes with two ways of accomplishing local number porting (LNP). One is populating the integrated LNP database with porting data, and the other is accessing external LNP databases via the Sipwise LNP daemon using the LNP API (PRO/CARRIER only).

12.1 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 *Settings*→*Number Porting* or via the API. The LNP configuration can be populated individually or via CSV import/export both on the panel and the API.

12.1.1 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 the sip:provider CE. 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.

12.1.2 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.

An optional *Start Date* and *End Date* allows 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.

12.1.3 Enabling local LNP support

In order to activate Local LNP during routing, the feature must be activated in *config.yml*. Set *kamailio→proxy→lnp→enabled* to *yes* and *kamailio→proxy→lnp→type* to *local*.

12.1.4 LNP Routing Procedure

Calls to non-authoritative Carriers

When a call arrives at the system, the calling and called party numbers are first normalized using the *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 solely to find the LNP prefix of the calling peer, which is then stored as *source_lnp_prefix* in the CDR. 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 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_basenum}`: 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 originating 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 `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.

12.1.5 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.

12.1.6 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:

carrier_name carrier_prefix number routing_number start end authoritative skip_rewrite

Table 8: LNP CSV Format

| Name | Description |
|----------------|---|
| Carrier Name | The <i>Name</i> in the LNP Carriers table (string, e.g. <i>My Carrier</i>) |
| Carrier Prefix | The <i>Prefix</i> in the LNP Carriers table (string, e.g. <i>DD55</i>) |
| Number | The <i>Number</i> in the LNP Numbers table (E164 number, e.g. <i>1800666</i>) |
| Routing Number | The <i>Routing Number</i> in the LNP Numbers table (E164 number or empty, e.g. <i>1555666</i>) |
| Start | The <i>Start</i> in the LNP Numbers table (YYYY-MM-DD or empty, e.g. <i>2016-01-01</i>) |
| End | The <i>End</i> in the LNP Numbers table (YYYY-MM-DD or empty, e.g. <i>2016-12-30</i>) |
| Authoritative | The <i>Authoritative</i> flag in the LNP Carriers table (0 or 1) |

Table 8: (continued)

| Skip Rewrite | The <i>Skip Rewrite</i> flag in the LNP Carriers table (0 or 1) |
|--------------|---|
|--------------|---|

13 Provisioning interfaces

The sip:provider CE provides two kinds of provisioning interfaces for easy interconnection with 3rd party tools. The one recommended by Sipwise is the REST API, and the other (soon deprecated) one is SOAP and XMLRPC. Any new functionality is only added to the REST interface, so do not base any new development on SOAP or XMLRPC.

13.1 REST API

The sip:provider CE 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.

13.1.1 API Workflows

The typical tasks done on the API involve managing customers and subscribers. The following chapter focuses on creating, changing and deleting these resources.

Managing Customers and Subscribers

The classical 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 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). In order 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 id from Location header
say "Created customer contact, id is $contact_id";

```



Important

To get the id of a just created resource, you need to parse the *Location* header. This will change in the future for POST requests to optionally also return the resource in the response, 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 the API is accessed via a reseller user. If you are accessing the API as admin user, you also have to provide a *reseller_id* parameter defining the reseller this contact belongs to.

Once the customer contact is created, you can create the actual customer.

```
$req = HTTP::Request->new('POST', "$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 id from Location header
say "Created customer, id is $customer_id";
```

Once the customer is created, 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.

```
$req = HTTP::Request->new('POST', "$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 id from Location header
say "Created subscriber, id is $subscriber_id";

```

**Important**

The domain has to exist prior to creating a subscriber and can be created 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.

```

$req = HTTP::Request->new('PUT', "$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, cft and cft 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 in order to trigger call forwards only during specific time periods.

To provision certain features of a subscriber, you can manipulate the subscriber preferences. A full list of preferences available for a subscriber is available at */api/subscriberpreferencedefs/*.

```

$req = HTTP::Request->new('GET', "$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.

```

$req = HTTP::Request->new('GET', "$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 basic information is still available on the internal database in order to perform billing/rating of calls done by this subscriber, but a subscriber is not able to connect to the system, login or do calls/chats, as the data is removed from the operational tables of the database.

Modifications to resources can not only be done via a GET/PUT combination, you can also add, modify or delete single properties of a resource without actually fetching the whole resource. An example is given below where we terminate the status of a customer using the PATCH method.

```

$req = HTTP::Request->new('PATCH', "$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";

```

13.2 SOAP and XMLRPC API



Important

SOAP and XMLRPC API are deprecated and disabled by default since mr3.6.1. Please consider using REST API as SOAP and XMLRPC API will be deleted in upcoming release(s). To enable SOAP and XMLRPC change */etc/ngcp-config/config.yml* by setting *ossbss→frontend→fcgi* and execute *ngcpcfg apply 'enable SOAP API'*.

The sip:provider CE provides two (soon deprecated) XML based provisioning interfaces - SOAP and XMLRPC. The server provides online documentation about all the functions available. To access the online documentation for the first time, you need to follow the following instructions:

- Generate a password for http access to the provisioning interfaces:

```
htpasswd -nbs myuser mypassword
```

Note

Also see `man 1 htpasswd` on how to generate crypt or MD5 passwords if you like. Of course you may use any other process to generate crypt, MD5 or SHA hashed passwords. But using `htpasswd` ensures the hashes are also understood by Nginx. To install `htpasswd` please run `apt-get install apache2-utils` on your system.

