



# The sip:provider CE Handbook mr4.5.7

Sipwise GmbH  
<support@sipwise.com>

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	About this Handbook . . . . .	1
1.2	What is the sip:provider CE? . . . . .	1
1.3	The Advantages of the sip:provider CE . . . . .	1
1.4	Who is the sip:provider CE for? . . . . .	2
1.5	Getting Help . . . . .	2
1.5.1	Community Support . . . . .	2
1.5.2	Commercial Support . . . . .	2
<b>2</b>	<b>Architecture</b>	<b>3</b>
2.1	SIP Signaling and Media Relay . . . . .	3
2.1.1	SIP Load-Balancer . . . . .	4
2.1.2	SIP Proxy/Registrar . . . . .	5
2.1.3	SIP Back-to-Back User-Agent (B2BUA) . . . . .	5
2.1.4	SIP App-Server . . . . .	6
2.1.5	Media Relay . . . . .	6
2.2	Redis Database . . . . .	7
<b>3</b>	<b>Initial Installation</b>	<b>8</b>
3.1	Prerequisites . . . . .	8
3.2	Using the NGCP install CD (recommended) . . . . .	8
3.3	Using the NGCP installer . . . . .	9
3.3.1	Installing the Operating System . . . . .	9
3.3.2	Installing the sip:provider CE . . . . .	10
3.4	Using a pre-installed virtual machine . . . . .	11
3.4.1	Vagrant box for VirtualBox . . . . .	11
3.4.2	VirtualBox image . . . . .	13
3.4.3	VMware image . . . . .	14

3.4.4	Amazon EC2 image	14
<b>4</b>	<b>Initial System Configuration</b>	<b>19</b>
4.1	Network Configuration	19
4.2	Apply Configuration Changes	20
4.3	Start Securing Your Server	20
4.4	Configuring the Email Server	21
4.5	Advanced Network Configuration	21
4.5.1	Audiocodes devices workaround	21
4.6	What's next?	22
<b>5</b>	<b>VoIP Service Configuration Scenario</b>	<b>23</b>
5.1	Creating a Customer	23
5.2	Creating a Subscriber	28
5.3	Domain Preferences	33
5.4	Subscriber Preferences	35
5.5	Creating Peerings	36
5.5.1	Creating Peering Groups	36
5.5.2	Creating Peering Servers	38
5.5.3	Authenticating and Registering against Peering Servers	45
5.6	Configuring Rewrite Rule Sets	48
5.6.1	Inbound Rewrite Rules for Caller	51
5.6.2	Inbound Rewrite Rules for Callee	53
5.6.3	Outbound Rewrite Rules for Caller	54
5.6.4	Outbound Rewrite Rules for Callee	55
5.6.5	Emergency Number Handling	55
5.6.6	Assigning Rewrite Rule Sets to Domains and Subscribers	58
5.6.7	Creating Dialplans for Peering Servers	59
5.6.8	Call Routing Verification	59

<b>6 Features</b>	<b>65</b>
6.1 Managing System Administrators	65
6.1.1 Configuring Administrators	65
6.1.2 Access Rights of Administrators	66
6.2 Access Control for SIP Calls	69
6.2.1 Block Lists	69
6.2.2 NCOS Levels	71
6.2.3 IP Address Restriction	78
6.3 Call Forwarding and Call Hunting	79
6.3.1 Setting a simple Call Forward	79
6.3.2 Advanced Call Hunting	79
6.4 Local Number Porting	82
6.4.1 Local LNP Database	83
6.5 Emergency Mapping	86
6.5.1 Emergency Mapping Description	88
6.5.2 Emergency Mapping Configuration	88
6.6 Header Manipulation	94
6.6.1 Header Filtering	94
6.6.2 Codec Filtering	95
6.6.3 Enable History and Diversion Headers	95
6.7 SIP Trunking with SIPconnect	96
6.7.1 User provisioning	96
6.7.2 Inbound calls routing	96
6.7.3 Number manipulations	96
6.7.4 Registration	99
6.8 Trusted Subscribers	100
6.9 Voicemail System	100
6.9.1 Accessing the IVR Menu	100



6.9.2	IVR Menu Structure	101
6.9.3	Type Of Messages	102
6.9.4	Folders	103
6.9.5	Flowcharts with Voice Prompts	103
6.10	Configuring Subscriber IVR Language	108
6.11	Sound Sets	108
6.11.1	Configuring Early Reject Sound Sets	109
6.12	Conference System	114
6.12.1	Configuring Call Forward to Conference	115
6.12.2	Configuring Conference Sound Sets	115
6.12.3	Joining the Conference	117
6.12.4	Conference Flowchart with Voice Prompts	117
6.13	Malicious Call Identification (MCID)	119
6.13.1	Setup	119
6.13.2	Usage	120
6.13.3	Advanced configuration	120
6.14	Subscriber Profiles	120
6.14.1	Subscriber Profile Sets	120
6.15	SIP Loop Detection	123
6.16	Invoices and Invoice Templates	123
6.16.1	Invoices Management	123
6.16.2	Invoice Templates	125
6.16.3	Invoices Generation	135
6.17	Email Reports and Notifications	137
6.17.1	Email events	137
6.17.2	Initial template values and template variables	137
6.17.3	Password reset email template	137
6.17.4	New subscriber notification email template	138

6.17.5 Invoice email template . . . . .	139
6.17.6 Email templates management . . . . .	140
6.18 The Vertical Service Code Interface . . . . .	142
6.18.1 Configuration of Vertical Service Codes . . . . .	142
6.18.2 Voice Prompts for Vertical Service Code Configuration . . . . .	143
6.19 Handling WebRTC Clients . . . . .	144
6.20 XMPP and Instant Messaging . . . . .	145
<b>7 Customer Self-Care Interface and Menus</b>	<b>146</b>
7.1 The Customer Self-Care Web Interface . . . . .	146
7.1.1 Login Procedure . . . . .	146
7.1.2 Site Customization . . . . .	146
7.2 The Voicemail Menu . . . . .	152
<b>8 Billing Configuration</b>	<b>153</b>
8.1 Billing Profiles . . . . .	153
8.1.1 Creating Billing Profiles . . . . .	153
8.1.2 Creating Billing Fees . . . . .	155
8.1.3 Creating Off-Peak Times . . . . .	157
8.2 Fraud Detection and Locking . . . . .	159
8.2.1 Fraud Lock Levels . . . . .	160
8.3 Billing Data Export . . . . .	160
8.3.1 Glossary of Terms . . . . .	161
8.3.2 File Name Format . . . . .	161
8.3.3 File Format . . . . .	162
8.3.4 File Transfer . . . . .	174
<b>9 Provisioning REST API Interface</b>	<b>175</b>
9.1 API Workflows for Customer and Subscriber Management . . . . .	175
<b>10 Configuration Framework</b>	<b>181</b>

---

10.1 Configuration templates . . . . .	181
10.1.1 .tt2 and .customtt.tt2 files . . . . .	181
10.1.2 .prebuild and .postbuild files . . . . .	182
10.1.3 .services files . . . . .	183
10.2 config.yml, constants.yml and network.yml files . . . . .	184
10.3 ngcpcfg and its command line options . . . . .	184
10.3.1 apply . . . . .	184
10.3.2 build . . . . .	184
10.3.3 commit . . . . .	184
10.3.4 decrypt . . . . .	185
10.3.5 diff . . . . .	185
10.3.6 encrypt . . . . .	185
10.3.7 help . . . . .	185
10.3.8 initialise . . . . .	185
10.3.9 pull . . . . .	185
10.3.10push . . . . .	185
10.3.11services . . . . .	185
10.3.12status . . . . .	186
<b>11 Network Configuration</b>	<b>187</b>
11.1 General Structure . . . . .	187
11.1.1 Available Host Options . . . . .	187
11.1.2 Interface Parameters . . . . .	188
11.2 Advanced Network Configuration . . . . .	189
11.2.1 Extra SIP Sockets . . . . .	189
11.2.2 Extra SIP and RTP Sockets . . . . .	190
<b>12 Software Upgrade</b>	<b>192</b>
12.1 Release Notes . . . . .	192
12.2 Upgrade sip:provider CE from previous mr4.4/mr4.5 versions to mr4.5.7 LTS version . . . . .	192

---

12.3 Upgrade sip:provider CE from previous mr3.8 LTS version to mr4.5.7 LTS version . . . . .	194
<b>13 Backup, Recovery and Database Maintenance</b>	<b>197</b>
13.1 sip:provider CE Backup . . . . .	197
13.1.1 What data to back up . . . . .	197
13.2 Recovery . . . . .	197
13.3 Reset Database . . . . .	198
13.4 Accounting Data (CDR) Cleanup . . . . .	198
13.4.1 Cleanuptools Configuration . . . . .	198
13.4.2 Accounting Database Cleanup . . . . .	198
13.4.3 Exported CDR Cleanup . . . . .	201
<b>14 Platform Security, Performance and Troubleshooting</b>	<b>203</b>
14.1 Sipwise SSH access to sip:provider CE . . . . .	203
14.2 Firewalling . . . . .	203
14.3 Password management . . . . .	204
14.4 SSL certificates. . . . .	205
14.5 Securing your sip:provider CE against SIP attacks . . . . .	206
14.5.1 Denial of Service . . . . .	206
14.5.2 Bruteforcing SIP credentials . . . . .	207
14.6 Topology Hiding . . . . .	207
14.6.1 Introduction to Topology Hiding on NGCP . . . . .	207
14.6.2 Configuration of Topology Hiding . . . . .	208
14.6.3 Considerations for Topology Hiding . . . . .	208
14.7 System Requirements and Performance . . . . .	209
14.8 Troubleshooting . . . . .	211
14.8.1 Collecting call information from logs . . . . .	213
14.8.2 Collecting SIP traces . . . . .	214
<b>A Basic Call Flows</b>	<b>215</b>

---

A.1	General Call Setup	215
A.2	Endpoint Registration	216
A.3	Basic Call	219
A.4	Session Keep-Alive	220
A.5	Voicebox Calls	221
<b>B</b>	<b>NGCP configs overview</b>	<b>223</b>
B.1	config.yml Overview	223
B.1.1	apps	223
B.1.2	asterisk	223
B.1.3	autoprov	224
B.1.4	backuptools	225
B.1.5	cdrexport	226
B.1.6	checktools	226
B.1.7	cleanuptools	228
B.1.8	cluster_sets	229
B.1.9	database	230
B.1.10	faxserver	230
B.1.11	general	230
B.1.12	heartbeat	230
B.1.13	intercept	231
B.1.14	kamailio	231
B.1.15	mediator	236
B.1.16	nginx	236
B.1.17	ntp	236
B.1.18	ossbss	236
B.1.19	pbx (only with additional cloud PBX module installed)	238
B.1.20	prosody	238
B.1.21	pushd	239

B.1.22 qos	240
B.1.23 rate-o-mat	240
B.1.24 redis	240
B.1.25 reminder	240
B.1.26 rsyslog	241
B.1.27 rtpproxy	241
B.1.28 security	242
B.1.29 sems	242
B.1.30 snmpagent	244
B.1.31 sshd	244
B.1.32 voisniff	244
B.1.33 www_admin	245
B.2 constants.yml Overview	247
B.3 network.yml Overview	247
<b>C RTC:engine</b>	<b>249</b>
C.1 Overview	249
C.2 RTC:engine enabling	249
C.2.1 Enabling services via CLI	249
C.2.2 Enabling via Panel for resellers and subscribers	250
C.2.3 Create RTC:engine session	250
C.3 RTC:engine protocol details	251
C.3.1 Terminology	251
C.3.2 Messages	252
C.3.3 Account	254
C.3.4 Call	258
C.3.5 Session	264
<b>D NGCP Internals</b>	<b>266</b>
D.1 Pending reboot marker	266

---

- D.2 Redis id constants . . . . . 266
  - D.2.1 Redis monitoring keys . . . . . 267
- D.3 Enum preferences . . . . . 268

# 1 Introduction

## 1.1 About this Handbook

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

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

## 1.3 The Advantages of 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, Kamailio, SEMS, Asterisk, etc. as its core building blocks. These blocks are glued together using optimized and proven configurations and workflows and are complemented by functionality developed by Sipwise to provide fully-featured and easy to operate VoIP services.

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 workflows.



## 1.4 Who is the sip:provider CE for?

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 implementing 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.

## 1.5 Getting Help

### 1.5.1 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.

### 1.5.2 Commercial Support

If you need professional help setting up and maintaining the sip:provider CE, send an email to [support@sipwise.com](mailto: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](http://www.sipwise.com) for more information on commercial offerings.

## 2 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:

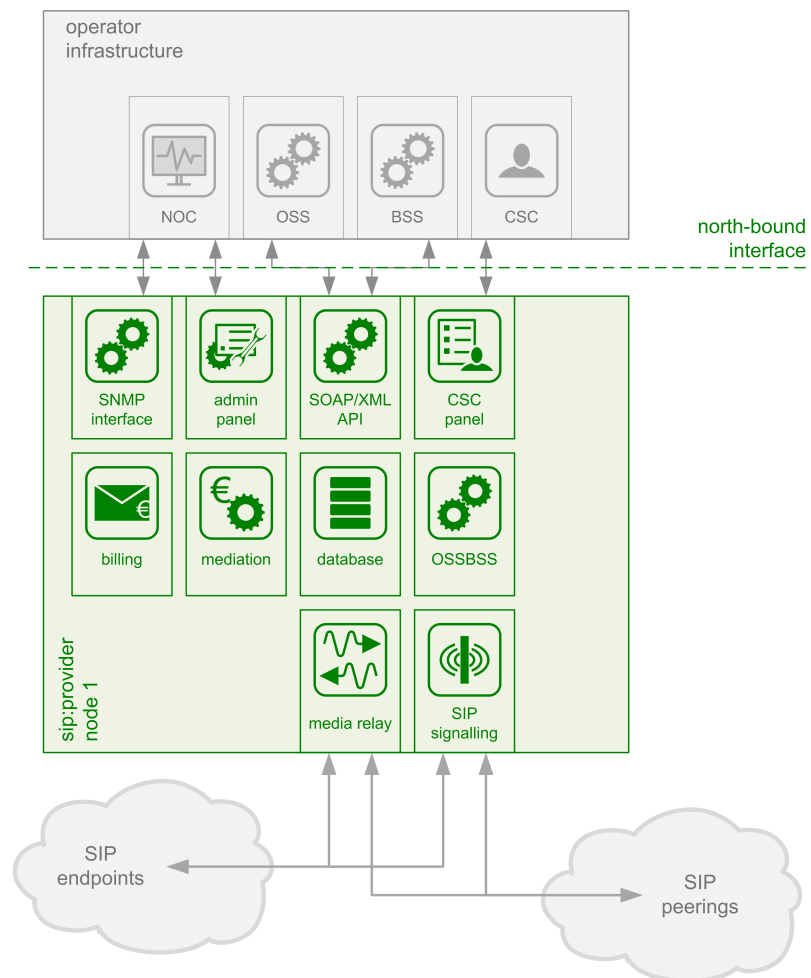


Figure 1: Architecture Overview

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.

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

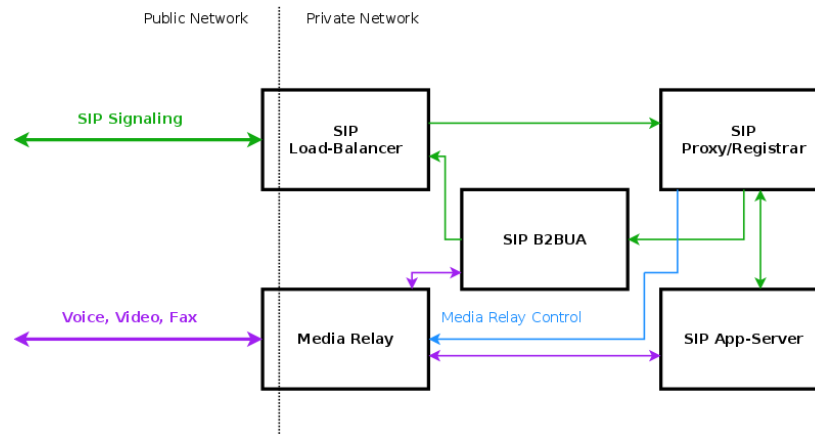


Figure 2: SIP and Media Relay Components

### 2.1.1 SIP Load-Balancer

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

---

### 2.1.2 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`.

---

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

The SIP B2BUA (also called SBC within the system) decouples the first call-leg (calling party to 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`

---

#### 2.1.4 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`

---

#### 2.1.5 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 even port numbers for the media data, and one pair on the next odd port numbers for metadata, e.g. RTCP in case of RTP streams. Each endpoint communicates with one dedicated port per media stream (opposed to some implementations which use one pair for both endpoints) to avoid issues in determining where to send a packet to. The rtpengine also sets the QoS/ToS/DSCP field of each IP packet it sends to a configured value, 184 (0xB8, *expedited forwarding*) by default.

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

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`

---

## 2.2 Redis Database

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

## 3 Initial Installation

### 3.1 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.

---

### 3.2 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.

### 3.3 Using the NGCP installer

#### 3.3.1 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.

---

##### 3.3.1.1 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.

---

### 3.3.2 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
```

## 3.4 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.

### 3.4.1 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-mr4.5.7 http://deb.sipwise.com/spce/images/sip_provider_CE_mr4.5.7 ↔  
  _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	<a href="https://127.0.0.1:1443/login/admin">https://127.0.0.1:1443/login/admin</a>	<a href="https://a.b.c.d:1443/login/admin">https://a.b.c.d:1443/login/admin</a> or <a href="https://x.y.z.p:1443/login/admin">https://x.y.z.p:1443/login/admin</a>	
New Customer self care interface	<a href="https://127.0.0.1:1443">https://127.0.0.1:1443</a>	<a href="https://a.b.c.d:1443">https://a.b.c.d:1443</a> or <a href="https://x.y.z.p:1443">https://x.y.z.p:1443</a>	new self-care interface based on powerful ngcp-panel framework
Old Customer self care interface	<a href="https://127.0.0.1:22443">https://127.0.0.1:22443</a>	<a href="https://a.b.c.d:443">https://a.b.c.d:443</a> or <a href="https://x.y.z.p:22443">https://x.y.z.p:22443</a>	will be removed in upcoming releases
Provisioning interfaces	<a href="https://127.0.0.1:2443">https://127.0.0.1:2443</a>	<a href="https://a.b.c.d:2443">https://a.b.c.d:2443</a> or <a href="https://x.y.z.p:2443">https://x.y.z.p:2443</a>	
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-mr4.5.7
```

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](#)).

### 3.4.2 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_mr4.5.7_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.

### 3.4.3 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.

### 3.4.4 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 3: Launch Amazon EC2 Instance for your region

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



Figure 4: 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):

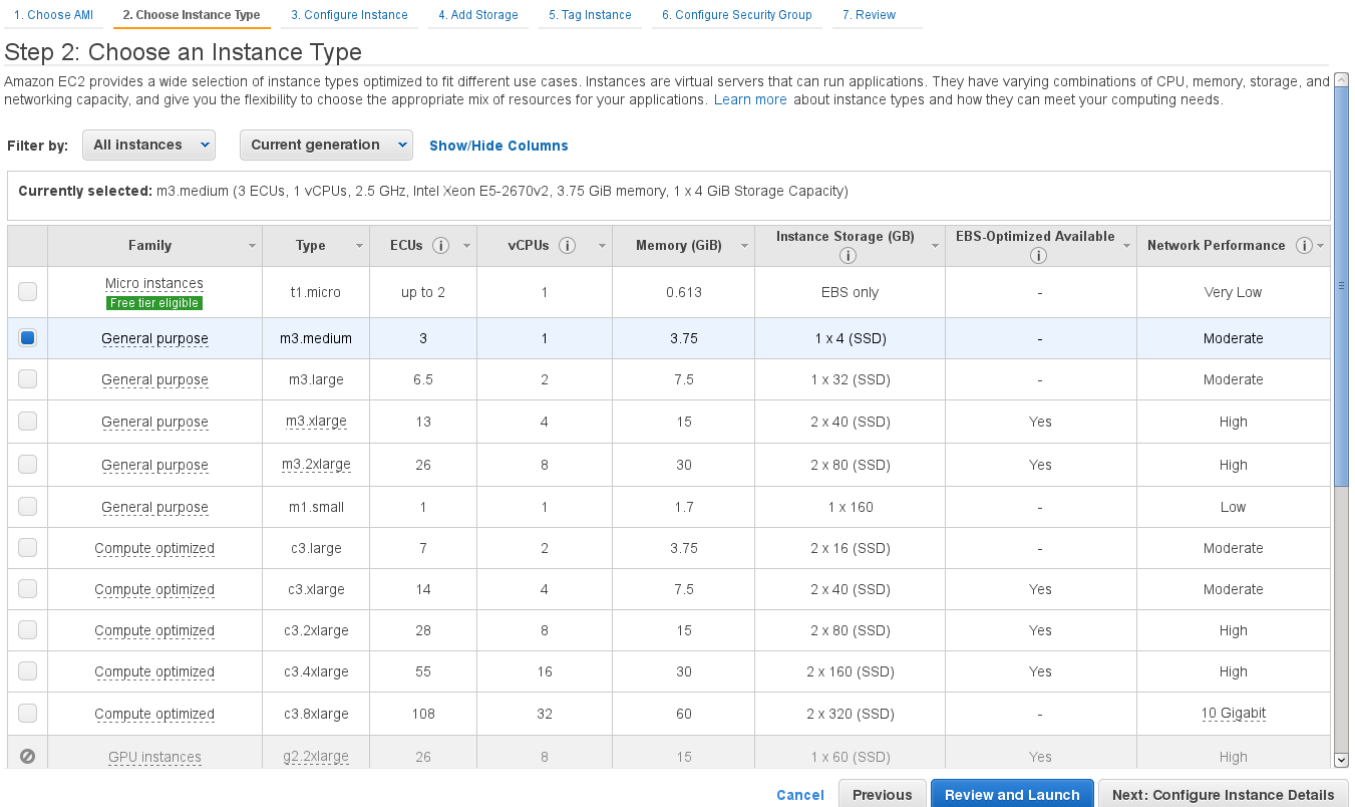


Figure 5: Choose Amazon EC2 instance

**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 14.7.

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 <sup>i</sup>	Protocol <sup>i</sup>	Port Range <sup>i</sup>	Source <sup>i</sup>
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 6: 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 7: 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 8: 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 9: 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 14 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/>).

---

## 4 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.

### 4.1 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 192.168.0.10
    netmask 255.255.255.0
    gateway 192.168.0.1
    dns-nameservers 8.8.8.8
    dns-search yourdomain.com
```

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

## 4.2 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.

---

## 4.3 Start Securing Your Server

During installation, the system user `cdrexpert` 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 cdrexpert
```

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.

## 4.4 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.

---

## 4.5 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.

### 4.5.1 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'
```

Ensure kernel module `dummy` will be loaded after the server reboot:

```
mkdir -p /etc/modprobe.d/
echo "options dummy numdummies=1" > /etc/modprobe.d/sipwise.conf