- Edit `/etc/ngcp-config/config.yml`. Under section `ossbss→htpasswd`, replace `user` and `pass` with your new values and execute `ngcpcfg apply 'change SOAP credentials'` as usual.
 - Access <https://<ip>:2443/SOAP/Provisioning.wsdl> and login with your new credentials.
-

Note

The default port for provisioning interfaces is 2443. You can change it in `/etc/ngcp-config/config.yml` by modifying `ossbss→apache→port` and execute `ngcpcfg apply`.

**Important**

The displayed online API documentation shows all the currently available functionalities. Enabling or disabling features in `/etc/ngcp-config/config.yml` will directly reflect in the functions being available via the APIs.

**Important**

If your SOAP client throws errors because of the inline `<documentation>` tags (e.g. Visual Studio and the stock PHP SOAP client complain about this), try to use the WSDL URL <https://<ip>:2443/SOAP/Provisioning.wsdl?plain> instead, which suppresses the output of these tags.

14 Configuration Framework

The sip:provider CE 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 the NGCP 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 6 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.

14.1 Configuration templates

The sip:provider CE 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/ngcp-sems/sems.conf` are stored in `/etc/ngcp-config/templates/etc/ngcp-sems/`.

There are different types of files in this template framework, which are described below.

14.1.1 .tt2 and .customtt.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 sip:provider CE 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 to change the IP used by kamailio load balancer on interface `eth0` to IP 1.2.3.4. This will change kamailio's listen address, when the configuration file is generated. A quick look to the template file under `/etc/ngcp-config/templates/etc/kamailio/lb/kamailio.conf` will show a line like this:

```
listen=udp:[% ip %]:[% kamailio.lb.port %]
```

After applying the changes with the `ngcpconfig 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. Anyways, 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. An advanced user should create a `.customtt.tt2` file from a copy of the corresponding `.tt2` template and leave the `.tt2` template untouched. This way, the user will have his personalized configuration and the system will continue providing a working, updated configuration template in `.tt2` format.

Example: We'll create `/etc/ngcp-config/templates/etc/lb/kamailio.cfg.customtt.tt2` and use it for our personalized configuration. In this example, we'll just 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
ngcpconfig apply 'my commit message'
```

The `ngcpconfig` command will generate `/etc/kamailio/kamailio.cfg` from our custom template instead of the general one.

```
tail -1 /etc/kamailio/kamailio.cfg
# This is my last line comment
```

Tip

The `tt2` files use the [Template Toolkit](#) language. Therefore you can use all the feature this excellent toolkit provides within `ngcpconfig`'s template files (all the ones with the `.tt2` suffix).

14.1.2 `.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
```

14.1.3 .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` which 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 '/etc/init.d/mysql restart' \
> /etc/ngcpcfg-config/templates/etc/mysql/my.cnf.services
echo '/etc/init.d/asterisk restart' \
> /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.

14.2 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.

The `/etc/ngcp-config/ngcpcfg.cfg` file is the main configuration file for `ngcpcfg` itself. Do not manually edit this file unless you really know what you're doing.

14.3 ngcpcfg and its command line options

The `ngcpcfg` utility supports the following command line options:

14.3.1 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.

14.3.2 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/
```

14.3.3 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.

14.3.4 decrypt

Decrypt `/etc/ngcp-config-encrypted.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.

14.3.5 diff

Show uncommitted changes between `ngcpcfg`'s Git repository and the working tree inside `/etc/ngcp-config`. If 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 available since `ngcp-ngcpcfg` version 0.11.0.

14.3.6 encrypt

Encrypt `/etc/ngcp-config` and all resulting configuration files with a user defined password and save the result as `/etc/ngcp-config-encrypted.tgz.gpg`. Note: This feature is only available if the `ngcp-ngcpcfg-locker` package is installed.

14.3.7 help

The *help* options displays `ngcpcfg`'s help screen and then exits without any further actions.

14.3.8 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.

14.3.9 pull

Retrieve modifications from shared storage. Note: This option is available in the High Availability setup only.

14.3.10 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. If no host has been specified then the hosts specified in `/etc/ngcp-config/systems.cfg` are used. Note: This option is available in the High Availability setup only.

14.3.11 services

The *services* option executes the service handlers for any modified configuration file(s)/directory.

14.3.12 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 just 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.
```

If the output doesn't say "OK" just 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.

15 Network Configuration

Starting with version 2.7, the sip:provider CE 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 to generate */etc/network/interfaces* automatically for all nodes based on this central configuration file.

15.1 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
```

In CE systems, there is only one host entry in the file, and it's always named *self*.

15.2 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 sip:provider CE, 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.

- *lb*: This entry means the host is acting as SIP load-balancer for the platform. In a sip:provider CE, this option must always been set. 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 sip:provider CE, this option must always been set. 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 sip:provider CE, this option must always be set. 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 sip:provider CE, this option must always be set. The RTP relay node runs the rtpengine.
- *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.
- *<interface name>*: After the interfaces are defined in the *interfaces* setting, each of those interfaces needs to be specified as a separate setting with the following options:
 - *ip*
 - *netmask*
 - *advertised_ip*
 - *type*

There are different *interface types*, which define the services on a particular *interface*. For example the type *ssh_ext* set for a specific interface defines that the SSH daemon will listen on that interface for incoming connections. The list of possible types is as follows (note that you can assign a type only once per node):

- *mon_ext*: interface for monitoring purposes, e.g. for snmpd
- *rtp_ext*: interface for external RTP relay
- *sip_ext*: interface for external SIP communication between the sip:provider CE and the end points
- *sip_ext_incoming*: extra listen interface for external SIP traffic (optional)
- *sip_int*: interface for internal SIP communication, e.g. between load-balancer, proxy and application servers
- *ssh_ext*: interface for SSH remote login
- *web_ext*: interface for the subscriber web panel and the subscriber's SOAP/REST APIs
- *web_int*: interface for the administrator web panel, his SOAP/REST APIs and internal API communication
- *aux_ext*: interface for potentially insecure external components like rsyslogd service; e.g. the CloudPBX module can use those services 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 rsyslog services.

16 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.

16.1 Extra SIP Sockets

By default, the load-balancer listens on the UDP and TCP ports 5060 (*kamailio*→*lb*→*port*) and TLS port 5061 (*kamailio*→*lb*→*tls*→*port*). If you need to setup one or more extra SIP listening ports or IP addresses in addition to those standard ports, please edit the *kamailio*→*lb*→*extra_sockets* option in your */etc/ngcp-config/config.yml* file.

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'
```

The direction of communication through this SIP extra socket is incoming+outgoing. The sip:provider CE 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 until the end of next chapter Section 16.2 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.

16.2 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:

```
ngcp-network --set-interface=eth0.100 --ip=auto --netmask=auto --type=sip_ext_incoming -- ↵
  type=rtp_int_100
```

The generated file should look like the following:

```

..
..
  eth0.100:
    hwaddr: ff:ff:ff:ff:ff:ff
    ip: 192.168.1.3
    netmask: 255.255.255.0
    type:
      - sip_ext_incoming
      - 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 sip:provider CE 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'
```

After applying the changes, a new SIP socket will listen on IP *192.168.1.3* and this socket can now be used as source socket to send SIP messages to your peer for example. In 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* 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.

17 Security and Maintenance

Once the sip:provider CE 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.

17.1 Sipwise SSH access to sip:provider CE

The sip:provider CE 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 sip:provider CE 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.
```

17.2 Firewalling

The sip:provider CE runs a wide range of services. Some of them need to interact with the user, while some others need to interact with the administrator or with nobody at all. Assuming that we trust the sip:provider CE server for outgoing connections, we'll focus only on incoming traffic to define the services that need to be open for interaction.

Table 9: Subscribers

| Service | Default port | Config option |
|------------------------------|----------------------|--|
| Customer self care interface | 443 TCP | <code>www_admin→http_csc→port</code> |
| SIP | 5060 UDP, TCP | <code>kamailio→lb→port</code> |
| SIP over TLS | 5061 TCP | <code>kamailio→lb→tls→port + kamailio→lb→tls→enable</code> |
| RTP | 30000-40000 UDP | <code>rtpproxy→minport + rtpproxy→maxport</code> |
| XCAP | 1080 TCP | <code>kamailio→proxy→presence→enable + nginx→xcap_port</code> |
| XMPP | 5222 and 5269 TCP | None, standard XMPP ports for clients (5222) and federation (5269) |

Table 10: Administrators

| Service | Default port | Config option |
|-------------------------|--------------|--|
| SSH/SFTP | 22 TCP | NA |
| Administrator interface | 1443 TCP | <code>www_admin→http_admin→port</code> |
| Provisioning interfaces | 2443 TCP | <code>ossbss→apache→port</code> |

Caution

To function correctly, the *rtengine* requires an additional *iptables* rule installed. This rule (with a target of `RTPENGINE`) is automatically installed and removed when the *rtengine* 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`

17.3 Password management

The sip:provider CE 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 document.

- The login for the system account *cdrexpert* 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 cdrexpert`.

- The *root* user in MySQL has no default password. A password should be set using the *mysqladmin password* command.
- The administrative web interface has a default user *administrator* with password *administrator*. It should be changed within this interface.
- Generate new password for user *ngcpssoap* to access the provisioning interfaces, see the details in Section 13.

The Vagrant/VirtualBox/VMWare sip:provider CE images come with more default credentials which should be changed immediately:

- The default password of the system account *root* is *sipwise*. A password must be changed immediately using command *passwd root*.
- SSH *authorized_keys* for users *root* and *sipwise* should be wiped out using command *rm ~root/.ssh/sipwise_vagrant_key ~sipwise/.ssh/sipwise_vagrant_key* for VirtualBox/VMWare images (skip the step if you use Vagrant).



Important

Many NGCP 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.

17.4 SSL certificates.

The sip:provider CE 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*:
 - *ossbss*→*apache*→*autoprov*→*sslcertfile* and *ossbss*→*apache*→*autoprov*→*sslcertkeyfile* for the *provisioning interface*.
 - *ossbss*→*apache*→*restapi*→*sslcertfile* and *ossbss*→*apache*→*restapi*→*sslcertkeyfile* for the *REST interface*.
 - *www_admin*→*http_admin*→*sslcertfile* and *www_admin*→*http_admin*→*sslcertkeyfile* for the *admin interface*.
 - *www_admin*→*http_csc*→*sslcertfile* and *www_admin*→*http_csc*→*sslcertkeyfile* for the *customer self care interface*.
- Apply the configuration changes with *ngcpcfg apply 'added web ssl certs'*.

The sip:provider CE 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 (*kamailio*→*lb*→*tls*→*enable* (disabled by default)).

- Generate the certificates.
- Upload the certificates to the system
- Set the path to the new certificates in `/etc/ngcp-config/config.yml`:
 - `kamailio→lb→tls→sslcrtfile` and `kamailio→lb→tls→sslcertkeyfile` .
- Apply the configuration changes with `ngcpcfg apply 'added kamailio certs'`.

17.5 Securing your sip:provider CE against SIP attacks

The sip:provider CE 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.

17.5.1 Denial of Service

As soon as you have packets arriving on your sip:provider CE 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. sip:provider CE allow you to block such kind of attack quite easily, by configuring the following section in your `/etc/ngcp-config/config.yml`:

```
security:
  dos_ban_enable: 'yes'
  dos_ban_time: 3600
  dos_reqs_density_per_unit: 50
  dos_sampling_time_unit: 2
```

Basically, as soon as sip:provider CE receives more than 50 messages from the same IP in a time window of 2 seconds, that IP will be block for 3600 sec, and you will see in the 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 sht_dump ipban
```

17.5.2 Bruteforcing SIP credentials

This is a very common attack you can easily detect checking your `/var/log/ngcp/kamailio-proxy.log`. You will see INVITE/REGISTER messages coming in with strange usernames. Attackers is 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 sip:provider CE allow you to detect and block such attacks quite easily, by configuring the following `/etc/ngcp-config/config.yml` section:

```
failed_auth_attempts: 3
failed_auth_ban_enable: 'yes'
failed_auth_ban_time: 3600
```

You may increase the number of failed attempt if you want (in some cases it's better to be safed, some users can be banned accidentally because they are not writing the right password) and adjust the ban time. If a user try to authenticate an INVITE (or REGISTER) for example and it fails more then 3 times, the "user@domain" (not the IP as for Denial of Service attack) will be block for 3600 seconds. In this case you will see in your `/var/log/ngcp/kamailio-lb.log` the following lines:

```
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 are shown in the Admin web interface, you can check them by accessing the **Security Bans** section in the main menu. You can check the banned user as well by retrieving the same info directly from kamailio memory, using the following commands:

```
# ngcp-kamctl lb fifo sht_dump auth
```

17.6 Backup and recovery

17.6.1 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. Hence, we strongly suggest you to configure a backup procedure. The sip:provider CE 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.

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.

- System configuration

The system configuration files such as `/etc/mysql/sipwise.cnf` and the `/etc/ngcp-config/` directory should be included in the backup as well. We suggest backing up the whole `/etc` folder.

- 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.

The built-in backup solution

The sip:provider CE 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/cdreexport` 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 “NGCP configs overview” chapter for the backup configuration options.

Once you set the required configuration options, apply the changes:

```
ngcpcfg apply 'enabled the backup feature'
```

Once you activate the feature, the sip:provider CE will create backups in the off-peak time and put them to the `/var/backup/ngcp_backup` directory. 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.

17.6.2 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 the sip:provider CE as explained in chapter 2.
- Restore the `/etc/ngcp-config/` directory and the `/etc/mysql/sipwise.cnf` file from the backup, overwriting your local files.
- 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'
```

17.7 Reset database

To reset database to its original state you can use the script provided by CE: * Execute `ngcp-reset-db`. It will assign new unique password for the NGCP services and restart all services. **IMPORTANT:** All existing data will be wiped out without possibility of restoring.

17.8 System requirements and performance

The sip:provider CE 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 the sip:provider CE has sometimes to be tuned for special environments, special hardware requirements or just 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 the sip:provider CE administrator tune the system requirements for optimum performance.

Table 11: Requirement_options

| Option | Default value | Requirement impact |
|--------------------------|-------------------------|--|
| cleanuptools→binlog_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. |
| database→bufferpoolsize | 64MB | 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. |
| kamailio→lb→pkg_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. |
| kamailio→lb→shm_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. |
| kamailio→lb→tcp_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. |
| kamailio→lb→tls→enable | 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. |
| kamailio→lb→udp_children | 8 | See <i>kamailio→lb→tcp_children</i> explanation |

Table 11: (continued)

| Option | Default value | Requirement impact |
|---|-------------------------|--|
| <code>kamailio→proxy→children</code> | 8 | See <code>kamailio→lb→tcp_children</code> 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. |
| <code>kamailio→proxy→*_expires</code> | | 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. |
| <code>kamailio→proxy→natping_interval</code> | 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 <code>kamailio→proxy→natping_processes</code> explanation. |
| <code>kamailio→proxy→natping_processes</code> | 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 <code>kamailio→proxy→natping_interval</code> parameter the number of workers may need to be adjusted. The number can be calculated like $\text{nated_subscribers}/\text{natping_interval}/\text{pings_per_second_per_process}$. 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. |
| <code>kamailio→proxy→shm_mem</code> | 1/16 * Total System RAM | See <code>kamailio→lb→shm_mem</code> explanation. |
| <code>rateomat→enable</code> | yes | Set this to no if the system shouldn't perform rating on the CDRs. This will save CPU usage. |
| <code>rsyslog→external_log</code> | 0 | If enabled, the system will send the log messages to an external server. Depending on the <code>rsyslog→external_loglevel</code> parameter this can increase dramatically the network traffic. |
| <code>rsyslog→ngcp_logs_preserve_days</code> | 93 | This setting will set the number of days ngcp logs under <code>/var/log/ngcp</code> 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 sip:provider CE configuration for high/low performance:

```
root@spce:~# /usr/sbin/ngcp-toggle-performance-config
```

```

/usr/sbin/ngcp-toggle-performance-config - tool to adjust sip:provider 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:~#

```

17.9 Troubleshooting

The sip:provider CE 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 sip:provider CE 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 sip:provider CE by Sipwise operational team while main log files for daily routine usage are:

| Log file | Content | Estimated size |
|--|---|----------------|
| <code>/var/log/ngcp/api.log</code> | API logs providing type and content of API requests and responses as well as potential errors | medium |
| <code>/var/log/ngcp/panel.log</code> <code>/var/log/ngcp/panel-debug.log</code> | Admin Web UI logs when performing operational tasks on the ngcp-panel | medium |

| Log file | Content | Estimated size |
|----------------------------------|---|----------------|
| /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 |
| /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 |
| /var/log/ngcp/sems.log | Overview of SIP requests and replies between lb, proxy and sems processes | small |

| Log file | Content | Estimated size |
|-----------------------|--|----------------|
| /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.