mkdir -p /etc/modules-load.d/sipwise.conf
echo "dummy" > /etc/modules-load.d/sipwise.conf
```

## 4.6 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.

## 5 VoIP Service Configuration Scenario

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 [Changing Administrator Password](#) [?simpara].

### 5.1 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' menu is open, with 'Customers' highlighted in orange. The dashboard contains four main sections: 'System Status' (All services running), 'Resellers' (9 Resellers), 'Billing' (6 Billing Profiles), and a fourth section with various metrics. Each section has a 'Configure' button.

System Status	Resellers	Billing	
All services running	9 Resellers	6 Billing Profiles	
Applications <b>Ok</b>	0 Domains	0.00 Peering Costs	
System <b>Ok</b>	0 Customers	0.00 Reseller Revenue	
Hardware <b>Ok</b>	0 Subscribers	0.00 Customer Revenue	
<a href="#">View Statistics</a>	<a href="#">Configure</a>	<a href="#">Configure</a>	<a href="#">Configure</a>

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. To the right of these buttons is a search input field labeled 'Search:'. Below the search field is a table with the following headers: '#', 'External #', 'Reseller', 'Contact Email', 'Billing Profile', and 'Status'. The table content is empty, displaying 'No data available in table'. Below the table, it says 'Showing 0 to 0 of 0 entries' and includes navigation arrows. At the bottom of the dashboard, there is a footer: '© 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 'Create Contract' dialog box in a web application. The dialog is titled 'Create Contract' and has a close button (X) in the top right corner. It is divided into two sections: 'Contact' and 'Billing Profile'. Each section has a search bar and a table of entries. The 'Contact' table has columns for '#', 'Reseller', 'First Name', 'Last Name', and 'Email'. The 'Billing Profile' table has columns for '#', 'Reseller', and 'Profile'. Below each table are navigation controls (back, forward, and a page indicator '1') and a 'Create' button. The 'Create Contact' button is highlighted with a red box. At the bottom right of the dialog is a 'Save' button.

Logged in as administrator Logout

sip:wise

### Create Contract

Contact Search:

#	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

Create Contact

Billing Profile Search:

#	Reseller	Profile
1	default	Default Billing Profile

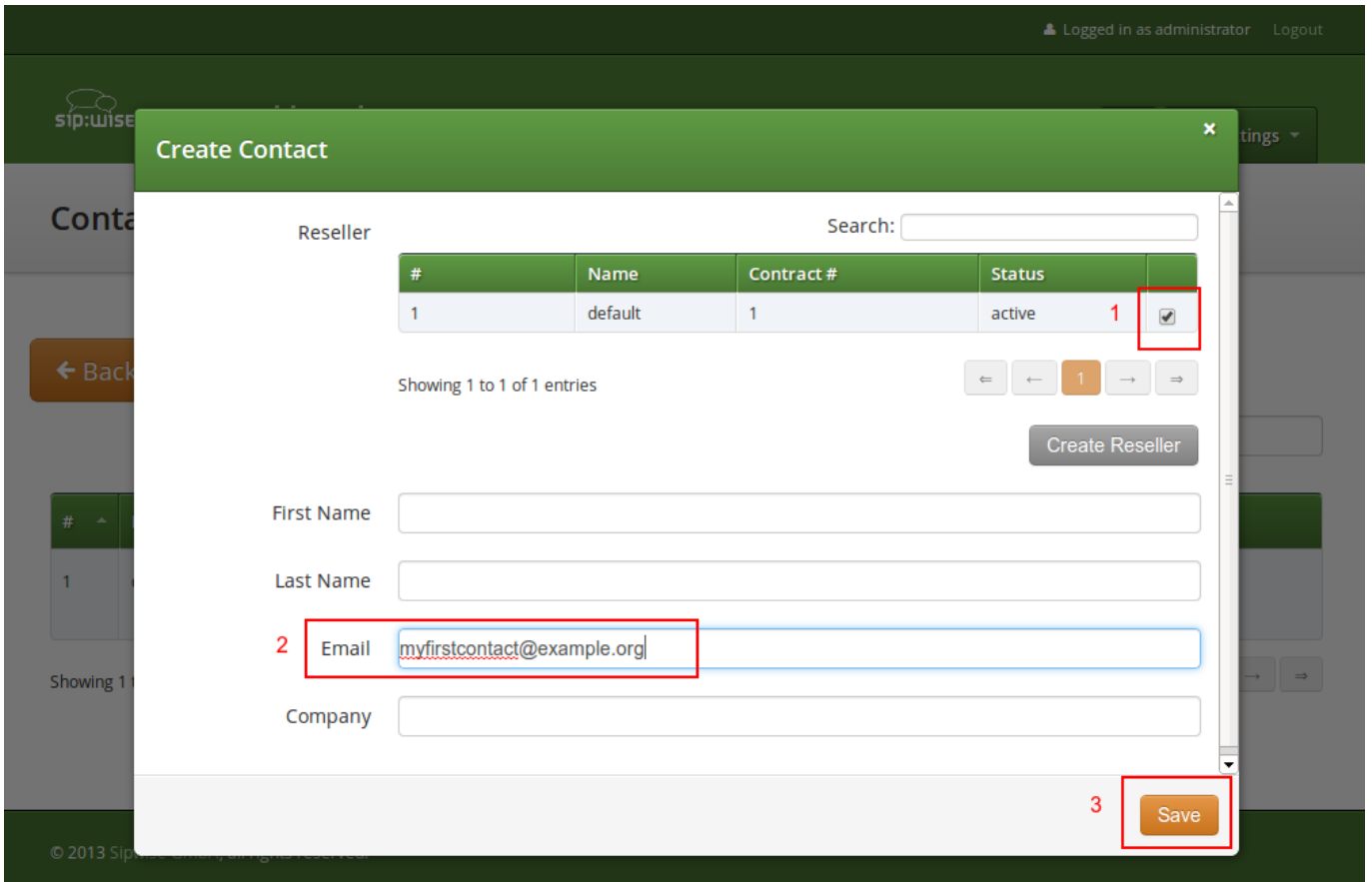
Showing 1 to 1 of 1 entries

Create Billing Profile

Save

© 2013 Sip

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 *Customer* 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.

Logged in as administrator Logout

sip:wise NGCP Dashboard

Home Settings

## Customers

← Back ★ Create Customer

Contract successfully created

Search:

#	External #	Reseller	Contact Email	Billing Profile	Status	
20		default	myfirstcontact@example.org	Default Billing Profile	active	<a href="#">Edit</a> <a href="#">Terminate</a> <a href="#">Details</a>

Showing 1 to 1 of 1 entries

## 5.2 Creating a Subscriber

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

Logged in as administrator Logout

sip:wise NGCP Dashboard

Home Settings

### Customer Details

Back Edit

Reseller

Contact Details

Billing Profiles

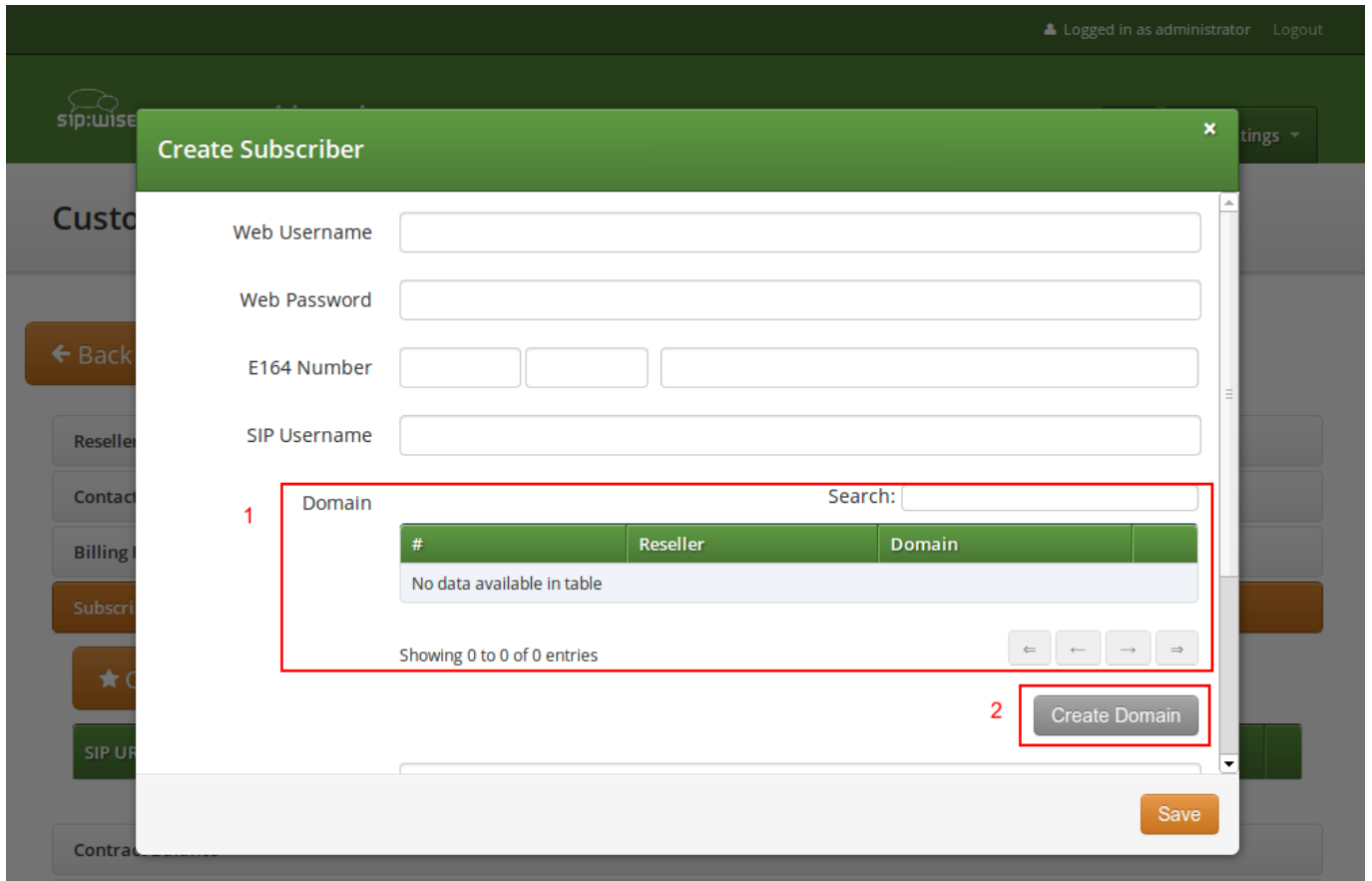
1 Subscribers

2 Create Subscriber

SIP URI	Primary Number	Registered Devices
---------	----------------	--------------------

Contract Balance

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 bar. 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 is a 'Create Reseller' button. Below that 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 5.6.

**Important**



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

- Web Password:** An empty text input field.
- E164 Number:** A field with three sub-inputs containing '43', '99', and '1001'.
- SIP Username:** A text input field containing 'testuser1'.
- Domain:** A table with columns '#', 'Reseller', and 'Domain'. It contains one entry: '# 6', 'Reseller default', 'Domain 1.2.3.4'. A search bar is above the table. A 'Create Domain' button is below the table.
- SIP Password:** A text input field containing 'mysecretpassword'.
- Save:** A button at the bottom right of the form.

Red boxes in the image highlight the E164 Number field (1), the SIP Username field (2), the SIP Password field (4), and the Save button (5).

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

---

**Tip**

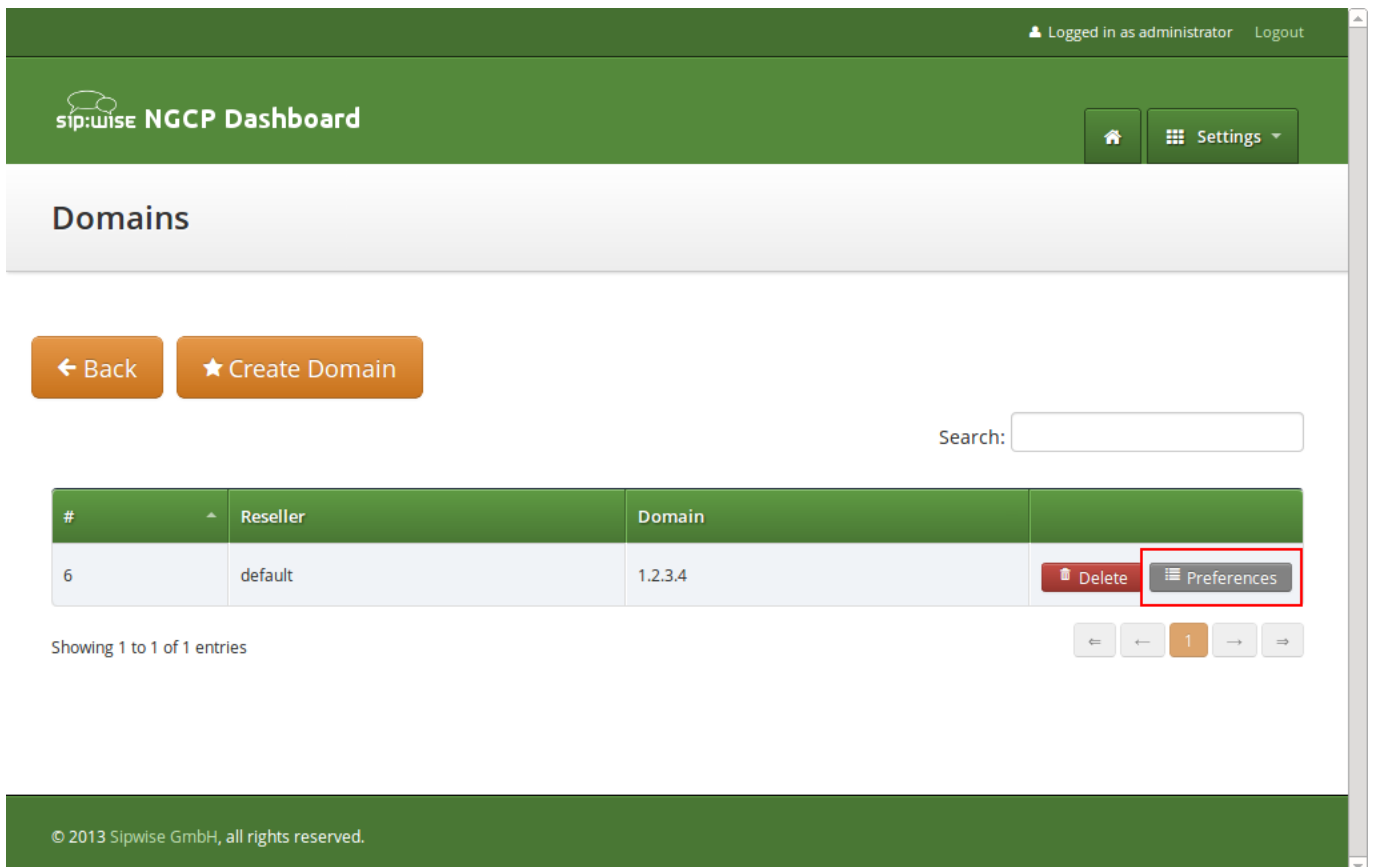
At this point, you're able to register your subscribers to the NGCP and place calls between these subscribers.

---

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

### 5.3 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 home icon and a "Settings" dropdown menu. Below the header, the "Domains" section is displayed. There are two orange buttons: "Back" and "Create Domain". A search bar is present. A table lists domains with columns for "#", "Reseller", and "Domain". The first entry is "# 6", "Reseller: default", "Domain: 1.2.3.4". To the right of this entry are "Delete" and "Preferences" buttons. The "Preferences" button is highlighted with a red box. Below the table, it says "Showing 1 to 1 of 1 entries" and there are navigation arrows. At the bottom, it says "© 2013 Sipwise GmbH, all rights reserved."

#	Reseller	Domain	
6	default	1.2.3.4	Delete Preferences

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

Here you can configure the following:

- for incoming calls - which SIP message headers to take numbers from
- for outgoing calls - where in the SIP messages to put certain numbers to



- for both - how these numbers are normalized to E164 format and vice versa

To assign a *Rewrite Rule Set* to a *Domain*, create a set first as described in Section 5.6, then assign it to the domain by editing the *rewrite\_rule\_set* preference.

## Domain "1.2.3.4" - Preferences

← Back

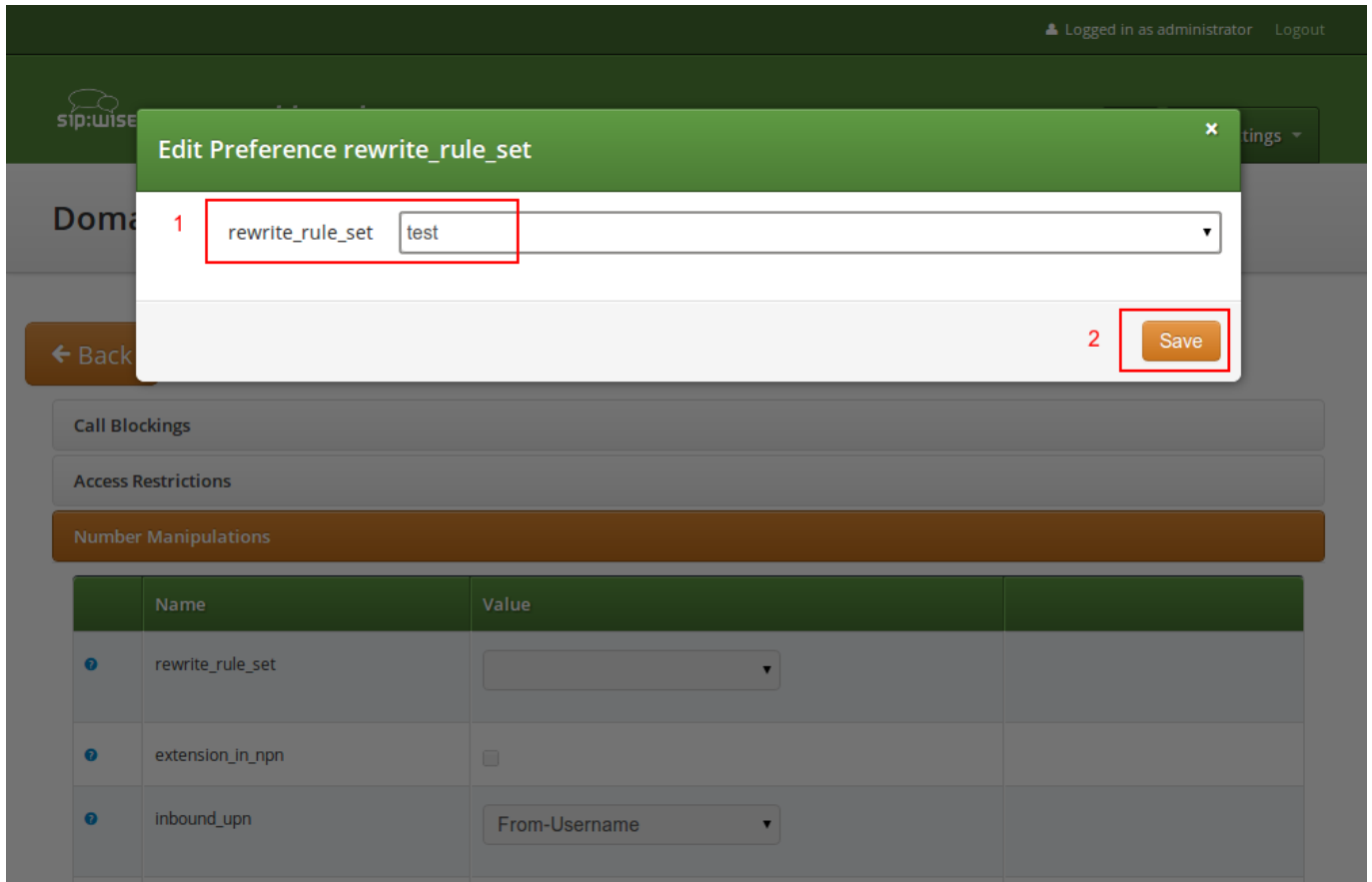
Call Blockings

Access Restrictions

1 **Number Manipulations**

	Name	Value	
i	rewrite_rule_set	<input type="text" value=""/>	2 <span style="background-color: #e67e22; color: white; padding: 2px 5px; border-radius: 3px; border: 1px solid #ccc; cursor: pointer;">✎ Edit</span>
i	extension_in_npn	<input type="checkbox"/>	
i	inbound_upn	<input type="text" value="From-Username"/>	
i	outbound_from_user	<input type="text" value="User-Provided-Number"/>	
i	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.

## 5.4 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 instead.
- You can override any user-provided number coming from the subscriber using the *user\_cli* preference.

**Note**

Subscribers preference *allowed\_clis* will be synchronized with subscribers primary number and aliases if *ossbss→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 if *ossbss→provisioning→auto\_sync\_cli* is set to **yes** in */etc/ngcp-config/config.yml*.

## 5.5 Creating Peerings

If you want to terminate calls at or allow calls from 3<sup>rd</sup> 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.

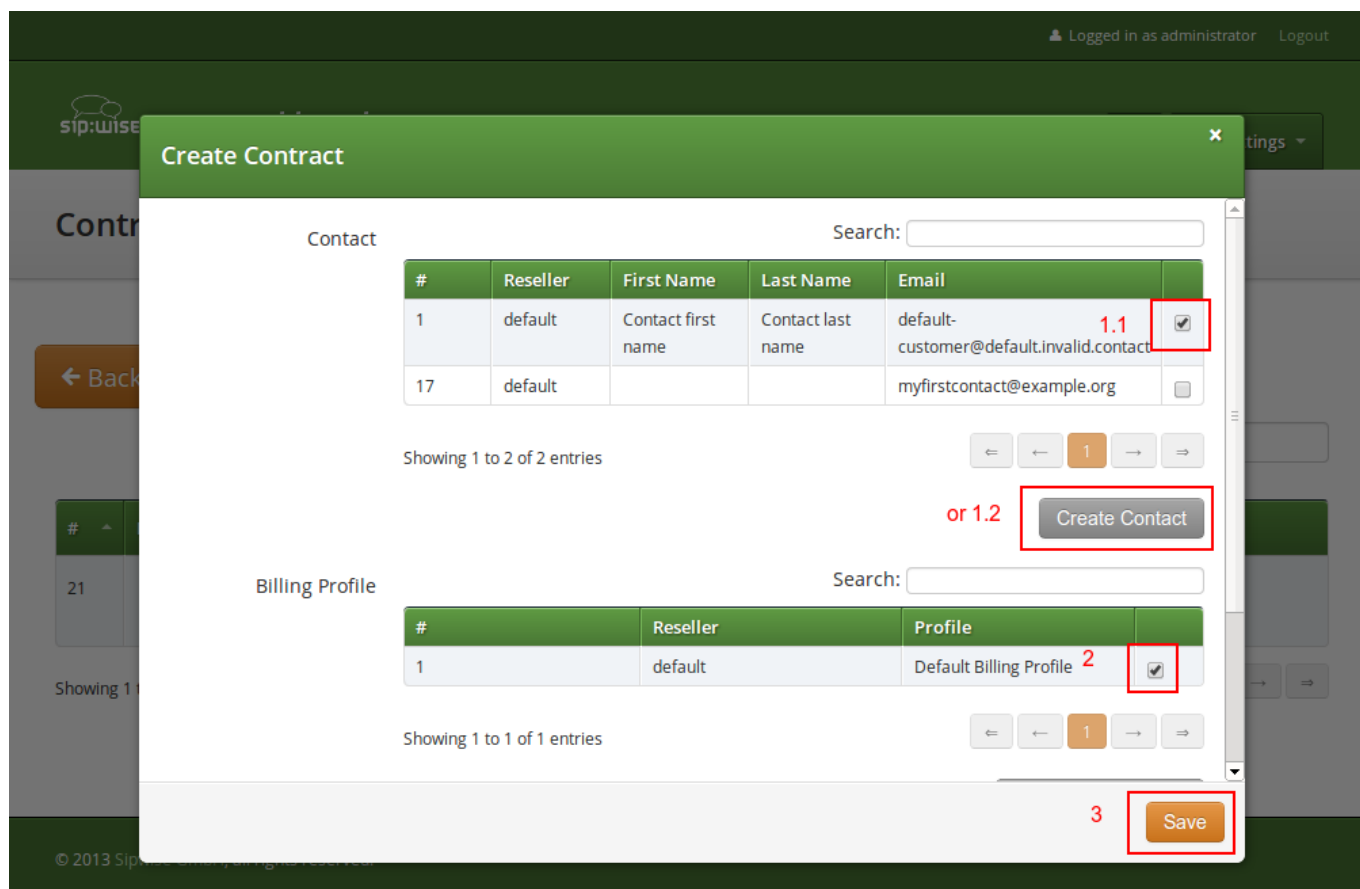
### 5.5.1 Creating Peering Groups

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.

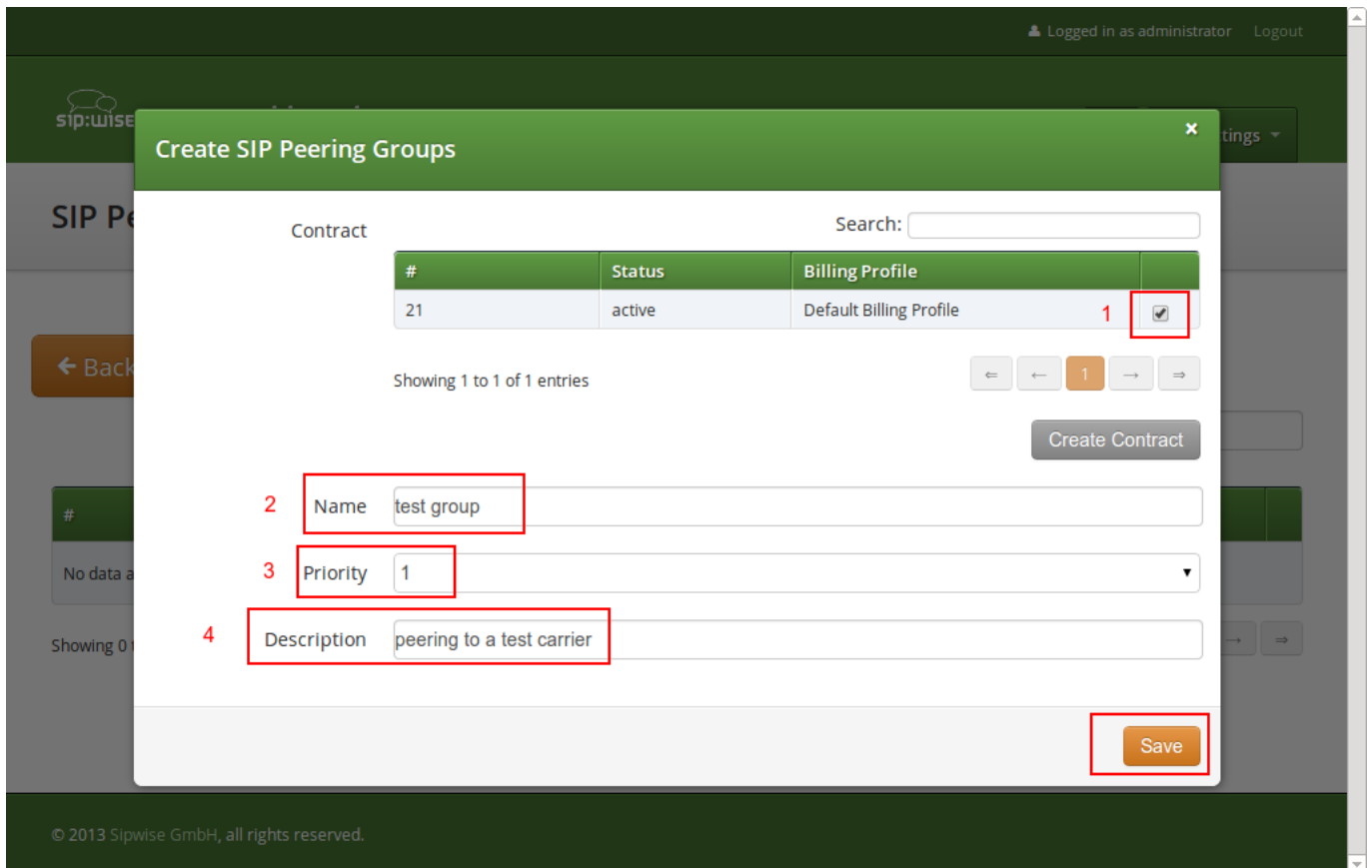
The screenshot shows the 'Create SIP Peering Groups' modal dialog in the Sipwise web interface. The dialog has a green header with the title 'Create SIP Peering Groups' and a close button. Below the header, there is a search bar labeled 'Contract' and a table with columns '#', 'Status', and 'Billing Profile'. The table is currently empty, displaying 'No data available in table' and 'Showing 0 to 0 of 0 entries'. A 'Create Contract' button is highlighted with a red box. Below the table, there are form fields for 'Name', 'Priority' (set to 1), and 'Description'. A 'Save' button is located at the bottom right of the dialog. The background shows the Sipwise interface with a user logged in as administrator.

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



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

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



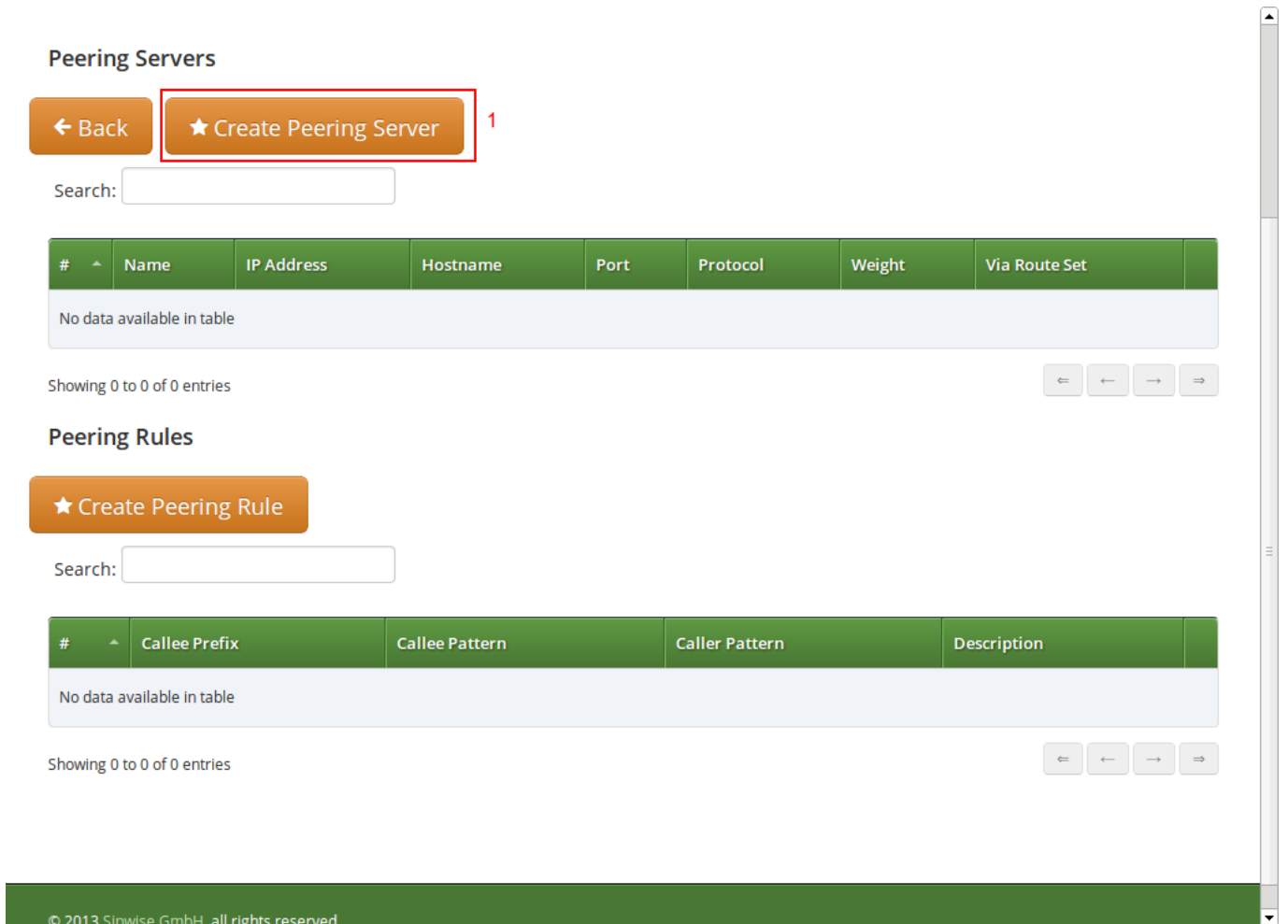
The *Priority* option defines which *Peering Group* to favor (Priority 1 gives the highest precedence) if two peering groups have peering rules matching an outbound call. *Peering Rules* are described below.

Then click *Save* to create the group.

### 5.5.2 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.



**Peering Servers**

[← Back](#) [★ Create Peering Server](#) <sup>1</sup>

Search:

#	Name	IP Address	Hostname	Port	Protocol	Weight	Via Route Set
No data available in table							

Showing 0 to 0 of 0 entries

**Peering Rules**

[★ Create Peering Rule](#)

Search:

#	Callee Prefix	Callee Pattern	Caller Pattern	Description
No data available in table				

Showing 0 to 0 of 0 entries

© 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' modal in the NGCP Dashboard. The form has the following fields:

- 1 Name: test-gw-1
- 2 IP Address: 2.3.4.5
- Hostname: (empty)
- 3 Port: 5060
- 4 Protocol: UDP
- 5 Weight: 1
- Via Route: None
- 6 Save button

Click **Save** to create the peering server.

### Tip

The *hostname* field for a peering server is optional. Usually, the IP address of the peer is used as the **domain** part of the Request URI. Fill in this field if a peer requires a particular hostname instead of the IP address. The IP address must always be given though as the request will always be sent to the specified 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**

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

**Peering Rules**

<sup>1</sup>

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 NGCP Dashboard. The modal has a green header with a close button. Below the header are four input fields: 'Callee prefix', 'Callee pattern', 'Caller pattern', and 'Description'. The 'Description' field contains the text 'Default Rule' and is highlighted with a red box and a red '1'. At the bottom right of the modal is a 'Save' button, which is also highlighted with a red box and a red '2'. The background shows the dashboard interface with a 'SIP Peering' section and a table of peering rules.

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 ⏪ ⏩

### 5.5.2.1 Routing Order Selection

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

- All peering groups that meet the following criteria configured in the outbound peering rule are added to the list of routes for a particular call:
  - Callee's username matches *callee prefix*
  - Callee's URI matches *callee pattern*
  - Caller's URI matches *caller pattern*
- When all matching peering groups are selected, they are ordered by *callee prefix* according to the **longest match basis** (sometimes referred to as the **longest pattern match** or **maximum pattern length match**). One or more peering group with longest *callee prefix* match will be given first positions on the list of routes.
- Peering groups with the same *callee prefix* length are further ordered by **Priority**. Peering group(s) with the higher priorities will occupy higher positions.

**Important**

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

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

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

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

**Important**

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

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

```
#!/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;
}
```

```
}

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

foreach ($sorted as $rand) {
    end($rand);
    $countts[key($rand)]++;
}

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

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

```
# 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 tried as a fallback. All the servers within the group are tried one after another until the call succeeds. If no more servers are left in the current peering group, the next group which matches the outbound peering rules is used.

### 5.5.3 Authenticating and Registering against Peering Servers

#### 5.5.3.1 Proxy-Authentication for outbound calls

If a peering server requires the sip:provider CE 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	<span>Edit</span> <span>Delete</span> <span>Preferences</span>

Showing 1 to 1 of 1 entries

←
←
1
→
→

### Peering Rules

★ Create Peering Rule

Search:

#	Callee Prefix	Callee Pattern	Callee Pattern	Description	
2				Default Rule	

Showing 1 to 1 of 1 entries

←
←
1
→
→

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`.

### 5.5.3.2 Registering at a Peering Server

Unfortunately, the credentials configured above are not yet automatically used to register the sip:provider CE 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'`.

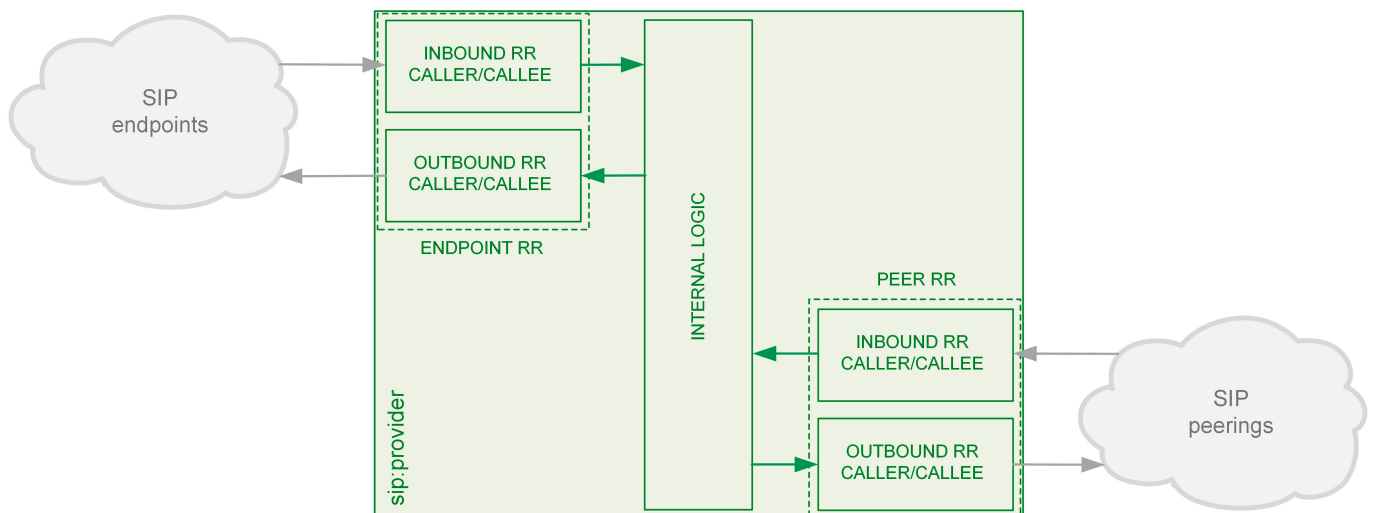


#### Important

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

## 5.6 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 of how to use them are also provided in the following sections:

- `${caller_cc}` : This is the value taken from the subscriber's preference CC value under Number Manipulation
- `${caller_ac}` : This is the value taken from the subscriber's preference AC value under Number Manipulation
- `${caller_emergency_cli}` : This is the value taken from the subscriber's preference emergency\_cli value under Number Manipulation
- `${caller_emergency_prefix}` : This is the value taken from the subscriber's preference emergency\_prefix value under Number Manipulation
- `${caller_emergency_suffix}` : This is the value taken from the subscriber's preference emergency\_suffix value under Number Manipulation

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 says 'Logged in as administrator' and 'Logout'. The dashboard header includes the sip:wise logo and 'NGCP Dashboard'. Below the header, there are navigation buttons for 'Home' and 'Settings'. 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. To the right of these buttons is a search input field labeled 'Search:'. Below the search field is a table with the following data:

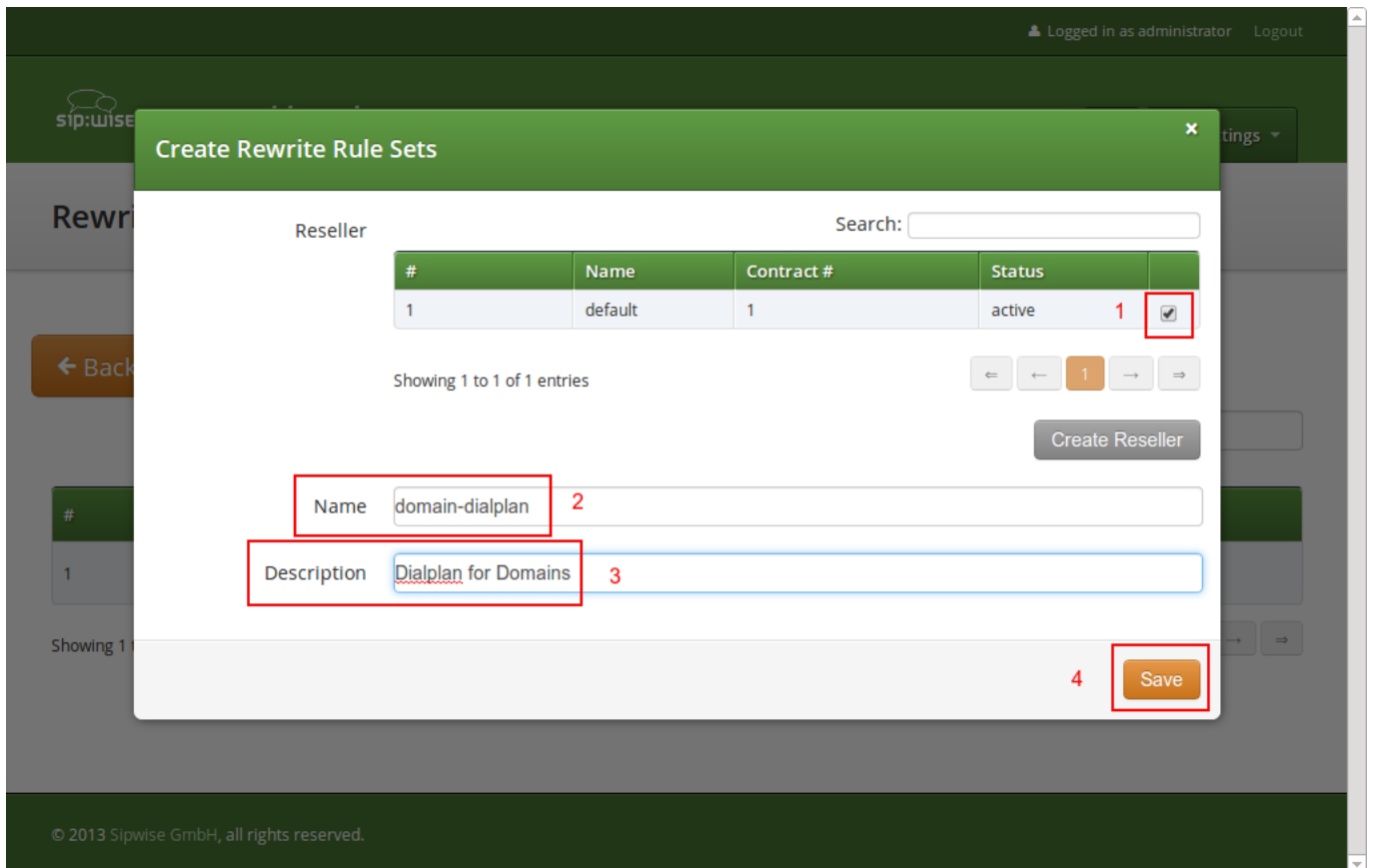
#	Reseller	Name	Description
1	default	defaultdom	Default Domain

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.





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	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Rules</a>

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.

#### 5.6.1 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 white body. It contains several fields: 'Match pattern' with the value '^([00|+)([1-9][0-9]+)\$' and a red box around it labeled '1'; 'Replacement Pattern' with the value '\2' and a red box around it labeled '2'; 'Description' with the value 'International to E.164' and a red box around it labeled '3'; 'Direction' with the value 'Inbound' and a red box around it labeled '4'; 'Field' with the value 'Caller' and a red box around it labeled '5'; and a 'Save' button at the bottom right with a red box around it labeled '6'. The background shows a blurred interface with a 'Logged in as administrator' status and a 'Logout' link.

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	↑ ↓ <input type="checkbox"/> <input type="checkbox"/>	^(00 \+)([1-9][0-9]+)\$	\2 International to E.164
	↑ ↓ <input type="checkbox"/> <input type="checkbox"/> 2	^0([1-9][0-9]+)\$	\$(caller_cc)\1 National to E.164
	↑ ↓ <input type="checkbox"/> <input type="checkbox"/>	^([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

### 5.6.2 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.

---

### 5.6.3 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).

---

### 5.6.4 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

### 5.6.5 Emergency Number Handling

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

- *Simple* emergency number handling: inbound rewrite rules append an emergency tag to the called number, this will be recognised by NGCP's call routing logic and the call is routed directly to a peer. Please read the next section for details of simple emergency number handling.
- An *emergency number mapping* is applied: a dedicated emergency number mapping database is consulted in order to obtain the most appropriate routing number of emergency services. This logic ensures that the caller will contact the geographically closest emergency service. Please visit the [Emergency Mapping](#) Section 6.5 section of the handbook for more details.

### **5.6.5.1 Simple Emergency Number Handling Overview**

The overview of emergency call processing is as follows:

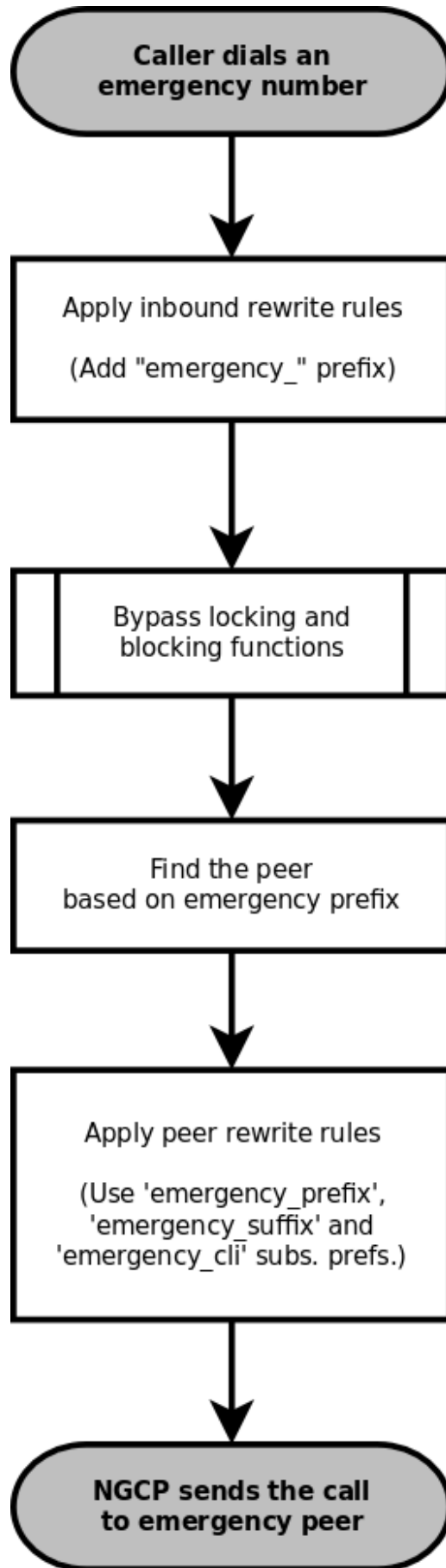


Figure 10: Simple Emergency Call Handling



Configuring Emergency Numbers is also done via Rewrite Rules.

### 5.6.5.2 Tagging Inbound Emergency Calls

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

#### TAG AN EMERGENCY CALL

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

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

### 5.6.5.3 Normalize Emergency Calls for Peers

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

#### NORMALIZE EMERGENCY CALL FOR PEER

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

### 5.6.6 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 'Number Manipulations' section is highlighted in orange. It contains a table with the following data:

Name	Value	
rewrite_rule_set	defaultdom	Edit
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.

### 5.6.7 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 5.6 (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.

### 5.6.8 Call Routing Verification

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

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

- What is considered during the test:
  - subscriber preferences: `cli` and `allowed_clis`
  - domain / subscriber / peer rewrite rules
- What is not taken into account during the test:
  - other subscriber or peer preferences
  - LNP (Local Number Portability) lookup on called numbers; LNP rewrite rules

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

### **Expected input data**

- `Caller number/uri`: 2 formats are accepted in this field:
  - A simple **phone number** in international (00431 . . , +431 . . ) or E.164 (431 . . ) format.
  - A SIP **URI** in `username@domain` format (without adding "sip:" at the beginning).
- `Callee number/uri`: The same applies as for `Caller number/uri`.
- `Caller Type`: Select `Subscriber` or `Peer`, depending on the source of the call.
- `Caller Subscriber` or `Caller Peer`: Optionally, you can select the subscriber or peer explicitly. Without the explicit selection, however, the *Call Routing Verification* tool is able to find the caller in the database, based on the provided number / URI.
- `Caller RWR Override`, `Callee RWR Override`, `Callee Peer Override`: The caller / callee rewrite rules and peer selection rules defined in domain, subscriber and peer preferences are used for call processing by default. But you can also override them by explicitly selecting another rewrite or peer selection rule.

### **Examples**

1. Using only phone numbers and explicit subscriber selection

- Input Data:

### Call Routing Verification

[← Back](#) [Expand Groups](#)

Caller number/url:   
Callee number/url:

Caller Type:  Subscriber  Peer

Caller Subscriber Search:

#	Username	Domain	UUID	Number	
295	43993002	10.15.18.227	51e32173-c8a9-44f1-af30-a1ed431eb2bf		<input checked="" type="checkbox"/>
297	43993003	10.15.18.227	6feb9ea-21c0-4f55-8828-80d546b8998f		<input type="checkbox"/>
299	43993004	10.15.18.227	3543a26e-861b-459f-a348-a8ef3e1e9eab		<input type="checkbox"/>
301	43993005	10.15.18.227	355773d2-1c08-475c-8858-eaf75bb58c73		<input type="checkbox"/>

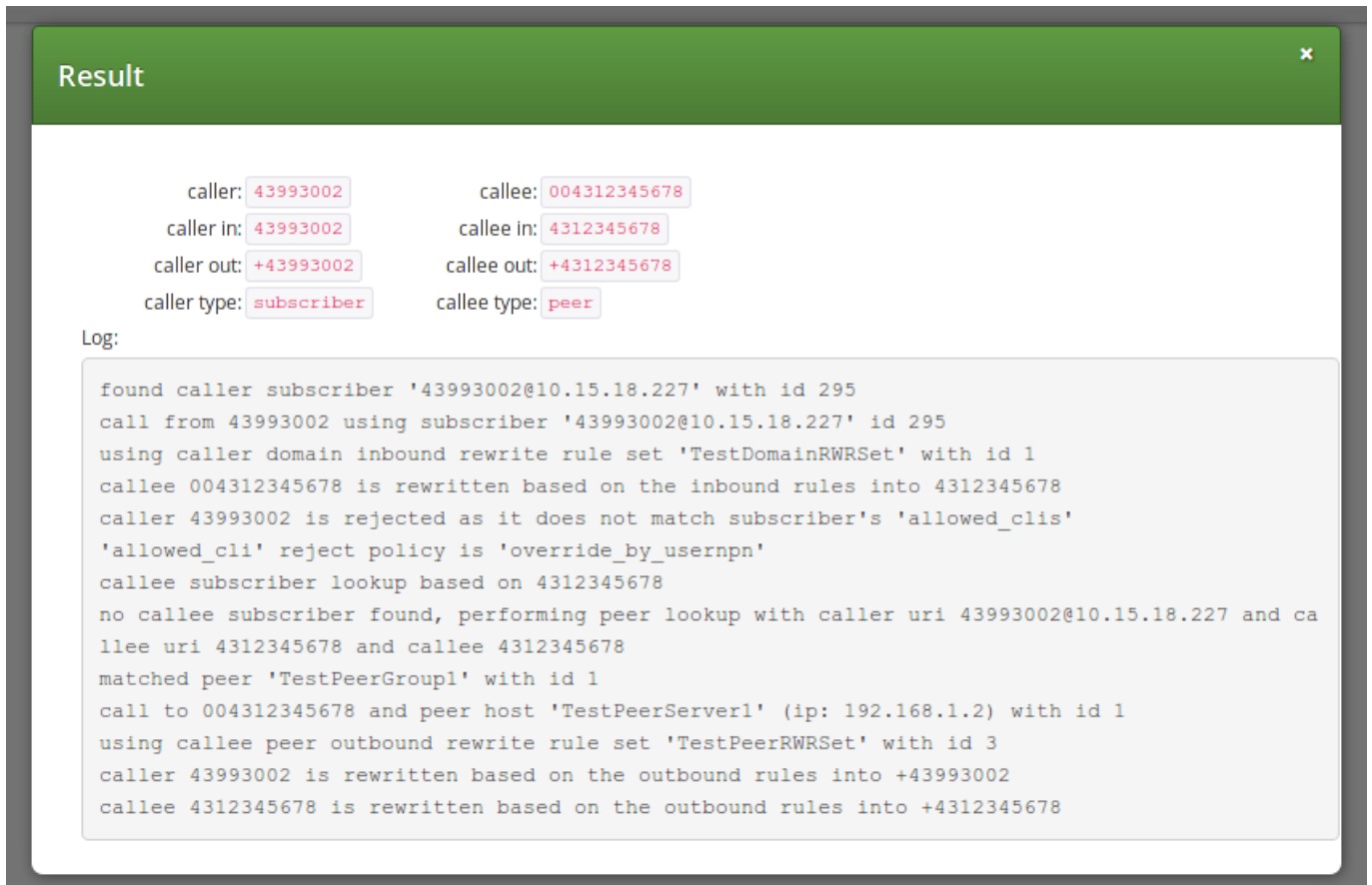
Showing 1 to 4 of 8 entries (filtered from 56 total entries) ← 1 2 →

Caller Rewrite Rules Override  
Callee Rewrite Rules Override  
Callee Peer Override

[Verify](#)

Figure 11: Call Routing Verif. - Only Numbers - Input

- Result:



The screenshot shows a window titled "Result" with a green header and a close button. The main content area displays call routing details in two columns:

caller:	43993002	callee:	004312345678
caller in:	43993002	callee in:	4312345678
caller out:	+43993002	callee out:	+4312345678
caller type:	subscriber	callee type:	peer

Below the details is a "Log:" section containing the following text:

```
found caller subscriber '43993002@10.15.18.227' with id 295
call from 43993002 using subscriber '43993002@10.15.18.227' id 295
using caller domain inbound rewrite rule set 'TestDomainRWRSet' with id 1
callee 004312345678 is rewritten based on the inbound rules into 4312345678
caller 43993002 is rejected as it does not match subscriber's 'allowed_clis'
'allowed_cli' reject policy is 'override_by_usernpn'
callee subscriber lookup based on 4312345678
no callee subscriber found, performing peer lookup with caller uri 43993002@10.15.18.227 and ca
llee uri 4312345678 and callee 4312345678
matched peer 'TestPeerGroup1' with id 1
call to 004312345678 and peer host 'TestPeerServer1' (ip: 192.168.1.2) with id 1
using callee peer outbound rewrite rule set 'TestPeerRWRSet' with id 3
caller 43993002 is rewritten based on the outbound rules into +43993002
callee 4312345678 is rewritten based on the outbound rules into +4312345678
```

Figure 12: Call Routing Verif. - Only Numbers - Result

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

- Input Data:

### Call Routing Verification

[← Back](#) [Expand Groups](#)

Caller number/uri:

Callee number/uri:

Caller Type:  Subscriber  Peer

Caller Subscriber Search:

#	Username	Domain	UUID	Number	
295	43993002	10.15.18.227	51e32173-c8a9-44f1-af30-a1ed431eb2bf		<input type="checkbox"/>
297	43993003	10.15.18.227	6feb9ea-21c0-4f55-8828-80d546b8998f		<input type="checkbox"/>
299	43993004	10.15.18.227	3543a26e-861b-459f-a348-a8ef3e1e9eab		<input type="checkbox"/>
301	43993005	10.15.18.227	355773d2-1c08-475c-8858-eaf75bb58c73		<input type="checkbox"/>

Showing 1 to 4 of 8 entries (filtered from 56 total entries) ◀ 1 2 ▶

Caller Rewrite Rules Override

Callee Rewrite Rules Override

Callee Peer Override

[Verify](#)

Figure 13: Call Routing Verif. - Number and URI - Input

- Result:

### Result

caller:	43993003	callee:	+431555666
caller in:	43993003	callee in:	431555666
caller out:	+43993003	callee out:	+431555666
caller type:	subscriber	callee type:	peer

Log:

```
no caller subscriber/peer was specified, using subscriber lookup based on caller 43993003@10.15.18.227
found caller subscriber '43993003@10.15.18.227' with id 297
call from 43993003 using subscriber '43993003@10.15.18.227' id 297
using caller domain inbound rewrite rule set 'TestDomainRWRSet' with id 1
callee +431555666 is rewritten based on the inbound rules into 431555666
caller 43993003 is rejected as it does not match subscriber's 'allowed_clis'
'allowed_cli' reject policy is 'override_by_usernpn'
callee subscriber lookup based on 431555666
no callee subscriber found, performing peer lookup with caller uri 43993003@10.15.18.227 and ca
llee uri 431555666 and callee 431555666
matched peer 'TestPeerGroup1' with id 1
call to +431555666 and peer host 'TestPeerServer1' (ip: 192.168.1.2) with id 1
using callee peer outbound rewrite rule set 'TestPeerRWRSet' with id 3
caller 43993003 is rewritten based on the outbound rules into +43993003
callee 431555666 is rewritten based on the outbound rules into +431555666
```

Figure 14: Call Routing Verif. - Number and URI - Result

## 6 Features

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

### 6.1 Managing System Administrators

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

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

There is a single administrator, whose account is enabled by default and who belongs to the *default reseller*. This user is the *superuser* of the NGCP administrative web interface (the so-called "admin panel"), and he has the right to modify administrators of other *Resellers* as well.

#### 6.1.1 Configuring Administrators

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

The screenshot shows the 'Administrators' management page. At the top, there are two buttons: '← Back' and '★ Create Administrator'. The 'Create Administrator' button is circled in red. Below the buttons, a light blue message bar states 'Administrator successfully updated'. Underneath, there is a search bar and a 'Show 5 entries' dropdown. The main content is a table with columns: #, Reseller, Login, Master, Active, Read Only, Show Passwords, Show CDRs, Show Billing Info, and Lawful Intercept. The table contains two entries. The second entry, for 'Demo Reseller', has an 'Edit' button circled in red, along with 'Delete' and 'API key' buttons. At the bottom, it says 'Showing 1 to 2 of 2 entries' and has pagination controls.

#	Reseller	Login	Master	Active	Read Only	Show Passwords	Show CDRs	Show Billing Info	Lawful Intercept
1	default	administrator	1	1	0	1	1	1	1
3	Demo Reseller	demoadmin	1	1	0	1	1	0	0

Figure 15: List of System Administrators

You have 2 options:

- If you'd like to **create** a new administrator user press *Create Administrator* button.



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

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

**Edit Administrator** [X]

Reseller Search:

#	Name	Contract #	Status	
16	Demo Reseller	200	active	<input checked="" type="checkbox"/>
1	default	1	active	<input type="checkbox"/>
		137	active	<input type="checkbox"/>

Showing 1 to 3 of 3 entries

← ← 1 → →

Login:

Password:

Is superuser:

Figure 16: Generic System Administrator Attributes

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

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

### 6.1.2 Access Rights of Administrators

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

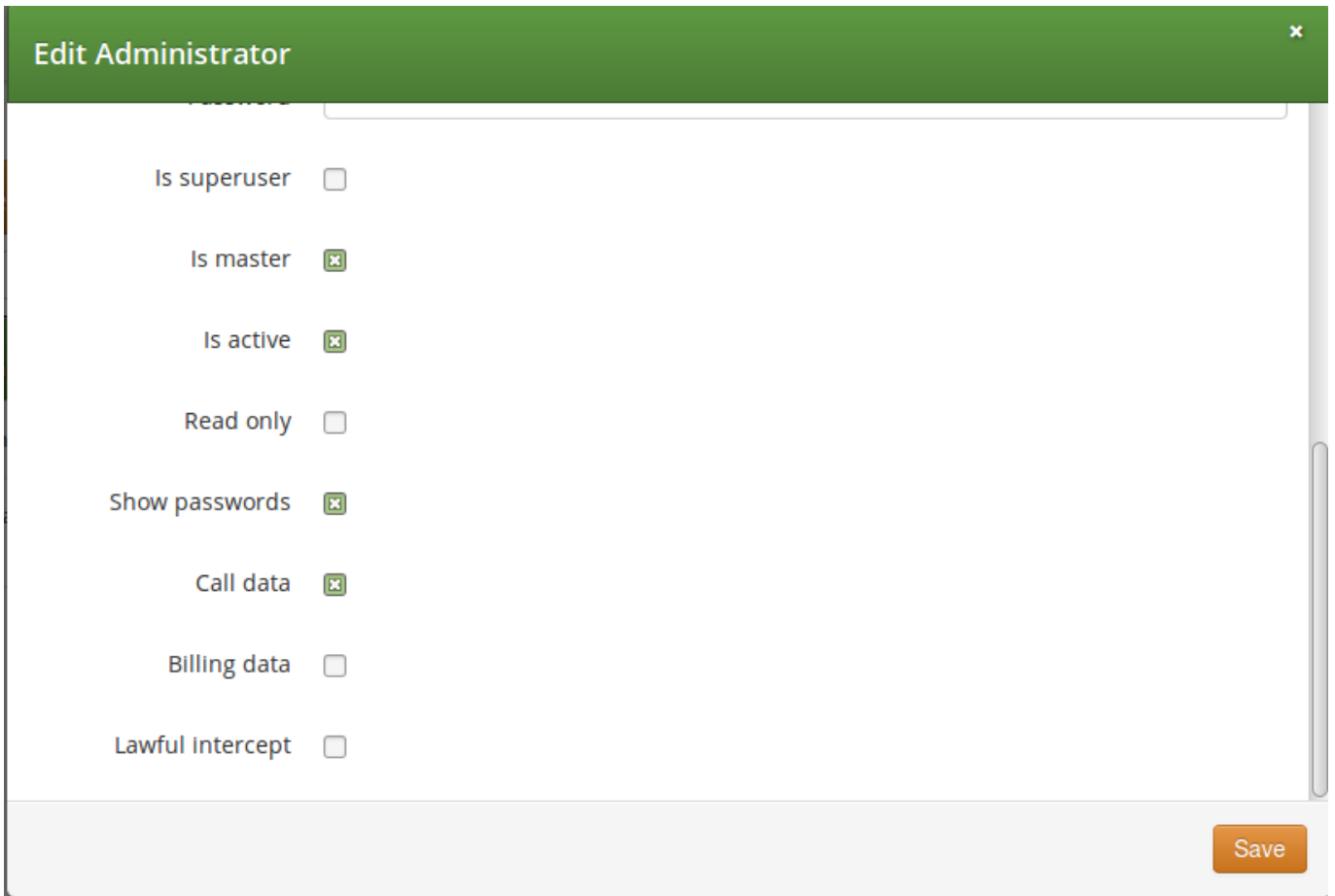


Figure 17: Access Rights of System Administrators

Table 2: Access Rights of System Administrators

Label in admin list	Access Right	Description
<i>not shown</i>	Is superuser	The user is allowed to modify data on Reseller level and — among others — is able to modify administrators of other resellers. There should be only 1 user on Sipwise NGCP with this privilege.
Master	Is master	The user is allowed to create, delete or modify other Admins who belong to the same Reseller.
Active	Is active	The user account is active, i.e. the admin user can login on the web panel or authenticate himself on REST API; otherwise user authentication will fail.

Table 2: (continued)

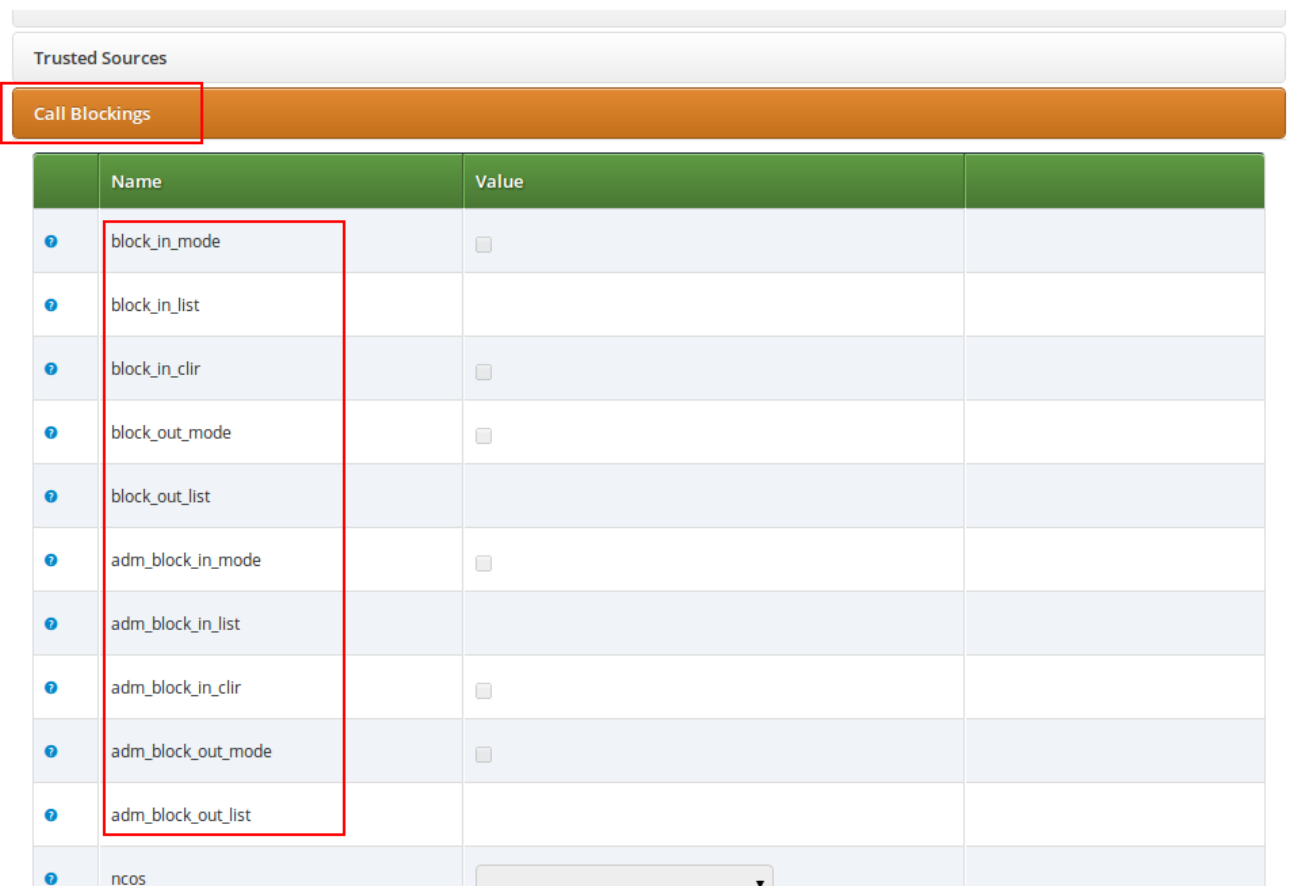
Label in admin list	Access Right	Description
Read Only	Read only	<p>The user will only be able to list various data but is not allowed to modify anything.</p> <ul style="list-style-type: none"> <li>For the <b>web interface</b> this means that <i>Create...</i> and <i>Edit</i> buttons will be hidden or disabled.</li> <li>For the <b>REST API</b> this means that only <code>GET</code>, <code>HEAD</code>, <code>OPTIONS</code> HTTP request methods are accepted, and NGCP will reject those targeting data modification: <code>PUT</code>, <code>PATCH</code>, <code>POST</code>, <code>DELETE</code>.</li> </ul>
Show Passwords	Show passwords	<p>The user sees subscriber passwords (in plain text) on the web interface.</p> <hr/> <p><b>Note</b></p> <p>Admin panel user passwords are stored in an unreadable way (cryptographic hash digest) in the database, while subscriber passwords are basically always stored in plain text. The latter happens on purpose, e.g. to make subscriber data migration possible.</p> <hr/>
Show CDRs	Call data	<p>This privilege has effect on 2 items that will be displayed on admin panel of NGCP, when <i>Subscriber</i> → <i>Details</i> is selected:</p> <ol style="list-style-type: none"> <li>1. <i>PBX Groups</i> list</li> <li>2. <i>Captured Dialogs</i> list</li> </ol>
Show Billing Info	Billing data	<p>Some REST API resources that are related to billing are disabled: HTTP requests on <code>/api/vouchers</code>, <code>/api/topupcash</code> and <code>/api/topupvoucher</code> resources are rejected.</p>
Lawful Intercept	Lawful intercept	<p>If the privilege is selected then the REST API for interceptions (that is: <code>/api/interceptions</code>) is enabled; if the privilege is not selected then the interceptions API is disabled.</p> <hr/> <p><b>Note</b></p> <p>This means that besides enabling LI in <code>config.yml</code> configuration file one also needs to enable the API via the LI privilege of an administrator user, so that NGCP can really provide LI service.</p> <hr/>

## 6.2 Access Control for SIP Calls

There are two different methods to provide fine-grained call admission control to both subscribers and admins. One is *Block Lists*, where you can define which numbers or patterns can be called from a subscriber to the outbound direction and which numbers or patterns are allowed to call a subscriber in the inbound direction. The other is *NCOS Levels*, where the admin predefines rules for outbound calls, which are grouped in certain levels. The subscriber can then just choose the level, or the admin can restrict a subscriber 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 sections describe these features in detail.

### 6.2.1 Block Lists

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



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

### 6.2.1.1 Block Modes

Block lists can either be *whitelists* or *blacklists* and are controlled by the User Preferences *block\_in\_mode*, *block\_out\_mode* and their administrative counterparts.

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

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

### 6.2.1.2 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.

The screenshot shows the 'Edit Preference block\_out\_list' dialog in the sip:wise interface. The dialog has a green header with the title 'Edit Preference block\_out\_list' and a close button (X) in a red box. Below the header, there are two input fields: the first contains '1\*' and has a red box around it, with a '1' next to it; the second contains '431234567' and has a red box around it, with a '2' next to it. To the right of these fields are two trash icons and an 'Add' button. Below the dialog, the main interface shows a sidebar with 'Call Blockings' selected, and a table with columns 'Name' and 'Value'.

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

### 6.2.1.3 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.

### 6.2.2 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.

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

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.

#### 6.2.2.1 Creating NCOS Levels

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

Logged in as administrator Logout

sip:wise NGCP Dashboard

Home Settings

## NCOS Levels

← Back **★ Create NCOS Level**

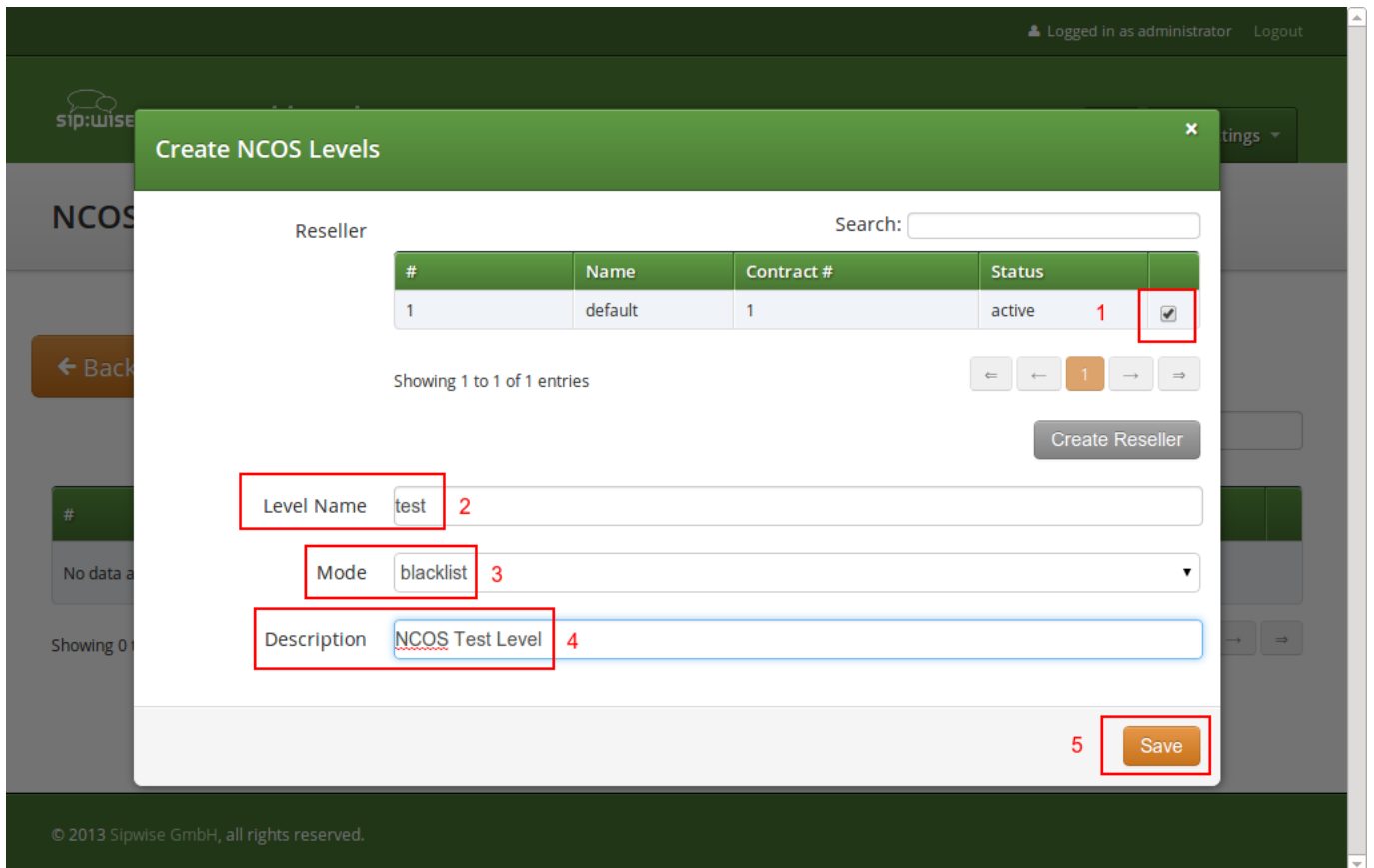
Search:

#	Reseller	Level Name	Mode	Description
No data available in table				

Showing 0 to 0 of 0 entries

© 2013 Sipwise GmbH, all rights reserved.

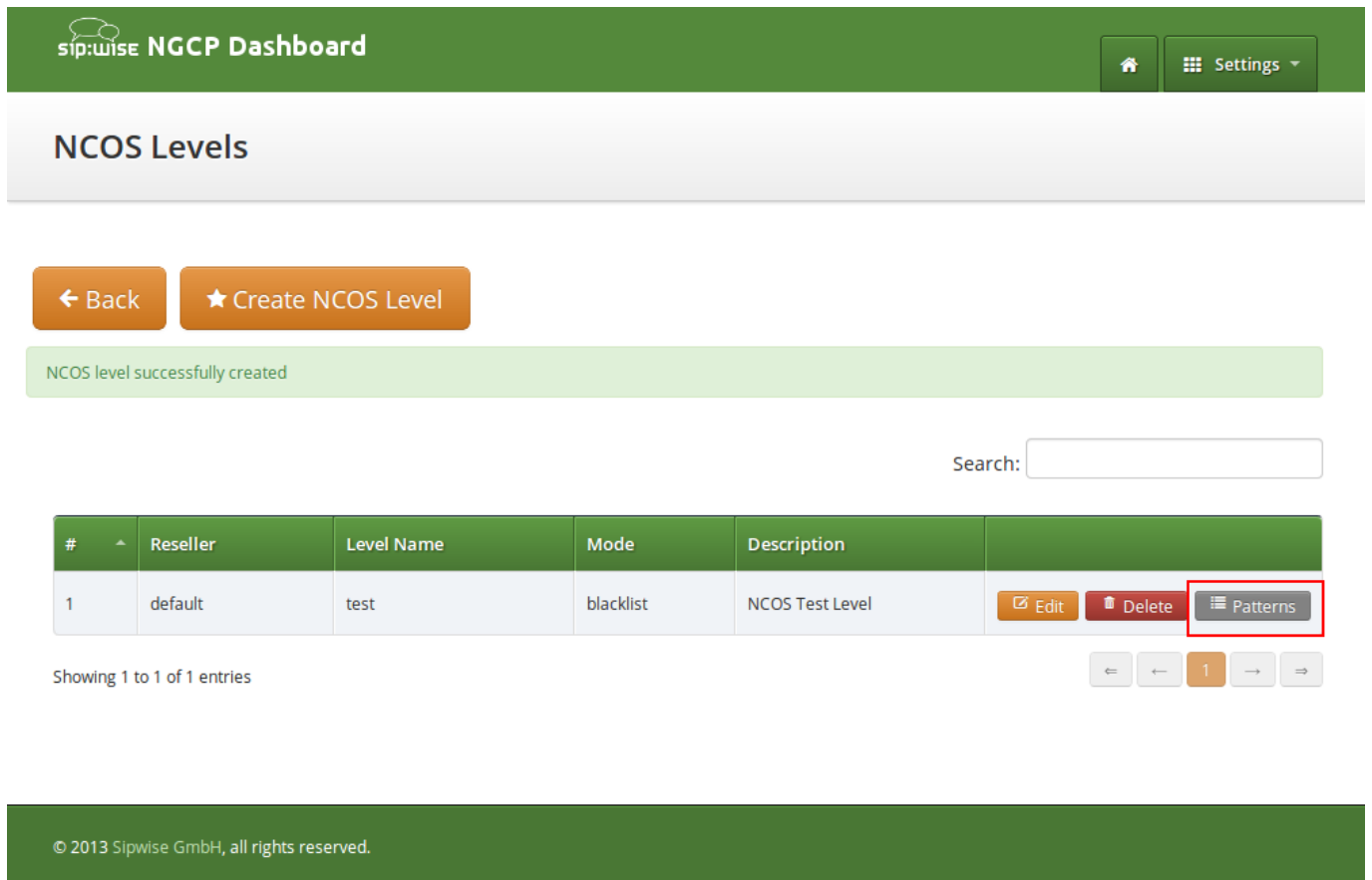
Select a reseller, enter a name, select the mode and add a description, then click the Save button.



### 6.2.2.2 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	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Patterns</a>

Showing 1 to 1 of 1 entries

© 2013 Sipwise GmbH, all rights reserved.

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

- **NCOS Number Patterns:** here you can define number patterns that will be matched against the called number and allowed or blocked, depending on whitelist / blacklist mode. The patterns are regular expressions.
- **NCOS LNP Carriers:** here you can select predefined *LNP Carriers* that will be allowed (whitelist mode) or prohibited (blacklist mode) to route calls to them. (See Section 6.4.1 in the handbook for the description of LNP functionality)

**NCOS Number Patterns**

← Back   ★ Create Pattern Entry

NCOS pattern successfully created

Show 5 entries   Search:

#	Pattern	Description
1	^439	Austrian Premium Numbers

Showing 1 to 1 of 1 entries

Include local area code  
 Intra PBX Calls within same customer

✎ Edit

**NCOS LNP Carriers**

★ Create LNP Entry

Show 5 entries   Search:

#	LNP Carrier	Description
1	LNP_Carr1	Rule for LNP Carrier 1

Showing 1 to 1 of 1 entries

Figure 18: NCOS Patterns List

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

**Create Number Pattern** ✕

Pattern: ^439

Description: Austrian Premium Numbers

Save

Figure 19: Create NCOS Number Pattern

In this example, we block (since the mode of the level is *blacklist*) all numbers starting with 439. Click the *Save* button to save the entry in the level.

There are **2 options** that help you to easily define specific number ranges that will be allowed or blocked, depending on whitelist / blacklist mode:

- *Include local area code*: all subscribers within the caller's local area, e.g. if a subscriber has country-code 43 and area-code 1, then selecting this checkbox would result in the implicit number pattern:  $\wedge 431$ .
- *Intra PBX calls within same customer*: all subscribers that belong to the same PBX customer as the caller himself.

In the **NCOS LNP Carriers** view you can select specific LNP Carriers—i.e. carriers that host the called ported numbers—that will be allowed or blocked for routing calls to them (whitelist / blacklist mode, respectively).

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

An example of *NCOS LNP Carrier* pattern definition:

The screenshot shows a web interface titled "Create LNP Carriers". It features a search bar, a table of LNP carriers, and a description field. The table has columns for ID, Name, Prefix, and a selection checkbox. The entry "LNP\_Carr1" with prefix "C1" is selected. Below the table are pagination controls and a "Create LNP Carrier" button. The description field contains "Rule for LNP Carrier 1" and a "Save" button is at the bottom right.

#	Name	Prefix	
11	test_lnp_carrier_4_1510288861	test1510288861	<input type="checkbox"/>
13	test_lnp_carrier_5_1510288862	test1510288862	<input type="checkbox"/>
15	test_lnp_carrier_6_1510288863	test1510288863	<input type="checkbox"/>
17	LNP_Carr1	C1	<input checked="" type="checkbox"/>

Showing 5 to 8 of 9 entries

Create LNP Carrier

Description: Rule for LNP Carrier 1

Save

Figure 20: Create NCOS LNP Carrier

In the above example we created a rule that blocks calls to "LNP\_Carr1" carrier, supposing we use blacklist mode of the NCOS Level.

#### Note

Currently NGCP does not support filtering of individual phone numbers in addition to LNP Carrier matching. In other words: combining phone number and LNP Carrier patterns is not possible.

**Tip**

There might be situations when phone number patterns may not be strictly aligned with telephony providers, for instance in case of full number portability in a country. In such cases using *NCOS LNP Carriers* patterns still allows for defining NCOS levels that allow / block calls to mobile numbers, for example. In order to achieve this goal you have to list all LNP carriers in the NCOS patterns that are known to host mobile numbers.

**6.2.2.3 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 *adm\_ncos* setting in the *Call Blockings* section.

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	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	<input type="text" value=""/>

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.

### 6.2.2.4 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.

### 6.2.3 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.

Call Blockings			
Access Restrictions			
1			
	Name	Value	
	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
	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.

## 6.3 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).

### 6.3.1 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*).

The screenshot shows the 'Edit Call Forward Unconditional' dialog box. It has a green header with the title and a close button. Below the title, there are three radio buttons for 'Destination': 'Voicemail', 'Conference', and 'URI/Number'. The 'URI/Number' option is selected and highlighted with a red box labeled '1'. Below the radio buttons, there is a text input field for 'URI/Number' containing '4312345', highlighted with a red box labeled '2'. Below that is a text input field for 'for (seconds)' containing '300'. At the bottom right, there are two buttons: 'Advanced View' and 'Save', with the 'Save' button highlighted by a red box labeled '3'.

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.

### 6.3.2 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*.

### 6.3.2.1 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.

1 Name my test set

Destination  Voicemail  URI/Number

2 URI/Number 12345

for (seconds) 300

4 Priority 1

optional 5 Add another destination

6 Save

Remove

When you close the *Destination Set Overview*, you can now assign your new set in addition or instead of the *quickset\_cfu* set.

1 Destination Set quickset\_cfu

during Time Set <always>

optional 2 via "Add more"

Destination Set my test set

during Time Set <always>

Add more

3 Save

Press *Save* to store your settings.

### 6.3.2.2 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.

## 6.4 Local Number Porting

The Sipwise NGCP platform comes with two ways of accomplishing local number porting (LNP):

- one is populating the integrated LNP database with porting data,
- the other is accessing external LNP databases via the Sipwise LNP daemon using the LNP API.

---

**Note**

Accessing external LNP databases is available for PRO and CARRIER products only.

---

### 6.4.1 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.

#### 6.4.1.1 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.

#### 6.4.1.2 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.

### 6.4.1.3 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*.

### 6.4.1.4 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.

#### 6.4.1.5 Blocking Calls Using LNP Data

The Sipwise NGCP provides means to allow or block calls towards ported numbers that are hosted by particular LNP carriers. Please visit Section 6.2.2.2 in the handbook to learn how this can be achieved.

#### 6.4.1.6 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.

#### 6.4.1.7 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 3: 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> )

Table 3: (continued)

Start	The <i>Start</i> in the LNP Numbers table (YYYY-MM-DD or empty, e.g. 2016-01-01)
End	The <i>End</i> in the LNP Numbers table (YYYY-MM-DD or empty, e.g. 2016-12-30)
Authoritative	The <i>Authoritative</i> flag in the LNP Carriers table (0 or 1)
Skip Rewrite	The <i>Skip Rewrite</i> flag in the LNP Carriers table (0 or 1)

## 6.5 Emergency Mapping

As opposed to the [Simple Emergency Number Handling](#) Section 5.6.5.1 solution, the Sipwise NGCP supports an advanced emergency call handling method, called *emergency mapping*. The main idea is: instead of obtaining a statically assigned emergency prefix / suffix from subscriber preferences, NGCP retrieves an emergency routing prefix from a central emergency call routing table, according to the current location of the calling subscriber.

The following figure shows the overview of emergency call processing when using *emergency mapping* feature:

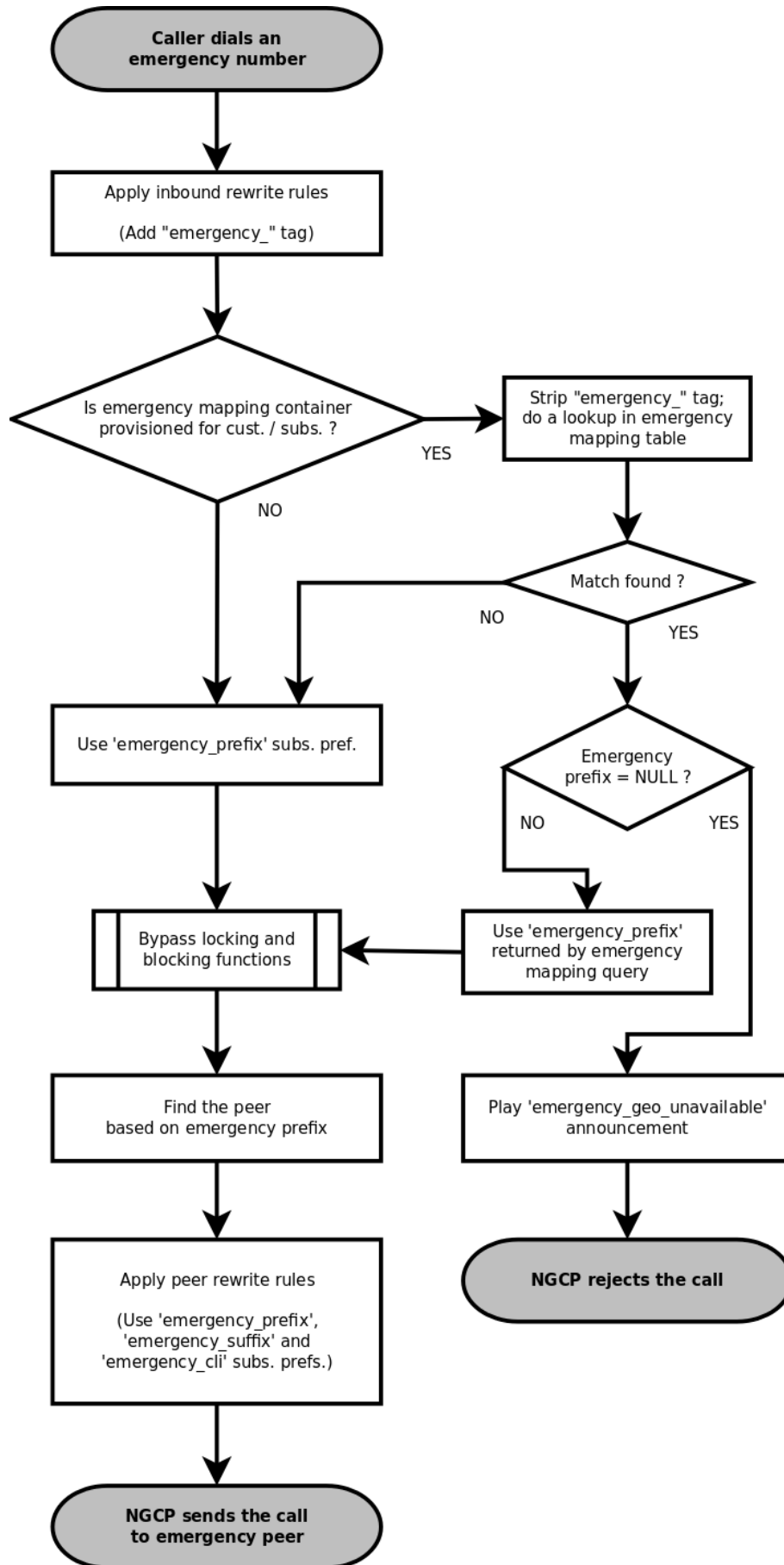


Figure 21: Emergency Call Handling with Mapping

### 6.5.1 Emergency Mapping Description

Emergency numbers per geographic location are mapped to different routing prefixes not deriveable from an area code or the emergency number itself. This is why a **global emergency mapping table** related to resellers is introduced, allowing to map emergency numbers to their geographically dependent routing numbers.

The geographic location is referenced by a location ID, which has to be populated by a north-bound provisioning system. No towns, areas or similar location data is stored on the NGCP platform. The locations are called *Emergency Containers* on NGCP.

The actual emergency number mapping is done per location (per *Emergency Container*), using the so-called *Emergency Mapping* entries. An *Emergency Mapping* entry assigns a routing prefix, valid only in a geographic area, to a generic emergency number (for example *112* in Europe, *911* in the U.S.A.) or a country specific one (for example *133*).

---

**Note**

As of mr4.5 version, the NGCP performs an exact match on the emergency number in the emergency routing table.

---

*Emergency Containers* may be assigned to various levels of the client hierarchy within NGCP. The following list shows such levels with each level overriding the settings of the previous one:

1. Customer or Domain
2. Customer Location, which is a territory representing a subset of the customer's subscribers, defined as one or more IP subnets.
3. Subscriber

---

**Note**

Please be aware that *Customer Location* is not necessarily identical to the "location" identified through an *Emergency Container*.

---

Once the emergency routing prefix has been retrieved from the emergency mapping table, call processing continues in the same way as in case of simple emergency call handling.

### 6.5.2 Emergency Mapping Configuration

The administrative web panel of NGCP provides the configuration interface for emergency mapping. Please navigate to *Settings* → *Emergency Mapping* menu item first, in order to start configuring the mapping.

An *Emergency Container* must be created, before the mapping entries can be defined. Press *Create Emergency Container* to start this. An example of a container is shown here:

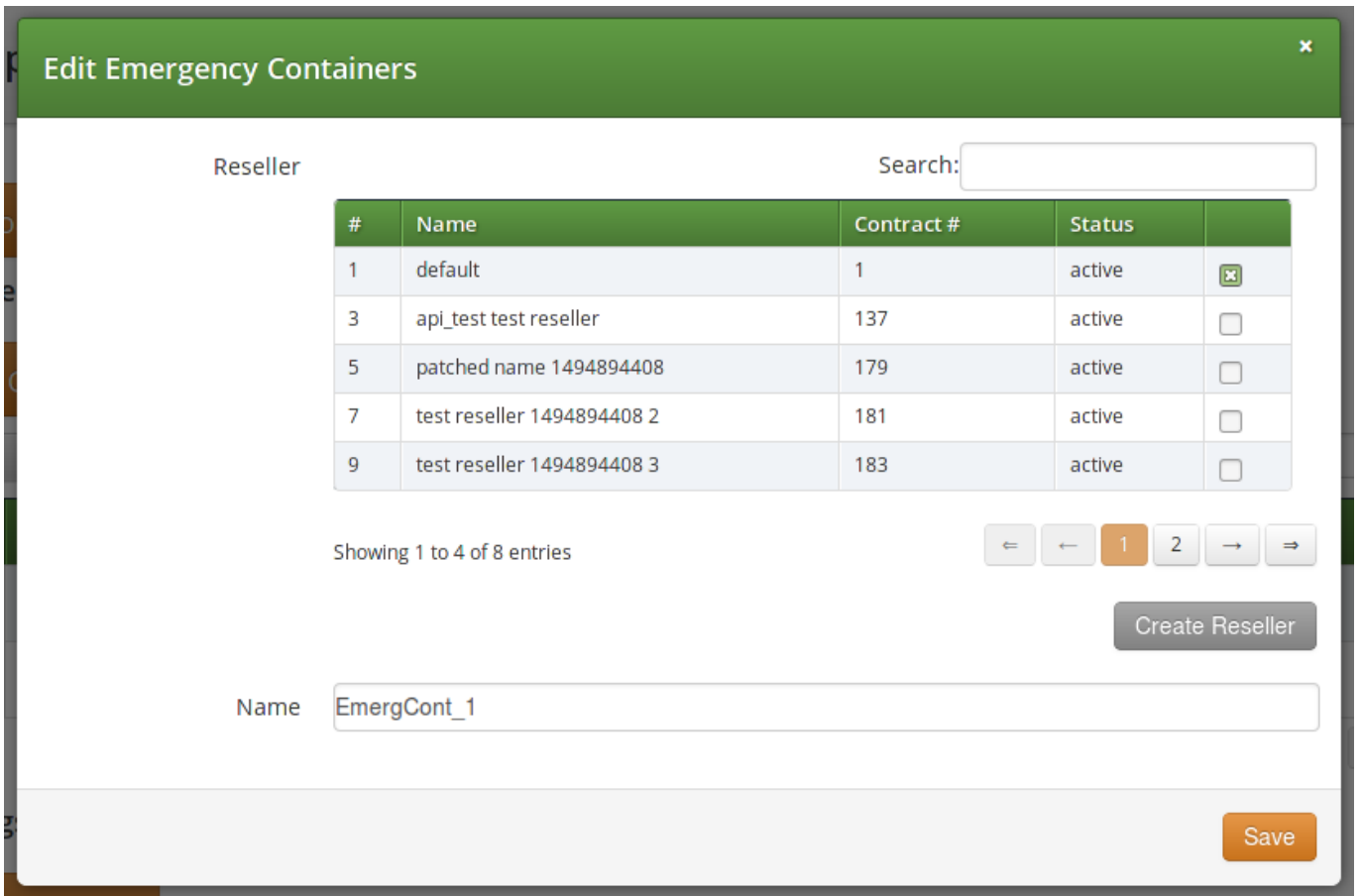


Figure 22: Creating an Emergency Container

You have to select a `Reseller` that this container belongs to, and enter a `Name` for the container, which is an arbitrary text.

**Tip**

The platform administrator has to create as many containers as the number of different geographic areas (locations) the subscribers are expected to be in.

As the second step of emergency mapping provisioning, the *Emergency Mapping* entries must be created. Press *Create Emergency Mapping* to start this step. An example is shown here:



Emergency Mapping Container

Search:

#	Reseller	Name	
1	default	EmergCont_1	<input checked="" type="checkbox"/>
3	default	EmergCont_2	<input type="checkbox"/>

Showing 1 to 2 of 2 entries

Create Emergency Mapping Container

Code

Prefix

Save

Figure 23: Creating an Emergency Mapping Entry

The following parameters must be set:

- **Container:** select an emergency mapping container (i.e. a location ID)
- **Code:** the emergency number that subscribers will dial
- **Prefix:** the routing prefix that belongs to the particular emergency service within the selected location

Once all the necessary emergency mappings have been defined, the platform administrator will see a list of containers and mapping entries:

## Emergency Mappings

← Back
★ Download CSV
★ Upload CSV

### Emergency Containers

★ Create Emergency Container

Show 5 entries Search:

#	Reseller	Name
1	default	EmergCont_1
3	default	EmergCont_2

Showing 1 to 2 of 2 entries

### Emergency Mappings

★ Create Emergency Mapping

Show 5 entries Search:

#	Container	Reseller	Emergency Number	Emergency Prefix
1	EmergCont_1	default	133	E1_133_
3	EmergCont_1	default	144	E1_144_
5	EmergCont_2	default	133	E2_133_

Figure 24: Emergency Mapping List

The emergency number mapping is now defined. As the next step, the platform administrator has to assign the emergency containers to *Customers / Domains / Customer Locations* or *Subscribers*. We'll take an example with a *Customer*: select the customer, then navigate to *Details* → *Preferences* → *Number Manipulations*. In order to assign a container, press the *Edit* button and then select one container from the drop-down list:

The screenshot shows the 'Customer #205 - Preferences' page. At the top, there is a 'Back' button and an 'Expand Groups' button. Below these are sections for 'Call Blockings', 'Access Restrictions', and 'Number Manipulations'. The 'Number Manipulations' section contains a table with the following data:

Attribute	Name	Value
emergency_prefix	Emergency Prefix variable	
emergency_suffix	Emergency Suffix variable	
emergency_cli	Emergency CLI	
emergency_mapping_container	Emergency Mapping Container	EmergCont_2

The 'Value' column for the 'emergency\_mapping\_container' row shows a dropdown menu with 'EmergCont\_2' selected. An 'Edit' button is located to the right of this row. Both the dropdown menu and the 'Edit' button are circled in red in the original image.

Figure 25: Assigning an Emergency Mapping Container

### Rewrite Rules for Emergency Mapping

Once emergency containers and emergency mapping entries are defined, the NGCP administrator has to ensure that the proper number manipulation takes place, before initiating any emergency call towards peers.



#### Important

Please don't forget to define the rewrite rules for peers—particularly: *Outbound Rewrite Rules for Callee*—as described in [Normalize Emergency Calls for Peers](#) Section 5.6.5.3 section of the handbook.

#### 6.5.2.1 Emergency Calls Not Allowed

There is a special case when the dialed number is recognized as an emergency number, but the emergency number is not available for the geographic area the calling party is located in.

In such a case the emergency mapping lookup will return an emergency prefix, but the value of this will be NULL. Therefore the call is rejected and an announcement is played. The announcement is a newly defined sound file referred as `emergency_geo_unavailable`.

It is possible to configure the rejection code and reason in `/etc/ngcp-config/config.yml` file, the parameters are: `kamailio.proxy.early_rejects.emergency_invalid.announce_code` and `kamailio.proxy.early_rejects.emergency_invalid.announce_reason`.

### 6.5.2.2 Bulk Upload or Download of Emergency Mapping Entries

The Sipwise NGCP offers the possibility to upload / download emergency mapping entries in form of CSV files. This operation is available for each reseller, and is very useful if a reseller has many mapping entries.

#### **Downloading Emergency Mapping List**

One has to navigate to *Settings* → *Emergency Mapping* menu and then press the *Download CSV* button to get the list of mapping entries in a CSV file. First the reseller must be selected, then the *Download* button must be pressed. As an example, the entries shown in "Emergency Mapping List" picture above would be written in the file like here below:

```
EmergCont_1,133,E1_133_  
EmergCont_1,144,E1_144_  
EmergCont_2,133,E2_133_
```

The **CSV file** has a plain text format, each line representing a mapping entry, and contains the following **fields**:

- Container name, as defined in *Emergency Containers*
- Emergency Number
- Emergency Prefix

#### **Uploading Emergency Mapping List**

Uploading a CSV file with emergency mapping entries may be started after pressing the *Upload CSV* button. The following data must be provided:

- `Reseller`: selected from the list
- `Upload mapping`: the CSV file must be selected after pressing the *Choose File* button
- `Purge existing`: an option to purge existing emergency mapping entries that belong to the selected reseller, before populating the new mapping data from the file

Upload mapping  (None)

Reseller Search:

#	Name	Contract #	Status	
1	default	1	active	<input type="checkbox"/>
3	api_test test reseller	137	active	<input type="checkbox"/>
5	patched name 1494894408	179	active	<input type="checkbox"/>
7	test reseller 1494894408 2	181	active	<input type="checkbox"/>

Showing 1 to 4 of 8 entries

Purge existing

Figure 26: Uploading Emergency Mapping Data

The CSV file for the upload has the same format as the one used for download.

## 6.6 Header Manipulation

### 6.6.1 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=[%IF kamailio.proxy.debug == "yes"%]P-NGCP-CFGTEST, [%END%]
P-R-Uri,P-D-Uri,P-Preferred-Identity,P-Asserted-Identity,Diversion,Privacy,
Allow,Supported,Require,RAck,RSeq,Rseq,User-Agent,History-Info,Call-Info
[%IF kamailio.proxy.presence.enable == "yes"%],Event,Expires,
Subscription-State,Accept[%END%] [%IF kamailio.proxy.allow_refer_method
== "yes"%],Referred-By,Refer-To,Replaces[%END%]
```

By default the system will remove from the second leg all the SIP headers which are not in the above list. If you want to keep some additional/custom SIP headers, coming from the first leg, into the second leg you 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.

---

### 6.6.2 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`.

### 6.6.3 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)

## 6.7 SIP Trunking with SIPconnect

### 6.7.1 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.

### 6.7.2 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).

### 6.7.3 Number manipulations

The following sections describe the recommended configuration for correct call routing and CLI presentation according to the SIPconnect 1.1 recommendation.

#### 6.7.3.1 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 5.6. 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 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.

---



### 6.7.3.2 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

### 6.7.3.3 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.

#### 6.7.3.4 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.

#### 6.7.4 Registration

SIP-PBX can use either Static or Registration Mode. While SIPconnect 1.1 continues to require TLS support at MUST strength, one should note that using TLS for signaling does not require the use of the SIPS URI scheme. SIPS URI scheme is obsolete for this purpose.

**Static Mode** While SIPconnect 1.1 allows the use of Static mode, this poses additional maintenance overhead on the operator. The administrator should create a static registration for the SIP-PBX: go to Subscribers, *Details*→*Registered Devices*→*Create Permanent Registration* and put address of the SIP-PBX in the following format: sip:username@ipaddress:5060 where username=username portion of SIP URI and ipaddress = IP address of the device.

**Registration Mode** It is recommended to use the Registration mode with SIP credentials defined for the SIP-PBX subscriber.



#### Important

The use of RFC 6140 style "bulk number registration" is discouraged. The SIP-PBX should register one AOR with email-style SIP URI. The sip:provider CE will take care of routing the aliases to the AOR with *e164\_to\_ruri* preference.

---

#### 6.7.4.1 Trusted Sources

If a SIP-PBX cannot perform the digest authentication, you can authenticate it by its source IP address in sip:provider CE. To configure the IP-based authentication, go to the subscriber's preferences (*Details*→*Preferences*→*Trusted Sources*) and specify the IP address of the SIP-PBX in the *Source IP* field.

To authenticate multiple subscribers from the same IP address, use the *From* field to distinguish these subscribers.

When this feature is configured for a subscriber, the sip:provider CE authenticates all calls that arrive from the specified IP address without challenging them.



#### Important

If the same IP address and the FROM field are mistakenly specified as trusted for different subscribers, the sip:provider CE will not know which subscriber to charge for the call and will randomly select one.

---

## 6.8 Trusted Subscribers

In some cases, when you have a device that cannot authenticate itself against sip:provider CE, you may need to create a *Trusted Subscriber*. Trusted Subscribers use IP-based authentication and they have a Permanent SIP Registration URI in order to receive messages from sip:provider CE.

In order to make a regular subscriber trusted, perform the following extra steps: \* Create a permanent registration via (*Subscribers*→*Details*→ *Registered Devices*→*Create Permanent Registration*) \* Add the IP address of the device as Trusted Source in your subscriber's preferences (*Details*→*Preferences*→*Trusted Sources*).

This way, all SIP messages coming from the device IP will be considered trusted (and get authenticated just by the source IP). All the SIP messages forwarded to the devices will be sent to the SIP URI specified in the subscriber's permanent registration.

## 6.9 Voicemail System

### 6.9.1 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 number, causing the system to prompt for a mailbox and the PIN. The PIN can be set in the *Voicemail and Voicebox* section of the *Subscriber Preferences*.

#### 6.9.1.1 Mapping numbers and codes to IVR access

Since access might need to be provided from external networks like PSTN/Mobile, and since certain SIP phones do not support calling alphanumeric numbers to dial `voicebox`, you can map any number to the voicebox URIs using rewrite rules.

To do so, you can provision a match pattern e.g. `^(00|\+)12345$` with a replace pattern `voicebox` or `voiceboxpass` to map a number to either password-less or password-based IVR access respectively. Create a new rewrite rule with the *Inbound* direction and the *Callee* field in the corresponding rewrite rule set.

For inbound calls from external networks, assign this rewrite rule set to the corresponding incoming peer. If you also need to map numbers for on-net calls, assign the rewrite rule set to subscribers or the whole SIP domain.

#### 6.9.1.2 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.

## 6.9.2 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

### 6.9.3 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.

#### 6.9.3.1 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".

#### 6.9.3.2 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."

### 6.9.3.3 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.

## 6.9.4 Folders

The Voicemail system allows you to save and organize your messages into folders. There can be up to ten folders.

### 6.9.4.1 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.

## 6.9.5 Flowcharts with Voice Prompts

This section shows flowcharts of calls to the voicemail system. Flowcharts contain the name of prompts as they are identified among *Asterisk* voice prompts.

### 6.9.5.1 Listening to New Messages

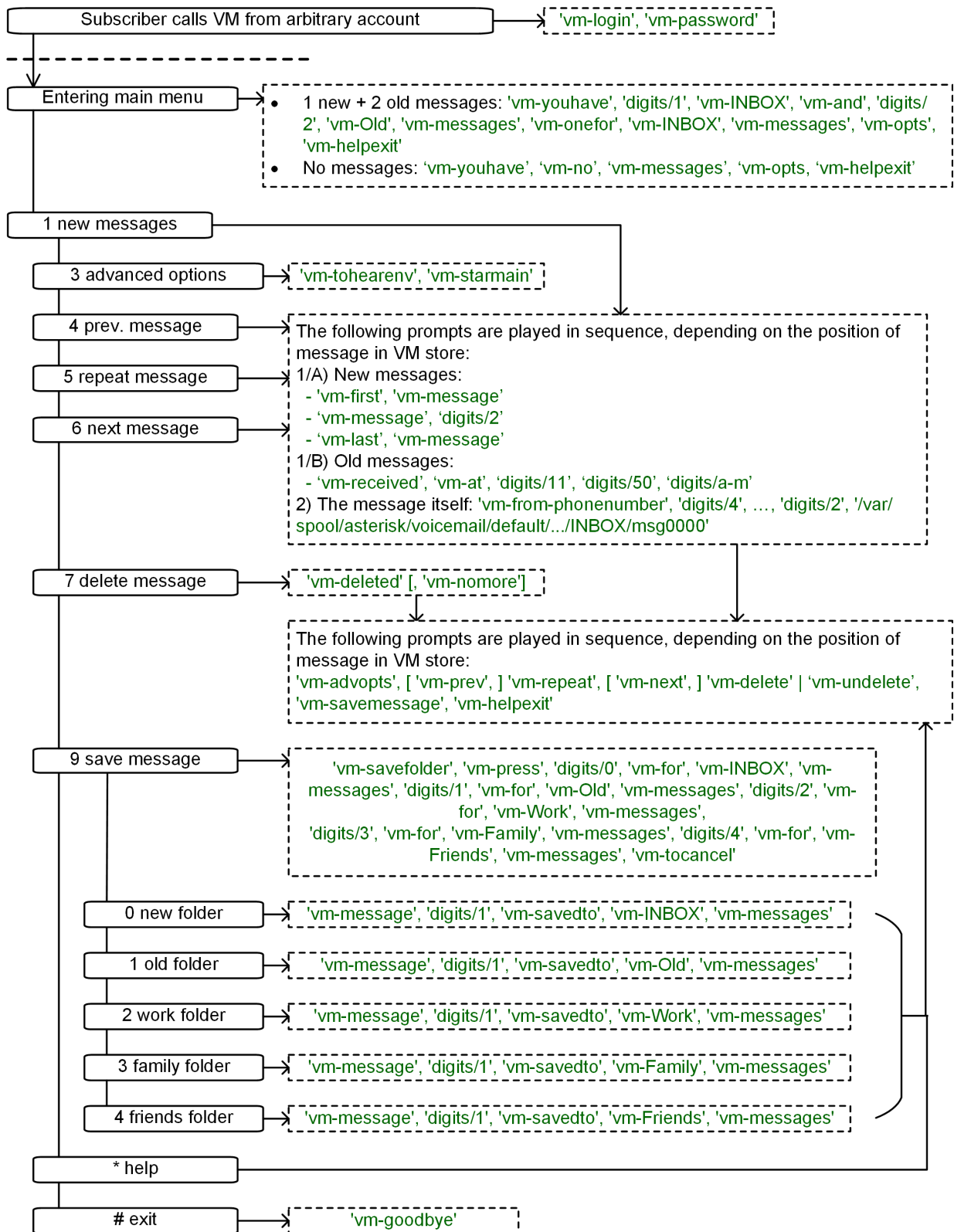


Figure 27: Flowchart of Listening to New Messages

### 6.9.5.2 Changing Voicemail Folders

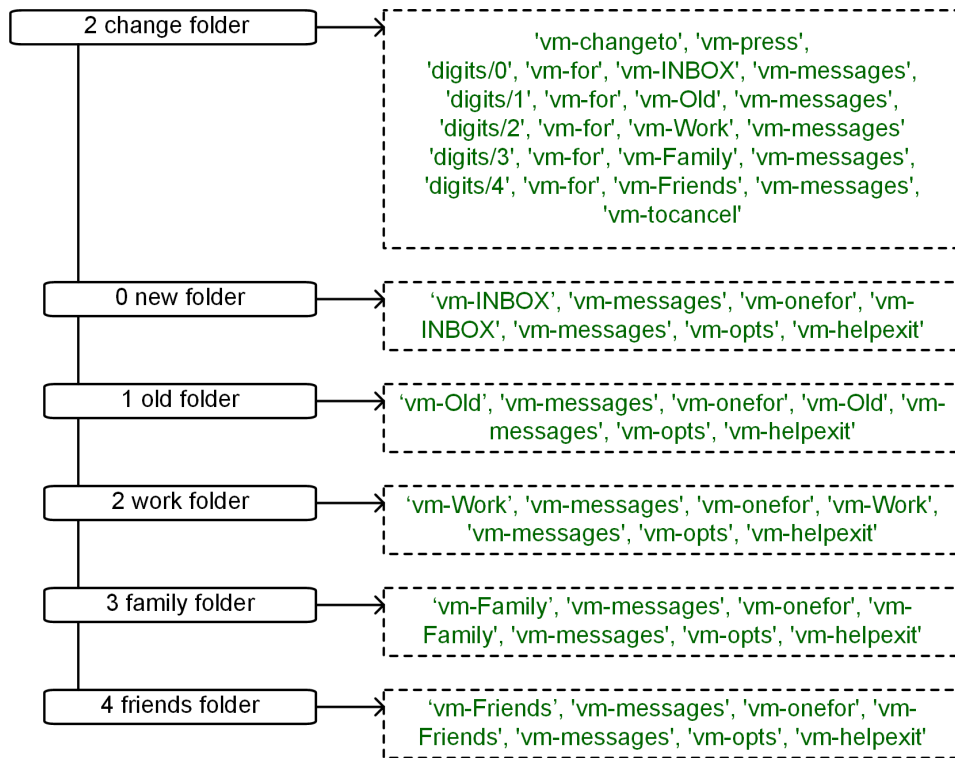


Figure 28: Flowchart of Changing Voicemail Folders



### 6.9.5.3 Mailbox Options

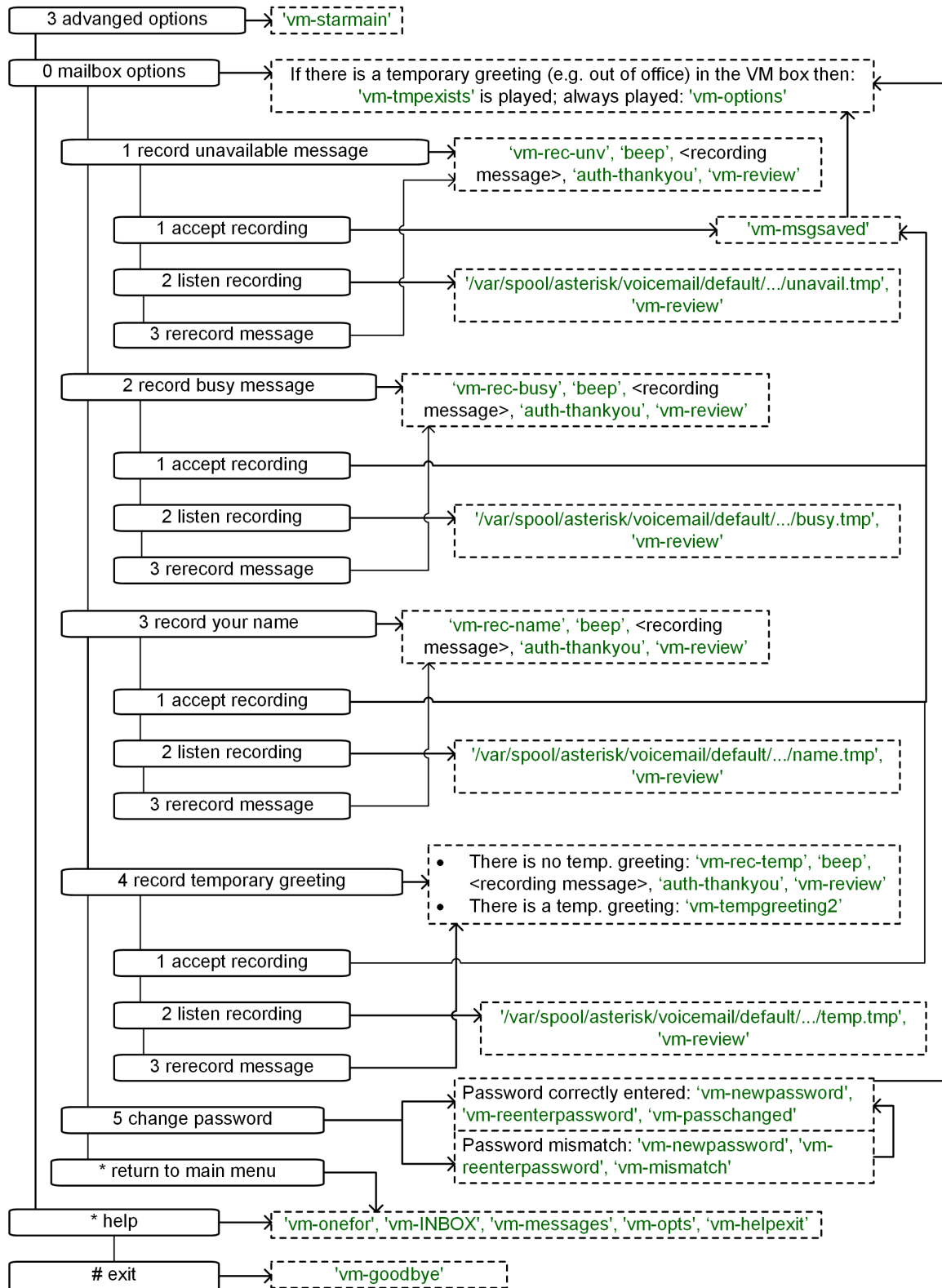


Figure 29: Flowchart of Changing Mailbox Options

6.9.5.4 Leaving a Message

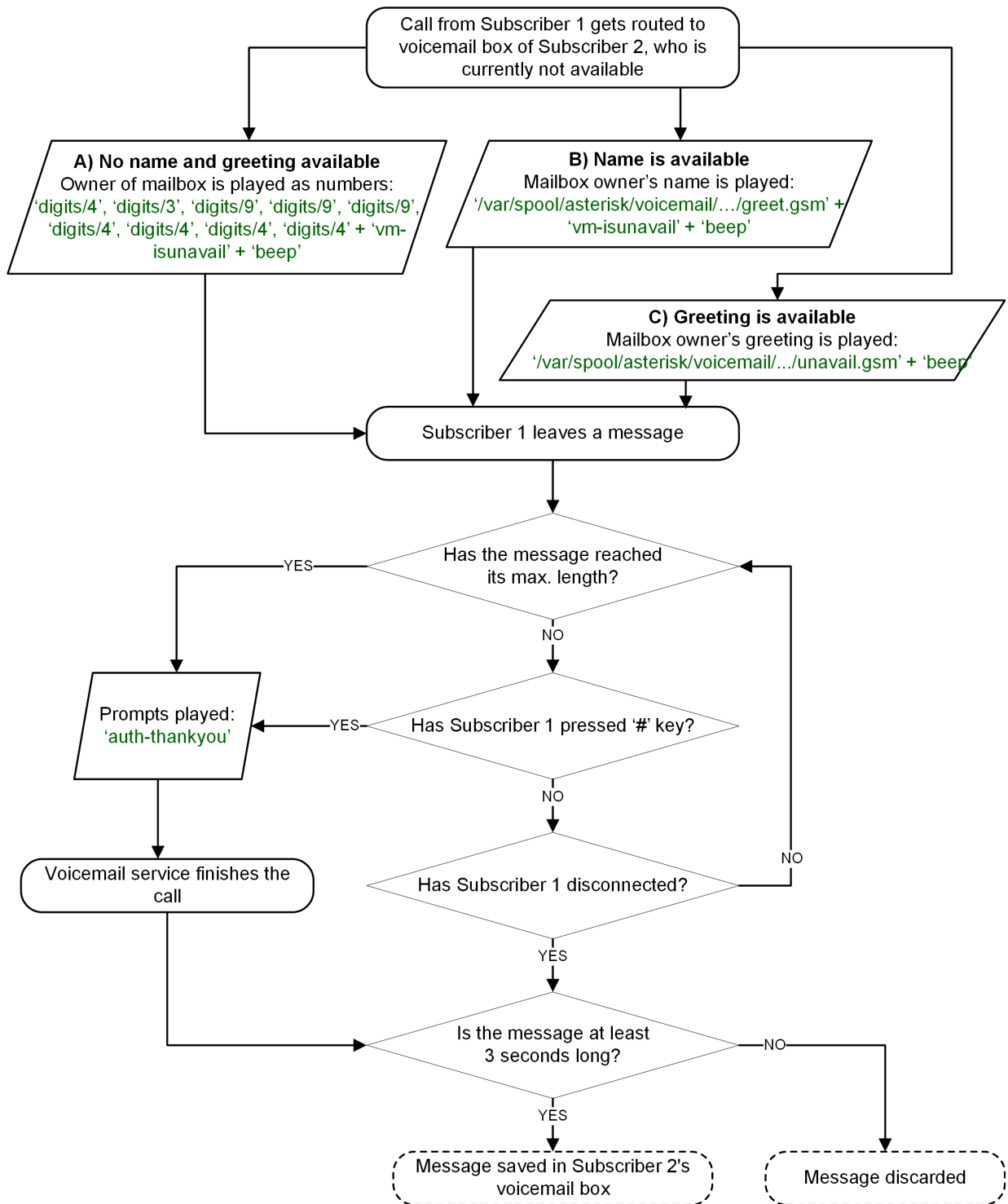
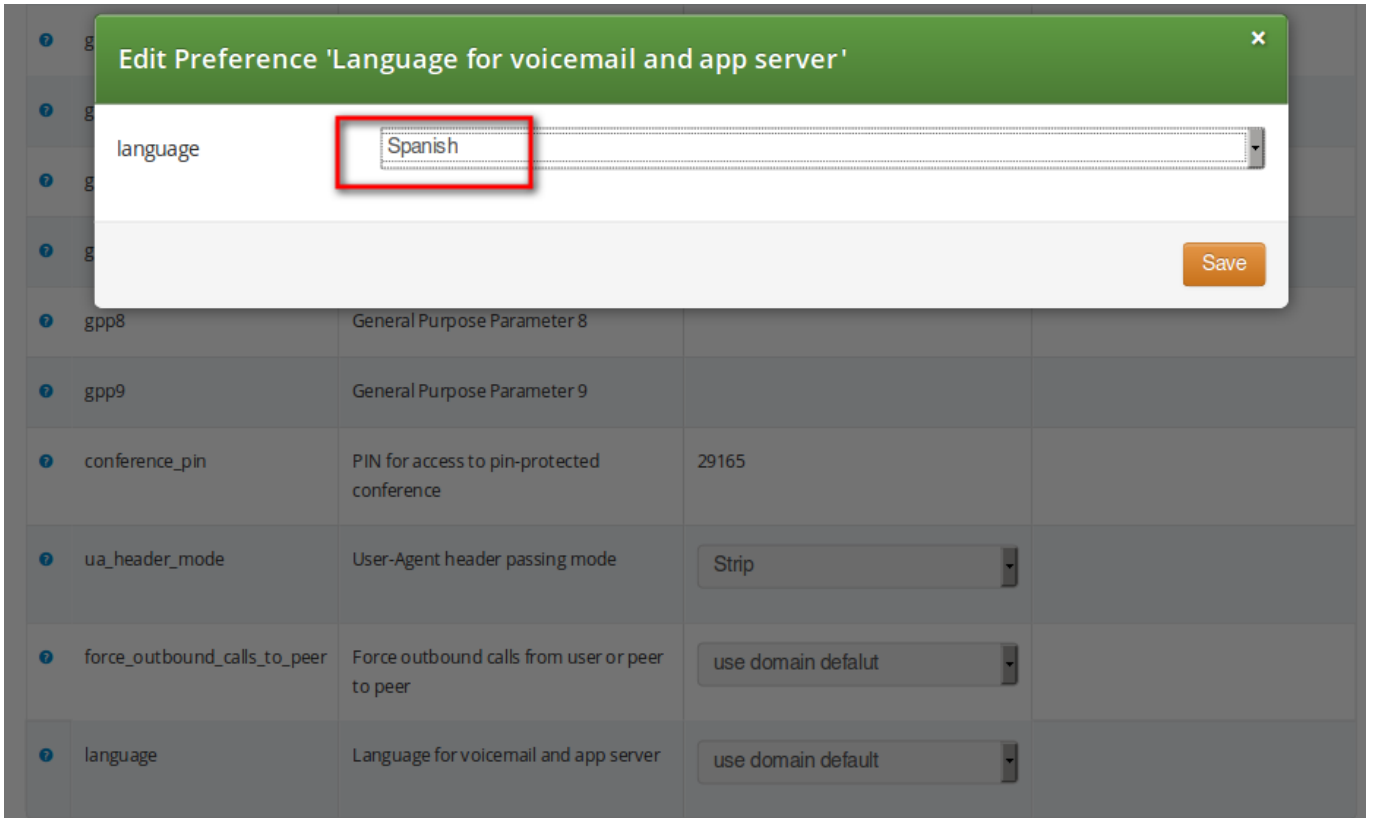


Figure 30: Flowchart of Leaving a Voice Message

## 6.10 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.

## 6.11 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* in the *NAT and Media Flow Control* section on Domain and Subscriber levels to link subscribers to the sound set that they should hear (as usual the subscriber preference overrides the domain one). 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** Home Monitoring & Statistics Settings

## Sound Sets

← Back ★ Create Sound Set

Sound set successfully created

Show 5 entries Search:

#	Reseller	Customer	Name	Description	
1	default		Conference		<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Files</a>
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.