17.9.1 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` 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 ↔
=364e4676776621034977934e055d19ea@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
- 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
```

17.9.2 Collecting SIP traces

The sip:provider CE platform provides several tools to collect SIP traces. It can be used the sip:provider CE *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
```

A NGCP configs overview

A.1 config.yml overview

config.yml is the main configuration YAML file used by Sipwise NGCP. 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.

A.1.1 asterisk

The following is the asterisk section:

```
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 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.

A.1.2 autoprov

The following is the autoprovisioning section:

```
autoprov:
  hardphone:
    skip_vendor_redirect: 'no'
  server:
    bootstrap_port: 1445
    ca_certfile: '/etc/ngcp-config/ssl/client-auth-ca.crt'
    host: localhost
    port: 1444
    server_certfile: '/etc/ngcp-config/ssl/myserver.crt'
    server_keyfile: '/etc/ngcp-config/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.

A.1.3 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'
  rotate_days: 7
  storage_dir: '/var/backup/ngcp_backup'
  temp_backup_dir: '/tmp/ngcp_backup'
```

- `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_subject`: 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.rotate_days`: Number of backups to keep stored.
- `backuptools.storage_dir`: Storage directory of backups.
- `backuptools.temp_backup_dir`: Temporary storage directory of backups.

A.1.4 cdrexport

The following is the cdr export section:

```
cdrexport:  
  daily_folder: 'yes'  
  export_failed: 'no'  
  export_incoming: 'no'  
  exportpath: '/home/jail/home/cdreexport'  
  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.exportpath`: The path to store CDRs in .csv format.
- `cdrexport.full_names`: Use full namen 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.

A.1.5 checktools

The following is the check tools section:

```
checktools:  
  collcheck:  
    cpuidle: 0.1  
    dfused: 0.9  
    eximmaxqueue: 15  
    loadlong: 2  
    loadmedium: 2  
    loadshort: 3  
    maxage: 600  
    memused: 0.7  
    siptimeout: 15  
    swapfree: 0.5  
  active_check_enable: 1  
  asr_nsr_statistics: 1  
  exim_check_enable: 0  
  force: 0  
  kamilio_check_concurrent_calls_enable: 0  
  kamilio_check_dialog_active_enable: 1  
  kamilio_check_dialog_early_enable: 1  
  kamilio_check_dialog_incoming_enable: 1  
  kamilio_check_dialog_local_enable: 1  
  kamilio_check_dialog_outgoing_enable: 1  
  kamilio_check_dialog_relay_enable: 1
```

```

kamailio_check_shmem_enable: 1
kamailio_check_usrloc_regdevices_enable: 1
kamailio_check_usrloc_regusers_enable: 1
mpt_check_enable: 1
mysql_check_enable: 1
mysql_check_replication: 1
oss_check_provisioned_subscribers_enable: 1
sip_check_enable: 1
sipstats_check_num_packets: 1
sipstats_check_num_packets_perday: 1
sipstats_check_partition_size: 1
snmpd:
  communities:
    public:
      - localhost

```

- `checktools.collcheck.cpuidle`: Sets the minimum value for CPU usage (0.1 means 10%).
- `checktools.collcheck.dfused`: Sets the maximum value for DISK usage (0.9 means 90%).
- `checktools.collcheck.loadlong/loadlong/loadshort`: Max values for load (long, short, medium term).
- `checktools.collcheck.maxage`: Max age in seconds.
- `checktools.collcheck.memused`: Sets the maximum value for MEM usage (0.7 means 70%).
- `checktools.collcheck.siptimeout`: Max timeout for sip options.
- `checktools.collcheck.swapfree`: Sets the minimum value for SWAP free (0.5 means 50%).
- `checktools.exim_check_enable`: Exim queue check plugin for collectd.
- `checktools.active_check_enable`: Active node check plugin for collectd.
- `checktools.asr_nsr_statistics`: enable/Disable ASR/NSR statistics.
- `checktools.force`: Perform checks even if not active from `ngcp-check_active` command.
- `checktools.kamailio_check_*`: Enable/Disable SNMP collective check plugin for Kamailio.
- `checktools.mpt_check_enable`: MPT raid SNMP check plugin.
- `checktools.mysql_check_enable`: MySQL SNMP check plugin.
- `checktools.mysql_check_replication`: MySQL replication check.
- `checktools.oss_check_provisioned_subscribers_enable`: OSS provisioned subscribers count plugin.
- `checktools.sip_check_enable/sipstats_check_*`: Enable/Disable SIP check plugins.
- `checktools.snmpd.communities`: Sets the snmp community and sources (separated by comma , - ex. source: 127.0.0.1, 10.10.10.2, 10.10.10.3).