### 6.11.1 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.

The exact error status code and text are configurable in the `/etc/ngcp-config/config.yml` file, in `kamailio.proxy.early_rejects` section. Please look for the announcement handle listed in below table in order to find it in the configuration file.

Table 4: Early Reject Announcements

Handle	Description	Message played
announce_before_cf	This is an announcement that the calling party hears before the call is being forwarded (Unconditional and Not Available cases) to the destination. The feature can be activated with Applications / <code>play_announce_before_cf</code> domain or subscriber preference.	N/A (custom message, no default)
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 ( <code>adm_block_in_list</code> , <code>block_in_list</code> customer/subscriber preferences)	Your call is blocked by the number you are trying to reach.

Table 4: (continued)

Handle	Description	Message played
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> customer/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). <i>PLEASE NOTE:</i> It is not possible to configure the status code and text.	Your call to the number you are trying to reach is not permitted.
block_override_pin_wrong	Announcement played to calling party if it used wrong PIN code to override the outgoing user block list or the NCOS level for this call (the PIN set by <i>block_out_override_pin</i> and <i>adm_block_out_override_pin</i> preferences)	The PIN code you have entered is not correct.
callee_busy	Announcement played on incoming call to the subscriber which is currently busy (486 response from the UAS)	The number you are trying to reach is currently busy. Please try again later.
callee_offline	Announcement played on incoming call to the subscriber which is currently not registered	The number you are trying to reach is currently not available. Please try again later.
callee_tmp_unavailable	Announcement played on incoming call to the subscriber which is currently unavailable (408, other 4xx or no response code or 30x with malformed contact)	The number you are trying to reach is currently not available. Please try again later.
callee_unknown	Announcement that is played on call to unknown or invalid number (not associated with any of our subscribers/hunt groups)	The number you are trying to reach is not in use.
cf_loop	Announcement played when the called subscriber has the call forwarding configured to itself	The number you are trying to reach is forwarded to an invalid destination.

Table 4: (continued)

Handle	Description	Message played
emergency_geo_unavailable	Announcement played when emergency destination is dialed but the destination is not provisioned for the location of the user. <i>PLEASE NOTE:</i> The configuration entry for this case in <code>/etc/ngcp-config/config.yml</code> file is <code>emergency_invalid</code> .	The emergency number you have dialed is not available in your region.
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.
error_please_try_later	Announcement played when the call is handled by 3rd party call control (PCC) and there was an error during call processing. <i>PLEASE NOTE:</i> This announcement may be configured in the sound set in <code>voucher_recharge</code> section.	An error has occurred. Please try again later.
invalid_speeddial	This is what the calling party hears when it calls an empty speed-dial slot	The speed dial slot you are trying to use is not available.
locked_in	Announcement played on incoming call to a subscriber that is locked for incoming calls	The number you are trying to reach is currently not permitted to receive calls.
locked_out	Announcement played on outgoing call to subscriber that is locked for outgoing calls	You are currently not allowed to place outbound calls.
max_calls_in	Announcement played on incoming call to a subscriber who has exceeded the <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.

Table 4: (continued)

Handle	Description	Message played
max_calls_peer	Announcement played on calls from the peering if that peer has reached the maximum number of concurrent calls (configured by admin in <i>concurrent_max</i> preference of peering server). <i>PLEASE NOTE</i> : There is no configuration option of the status code and text in <i>config.yml</i> file for this case.	The network you are trying to reach is currently busy. Please try again later.
no_credit	Announcement played when prepaid account has insufficient balance to make a call to this destination	You don't have sufficient credit balance for the number you are trying to reach.
peering_unavailable	Announcement played in case of outgoing off-net call when there is no peering rule matching this destination and/or source	The network you are trying to reach is not available.
reject_vsc	When the VSC (Vertical Service Code) service is disabled in domain or subscriber preferences (Access Restrictions / <i>reject_vsc</i> is set to TRUE) and a subscriber tries to make a call with VSC, an announcement is played.	N/A (custom message, no default)
relaying_denied	Announcement played on inbound call from trusted IP (e.g. external PBX) with non-local Request-URI domain	The network you are trying to reach is not available.
unauth_caller_ip	This is what the calling party hears when it tries to make a call from unauthorized IP address or network ( <i>allowed_ips</i> , <i>man_allowed_ips</i> preferences)	You are not allowed to place calls from your current network location.
voicebox_unavailable	<i>PLEASE NOTE</i> : This announcement is already obsolete, as of NGCP version mr5.3	The voicemail of the number you are trying to reach is currently not available. Please try again later.

There are some early reject scenarios when either **no voice announcement is played, or a fixed announcement is played**. In either case a SIP error status message is sent from NGCP to the calling party. It is possible to configure the exact status code and text for such cases in the */etc/ngcp-config/config.yml* file, in *kamailio.proxy.early\_rejects* section. The below table gives an overview of those early reject cases.



Table 5: Additional Early Reject Reason Codes

Handle	Description
block_admin	Caller blocked by adm_block_in_list, adm_block_in_clir and callee blocked by adm_block_out_list (customer or subscriber preference)
block_callee	Callee blocked by subscriber preference block_out_list
block_caller	Caller blocked by subscriber preference block_in_list, block_in_clir
block_contract	Caller blocked by customer preference block_in_list, block_in_clir and callee blocked by customer preference block_out_list
callee_tmp_unavailable_gp	Callee is a PBX group with 0 members. Announcement callee_tmp_unavailable is played; status code and text can be configured.
callee_tmp_unavailable_tm	Callee is a PBX group and we have a timeout (i.e. no group member could be reached). Announcement callee_tmp_unavailable is played; status code and text can be configured.
emergency_invalid	<i>PLEASE NOTE:</i> This handle refers to the same early reject case as emergency_geo_unavailable, but is labeled differently in the configuration file.

## 6.12 Conference System

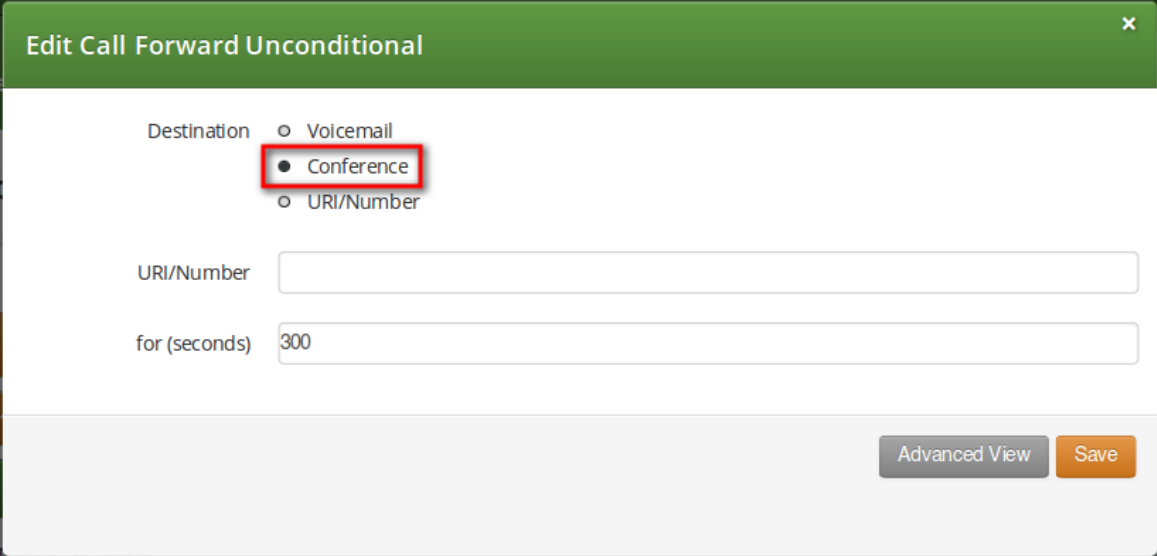
The sip:provider CE provides the simple pin-protected conferencing service built using the SEMS DSM scripting language. Hence it is open for all kinds of modifications and extensions.

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`

### 6.12.1 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*).

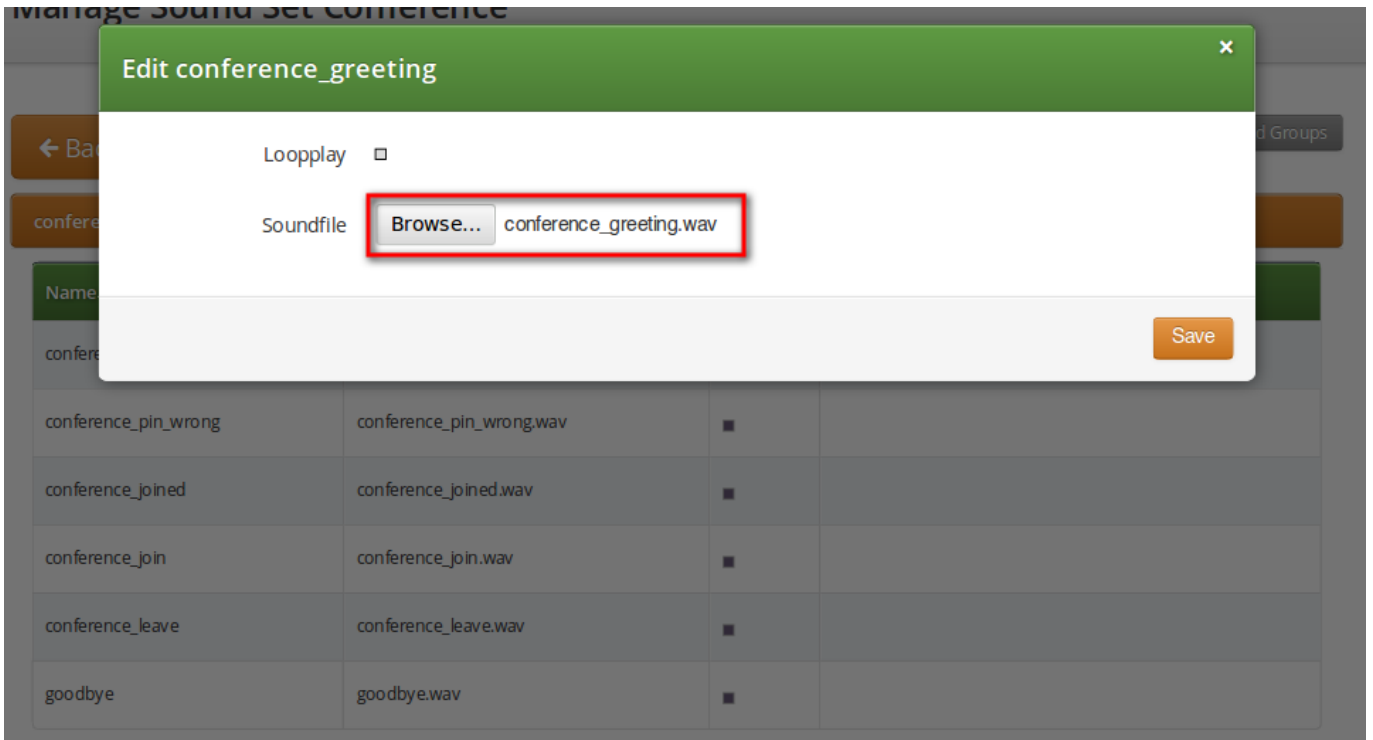


The screenshot shows the 'Edit Call Forward Unconditional' dialog box. The 'Destination' field has three radio button options: 'Voicemail', 'Conference' (selected and highlighted with a red box), and 'URI/Number'. The 'URI/Number' field is empty. The 'for (seconds)' field contains the value '300'. At the bottom right, there 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.

### 6.12.2 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 6: Conference Sound Sets

Handle	Message played
conference_greeting	Welcome to the conferencing service.
conference_pin	Please enter your PIN, followed by the pound key.
conference_pin_wrong	You have entered an invalid PIN number. Please try again.
conference_joined	You will be placed into the conference.
conference_first	You are the first person in the conference.
conference_join	A person has joined the conference.
conference_leave	A person has left the conference.
conference_max_participants	All conference lines are currently in use. Please try again later.
conference_waiting_music	... waiting music...
goodbye	Goodbye.

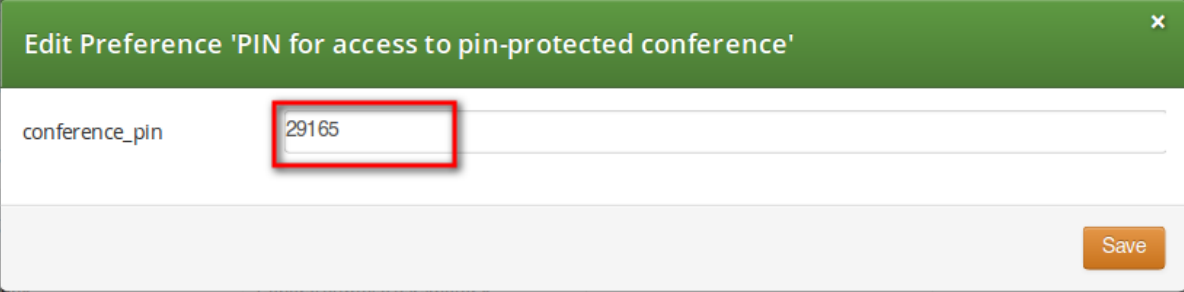
**Note**

You may use 8 or 16 bit mono WAV audio files.

Then set the preference *sound\_set* on the Domain or Subscriber level in order to assign the Sound Set you have just created to the subscriber (as usual the subscriber preference overrides the domain one).

### 6.12.3 Joining the Conference

There are 2 ways of joining a conference: with or without PIN code. The actual way of joining the conference depends on *Subscriber* settings. A subscriber who has activated the conference through call forwarding may set a PIN in order to protect the conference from unauthorized access. To activate the PIN one has to enter a value in *Subscriber* → *Details* → *Preferences* → *Internals* → *conference\_pin* field.



The image shows a screenshot of a web interface for editing preferences. A modal dialog titled "Edit Preference 'PIN for access to pin-protected conference'" is open. Inside the dialog, there is a text input field labeled "conference\_pin" containing the value "29165". A red rectangular box highlights the input field. To the right of the input field is an orange "Save" button. Below the dialog, a table of preferences is visible, with the "conference\_pin" row highlighted. The table has four columns: a name column, a description column, a value column, and an empty column.

gpp0	General Purpose Parameter 0		
gpp9	General Purpose Parameter 9		
conference_pin	PIN for access to pin-protected conference	29165	
ua_header_mode	User-Agent header passing mode	Strip	
force_outbound_calls_to_peer	Force outbound calls from user or peer to peer	use domain default	
language	Language for voicemail and app server	use domain default	

Figure 31: Setting Conference PIN

In case the PIN protection for the conference is activated, when someone calls the subscriber who has enabled the conference, the caller is prompted to enter the PIN of the conference. Upon the successful entry of the PIN the caller hears the announcement that he is going to be placed into the conference and at the same time this is announced to all participants already in the conference.

### 6.12.4 Conference Flowchart with Voice Prompts

The following 2 sections show flowcharts with voice prompts that are played to a caller when he dials the conference.

6.12.4.1 Conference Flowchart with PIN Validation

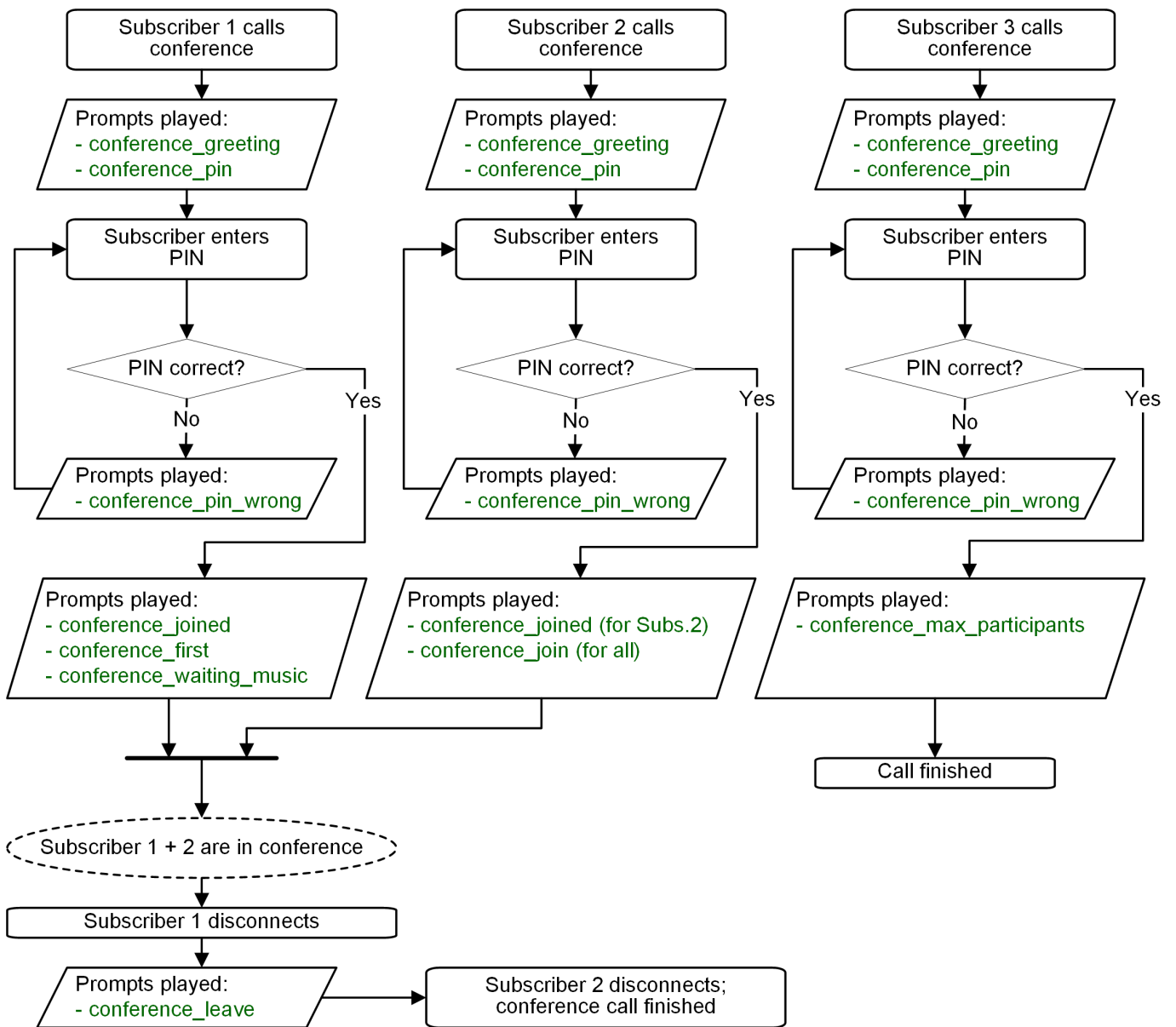


Figure 32: Flowchart of Conference with PIN Validation

### 6.12.4.2 Conference Flowchart without PIN

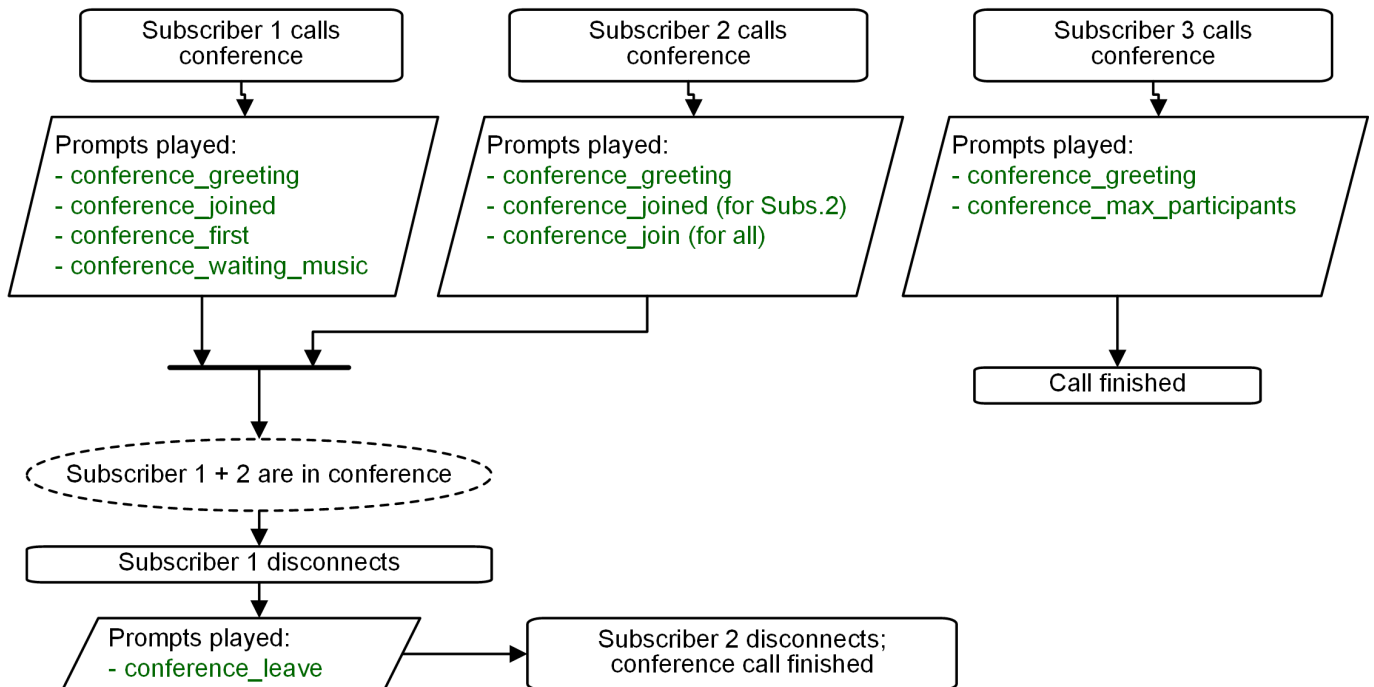


Figure 33: Flowchart of Conference without PIN

## 6.13 Malicious Call Identification (MCID)

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

### 6.13.1 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 *Callee Inbound* rewrite rule `^(\\*123)$ → malicious_call`

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

### 6.13.2 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.

### 6.13.3 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 *ngcpcfg apply Enabling MCID* afterwards.

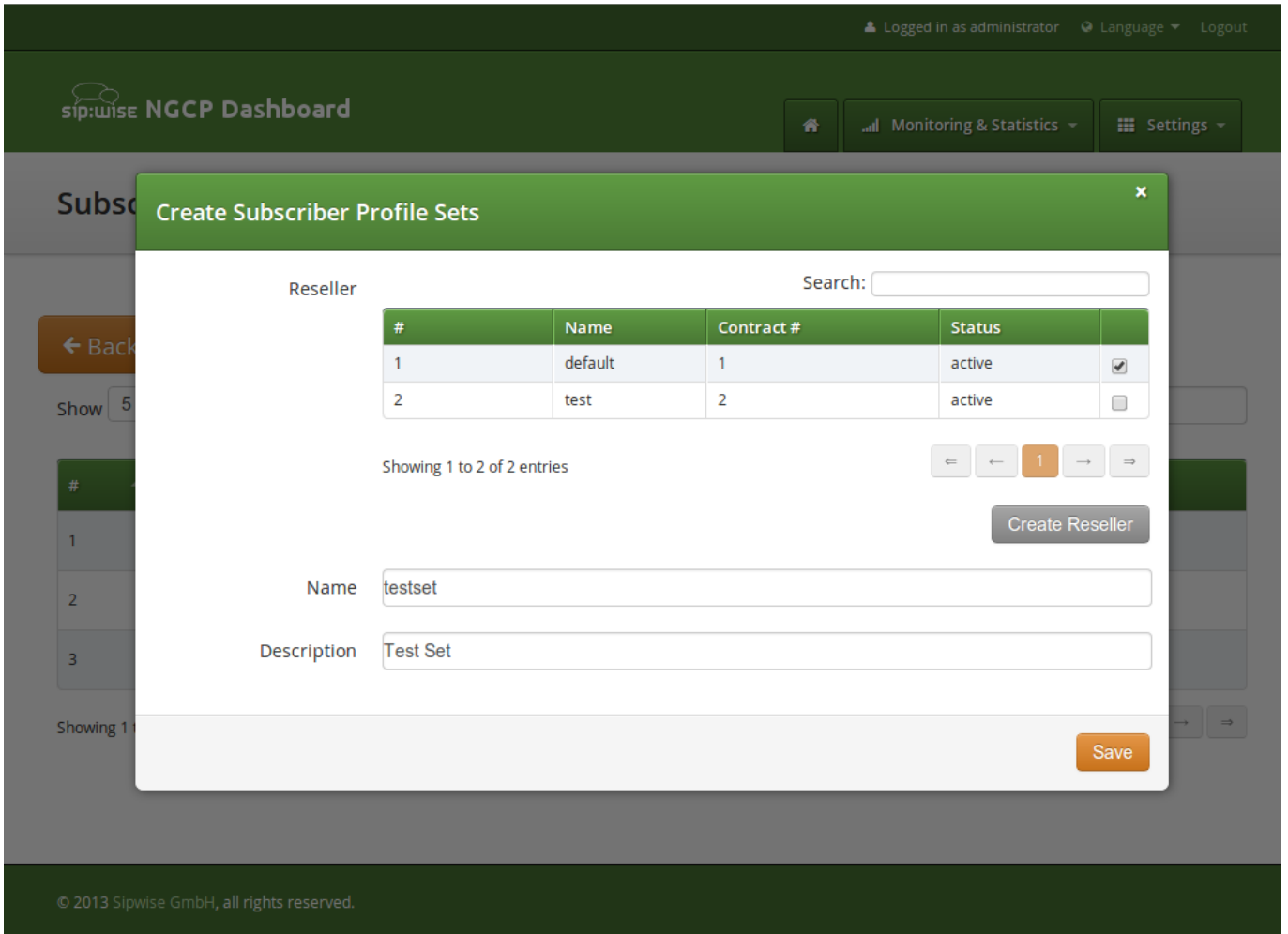
## 6.14 Subscriber Profiles

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

### 6.14.1 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 features a search bar and a table of resellers. The table has columns for '#', 'Name', 'Contract #', 'Status', and a checkbox. Below the table, there are navigation controls and a 'Create Reseller' button. At the bottom of the modal, there are input fields for 'Name' (containing 'testset') and 'Description' (containing 'Test Set'), along with a 'Save' button.

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

Showing 1 to 2 of 2 entries

Create Reseller

Name: testset

Description: Test Set

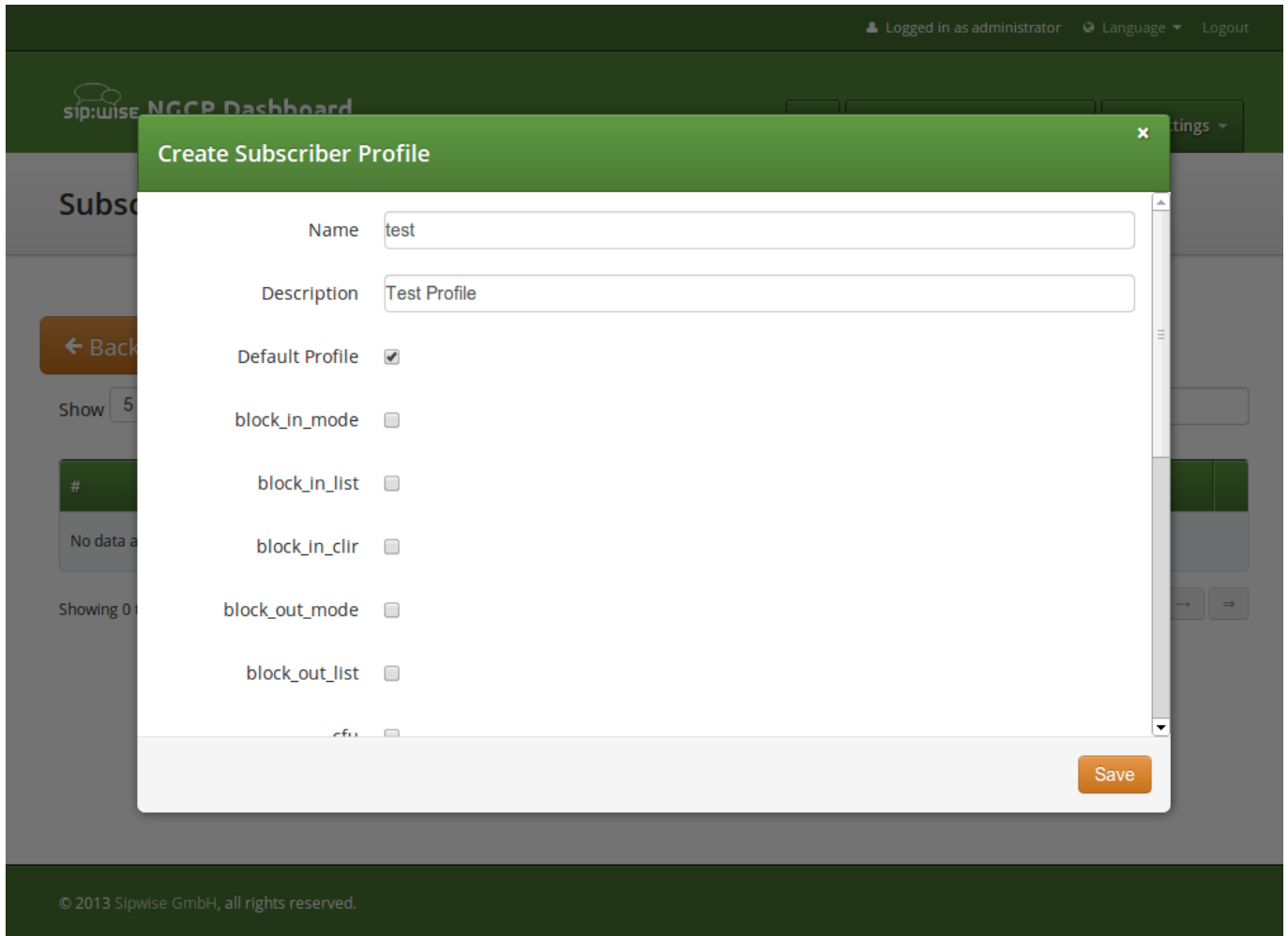
Save

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.





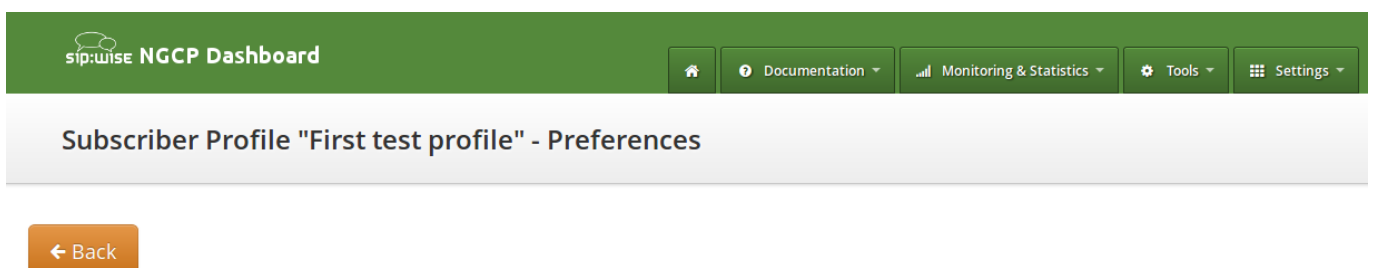
Checking the *Default Profile* option causes this profile to get assigned automatically to all subscribers, who have the profile set assigned. Other options define the user preferences which should be made available to the subscriber.

---

#### Note

When the platform administrator selects *Preferences* of the Subscriber Profile he will get an empty page like in the picture below, if none or only certain options are selected in the Subscriber Profile.

---



Some of the options, like `ncos` (NCOS level), will enable the definition of that preference within the Subscriber Profile Preferences. Thus all subscribers who have this profile assigned to will have the preference activated by default. The below picture shows the preferences linked to the sample Subscriber Profile:

sip:wise NGCP Dashboard

Home Documentation Monitoring & Statistics Tools Settings

Subscriber Profile "Test profile 1 for NCOS" - Preferences

Back Expand Groups

Call Blockings

	Attribute	Name	Value
	ncos	NCOS Level	Test NCOS for blocking outca

## 6.15 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_detection.expire` seconds. Higher occurrence of loops will be reported with a SIP 482 "Loop Detected" error message

## 6.16 Invoices and Invoice Templates

Content and vision of the invoices are customizable by [invoice templates](#) Section [6.16.2](#).

---

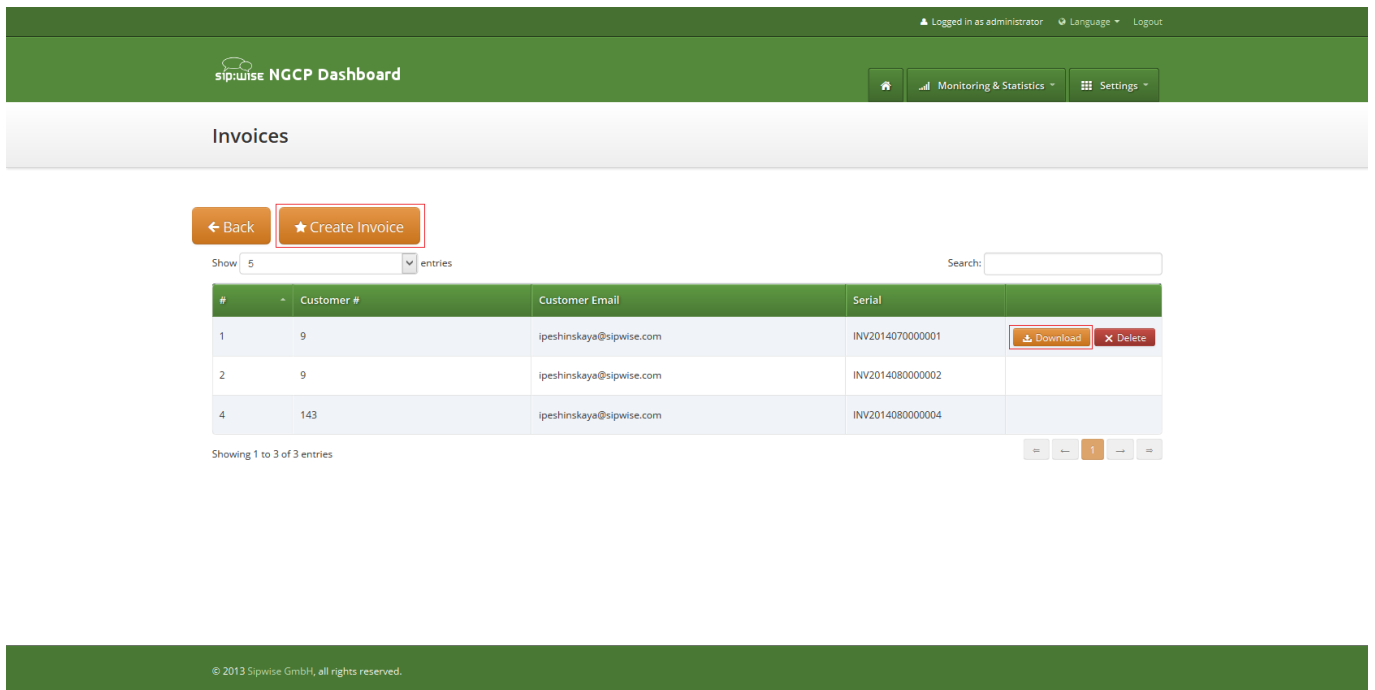
### Note

The sip:provider CE generates invoices in pdf format.