A.1.6 cleanuptools

The following is the cleanup tools section:

```
cleanuptools:
  acc_cleanup_days: 90
  archive_targetdir: '/var/backups/cdr'
  binlog_days: 15
  cdr_archive_months: 2
  cdr_backup_months: 2
  cdr_backup_retro: 3
  compress: gzip
  sql_batch: 10000
  trash_cleanup_days: 30
```

- `cleanuptools.acc_cleanup_days`: Clean up ACC entry older then 90 days.
- `cleanuptools.binlog_days`: Expire MySQL binlogs after 15 days.
- `cleanuptools.cdr_archive_months`: How many months worth of records to keep in the table and not move into the monthly archive tables.
- `cleanuptools.cdr_backup_months`: How many months worth of records to keep in the table and not move into the monthly backup tables.
- `cleanuptools.cdr_backup_retro`: How many months to process for backups, going backwards in time. Using the example above, with this value set to "3", the months October, September and August would be backed up, while any older records would be left untouched.
- `cleanuptools.sql_batch`: How many records to process within a single statement.
- `cleanuptools.trash_cleanup_days`: Clean up `acc_trash` and `acc_backup` entry after 30 days.

A.1.7 database

The following is the database section:

```
database:
  bufferpoolsize: 24768M
```

- `database.bufferpoolsize`: `InnoDB_buffer_pool_size` value in `/etc/mysql/my.cnf`

A.1.8 faxserver

The following is the fax server section:

```
faxserver:  
  enable: yes  
  fail_attempts: '3'  
  fail_retry_secs: '60'  
  mail_from: 'Sipwise NGCP 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_attempts".
- `faxserver.mail_from`: Sets the e-mail From Header for incoming fax.

A.1.9 general

The following is the general section:

```
general:  
  adminmail: adjust@example.org  
  companyname: sipwise  
  lang: en
```

- `general.adminmail`: Email address used by `monit` to send notifications to.
- `general.lang`: Sets sounds language (e.g: *de* for German)

A.1.10 heartbeat

The following is the heartbeat section:

```
heartbeat:  
  hb_watchdog:  
    action_max: 5  
    enable: 'yes'  
    interval: 10  
    transition_max: 10  
  pingnodes:  
    - 10.60.1.1  
    - 192.168.3.4
```

- `heartbeat.hb_watchdog.enable`: Enable heartbeat watchdog in order to prevent and fix split brain scenario.
- `heartbeat.hb_watchdog.action_max`: Max errors before taking any action.
- `heartbeat.hb_watchdog.interval`: Interval in secs for the check.

- heartbeat.hb_watchdog.transition_max: Max checks in transition state.
- heartbeat.pingnodes: List of pingnodes for heartbeat. Minimum 2 entries, otherwise by default NGCP will set the default gateway and DNS servers as pingnodes.

A.1.11 intercept

The following is the legal intercept section:

```
intercept:  
  captagent:  
    port: 18090  
    schema: http  
  enabled: 'no'
```

- intercept.captagent.enable: Enable captagent for Lawful Interception (additional NGCP module).

A.1.12 kamailio

The following is the kamailio section:

```
kamailio:  
  lb:  
    debug: 'no'  
    extra_sockets: ~  
    max_forwards: 70  
    nattetest_exception_ips:  
      - 1.2.3.4  
      - 5.6.7.8  
    pkg_mem: 16  
    port: 5060  
    security:  
      dos_ban_enable: 'yes'  
      dos_ban_time: 300  
      dos_reqs_density_per_unit: 50  
      dos_sampling_time_unit: 5  
      dos_whitelisted_ips: ~  
      dos_whitelisted_subnets: ~  
      failed_auth_attempts: 3  
      failed_auth_ban_enable: 'yes'  
      failed_auth_ban_time: 3600  
    shm_mem: 2012  
    start: 'yes'  
    strict_routing_safe: 'no'  
    tcp_children: 8  
    tcp_max_connections: 2048  
    tls:
```

```
enable: 'no'
port: 5061
sslcertfile: '/etc/kamailio/kamailio-selfsigned.pem'
sslcertkeyfile: '/etc/kamailio/kamailio-selfsigned.key'
udp_children: 8
use_dns_cache: 'on'
proxy:
  allow_info_method: 'no'
  allow_peer_relay: 'no'
  allow_refer_method: 'no'
  authenticate_bye: 'no'
  cf_depth_limit: 10
  children: 8
  debug: 'no'
  default_expires: 3600
  enum_suffix: e164.arpa.
  filter_100rel_from_supported: 'yes'
  fritzbox:
    enable: 'no'
    prefixes:
      - 112
      - 110
      - 118[0-9]{2}
  foreign_domain_via_peer: 'no'
  ignore_auth_realm: 'no'
  keep_original_to: 'no'
  lnp:
    api:
      invalid_lnp_routing_codes:
        - ^EE00
        - ^DD00
      lnp_request_blacklist: []
      lnp_request_whitelist: []
      request_timeout: '1000'
    enabled: no
    type: api
  max_expires: 43200
  max_gw_lcr: 128
  max_registrations_per_subscriber: 5
  min_expires: 60
  nathelper_dbro: 'no'
  natping_interval: 30
  natping_processes: 7
  nonce_expire: 300
  pbx:
    hunt_display_indicator: '[h]'
  perform_peer_lcr: 0
  pkg_mem: 16
```