---

### 6.16.1 Invoices Management

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



Logged in as administrator | Language | Logout

**sip:wise NGCP Dashboard** | Monitoring & Statistics | Settings

## Invoices

← Back | ★ Create Invoice

Show 5 entries | Search:

#	Customer #	Customer Email	Serial	
1	9	ipeshinskaya@sipwise.com	INV2014070000001	<a href="#">Download</a> <a href="#">Delete</a>
2	9	ipeshinskaya@sipwise.com	INV2014080000002	
4	143	ipeshinskaya@sipwise.com	INV2014080000004	

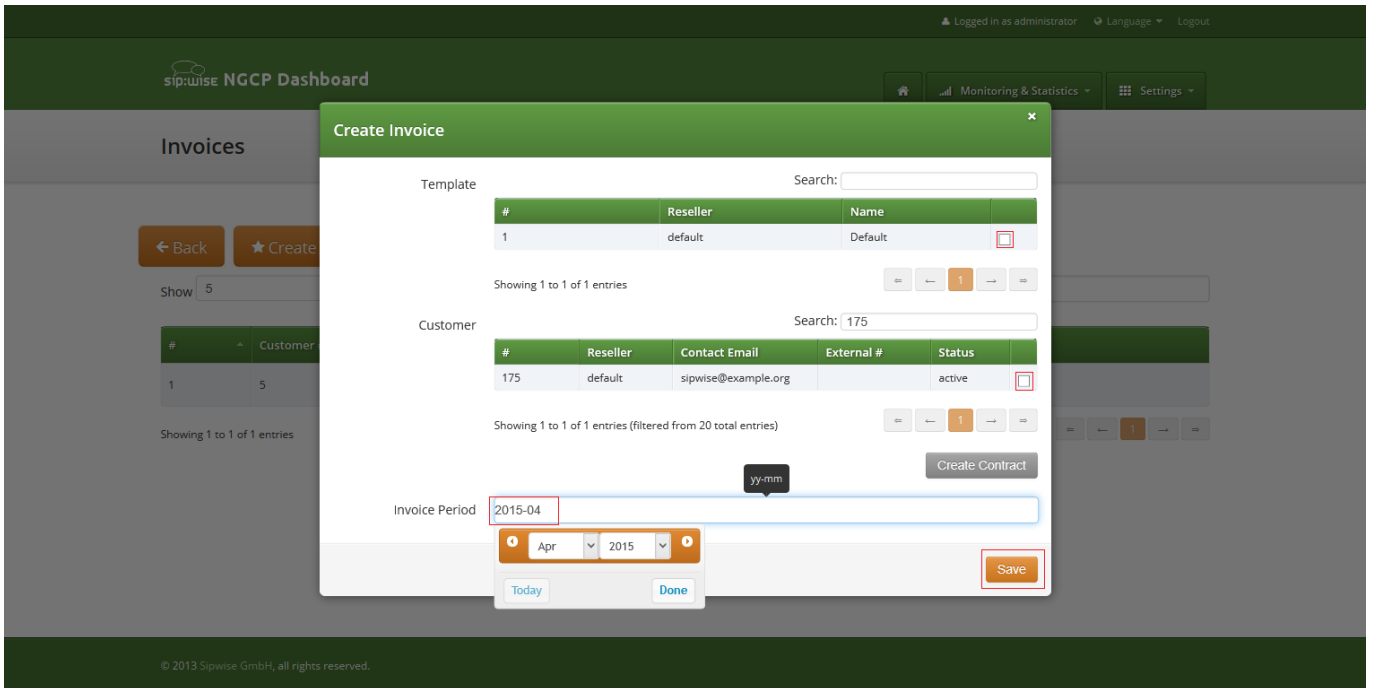
Showing 1 to 3 of 3 entries

© 2013 Sipwise GmbH, all rights reserved.

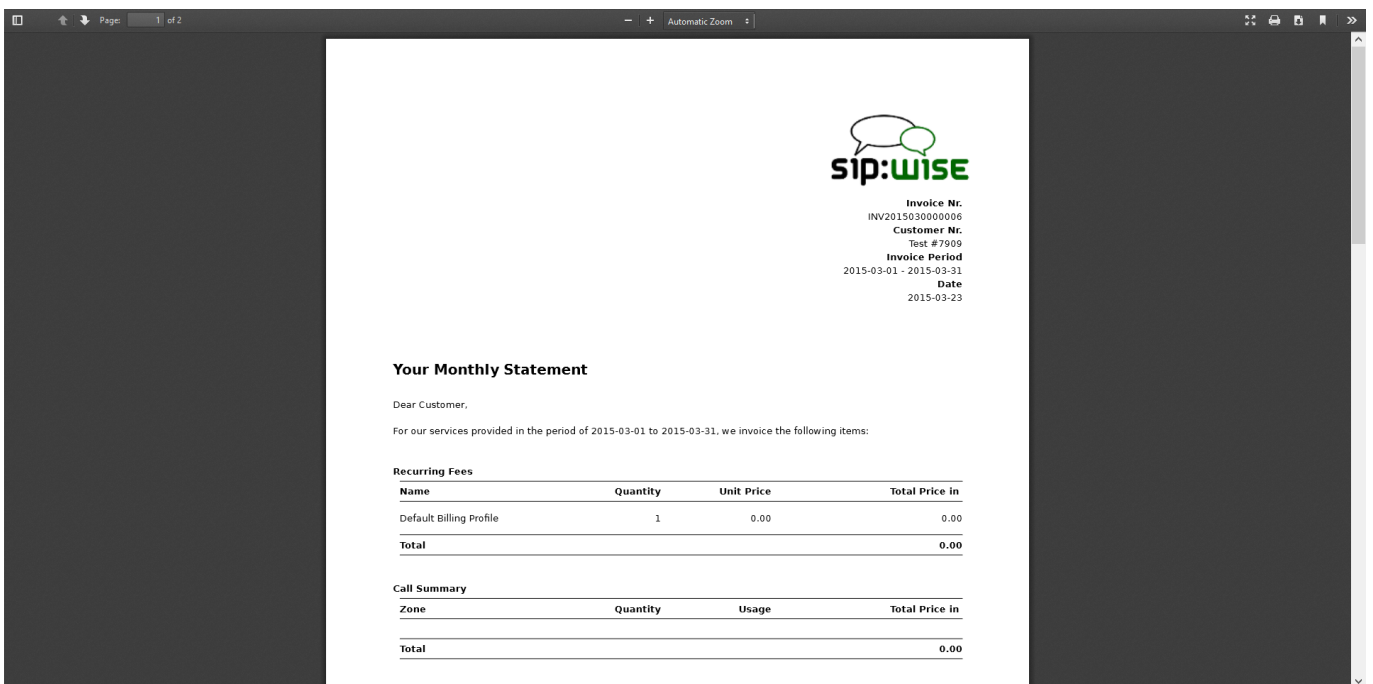
To request invoice generation for the particular customer and period press "Create invoice" button. On the invoice creation form following parameters are available for selection:

- **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.

### 6.16.2 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.

#### 6.16.2.1 Invoice Templates Management

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

© 2013 Sipwise GmbH, all rights reserved.

Invoice template creation is separated on two steps:

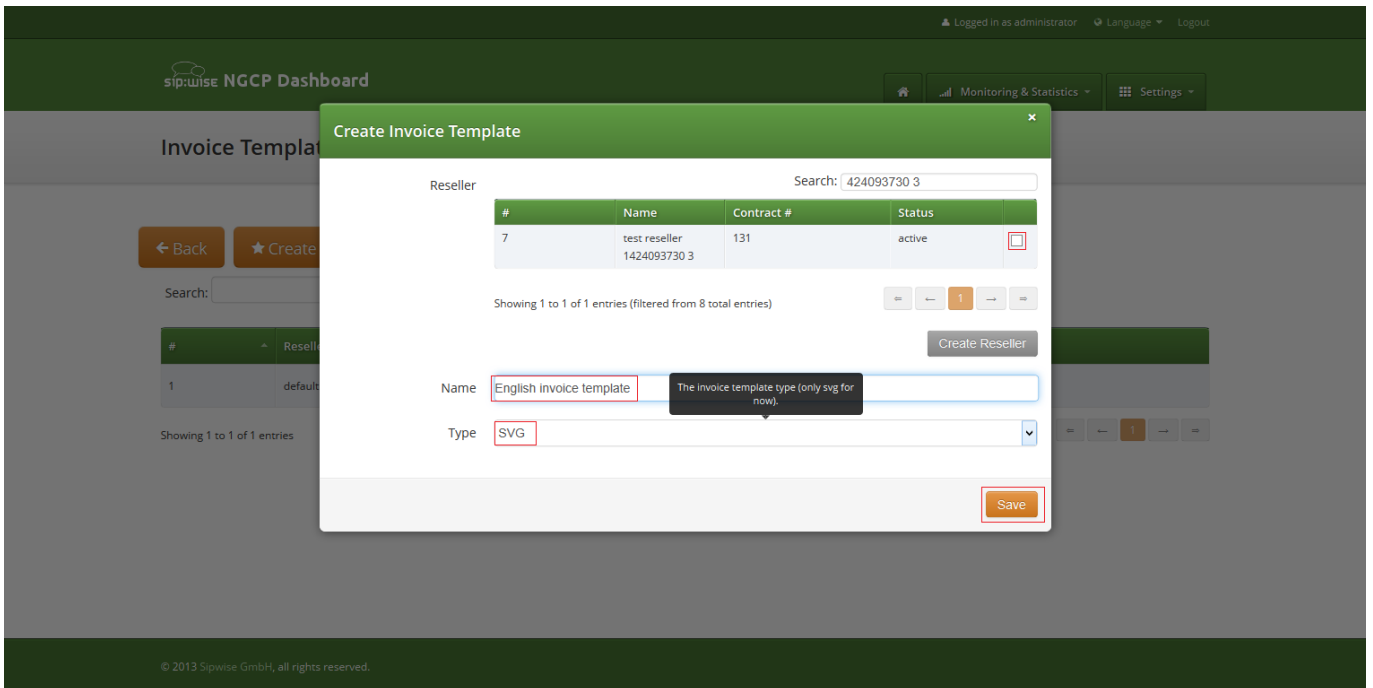
- Register new invoice template meta information.
- Edit content (template itself) of the invoice template.

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.

### 6.16.2.2 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 6.16.2.3.

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

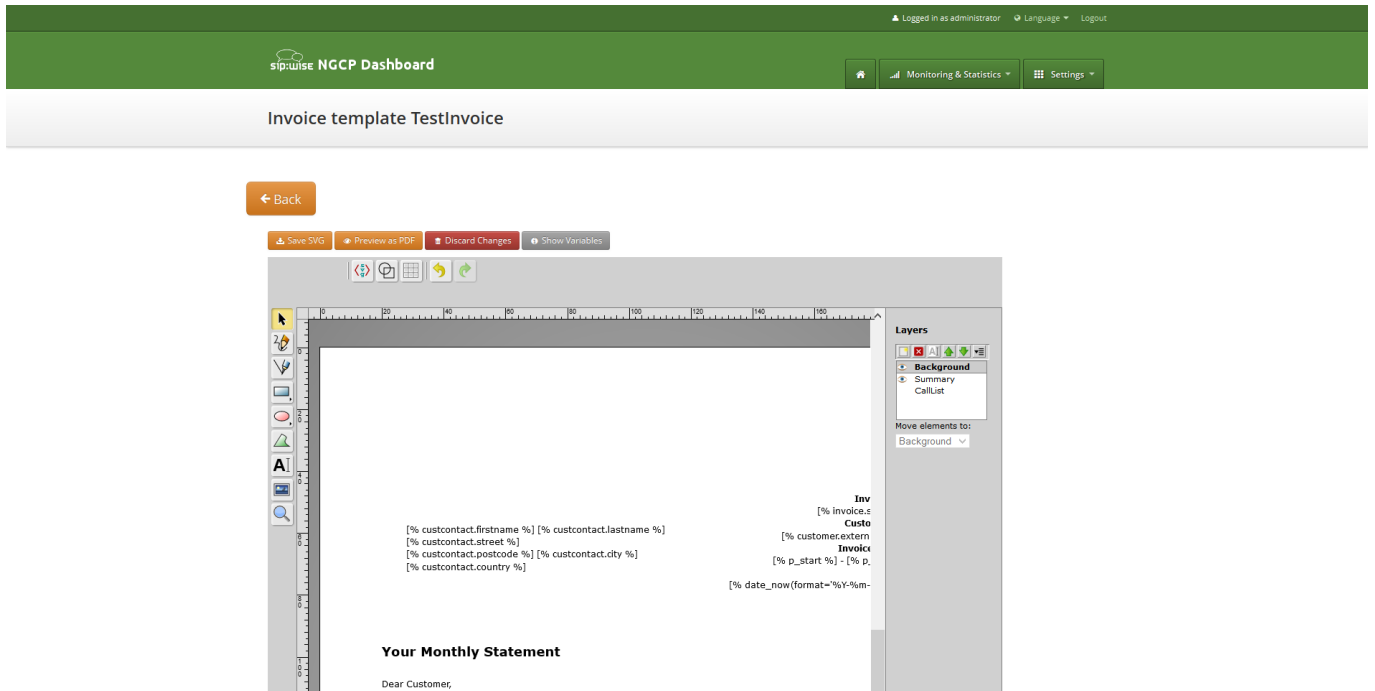
Default template contains three groups elements (<g/>), which can be thought of as pages, or in terms of svg-edit - layers. Layers are:

- **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:







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 displays the 'sip:wise NGCP Dashboard' interface. At the top, there's a navigation bar with 'Monitoring & Statistics' and 'Settings' menus. The main heading is 'Invoice template Rechnung\_v1'. Below this, a toolbar contains buttons for 'Back', 'Save SVG', 'Preview as PDF', 'Discard Changes', and 'Show Variables'. The central editor area shows a preview of an invoice template with the following content:

```

[% 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:

The screenshot shows the 'Invoice template Default' configuration page in the sip:wise NGCP Dashboard. The dashboard header is green and contains the logo, user information, and navigation links. The main content area is white and features a 'Back' button and a toolbar with 'Save SVG', 'Preview as PDF', 'Discard Changes', and 'Show Variables'. A dialog box is open, displaying XML code for the invoice template.

```

<!--{
  [%
    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>

```

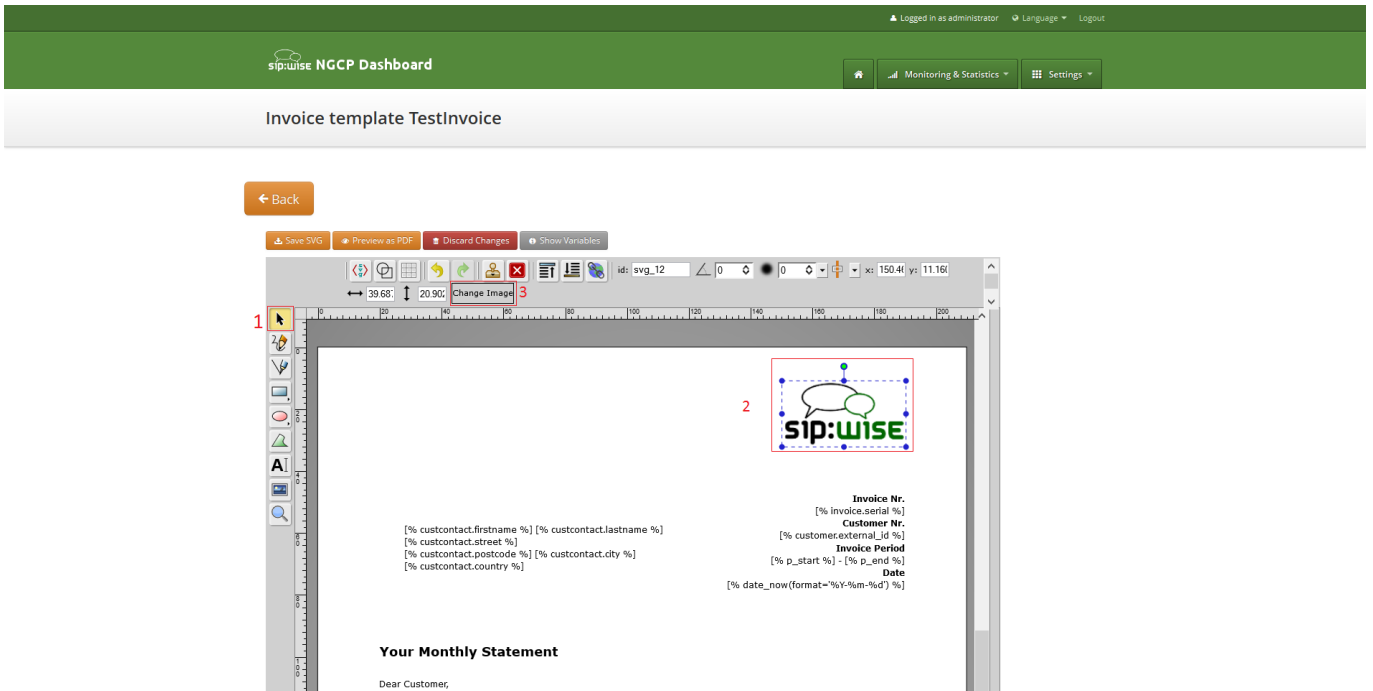
And then [save invoice template changes](#) Section 6.16.2.3.

### 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

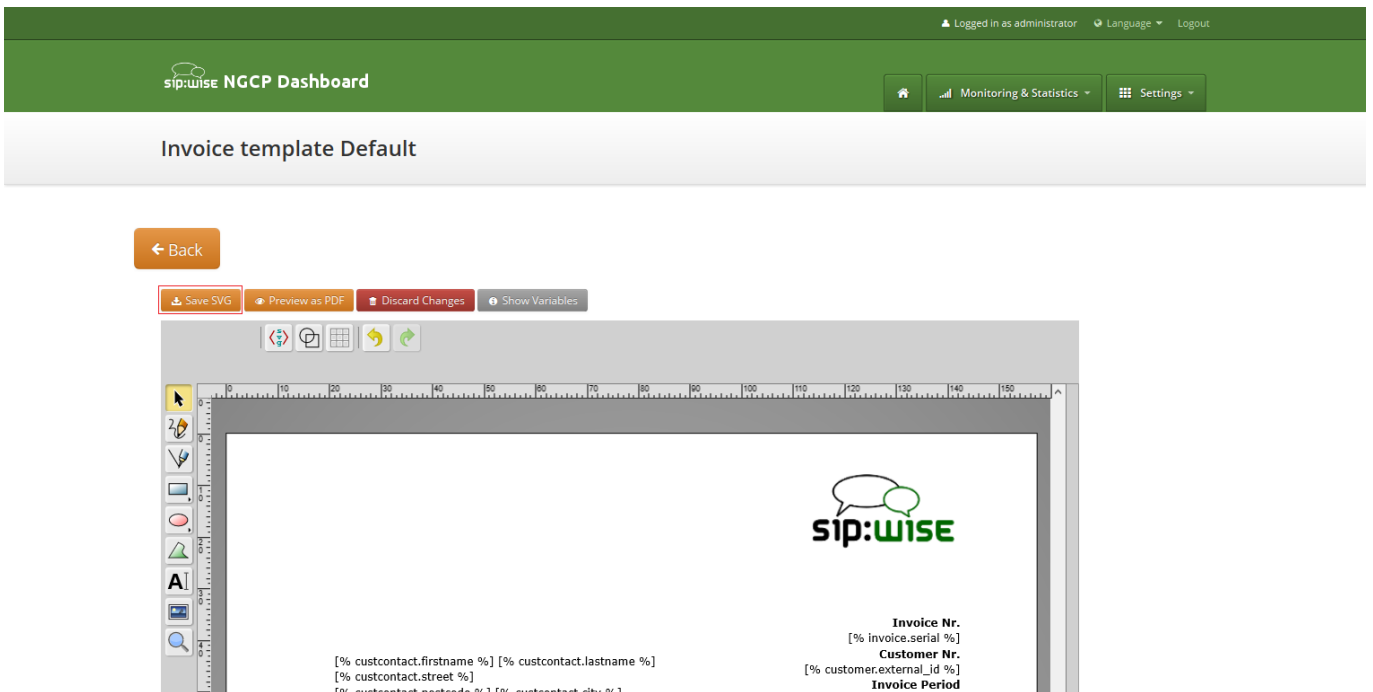
- Make sure that "Select tool" is active.
- Select default logo, clicking on the logo image.
- Press "Change image" button, which should appear on the top toolbar.



After image uploaded [save invoice template changes](#) Section 6.16.2.3.

### 6.16.2.3 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** | Home | 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 web browser window displaying a preview of an invoice. The invoice includes the sip:wise logo and the following details:

**Customer Information:**  
 Customerfirst Customerlast  
 Customerstreet 12/3  
 12345 Customercity  
 Customercountry

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

**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
<b>Total</b>			<b>29.90</b>

### 6.16.3 Invoices Generation

Besides generating invoices on demand using web interface, Sipwise NGCP contains an *invoice generator script* that allows for producing invoices automatically, at regular intervals, for all customers, using the *cron* system tool.



#### Warning

**Automated invoice generation is deprecated since mr4.0 release of NGCP. The invoice generator script will damage billing records in the database.** The rest of the description in "Invoices Generation" section is kept in the handbook for reference purposes only.

---

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 6.16.1.

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.

## 6.17 Email Reports and Notifications

### 6.17.1 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.

### 6.17.2 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 corresponding 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.

### 6.17.3 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:

<b>Template name</b>	<b>passreset_default_email</b>
<b>From</b>	default@sipwise.com
<b>Subject</b>	Password reset email



<b>Body</b>	<p>Dear Customer,</p> <p>Please go to [%url%] to set your password and log into your self-care ← interface.</p> <p>Your faithful Sipwise system</p> <p>--</p> <p>This is an automatically generated message. Do not reply.</p>
-------------	--

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.

#### 6.17.4 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.

---

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

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.

### 6.17.5 Invoice email template

<b>Template name</b>	<b>invoice_default_email</b>
<b>From</b>	<code>default@sipwise.com</code>
<b>Subject</b>	Invoice #[%invoice.serial%] from [%invoice.period_start_obj.ymd%] to [%invoice.period_end_obj.ymd%]
<b>Body</b>	<pre> Dear Customer,  Please find your invoice #[%invoice.serial%] for [%invoice. ←     period_start_obj.month_name%], [%invoice.period_start_obj.year%] in attachment     letter.  Your faithful Sipwise system  --  This is an automatically generated message. Do not reply.</pre>

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.

---

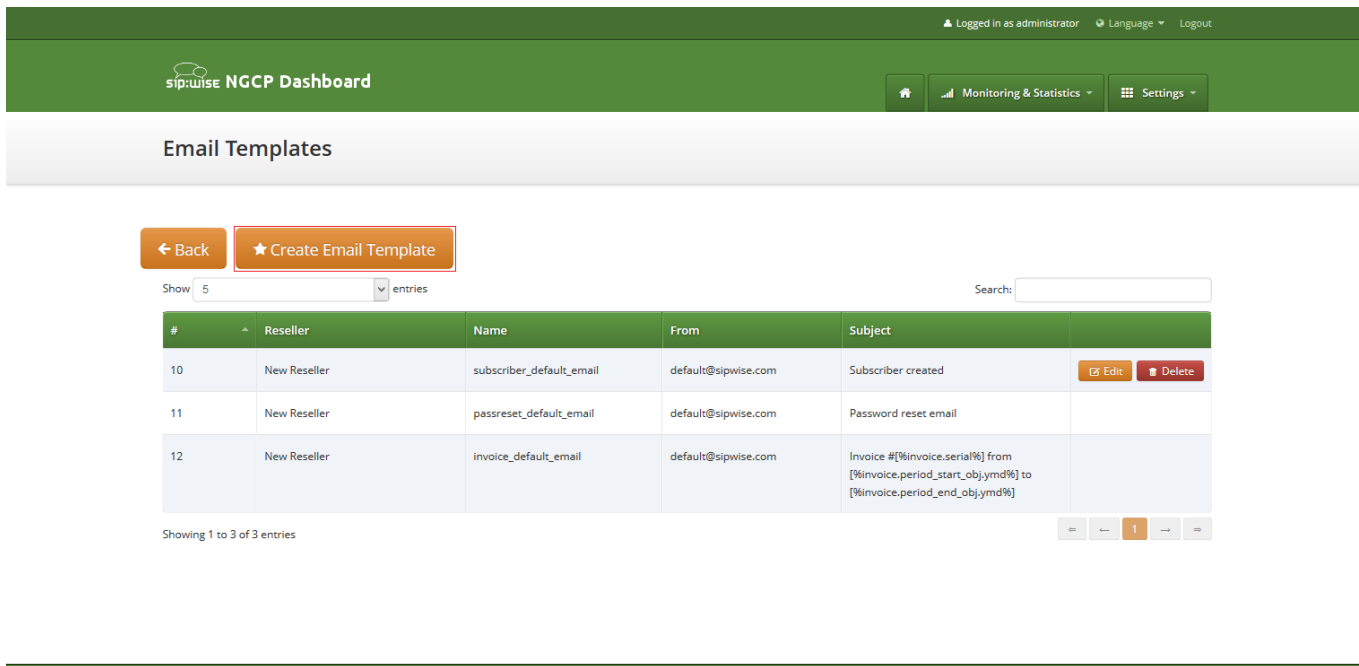
**Contact fields example for the "provider". Replace "provider" to client to access proper "customer" contact fields.**

- [%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%]

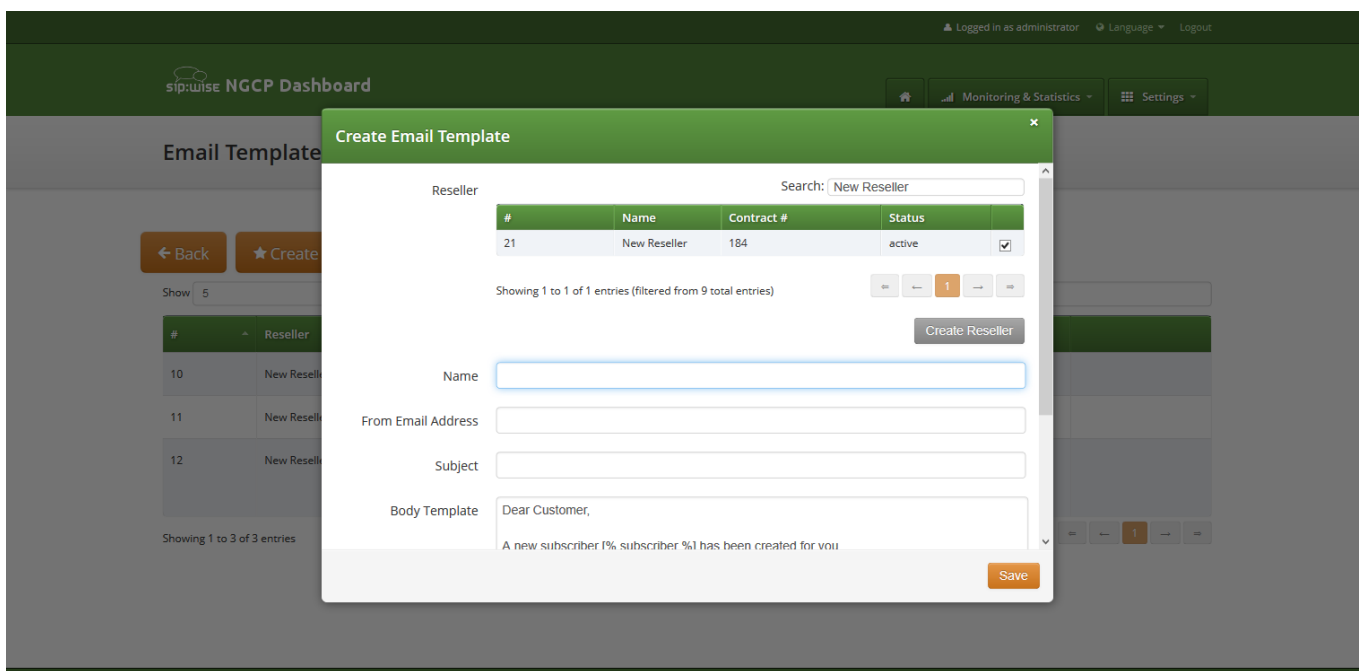
---

### 6.17.6 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 6.17.1 :
  - 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.

## 6.18 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 5.6.

---

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`.
- **32** - enable *Block Incoming Anonymous Calls* by dialing `*32*`, and disable it by dialing `#32`.
- **80** - call using *Call Block Override PIN*, number should be prefixed with a block override PIN configured in admin panel to disable the outgoing user/admin block list and NCOS level for a call. For example, when override PIN is set to 7890, dial `*80*789001000` to call 431000 bypassing block lists.

### 6.18.1 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.

### 6.18.2 Voice Prompts for Vertical Service Code Configuration

Table 7: VSC Voice Prompts

Prompt Handle	Related VSC	Message
vsc_error	any	An error has occurred. Please try again later.
vsc_invalid	wrong code	Invalid feature code.
reject_vsc	any	Vertical service codes are disabled for this line.
vsc_cfu_on	72 (Call Forward Unconditional)	Your unconditional call forward has successfully been activated.
vsc_cfu_off	72 (Call Forward Unconditional)	Your unconditional call forward has successfully been deactivated.
vsc_cfb_on	90 (Call Forward Busy)	Your call forward on busy has successfully been activated.
vsc_cfb_off	90 (Call Forward Busy)	Your call forward on busy has successfully been deactivated.
vsc_cft_on	92 (Call Forward on Timeout)	Your call forward on ring timeout has successfully been activated.
vsc_cft_off	92 (Call Forward on Timeout)	Your call forward on ring timeout has successfully been deactivated.
vsc_cfna_on	93 (Call Forward on Not Available)	Your call forward while not reachable has successfully been activated.
vsc_cfna_off	93 (Call Forward on Not Available)	Your call forward while not reachable has successfully been deactivated.
vsc_speeddial	50 (Speed Dial Slot)	Your speed dial slot has successfully been stored.
vsc_reminder_on	55 (One-Shot Reminder Call)	Your reminder has successfully been activated.
vsc_reminder_off	55 (One-Shot Reminder Call)	Your reminder has successfully been deactivated.
vsc_blockinclr_on	32 (Block Incoming Anonymous Calls)	Your rejection of anonymous calls has successfully been activated.
vsc_blockinclr_off	32 (Block Incoming Anonymous Calls)	Your rejection of anonymous calls has successfully been deactivated.

## 6.19 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\_rtproxy**: Always with rtproxy 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.

## 6.20 XMPP and 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 Customer Self-Care Interface and Menus

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.

### 7.1 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://<ngcp-ip>*. Every subscriber can log in there, change subscriber feature settings, view their call lists, retrieve voicemail messages and trigger calls using the click-to-dial feature.

#### 7.1.1 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 5.2. 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.

#### 7.1.2 Site Customization

As an operator (as well as a Reseller), you can change the branding logo of the Customer Self-Care (CSC) panel and the available languages on the CSC panel. This is possible via the admin web interface.

##### 7.1.2.1 Changing the Logo

For changing the branding logo on a reseller's admin web page and on the CSC panel you just need to access the web interface **as Administrator** and navigate to *Reseller* menu. Once there click on the *Details* button for your selected reseller, finally select *Branding*.

In order to do the same **as Reseller**, login on the admin web interface with the reseller's web credentials, then access the *Panel Branding* menu.

The web panel customisation happens as follows:

1. Press the *Edit Branding* button to start the customisation process.
2. Press the *Browse* button to select an image for the new logo:

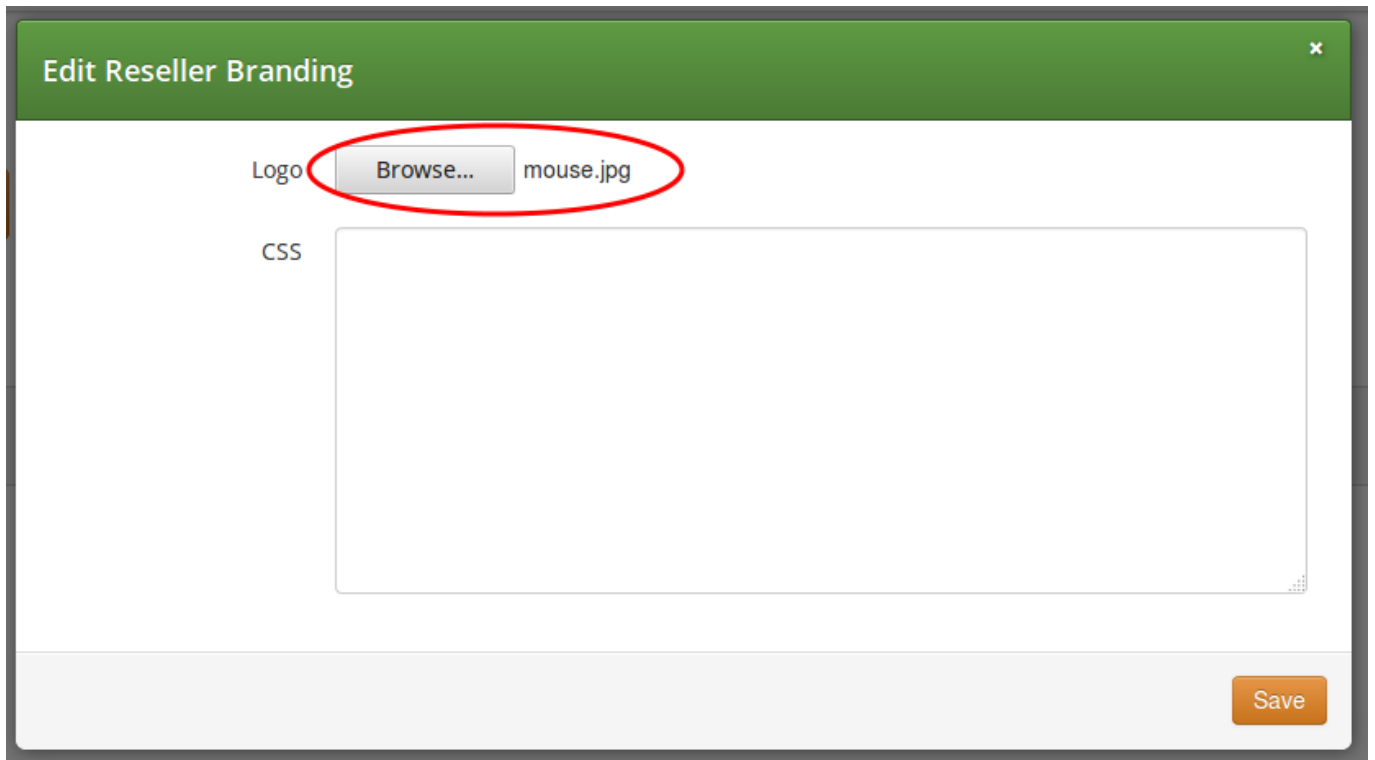



Figure 34: CSC Customisation Step 1: Select an image

3. Press the *Save* button to save changes.
4. Select and copy the auto-generated CSS code from the text box below the uploaded image:

Reseller branding successfully updated

[Edit Branding](#) [Delete Logo](#)

### Custom Logo



You can use the logo by adding the following CSS to the Custom CSS below.

```
#header .brand {  
  background: url(https://10.15.18.227:1443/reseller/3/css/logo/download) no-repeat 0 0;  
  background-size: 280px 32px;  
}
```

### Custom CSS

Figure 35: CSC Customisation Step 2: Copy CSS code

5. Press the *Edit Branding* button again.
6. Paste the CSS code into CSS text box and Save the changes:

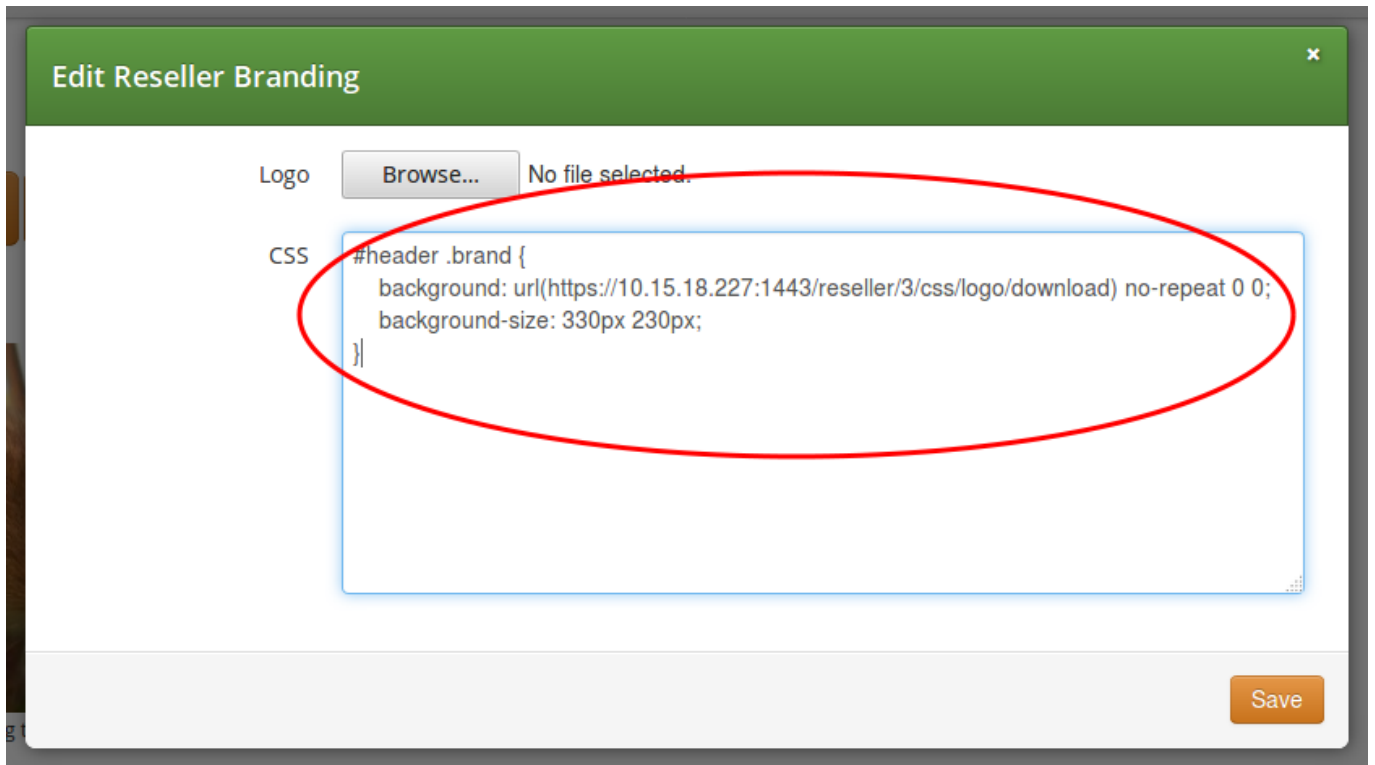


Figure 36: CSC Customisation Step 3: Paste CSS code

7. Now the new logo is already visible on the admin / CSC panel. If you want to hide the Sipwise copyright notice at the bottom of the web panels, add a line of CSS code as shown here:

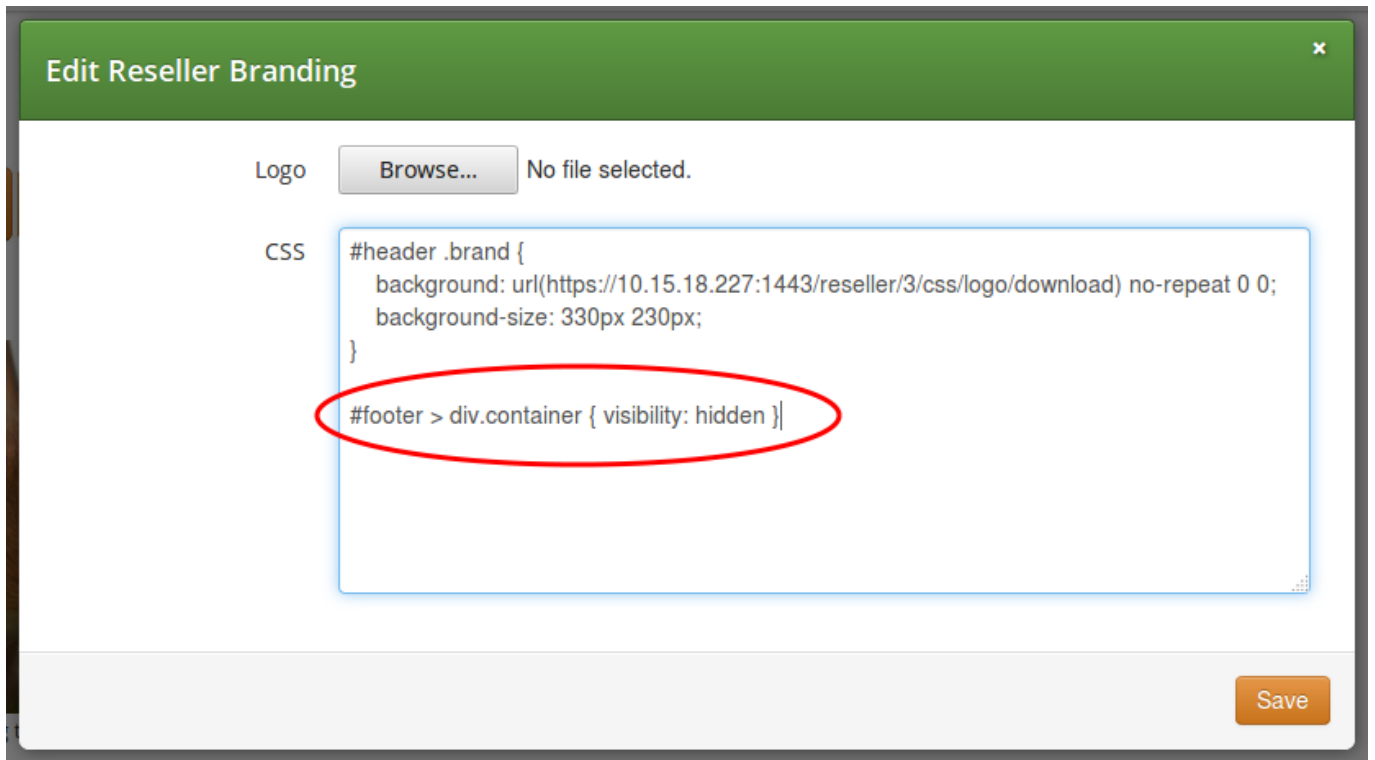


Figure 37: CSC Customisation: Hide copyright notice

8. The final branding data is shown on the admin web panel:

Reseller branding successfully updated

[Edit Branding](#) [Delete Logo](#)

Custom Logo



You can use the logo by adding the following CSS to the Custom CSS below:

```
#header .brand {
  background: url(https://10.15.18.227:1443/reseller/3/css/logo/download) no-repeat 0 0;
  background-size: 280px 32px;
}
```

Custom CSS

```
#header .brand {
  background: url(https://10.15.18.227:1443/reseller/3/css/logo/download) no-repeat 0 0;
  background-size: 330px 230px;
}

#footer > div.container { visibility: hidden }
```

Figure 38: CSC Customisation: Custom data on panel

### 7.1.2.2 Other Website Customisations

The layout and style of NGCP's admin and CSC web panel is determined by a single CSS file: `/usr/share/ngcp-panel/static/css/application.css`

More complex changes, like replacing colour of some web panel components, is possible via the modification of the CSS file.



#### Warning

Only experienced users with profound CSS knowledge are advised to change web panel properties in the main CSS file. *Sipwise does not recommend and also does not support the modification of the main CSS file.*

### 7.1.2.3 Selecting Available Languages

You can also enable/disable specific languages a user can choose from in the CSC panel. Currently, English (`en`), German (`de`), Italian (`it`), Spanish (`es`) and Russian (`ru`) are supported, and the default language is the same as the browser's preferred one.

You can select the *default language* provided by CSC by changing the parameter `www_admin.force_language` in `/etc/ngcp-config/config.yml` file. An example to set the English language as default: `force_language:en`

## 7.2 The Voicemail Menu

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 5.6.) 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.

## 8 Billing Configuration

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

### 8.1 Billing Profiles

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

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

#### 8.1.1 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

sip:wise

### 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 Save

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

- **Reseller:** The reseller this billing profile belongs to.
- **Handle:** A unique, permanently fixed string which is used to attach the billing profile to a customer or SIP peering contract.
- **Name:** A free form string used to identify the billing profile in the *Admin Panel*. This may be changed at any time.
- **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.

### 8.1.2 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**

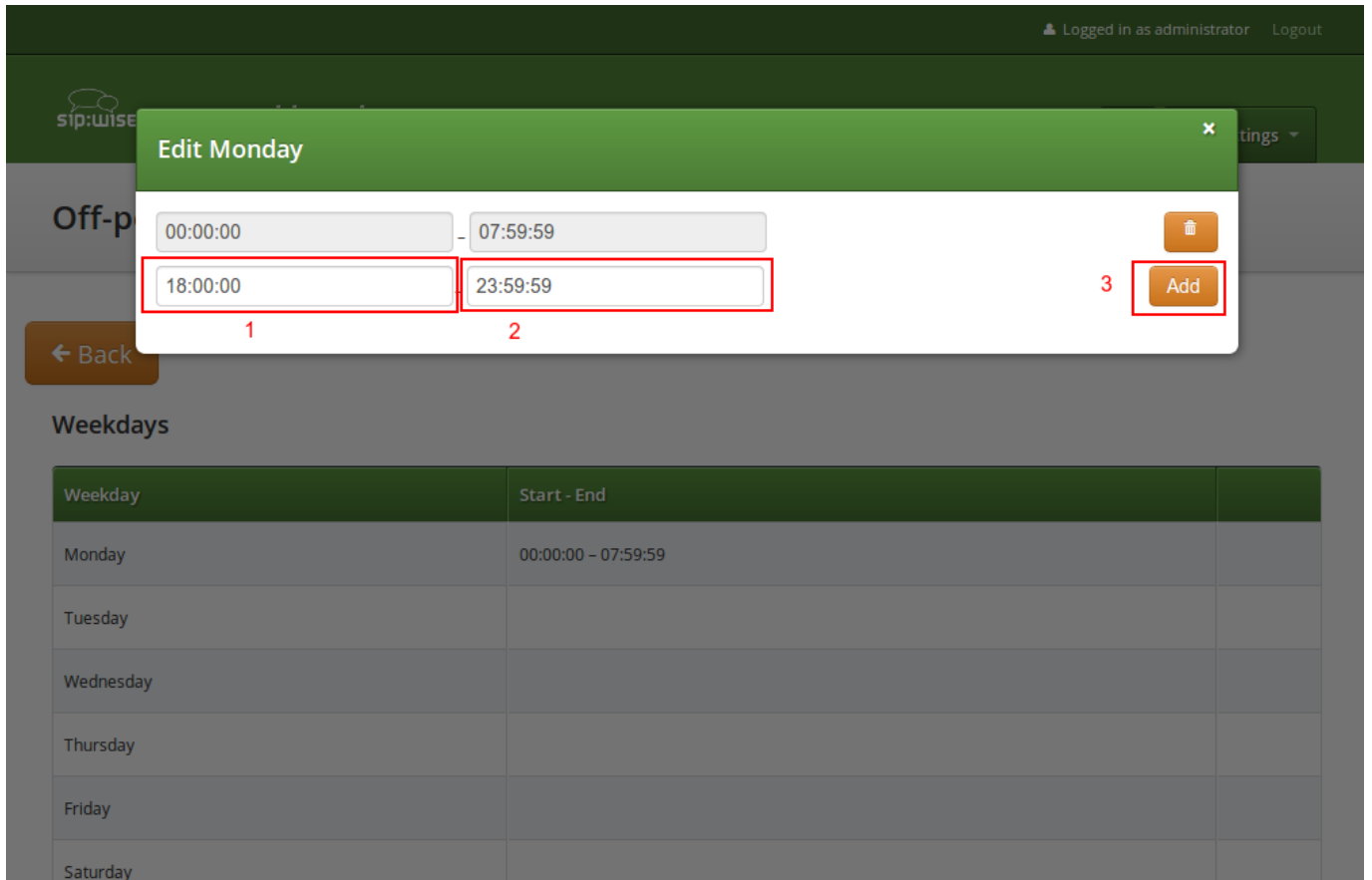
There are several internal services (vsc, conference, voicebox) which will need a specific destination entry with a domain-based destination. If you don't want to charge the same (or nothing) for those services, add a fee for destination `\.local$` there. If you want to charge different amounts for those services, break it down into separate fee entries for `@vsc\.local$`, `@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.

**8.1.3 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 *Off-Peaktimes* button on the billing profile you want to configure.

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

## 8.2 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 8.1.1, 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.

### 8.2.1 Fraud Lock Levels

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

The following lock levels are available:

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



#### Important

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

---



#### Caution

Accounts that were automatically locked by the fraud detection feature will **not** be automatically unlocked 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.

---

#### Note

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

---

## 8.3 Billing Data Export

Regular billing data export is done using CSV (*comma separated values*) files which may be downloaded from the platform using the `cdreexport` 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.

### 8.3.1 Glossary of Terms

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

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

### 8.3.2 File Name Format

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

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

The definition of the specific fields is as follows:

Table 8: 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

### 8.3.3 File Format

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

#### 8.3.3.1 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 9: 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

#### 8.3.3.2 File Body Format for Call Detail Records (CDR)

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

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

Table 10: Default set of system CDR fields

Body Element	Length	Type	Description
CDR_ID	1-10	uint	Internal CDR ID.
UPDATE_TIME	19	timestamp	Timestamp of last modification, including date and time (with seconds precision).
SOURCE_USER_ID	36	string	Internal UUID of calling party subscriber. Value is 0 if calling party is external.
SOURCE_PROVIDER_ID	0-255	string	Internal ID of the contract of calling party provider (i.e. reseller or peer).
SOURCE_EXTERNAL_SUBSCRIBER_ID	0-255	string	External, arbitrary ID of calling party subscriber. (A string value shown as "External ID" property of an NGCP subscriber.)
SOURCE_SUBSCRIBER_ID	1-11	uint	Internal ID of calling party subscriber. Value is 0 if calling party is external.
SOURCE_EXTERNAL_CONTRACT_ID	0-255	string	External, arbitrary ID of calling party customer. (A string value shown as "External ID" property of an NGCP customer/peer.)
SOURCE_ACCOUNT_ID	1-11	uint	Internal ID of calling party customer.
SOURCE_USER	0-255	string	SIP username of calling party.
SOURCE_DOMAIN	0-255	string	SIP domain of calling party.
SOURCE_CLI	0-64	string	CLI of calling party in E.164 format.
SOURCE_CLIR	1	uint	1 for calls with CLIR, 0 otherwise.
SOURCE_IP	0-64	string	IP Address of the calling party.
DESTINATION_USER_ID	36	string	Internal UUID of called party subscriber. Value is 0 if called party is external.
DESTINATION_PROVIDER_ID	0-255	string	Internal ID of the contract of called party provider (i.e. reseller or peer).
DESTINATION_EXTERNAL_SUBSCRIBER_ID	0-255	string	External, arbitrary ID of called party subscriber. (A string value shown as "External ID" property of an NGCP subscriber.)
DESTINATION_SUBSCRIBER_ID	1-11	uint	Internal ID of called party subscriber. Value is 0 if calling party is external.
DESTINATION_EXTERNAL_CONTRACT_ID	0-255	string	External, arbitrary ID of called party customer. (A string value shown as "External ID" property of an NGCP customer/peer.)
DESTINATION_ACCOUNT_ID	1-11	uint	Internal ID of called party customer.

Table 10: (continued)

Body Element	Length	Type	Description
DESTINATION_USER	0-255	string	Final SIP username of called party.
DESTINATION_DOMAIN	0-255	string	Final SIP domain of called party.
DESTINATION_USER_IN	0-255	string	Incoming SIP username of called party, after applying inbound rewrite rules.
DESTINATION_DOMAIN_IN	0-255	string	Incoming SIP domain of called party, after applying inbound rewrite rules.
DESTINATION_USER_DIALED	0-255	string	The user-part of the SIP Request URI as received by NGCP.
PEER_AUTH_USER	0-255	string	Username used to authenticate towards peer.
PEER_AUTH_REALM	0-255	string	Realm used to authenticate towards peer.
CALL_TYPE	3-4	string	The type of the call - one of: call: normal call cfu: call forward unconditional cfb: call forward busy cft: call forward timeout cfna: call forward not available cfs: call forward for SMS
CALL_STATUS	2-8	string	The final call status - one of: ok: successful call busy: called party busy noanswer: no answer from called party cancel: cancel from caller offline: called party offline timeout: no reply from called party other: unspecified, see CALL_CODE field for details
CALL_CODE	3	string	The final SIP status code.
INIT_TIME	23	timestamp	Timestamp of call initiation (SIP <i>INVITE</i> received from calling party). Includes date, time with milliseconds (3 decimals).
START_TIME	23	timestamp	Timestamp of call establishment (final SIP response received from called party). Includes date, time with milliseconds (3 decimals).
DURATION	4-13	fixed precision (3 decimals)	Length of call (calculated from START_TIME) including milliseconds (3 decimals).

Table 10: (continued)

Body Element	Length	Type	Description
CALL_ID	0-255	string	The SIP Call-ID.
RATING_STATUS	2-7	string	The internal rating status of the CDR - one of: unrated: not rated ok: successfully rated failed: error while rating Currently always ok or unrated, depending on whether rating is enabled or not.
RATED_AT	0-19	datetime	Time of rating, including date and time (with seconds precision). Empty if CDR is not rated.
SOURCE_CARRIER_COST	7-14	fixed precision (6 decimals)	The originating carrier cost that the carrier (i.e. SIP peer) charges for the calls routed to his network, or empty if CDR is not rated. <i>PLEASE NOTE: Only available in system exports, not for resellers.</i>
SOURCE_CUSTOMER_COST	7-14	fixed precision (6 decimals)	The originating customer cost, or empty if CDR is not rated.
SOURCE_CARRIER_ZONE	0-127	string	Name of the originating carrier billing zone, or onnet if data is not available. <i>PLEASE NOTE: Only available in system exports, not for resellers.</i>
SOURCE_CUSTOMER_ZONE	0-127	string	Name of the originating customer billing zone, or empty if CDR is not rated.
SOURCE_CARRIER_DETAIL	0-127	string	Description of the originating carrier billing zone, or platform internal if data is not available. <i>PLEASE NOTE: Only available in system exports, not for resellers.</i>
SOURCE_CUSTOMER_DETAIL	0-127	string	Description of the originating customer billing zone, or empty if CDR is not rated.
SOURCE_CARRIER_FREE_TIME	1-10	uint	The number of free time seconds used on originating carrier side, or empty if CDR is not rated. <i>PLEASE NOTE: Only available in system exports, not for resellers.</i>

Table 10: (continued)

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

Table 10: (continued)

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

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

```
'15','2013-03-26 22:09:11','a84508a8-d256-4c80-a84e-820099a827b0','1','','1','','2','testuser1','192.168.51.133','4311001','0','192.168.51.1',
```

```
'94d85b63-8f4b-43f0-b3b0-221c9e3373f2','1','','3','','4','testuser3',
'192.168.51.133','testuser3','192.168.51.133','testuser3','','','call','ok','200',
'2013-03-25 20:24:50.890','2013-03-25 20:24:51.460','10.880','44449842',
'ok','2013-03-25 20:25:27','0.00','24.00','onnet','testzone','platform internal',
'testzone','0','0','0.00','200.00','','foo','','foo','0','0',
'0.00','','','0','0.00','','','0'
```

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

---

#### Note

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

---

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

### 8.3.3.3 Extra fields that can be exported to CDRs

#### Supplementary Data

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

---

#### Note

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

---

Table 11: Supplementary data in CDR fields

Body Element	Length	Type	Description
SOURCE_GPP0	0-255	string	Supplementary data field 0 of calling party.
SOURCE_GPP1	0-255	string	Supplementary data field 1 of calling party.
SOURCE_GPP2	0-255	string	Supplementary data field 2 of calling party.
SOURCE_GPP3	0-255	string	Supplementary data field 3 of calling party.
SOURCE_GPP4	0-255	string	Supplementary data field 4 of calling party.
SOURCE_GPP5	0-255	string	Supplementary data field 5 of calling party.
SOURCE_GPP6	0-255	string	Supplementary data field 6 of calling party.
SOURCE_GPP7	0-255	string	Supplementary data field 7 of calling party.
SOURCE_GPP8	0-255	string	Supplementary data field 8 of calling party.
SOURCE_GPP9	0-255	string	Supplementary data field 9 of calling party.
DESTINATION_GPP0	0-255	string	Supplementary data field 0 of called party.

Table 11: (continued)

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

### **Account balance details (prepaid calls)**

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

---

#### **Tip**

Such fields may also be required to integrate sip:provider CE with legacy billing systems.

---

#### **Note**

Please be aware that pre-paid billing functionality is only available in Sipwise *sip:provider PRO* and *sip:carrier* products.

---

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

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

Examples:

- `source_customer_cash_balance_before`
-



- destination\_customer\_profile\_package\_id



### Important

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

### 8.3.3.4 File Body Format for Event Detail Records (EDR)

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

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

Table 12: Default set of system EDR fields

Body Element	Length	Type	Description
EVENT_ID	1-11	uint	Internal EDR ID.
TYPE	0-255	string	The type of the event - one of: <i>start_profile</i> : A subscriber profile has been newly assigned to a subscriber. <i>end_profile</i> : A subscriber profile has been removed from a subscriber. <i>update_profile</i> : A subscriber profile has been changed for a subscriber. <i>start_huntgroup</i> : A subscriber has been provisioned as PBX / hunting group. <i>end_huntgroup</i> : A subscriber has been deprovisioned as PBX / hunting group. <i>start_ivr</i> : A subscriber has a new call-forward to Auto-Attendant. <i>end_ivr</i> : A subscriber has removed a call-forward to Auto-Attendant.
CONTRACT_EXTERNAL_ID	0-255	string	The external ID of the customer. (A string value shown as "External ID" property of an NGCP customer.)
COMPANY	0-127	string	The company name of the customer's contact.
SUBSCRIBER_EXTERNAL_ID	0-255	string	The external ID of the subscriber. (A string value shown as "External ID" property of an NGCP subscriber.) <i>PLEASE NOTE: This field is empty in case of <i>start_huntgroup</i> and <i>end_huntgroup</i> events.</i>

Table 12: (continued)

Body Element	Length	Type	Description
PILOT_PRIMARY_NUMBER	0-64	string	The pilot subscriber's primary number (HPBX subscribers). <i>PLEASE NOTE: This is not included in default set of EDR fields from NGCP version mr5.0 upwards.</i>
PRIMARY_NUMBER	0-64	string	The VoIP number of the subscriber with the highest ID (DID or primary number).
OLD_PROFILE_NAME	0-255	string	The old status of the event. Depending on the event_type: start_profile: Empty. end_profile: The name of the subscriber profile which got removed from the subscriber. update_profile: The name of the former subscriber profile which got updated. start_huntgroup: Empty. end_huntgroup: Empty. start_ivr: Empty. end_ivr: Empty.
NEW_PROFILE_NAME	0-255	string	The new status of the event. Depending on the event_type: start_profile: The name of the subscriber profile which got assigned to the subscriber. end_profile: Empty. update_profile: The name of the new subscriber profile which got applied. start_huntgroup: Empty. end_huntgroup: Empty. start_ivr: Empty. end_ivr: Empty.
TIMESTAMP	23	timestamp	Timestamp of event. Includes date, time with milliseconds (3 decimals).
RESELLER_ID	1-11	uint	Internal ID of the reseller which the event belongs to. <i>PLEASE NOTE: Only available in system exports, not for resellers.</i>
<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", "1"

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

---

#### Note

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

---

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

#### 8.3.3.5 Extra fields that can be exported to EDRs

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

Table 13: Supplementary data in EDR fields

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

Table 13: (continued)

Body Element	Length	Type	Description
PILOT_PRIMARY_ALIAS_US ERNAME_BEFORE	0-255	string	The primary alias of the subscriber's pilot subscriber, before number updates during the operation.
PILOT_PRIMARY_ALIAS_US ERNAME_AFTER	0-255	string	The primary alias of the subscriber's pilot subscriber, after number updates during the operation.
FIRST_NON_PRIMARY_ALIA S_USERNAME_BEFORE_AF TER	0-255	string	Equals FIRST_NON_PRIMARY_ALIAS_USERNAME_BEFORE, if the value is not NULL, otherwise it's the same as FIRST_NON_PRIMARY_ALIAS_USERNAME_AFTER.
PILOT_FIRST_NON_PRIMAR Y_ALIAS_USERNAME_BEFOR E_AFTER	0-255	string	Equals PILOT_FIRST_NON_PRIMARY_ALIAS_USERNAME_BEFORE, if the value is not NULL, otherwise it's the same as PILOT_FIRST_NON_PRIMARY_ALIAS_USERNAME_AFTER.

### 8.3.3.6 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
```

### 8.3.4 File Transfer

Billing files are created twice per hour at minutes 25 and 55 and are stored in the home directory of the `cdrexp` 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 `cdrexp`.

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 `cdrexp` 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 `cdrexp` user is kept in a jailed environment on the system, so it has only access to a very limited set of commandline utilities.

---

## 9 Provisioning REST API Interface

The sip:provider CE provides the REST API interface for interconnection with 3rd party tools.

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.

### 9.1 API Workflows for Customer and Subscriber Management

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

The standard life cycle of a customer and subscriber is:

1. Create customer contact
2. Create customer
3. Create subscribers within customer
4. Modify subscribers
5. Modify subscriber preferences (features)
6. Terminate subscriber
7. Terminate customer

The boiler-plate to access the REST API is described in the online API documentation at [/api/#auth](#). A simple example in Perl using password authentication looks as follows:

```
#!/usr/bin/perl -w
use strict;
use v5.10;

use LWP::UserAgent;
use JSON qw();

my $uri = 'https://ngcp.example.com:1443';
my $ua = LWP::UserAgent->new;
my $user = 'myusername';
my $pass = 'mypassword';
$ua->credentials('ngcp.example.com:1443', 'api_admin_http', $user, $pass);
my ($req, $res);
```

For each customer you create, you need to assign a billing profile id. You either have the ID stored somewhere else, or you need to fetch it by searching for the billing profile handle.

```
my $billing_profile_handle = 'my_test_profile';
$req = HTTP::Request->new('GET', "$uri/api/billingprofiles/?handle=$billing_profile_handle" ←
);
$res = $ua->request($req);
if($res->code != 200) {
    die "Failed to fetch billing profile: ".$res->decoded_content."\n";
}
my $billing_profile = JSON::from_json($res->decoded_content);
my $billing_profile_id = $billing_profile->{_embedded}->{'ngcp:billingprofiles'}->{id};
say "Fetched billing profile, id is $billing_profile_id";
```

A customer is mainly a billing container for subscribers without a real identification other than the *external\_id* property you might have stored somewhere else (e.g. the ID of the customer in your CRM). To still easily identify a customer, a customer contact is required. It is created using the */api/customercontacts/* resource.

```
$req = HTTP::Request->new('POST', "$uri/api/customercontacts/");
$req->header('Content-Type' => 'application/json');
$req->content(JSON::to_json({
    firstname => 'John',
    lastname => 'Doe',
    email => 'john.doe@example.com'
}));
$res = $ua->request($req);
if($res->code != 201) {
    die "Failed to create customer contact: ".$res->decoded_content."\n";
}
my $contact_id = $res->header('Location');
$contact_id =~ s/^.+\/(\d+)\$\/$1/; # extract the ID from the Location header
say "Created customer contact, id is $contact_id";
```



### Important

To get the ID of the recently created resource, you need to parse the *Location* header. In future, this approach will be changed for POST requests. The response will also optionally return the ID of the resource. It will be controlled via the *Prefer: return=representation* header as it is already the case for PUT and PATCH.



### Warning

The example above implies the fact that you access the API via a reseller user. If you are accessing the API as the admin user, you also have to provide a *reseller\_id* parameter defining the reseller this contact belongs to.

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

```
$req = HTTP::Request->new('POST', "$uri/api/customers/");
```

```

$req->header('Content-Type' => 'application/json');
$req->content(JSON::to_json({
    status => 'active',
    contact_id => $contact_id,
    billing_profile_id => $billing_profile_id,
    type => 'sipaccount',
    external_id => undef, # can be set to your crm's customer id
}));
$res = $ua->request($req);
if($res->code != 201) {
    die "Failed to create customer: ".$res->decoded_content."\n";
}
my $customer_id = $res->header('Location');
$customer_id =~ s/^.+\(/(\d+)\$/1/; # extract the ID from the Location header
say "Created customer, id is $customer_id";

```

Once you have created the customer, you can add subscribers to it. One customer can hold multiple subscribers, up to the *max\_subscribers* property which can be set via */api/customers/*. If this property is not defined, a virtually unlimited number of subscribers can be added.

```

$req = HTTP::Request->new('POST', "$uri/api/subscribers/");
$req->header('Content-Type' => 'application/json');
$req->content(JSON::to_json({
    status => 'active',
    customer_id => $customer_id,
    primary_number => { cc => 43, ac => 9876, sn => 10001 }, # the main number
    alias_numbers => [ # as many alias numbers the subscriber can be reached at (or skip ←
        param if none)
        { cc => 43, ac => 9877, sn => 10001 },
        { cc => 43, ac => 9878, sn => 10001 }
    ],
    username => 'test_10001',
    domain => 'ngcp.example.com',
    password => 'secret subscriber pass',
    webusername => 'test_10001',
    webpassword => undef, # set undef if subscriber shouldn't be able to log into sipwise ←
        csc
    external_id => undef, # can be set to the operator crm's subscriber id
}));
$res = $ua->request($req);
if($res->code != 201) {
    die "Failed to create subscriber: ".$res->decoded_content."\n";
}
my $subscriber_id = $res->header('Location');
$subscriber_id =~ s/^.+\(/(\d+)\$/1/; # extract the ID from the Location header
say "Created subscriber, id is $subscriber_id";

```



**Important**

A domain must exist before creating a subscriber. You can create the domain via `/api/domains/`.

At that stage, the subscriber can connect both via SIP and XMPP, and can be reached via the primary number, all alias numbers, as well as via the SIP URI.

If you want to set call forwards for the subscribers, then perform an API call as follows.

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

To provision certain features of a subscriber, you can manipulate the subscriber preferences. You can find a full list of preferences available for a subscriber at `/api/subscriberpreferencedefs/`.

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

Resources modification can be done via the GET/PUT combination. Alternatively, you can add, modify or delete single properties of a resource without actually fetching the whole resource. See an example below where we terminate the status of a customer using the PATCH method.

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

## 10 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 7 steps:

- Generation or editing of configuration templates and/or configuration values.
- Generation of the configuration files based on configuration templates and configuration values defined in `config.yml`, `constants.yml` and `network.yml` files.
- Execution of `prebuild` commands if defined for a particular configuration file or configuration directory.
- Placement of the generated configuration file in the target directory. This step is called `build` in the configuration framework.
- Execution of `postbuild` commands if defined for that configuration file or configuration directory.
- Execution of `services` commands if defined for that configuration file or configuration directory. This step is called `services` in the configuration framework.
- Saving of the generated changes. This step is called `commit` in the configuration framework.

### 10.1 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.

#### 10.1.1 .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 we are changing the IP used by kamailio load balancer on interface `eth0` to IP 1.2.3.4. This will change kamailio's listen IP address, when the configuration file is generated. A quick look to the template file under `/etc/ngcp-config/templates/etc/kamailio/` will show a line like this:

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

After applying the changes with the `ngcpcfg apply 'my commit message'` command, a new configuration file will be created under `/etc/kamailio/lb/kamailio.cfg` with the proper values taken from the main configuration files (in this case `network.yml`):

```
listen=udp:1.2.3.4:5060
```

All the low-level configuration is provided by these `.tt2` template files and the corresponding `config.yml` file. Anyway, advanced users might require a more particular configuration.

Instead of editing `.tt2` files, the configuration framework recognises `.customtt.tt2` files. These files are the same as `.tt2`, but they have higher priority when the configuration framework creates the final configuration files. 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
ngcpcfg apply 'my commit message'
```

The `ngcpcfg` 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 `ngcpcfg`'s template files (all the ones with the `.tt2` suffix).

---

### 10.1.2 `.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
```

### 10.1.3 .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.

## 10.2 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.

## 10.3 ngcpcfg and its command line options

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

### 10.3.1 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.

### 10.3.2 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/
```

### 10.3.3 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.

### 10.3.4 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.

### 10.3.5 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.

### 10.3.6 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.

### 10.3.7 help

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

### 10.3.8 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.

### 10.3.9 pull

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

### 10.3.10 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.

### 10.3.11 services

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



### 10.3.12 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.

## 11 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.

### 11.1 General Structure

The basic structure of the file looks like this:

```
hosts:
  self:
    role:
      - proxy
      - lb
      - mgmt
    interfaces:
      - eth0
      - lo
    eth0:
      ip: 192.168.51.213
      netmask: 255.255.255.0
      type:
        - sip_ext
        - rtp_ext
        - web_ext
        - web_int
    lo:
      ip: 127.0.0.1
      netmask: 255.255.255.0
      type:
        - sip_int
        - ha_int
```

Some more complete, sample configuration is shown in [network.yml Overview](#) Section [B.3](#) section of the handbook.

The file contains all configuration parameters under the main key: `hosts`

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

#### 11.1.1 Available Host Options

There are three different main sections for a host in the config file, which are *role*, *interfaces* and the actual interface definitions.

- *role*: The role setting is an array defining which logical roles a node will act as. Possible entries for this setting are:

- *mgmt*: This entry means the host is acting as management node for the platform. In a sip:provider CE system this option must always be set. The management node exposes the admin and CSC panels to the users and the APIs to external applications and is used to export CDRs.
- *lb*: This entry means the host is acting as SIP load-balancer for the platform. In a sip:provider CE system this option must always be 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 system this option must always be 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 system 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 system this option must always be set. The RTP relay node runs the *rtpengine* NGCP component.
- *interfaces*: The interfaces setting is an array defining all interface names in the system. The actual interface details are set in the actual interface settings below. It typically includes `lo`, `eth0`, `eth1` physical and a number of virtual interfaces, like: `bond0`, `vlanXXX`
- *<interface name>*: After the interfaces are defined in the *interfaces* setting, each of those interfaces needs to be specified as a separate set of parameters.

Additional main parameters of a node:

- *dbnode*: the sequence number (unique ID) of the node in the database cluster; not used in sip:provider CE system

### 11.1.2 Interface Parameters

- *hwaddr*: MAC address of the interface
- *ip*: IPv4 address of the node
- *v6ip*: IPv6 address of the node; optional
- *netmask*: IPv4 netmask
- *advertised\_ip*: the IP address that is used in SIP messages when the NGCP system is behind NAT/SBC. An example of such a deployment is *Amazon AMI*, where the server doesn't have a public IP, so *load-balancer* component of NGCP needs to know what his public domain is (→ *advertised\_ip*).
- *type*: type of services that the node provides; these are usually the VLANs defined for a particular NGCP system.

---

#### Note

You can assign a type only once per node.

---

Available types are:

- `api_int`: internal, API-based communication interface. It is used for the internal communication of such services as faxserver, fraud detection and others.
- `aux_ext`: interface for potentially insecure external components like remote system log collection service.
- `mon_ext`: remote monitoring interface (e.g. SNMP)
- `rtp_ext`: main (external) interface for media traffic
- `sip_ext`: main (external) interface for SIP signalling traffic between NGCP and other SIP endpoints
- `sip_ext_incoming`: additional, optional interface for incoming SIP signalling traffic
- `sip_int`: internal SIP interface used by NGCP components (*lb*, *proxy*, etc.)
- `ssh_ext`: command line (SSH) remote access interface
- `web_ext`: interface for web-based or API-based provisioning and administration
- `web_int`: interface for the administrator's web panel, his API and generic internal API communication

---

### Note

Please note that, apart from the standard ones described so far, there might be other *types* defined for a particular NGCP system.

---

- `vlan_raw_device`: tells which physical interface is used by the particular VLAN
- `post_up`: routes can be defined here (interface-based routing)
- `bond_XY`: specific to "bond0" interface only; these contain Ethernet bonding properties

## 11.2 Advanced Network Configuration

You have a typical deployment now and you are good to go, however you may need to do extra configuration depending on the devices you are using and functionality you want to achieve.

### 11.2.1 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 `ngcpconfig 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 the next section Section 11.2.2 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.

## 11.2.2 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.

## 12 Software Upgrade

### 12.1 Release Notes

The sip:provider CE version mr4.5.7 has several important changes comparing to the previous release. Please find the complete changelog in our release notes [on our WEB site](#).

### 12.2 Upgrade sip:provider CE from previous mr4.4/mr4.5 versions to mr4.5.7 LTS version

The sip:provider CE system upgrade to mr4.5.7 will perform a couple of fundamental steps:

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



#### Warning

ensure you are using Sipwise APT repositories. Public Debian mirrors may not provide packages for old Debian releases anymore. Also they can be outdated. Consider to use Sipwise repositories for the time of upgrade.

---

Execute the following commands as *root*:

```
echo "# Please visit /etc/apt/sources.list.d/ instead." > /etc/apt/sources.list

mkdir -p /etc/apt/sources.list.d
for file in /etc/apt/sources.list.d/*.list ; do mv "${file}" "${file}.DISABLED" ; done

NGCP_CURRENT_VERSION=$(cat /etc/ngcp_version)
cat > /etc/apt/sources.list.d/debian.list << EOF
# Debian repositories, deployed via upgrade ${NGCP_CURRENT_VERSION}->mr4.5.7
deb https://debian.sipwise.com/debian/ jessie main contrib non-free
#deb-src https://debian.sipwise.com/debian/ jessie main contrib non-free
#
deb https://debian.sipwise.com/debian-security/ jessie-security main contrib non-free
#deb-src https://debian.sipwise.com/debian-security/ jessie-security main contrib non-free
#
deb https://debian.sipwise.com/debian/ jessie-updates main contrib non-free
#deb-src https://debian.sipwise.com/debian/ jessie-updates main contrib non-free
EOF
```

```
NGCP_CURRENT_VERSION=$(cat /etc/ngcp_version)
cat > /etc/apt/sources.list.d/sipwise.list << EOF
# NGCP_MANAGED_FILE
# Sipwise repository, deployed via upgrade ${NGCP_CURRENT_VERSION}->mr4.5.7
deb https://deb.sipwise.com/spce/${NGCP_CURRENT_VERSION}/ jessie main
#deb-src https://deb.sipwise.com/spce/${NGCP_CURRENT_VERSION}/ jessie main
EOF
```

Run "apt-get update" and ensure you have no warnings/errors here.

For upgrading the sip:provider CE to release mr4.5.7, execute the following commands:

```
NGCP_CURRENT_VERSION=$(cat /etc/ngcp_version)
sed -i "s/${NGCP_CURRENT_VERSION}/mr4.5.7/" /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 mr4.5.7 from previous release mr4.4.\* or previous builds mr4.5.\* only. The script ngcp-upgrade will find all the possible target releases for the upgrade and allow to choose the proper one.