```
port: 5062
presence:
  enable: 'yes'
  max_expires: '3600'
  reginfo_domain: example.org
proxy_lookup: 'no'
set_ruri_to_peer_auth_realm: 'no'
shm_mem: 2012
start: 'yes'
tcp_children: 4
use_enum: 'no'
usrloc_dbmode: 1
```

- `kamailio.lb.debug`: Enable intensive debug level.
- `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.nattest_exception_ips`: List of IPs that don't need the NAT test.
- `kamailio.lb.shm_mem`: Shared memory used by Kamailio Load Balancer. The default value is auto generated by the system, depending on your system architecture.
- `kamailio.lb.pkg_mem`: PKG memory used by Kamailio Load Balancer. The default value is auto generated by the system, depending on your system architecture.
- `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_u` 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.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.strict_routing_safe`: Enable strict routing handle feature.
- `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.proxy.allow_info_method`: Allow INFO 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.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.debug`: Enable intensive debug level.
- `kamailio.proxy.default_expires`: Default expires value in seconds for REGISTER messages.
- `kamailio.proxy.foreign_domain_via_peer`: Enable calls to foreign domains via peers.
- `kamailio.proxy.shm_mem`: Shared memory used by Kamailio Proxy. The default value is auto generated by the system, depending on your system architecture.
- `kamailio.proxy.pkg_mem`: PKG memory used by Kamailio Proxy. The default value is auto generated by the system, depending on your system architecture.
- `kamailio.proxy.enum_suffix`: Sets ENUM suffix - don't forget . (dot).
- `kamailio.proxy.filter_100rel_from_supported`: Enable filtering of *100rel* from Supported header, to disable PRACK.
- `kamailio.proxy.fritzbox.enable`: Enable detection for Fritzbox special numbers. Ex. Fritzbox add the AC prefix to emergency numbers.
- `kamailio.proxy.fritzbox.prefixes`: Specifies special prefixes to detect in order to remove the AC prefix added by Fritzbox.
- `kamailio.proxy.ignore_auth_realm`: Ignore SIP authentication realm.
- `kamailio.proxy.keep_original_to`: Not used now.
- `kamailio.proxy.lnp.enabled`: Enable/disable LNP (local number portability) lookup during call setup
- `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 NGCP PRO / CARRIER installations.
- `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.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.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.max_expires`: Sets the maximum expires in seconds for registration.

- `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.min_expires`: Sets the minimum expires in seconds for registration.
- `kamailio.proxy.natping_interval`: Sets the NAT ping interval in seconds.
- `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.nonce_expire`: Nonce expire time in seconds.
- `kamailio.proxy.perform_peer_lcr`: Enable/Disable Least Cost Routing based on peering fees.
- `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 the NGCP domain used in callback for mobile push.
- `kamailio.proxy.set_ruri_to_peer_auth_realm`: Set R-URI using peer auth realm
- `kamailio.proxy.use_enum`: Enable/Disable ENUM feature.

A.1.13 mediator

The following is the mediator section:

```
mediator:  
  interval: 10
```

- `mediator.interval`: Running interval of mediator.

A.1.14 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

A.1.15 ntp

The following is the ntp server section:

```
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.

A.1.16 ossbss

The following is the ossbss section:

```
ossbss:
  apache:
    port: 2443
    proxylisten: 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/ssl/myserver.crt'
    sslcertkeyfile: '/etc/ngcp-config/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
  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).

A.1.17 pbx (only with additional cloud PBX module installed)

The following is the PBX section:

```

pbx:
  bindport: 5085

```

```
enable: 'no'
highport: 55000
lowport: 50001
media_processor_threads: 10
session_processor_threads: 10
xmlrpcport: 8095
```

- `pbx.enable`: Enable Cloud PBX module.

A.1.18 prosody

The following is the prosody section:

```
prosody:
  ctrl_port: 5582
  log_level: info
```

- `prosody.ctrl_port`: XMPP server control port.
- `prosody.log_level`: Prosody loglevel.

A.1.19 pushd

The following is the pushd section:

```
pushd:
  apns:
    certificate: ''
    enable: 'no'
    endpoint: gateway.sandbox.push.apple.com
    feedback_endpoint: feedback.sandbox.push.apple.com
    feedback_interval: 3600
    key: ''
    socket_timeout: 0
  enable: 'no'
  gcm:
    enable: 'no'
    key: ''
  port: 45060
  processes: 4
  ssl: 'yes'
  unique_device_ids: 'no'
```

- `pushd.enable`: Enable/Disable Push Notification feature.

- `pushd.apns.certificate`: Specify the Apple certificate for push notification.
- `pushd.apns.enable`: Enable/Disable Apple push notification.
- `pushd.apns.key`: Specify the Apple key for push notification.
- `pushd.gcm.enable`: Enable/Disable Google push notification.
- `pushd.gcm.key`: Specify the Google key for push notification.

A.1.20 qos

The following is the QOS section:

```
qos:
  tos_rtp: 184
  tos_sip: 184
```

- `qos.tos_rtp`: TOS value for RTP traffic.
- `qos.tos_sip`: TOS value for SIP traffic.

A.1.21 rate-o-mat

The following is the rate-o-mat section:

```
rateomat:
  enable: 'yes'
  loopinterval: 10
  splitpeakparts: 0
```

- `rateomat.enable`: Enable/Disable Rate-o-mat
- `rateomat.loopinterval`: How long we shall sleep before looking for unrated CDRs again.
- `rateomat.splitpeakparts`: Whether we should split CDRs on peakttime borders.

A.1.22 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.

A.1.23 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'
```

- `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.

A.1.24 rsyslog

The following is the rsyslog section:

```
rsyslog:
  elasticsearch:
    action:
      resumeretrycount: '-1'
    bulkmode: 'on'
    dynSearchIndex: 'on'
    enable: 'yes'
    queue:
      dequeuebatchsize: 300
      size: 5000
      type: linkedlist
  external_address:
  external_log: 0
  external_loglevel: warning
  external_port: 514
  external_proto: udp
  ngcp_logs_preserve_days: 93
```

- `rsyslog.elasticsearch.enable`: Enable/Disable Elasticsearch web interface
- `rsyslog.external_address`: Set the remote rsyslog server.
- `rsyslog.ngcp_logs_preserve_days`: Specify how many days to preserve old rotated log files in `/var/log/ngcp/old` path.

A.1.25 rtpproxy

The following is the rtp proxy section:

```
rtpproxy:
  allow_userspace_only: 'yes'
  maxport: 40000
  minport: 30000
  rtp_timeout: 21600
  rtp_timeout_onhold: 3600
```

- `rtpproxy.allow_userspace_only`: Enable/Disable the user space failover for rtpengine (*yes* means enable). By default rtpengine works in kernel space.
- `rtpproxy.maxport`: Maximum port used by rtpengine for RTP traffic.
- `rtpproxy.minport`: Minimum port used by rtpengine for RTP traffic.
- `rtpproxy.rtp_timeout`: Maximum limit in seconds for a call (6h).
- `rtpproxy.rtp_timeout_onhold`: Maximum limit in seconds for an onhold (1h).

A.1.26 security

The following is the security section:

```
security:
  firewall:
    blacklist_networks_4: ~
    blacklist_networks_6: ~
    enable: 'yes'
    sipwise_support_access: 'no'
    whitelist_networks_4: ~
    whitelist_networks_6: ~
```

- `security.firewall.enable`: Enable/Disable security configuration for IPv6 and IPv4 (`sysctl_ipv6.conf`, `sysctl_ipv4.conf`).

A.1.27 sems

The following is the SEMS section:

```
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
  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
```

- `sems.conference.enable`: Enable/Disable conference feature.
- `sems.conference.max_participants`: Sets the number of concurrent participant.
- `sems.highport`: Maximum ports used by sems for RTP traffic.
- `sems.debug`: Enable/Disable debug mode.
- `sems.lowport`: Minimum ports used by sems for RTP traffic.
- `sems.prepaid.enable`: Enable/Disable prepaid feature.
- `sems.sbc.calltimer_max`: Sets the maximum call duration for inter-domain calls.
- `sems.sbc.outbound_timeout`:: Sets the maximum call duration for outboud calls.

- `sems.sbc.session_timer.enable`: Enable/Disable session timers (deprecated, use the web interface configuration).
- `sems.vsc.*`: Define here the VSC codes.

A.1.28 snmpagent

The following is the SNMP Agent section:

```
snmpagent:
  daemonize: '1'
  debug: '0'
  update_interval: '30'
```

- `daemonize`: Enable/Disable `ngcp-snmp-agent` daemonization.
- `debug`: Enable/Disable debug output.
- `update_interval`: Sets the interval in seconds used to update the fetched data.

A.1.29 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`).

A.1.30 voisniff

The following is the voice sniffer section:

```
voisniff:
  admin_panel: 'no'
  daemon:
    bpf: 'port 5060 or 5062 or ip6 proto 44 or ip[6:2] & 0x1fff != 0'
    external_interfaces: 'eth0 eth1'
  filter:
    exclude:
      -
        active: 0
        case_insensitive: 1
        pattern: '\ncseq: *\d+ +(register|notify|options)'
```

```
include: []
internal_interfaces: lo
mysql_dump_threads: 4
start: 'no'
threads_per_interface: 10
partitions:
  increment: 700000
  keep: 10
```

- `voisniff.admin_panel`: Enable/Disable SIP STATS on Admin interface. Default is *no*.
- `voisniff.daemon.external_interfaces`: Define binding interfaces.
- `voisniff.daemon.start`: Change to *yes* if you want `voisniff` start at boot. Default is *no*.

A.1.31 `www_admin`

The following is the WEB Admin interface (`www_admin`) section:

```
www_admin:
  ac_dial_prefix: 0
  apache:
    autoprov_port: 1444
  billing_features: 1
  callingcard_features: 0
  callthru_features: 0
  cc_dial_prefix: 00
  conference_features: 1
  contactmail: adjust@example.org
  dashboard:
    enabled: 1
  default_admin_settings:
    call_data: 0
    is_active: 1
    is_master: 0
    read_only: 0
    show_passwords: 1
  domain:
    preference_features: 1
    rewrite_features: 1
    vsc_features: 0
  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/ssl/myserver.crt'
  sslcertkeyfile: '/etc/ngcp-config/ssl/myserver.key'
http_csc:
  autoprov_bootstrap_port: 1445
  autoprov_port: 1444
  port: 443
  serveradmin: support@sipwise.com
  servername: "\"myserver\""
  ssl_enable: 'yes'
  sslcertfile: '/etc/ngcp-config/ssl/myserver.crt'
  sslcertkeyfile: '/etc/ngcp-config/ssl/myserver.key'
logging:
  apache:
    acc:
      facility: daemon
      identity: oss
      level: info
    err:
      facility: local7
      level: info
peer:
  preference_features: 1
peering_features: 1
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'
subscriber:
  auto_allow_cli: 0
  extension_features: 0
  voicemail_features: 1
```

- `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.