---

---

**Note**

If there is an error during upgrade, the ngcp-upgrade script will request you to solve it. Once you've fixed the problem just re-execute ngcp-upgrade again and it will continue from the previous step.

---

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

The upgrade process will take several minutes, depending on your network connection and server performance. After everything has been updated successfully, it will finally ask you to reboot your system. Confirm to let the system reboot (it will boot with an updated kernel).

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.

When all finishes successfully check that replication is running. Check `ngcp-status`. Finally, do a basic functionality test. Check web interface, register two test subscribers and perform a test call between them to ensure call routing works.

---

**Note**

You can find a backup of some important configuration files of your existing installation under `/var/backup/ngcp-mr4.5.7-*` (where \* is a place holder for a timestamp) in case you need to roll back something at any time. A log file of the upgrade procedure is available at `/var/backup/ngcp-mr4.5.7-*/upgrade.log`.

---



## 12.3 Upgrade sip:provider CE from previous mr3.8 LTS version to mr4.5.7 LTS version

The sip:provider CE system upgrade from mr3.8 LTS version to mr4.5.7 LTS version will perform a couple of fundamental steps:

- Verify APT source lists
- Upgrade Debian from version 7 (wheezy) to version 8 (jessie)
- Upgrade NGCP software packages
- Upgrade NGCP configuration templates
- Upgrade NGCP DB schema
- Upgrade the base system within Debian 8 (jessie) to the latest package versions



### Warning

ensure you are using Sipwise APT repositories. Some public Debian mirrors may not provide wheezy packages anymore! Execute the following commands as *root*:

```
echo "# Please visit /etc/apt/sources.list.d/ instead." > /etc/apt/sources.list

mkdir -p /etc/apt/sources.list.d
for file in /etc/apt/sources.list.d/*.list ; do mv "${file}" "${file}.DISABLED" ; done

NGCP_CURRENT_VERSION=$(cat /etc/ngcp_version)
cat > /etc/apt/sources.list.d/debian.list << EOF
## custom sources.list, deployed via upgrade ${NGCP_CURRENT_VERSION}->mr4.5.7
#
# Debian repositories
deb http://debian.sipwise.com/debian/ wheezy main contrib non-free
#deb-src http://debian.sipwise.com/debian/ wheezy main contrib non-free
#
deb http://debian.sipwise.com/debian-security/ wheezy-security main contrib non-free
#deb-src http://debian.sipwise.com/debian-security/ wheezy-security main contrib non-free
#
deb http://debian.sipwise.com/debian/ wheezy-updates main contrib non-free
#deb-src http://debian.sipwise.com/debian/ wheezy-updates main contrib non-free
EOF

NGCP_CURRENT_VERSION=$(cat /etc/ngcp_version)
cat > /etc/apt/sources.list.d/sipwise.list << EOF
# NGCP_MANAGED_FILE - do not remove this line if it should be automatically handled
#
# Sipwise repository
deb http://deb.sipwise.com/spce/${NGCP_CURRENT_VERSION}/ wheezy main
#deb-src http://deb.sipwise.com/spce/${NGCP_CURRENT_VERSION}/ wheezy main
EOF
```

Run "apt-get update" and ensure you have no warnings/errors here.

---

**Note**

Re-check you have all the latest packages/hotfixes installed, execute as *root*:

---

```
apt-get update && apt-get upgrade && \  
ngcp-update-db-schema && ngcp-update-cfg-schema && \  
ngcpcfg apply
```

ensure you have 0 everywhere below *apt-get upgrade* output:

```
> ... && apt-get upgrade  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

For upgrading the sip:provider CE from mr3.8 LTS version to mr4.5.7 LTS release, execute the following commands as *root*:

```
NGCP_CURRENT_VERSION=$(cat /etc/ngcp_version)  
sed -i "s/$NGCP_CURRENT_VERSION/mr4.5.7/" /etc/apt/sources.list.d/sipwise.list  
sed -i 's/wheezy/jessie/g' /etc/apt/sources.list.d/sipwise.list  
rm -f /etc/apt/sources.list.d/dell.list  
apt-get update  
apt-get install ngcp-upgrade-ce
```

---

**Note**

sip:provider CE can be upgraded to mr4.5.7 from mr3.8 LTS only. Run the upgrade script as *root* like this:

---

```
ngcp-upgrade
```

---

**Note**

If there is an error during upgrade, the ngcp-upgrade script will request you to solve it. Once you've fixed the problem just re-execute ngcp-upgrade again and it will continue from the previous step.

---

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

The upgrade process will take several minutes, depending on your network connection and server performance. After everything has been updated successfully, it will finally ask you to reboot your system. Confirm to let the system reboot (it will boot with an updated kernel).

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.

When all finishes successfully check that replication is running. Check `ngcp-status`. Finally, do a basic functionality test. Check web interface, register two test subscribers and perform a test call between them to ensure call routing works.

---

**Note**

You can find a backup of some important configuration files of your existing installation under `/var/backup/ngcp-mr4.5.7-*` (where `*` is a place holder for a timestamp) in case you need to roll back something at any time. A log file of the upgrade procedure is available at `/var/backup/ngcp-mr4.5.7-*/upgrade.log`.

---

## 13 Backup, Recovery and Database Maintenance

### 13.1 sip:provider CE 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 can be integrated with any Debian compatible backup software.

#### 13.1.1 What data to back up

- The database

This is the most important data in the system. All subscriber and billing information, CDRs, user preferences, etc. are stored in the MySQL server. It is strongly recommended to have up-to-date dumps of all the databases.

- 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/cdrexpert* directory contains the exported CDRs. It depends on your call data retention policy whether or not to remove these files after exporting them to an external system.

### 13.2 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'
```

### 13.3 Reset Database

**Important**

All existing data will be wiped out! Use this script only if you want to clear all previously configured services and start configuration from scratch.

To reset database to its original state you can use a script provided by CE: \* Execute *ngcp-reset-db*. It will assign new unique passwords for the NGCP services and reset all services. The script will also create dumps for all NGCP databases.

### 13.4 Accounting Data (CDR) Cleanup

Sipwise sip:provider CE offers an easy way to cleanup, backup or archive old accounting data — i.e. CDRs — that is not necessary for further processing any more, or must be deleted according to the law. There are some NGCP components designed for this purpose and they are commonly called *cleantools*. These are basically configurable scripts that interact with NGCP's *accounting* and *kamailio* databases, or remove exported CDR files in order to clean or archive the unnecessary data.

#### 13.4.1 Cleanuptools Configuration

The configuration parameters of *cleantools* are located in the main NGCP configuration file: `/etc/ngcp-config/config.yml`. Please refer to the `config.yml` file description: [Cleanuptools Configuration Data](#) Section [B.1.7](#) for configuration parameter details.

In case the system administrator needs to modify some configuration value, the new configuration must be activated in the usual way, by running the following commands:

```
> ngcpcfg apply 'Modified cleantools config'
```

As a result new configuration files will be generated for the accounting database and the exported CDR cleanup tools. Please read detailed description of those tools in subsequent sections of the handbook.

The NGCP system administrator can also select the time when cleanup scripts are run, by modifying the schedule here: `/etc/cron.d/cleanup-tools`

#### 13.4.2 Accounting Database Cleanup

The script responsible for cleaning up the database is: `/usr/sbin/acc-cleanup.pl`

The configuration file used by the script is: `/etc/ngcp-cleanup-tools/acc-cleanup.conf`

An extract from a sample configuration file is provided here:

```
#####  
  
batch = 10000
```

```
archive-target = /var/backup/cdr
compress = gzip

username = dbcleaner
password = rcKamRdHhx7saYRbkJfP
host = localhost

connect accounting
time-column = from_unixtime(start_time)
backup-months = 2
backup-retro = 2
backup cdr

connect accounting
archive-months = 2
archive cdr

connect kamailio
time-column = time
cleanup-days = 90
cleanup acc

# Clean up after mediator by deleting old leftover acc entries and deleting
# old entries out of acc_trash and acc_backup
connect kamailio
time-column = time
cleanup-days = 30
cleanup acc_trash
cleanup acc_backup
```

The configuration file itself contains a detailed description of how database cleanup script works. It consists of a series of statements, one per line, which are going to be executed in sequence. A statement can either just set a variable to some value, or perform an action.

There are 3 types of actions the database cleanup script can take:

- backup CDRs
- archive CDRs
- cleanup CDRs

These actions are discussed in following sections.

A generic action is connecting to the proper database: `connect <database name>`

### 13.4.2.1 Backup CDRs

The database cleanup tool can create *monthly backups* of CDRs in the `accounting` database and store those data records in separate tables named: `cdr_YYYYMM`. The instruction in the configuration file looks like: `backup <table name>`, by default and typically it is: `backup cdr`

Configuration values that govern the backup procedure are:

- `time-column`: Which column in `cdr` table shows the month which a CDR belongs to.
- `batch`: How many records to process within a single SQL statement. If unset, less than or equals 0, all of them are processed at once.
- `backup-months`: How many months worth of records to keep in the `cdr` table—where current CDRs are stored—and not move into the monthly backup tables.



#### Important

Months are always processed as a whole, thus the value specifies how many months to keep AT MOST. In other words, if the script is started on December 15th and this value is set to "2", then all of December and November is kept, and all of October will be backed up.

---

- `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.

### 13.4.2.2 Archive CDRs

The database cleanup tool can archive (dump) old monthly backup tables. The statement used for this purpose is: `archive <table name>`, by default and typically it is: `archive cdr`

This creates an SQL dump out of too old tables created by the `backup` statement and drop them afterwards from database. Archiving uses the following configuration values:

- `archive-months`: Uses the same logic as the `backup-months` variable above. If set to "12" and the script was started on December 15th, it will start archiving with the December table of the previous year.



#### Important

Note that the sum of `backup-retro + backup-months` values cannot be larger than `archive-months` value for the same table. Otherwise you end up creating empty monthly backup tables, only to dump and delete them right afterwards.

---

- `archive-target`: Target directory for writing the SQL dump files into. If explicitly specified as `"/dev/null"`, then no actual archiving will be performed, but instead the tables will only be dropped from database.
- `compress`: If set to "gzip", then gzip the dump files after creation. If unset, do not compress.
- `host`, `username` and `password`: As dumping is performed by an external command, those variables are reused from the `connect` statement.

### 13.4.2.3 Cleanup CDRs

The database cleanup tool may do database table cleanup without performing backup. In order to do that, the statement: `cleanup <table name>` is used. Typically this has to be done in `kamailio` database, examples:

- `cleanup acc`
- `cleanup acc_trash`
- `cleanup acc_backup`

Basically the `cleanup` statement works just like the `backup` statement, but doesn't actually backup anything, but rather just deletes old records. Configuration values used by the procedure:

- `time-column`: Gives the database column name that shows the time of CDR creation.
- `batch`: The same as with `backup` statement.
- `cleanup-days`: Any record older than this many days will be deleted.

### 13.4.3 Exported CDR Cleanup

The script responsible for cleaning up exported CDR files is: `/usr/sbin/cleanup-old-cdr-files.pl`

The configuration file used by exported CDR cleanup script is: `/etc/ngcp-cleanup-tools/cdr-files-cleanup.yml`

A sample configuration file is provided here:

```
enabled: no
max_age_days: 30
paths:
-
  path: /home/jail/home/*/20[0-9][0-9][0-9][0-9]/[0-9][0-9]
  wildcard: yes
  remove_empty_directories: yes
  max_age_days: ~
-
  path: /home/jail/home/cdrexpert/resellers/*/20[0-9][0-9][0-9][0-9]/[0-9][0-9]
  wildcard: yes
  remove_empty_directories: yes
  max_age_days: ~
-
  path: /home/jail/home/cdrexpert/system/20[0-9][0-9][0-9][0-9]/[0-9][0-9]
  wildcard: yes
  remove_empty_directories: yes
  max_age_days: ~
```



The exported CDR cleanup tool simply deletes CDR files in the directories provided in the configuration file, if those have already expired.

Configuration values that define the files to be deleted:

- `enabled`: Enable (`yes`) or disable (`no`) exported CDR cleanup.
- `max_age_days`: Gives the expiration time of the exported CDR files in days. There is a general value which may be overridden by a local value provided at a specific path. The local value is valid for the particular path only.
- `paths`: an array of path definitions
  - `path`: a path where CDR files are to be found and deleted; this may contain wildcard characters
  - `wildcard`: Enable (`yes`) or disable (`no`) using wildcards in the `path`
  - `remove_empty_directories`: Enable (`yes`) or disable (`no`) removing empty directories if those are found in the given `path`
  - `max_age_days`: the local expiration time value for files in the particular `path`

## 14 Platform Security, Performance and Troubleshooting

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.

### 14.1 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.
```

### 14.2 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 14: 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 15: 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`

### 14.3 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 handbook.

- 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 9.

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.

---

## 14.4 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→sslcrtkeyfile` .
- Apply the configuration changes with `ngcpcfg apply 'added kamailio certs'`.

## 14.5 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.

### 14.5.1 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 allows you to block such kind of attacks 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
  dos_whitelisted_ips: []
  dos_whitelisted_subnets: []
```

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 blocked for 3600 sec, and you will see in the `kamailio-lb.log` a line saying:

```
Nov 9 00:11:53 sp1 lb[41958]: WARNING: <script>: IP '1.2.3.4' is blocked and banned - R=< ↔
null> ID=304153-3624477113-19168@tedadg.testlab.local
```

The banned IP will be stored in kamailio memory, you can check the list via web interface or via the following command:

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

### Excluding SIP endpoints from banning

There may be some SIP endpoints that send a huge traffic towards NGCP from a specific IP address. A typical example is a *SIP Peering Server*.

**Caution**

sip:provider CE supports handling such situations by excluding all defined *SIP Peering Servers* from DoS protection mechanism.

The NGCP platform administrator may also add whitelisted IP addresses manually in `/etc/ngcp-config/config.yml` at `kamailio.lb.security.dos_whitelisted_ips` and `kamailio.lb.security.dos_whitelisted_subnets` parameters.

## 14.5.2 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
```

## 14.6 Topology Hiding

### 14.6.1 Introduction to Topology Hiding on NGCP

The term "topology hiding" in SIP is used to describe the measures taken by typically an SBC (Session Border Controller) to hide detailed information of the internal network at the border of which it is located. Pieces of information such as IP addresses and port numbers used by SIP endpoints and intermediaries within the network are considered sensitive, as these can give some hints to potential attackers about the topology of the network.

In a typical SIP session the mandatory headers may carry that sensitive information, for example: *Contact*, *Via*, *Record-Route*, *To*, *From*, *Call-ID*. An SBC applying topology hiding will mangle the content of those headers.

Concealment of sensitive information is achieved through encoding the original content of selected SIP headers. Then NGCP will create a new SIP URI using a preselected IP address and the encoded content as URI parameter, finally re-assembling the SIP header.

Examples for encoded SIP headers:

```
Record-Route: <sip:127.0.0.8;line=sr-NvaAlWtecghucEhu6WtAcu...>
Contact: <sip:127.0.0.8;line=sr-NvaAli-1VeL.kRxLcbN86W...>
```

The *load-balancer* element of the Sipwise NGCP has an SBC role, from the SIP peers point of view. The *LB* offers topology hiding function that can be simply activated through a configuration change. By default the function is disabled.

### 14.6.2 Configuration of Topology Hiding

Activating topology hiding function is possible through the modification of the following configuration parameters in `/etc/ngcp-config/config.yml` file (shown below with default values of parameters):

```
kamailio:
  lb:
    security:
      topoh:
        enable: no
        mask_callid: no
        mask_ip: 127.0.0.8
```

Meaning of the configuration parameters:

- `enable`: if set to `yes`, the topology hiding will be activated
- `mask_callid`: if set to `yes`, the SIP Call-ID header will also be encoded
- `mask_ip`: an IP address that will be used to create valid SIP URIs, after encoding the real/original header content.

---

#### Tip

Any valid, preferably private network address can be used. The suggestion is however to use an address that is not used by any other SIP endpoint or intermediary element in the network.

---

### 14.6.3 Considerations for Topology Hiding

Although hiding sensitive information about a SIP provider's network is desired, there are some potential side effects caused by topology hiding.

The most common example is the consequence that **SIP message size may grow** when applying topology hiding. The fact that SIP messages become larger may even prevent NGCP from communicating successfully with another SIP entity (a peer SBC, for example). This can be expected under following circumstances:

- SIP transport protocol is UDP
- SIP messages have more *Via and Record-Route* headers
- IP packets of SIP messages without the topology hiding feature already have a size close to the MTU

In such a case the IP packets carrying SIP messages with encoded headers will have a size exceeding the MTU, that will cause loss of data.

The recommended solution in such a case is to use TCP transport for SIP messages.

## 14.7 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 16: 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.



Table 16: (continued)

Option	Default value	Requirement impact
<code>kamailio→lb→tcp_children</code>	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.
<code>kamailio→lb→tls→enable</code>	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.
<code>kamailio→lb→udp_children</code>	8	See <code>kamailio→lb→tcp_children</code> explanation
<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:~#
```

**14.8 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
<i>/var/log/ngcp/api.log</i>	API logs providing type and content of API requests and responses as well as potential errors	medium
<i>/var/log/ngcp/panel.log</i> <i>/var/log/ngcp/panel-debug.log</i>	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.

### 14.8.1 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
```

## 14.8.2 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 Basic Call Flows

### A.1 General Call Setup

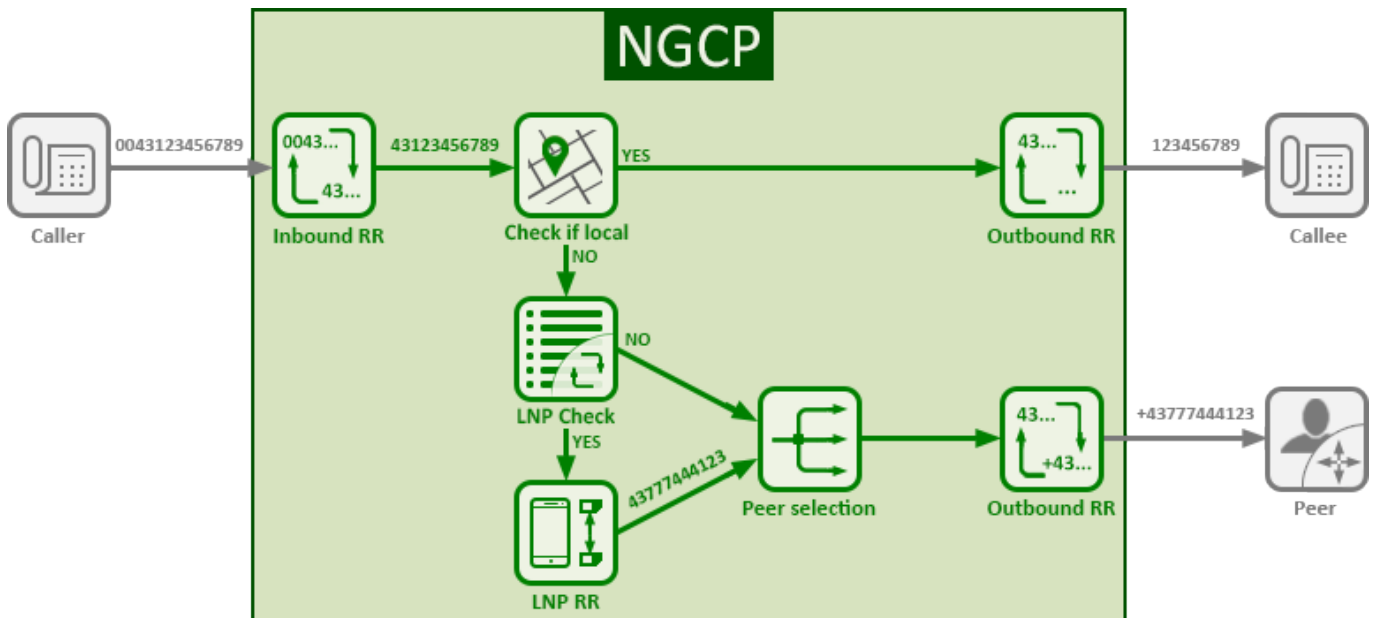


Figure 39: General Call Setup

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 refer to the [Local Number Porting](#) Section 6.4 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 Section 5.5.2.1 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.

## A.2 Endpoint Registration

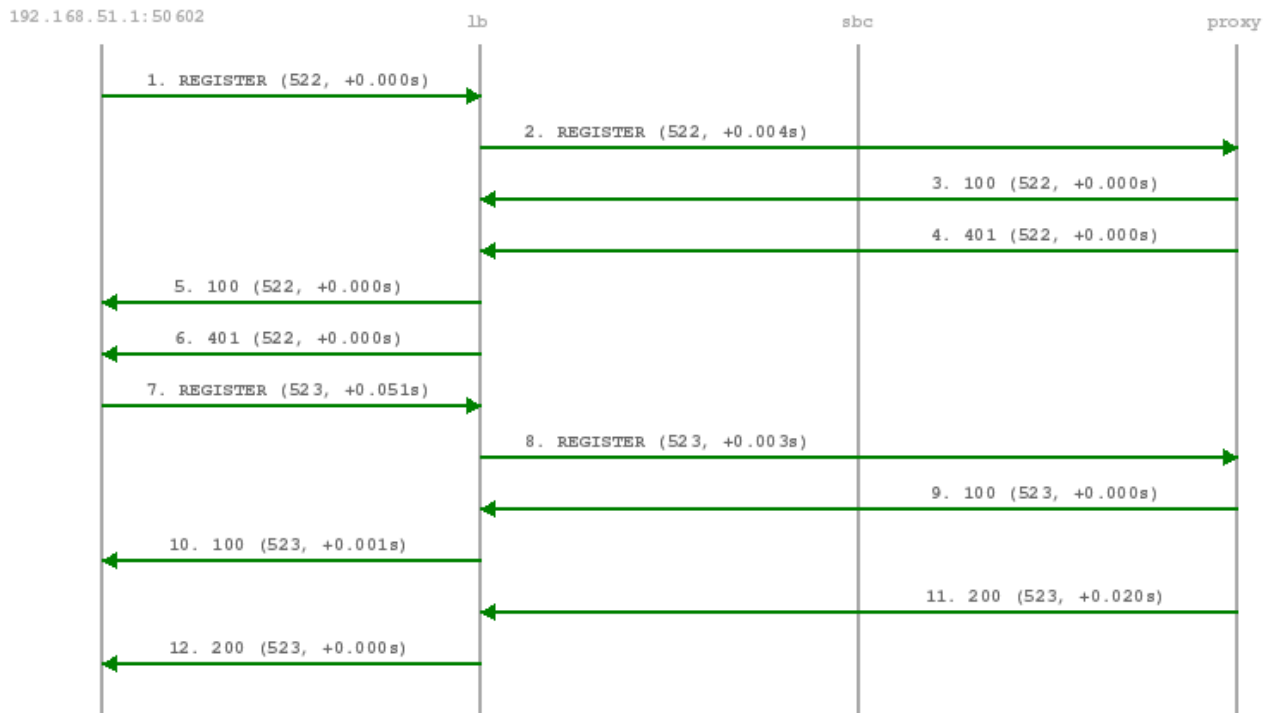


Figure 40: Registration Call-Flow

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).

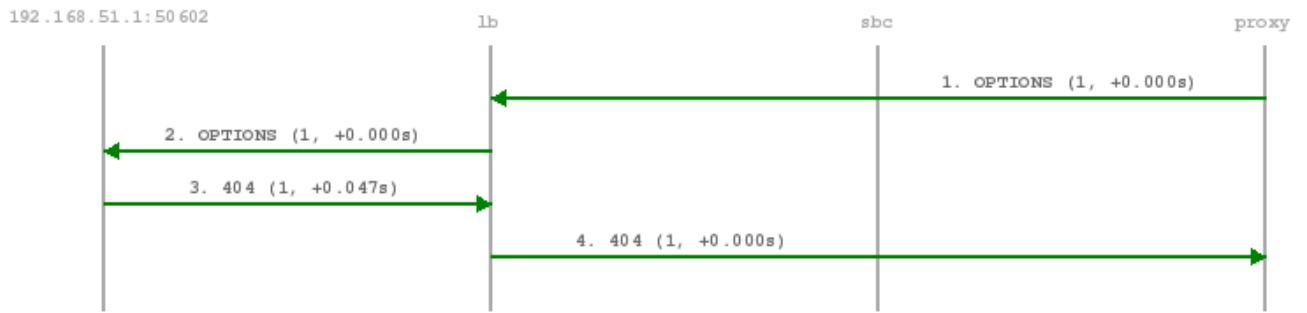


Figure 41: NAT-Ping Call-Flow

By default, a subscriber can register 5 contacts for an Address of Record (AoR, e.g. sip:someuser@example.org).





### A.3 Basic Call

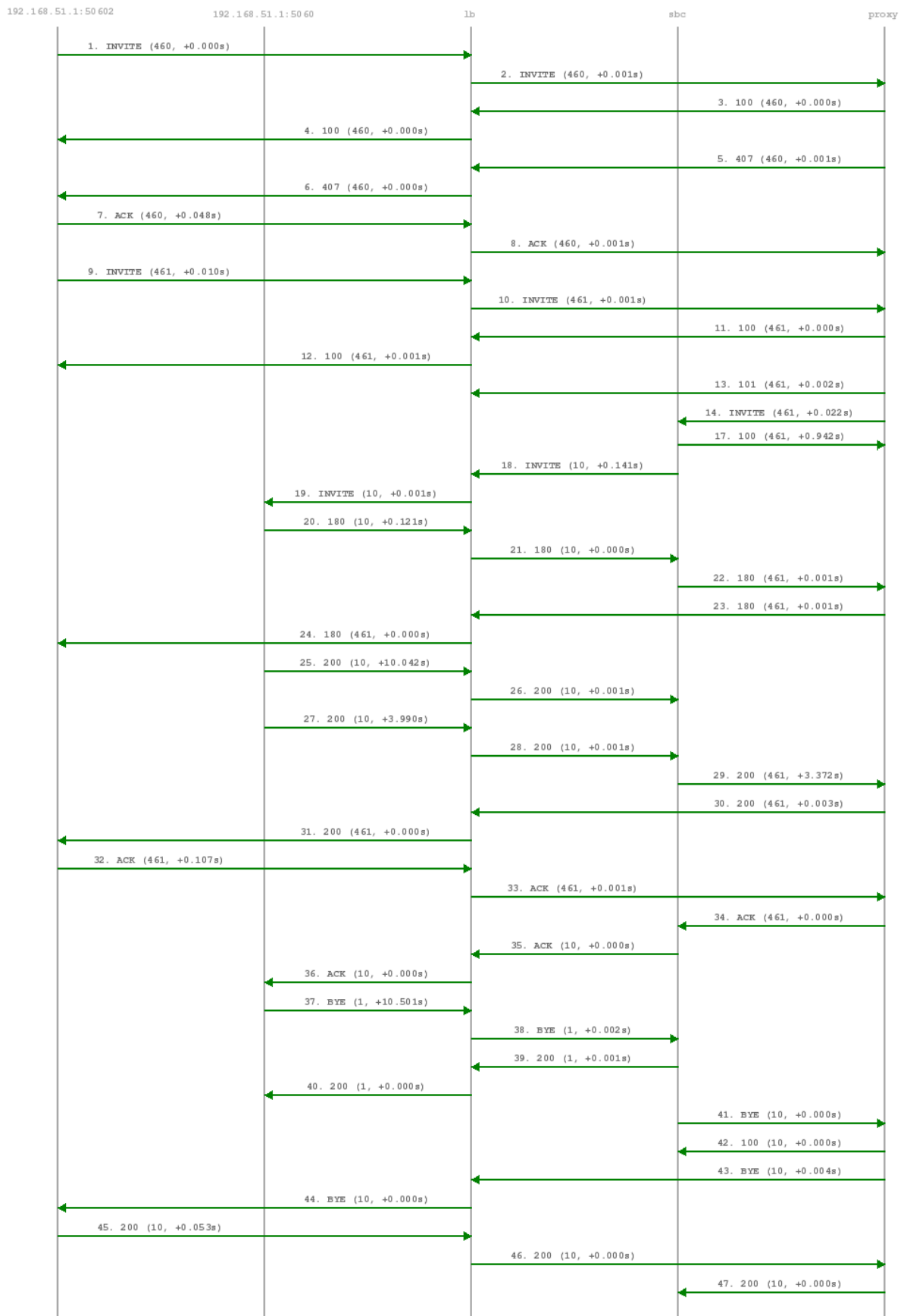


Figure 42: Basic Call Call-Flow

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.

## A.4 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.

### A.5 Voicebox Calls

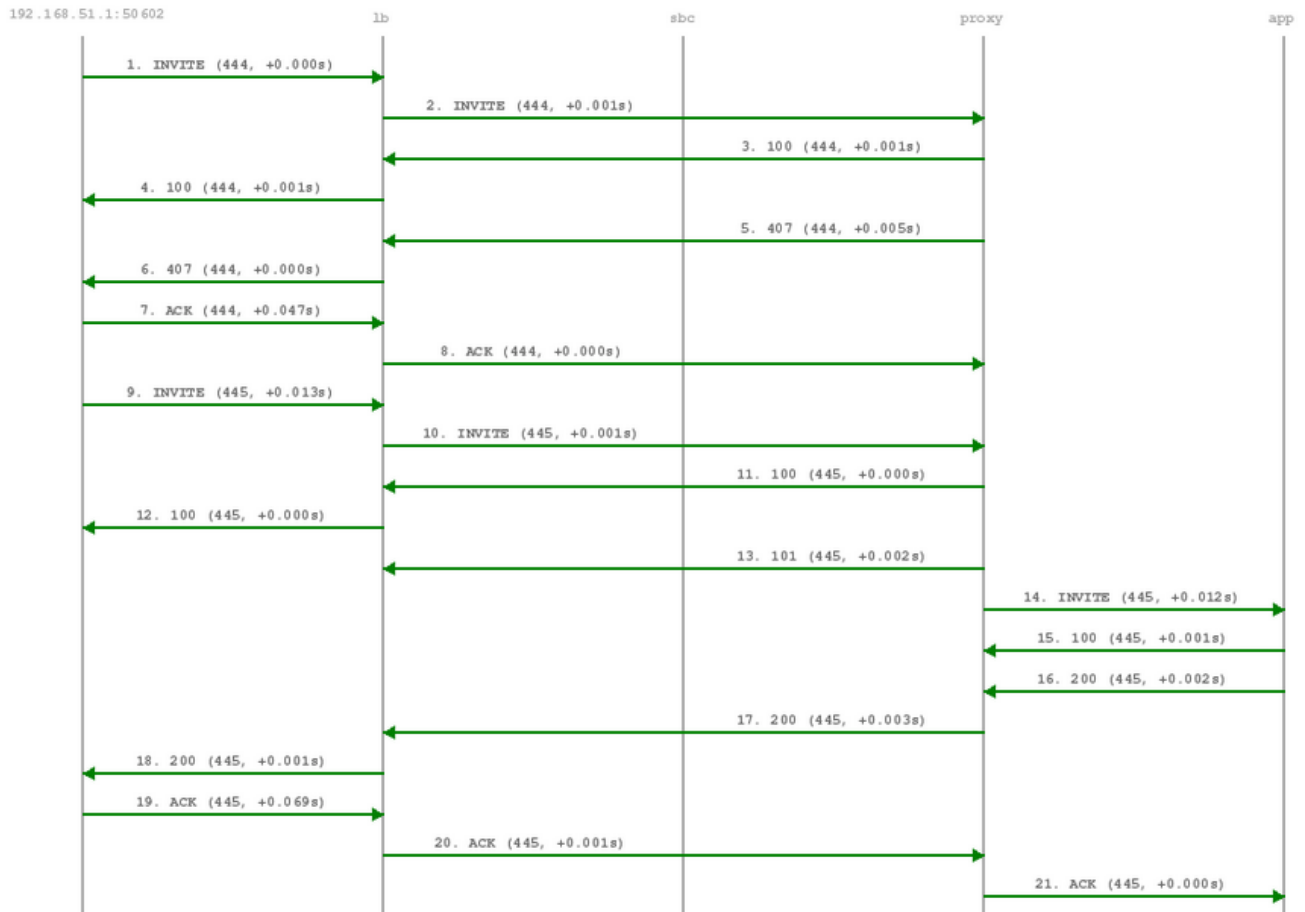


Figure 43: Voicebox Call-Flow

Calls to the Voicebox (both for callers leaving a voicemail message and for voicebox owners managing it via the IVR menu) are passed directly from the SIP proxy to the App-Server without a B2BUA. The App-Server maintains its own timers, so there is no risk of over-billing or overly long calls.

In such a case where an endpoint talks via the Media Relay to a system-internal endpoint, the Media Relay bridges the media streams between the public in the system-internal network.

In case of an endpoint leaving a new message on the voicebox, the Message-Waiting-Indication (MWI) mechanism triggers the sending of a unsolicited NOTIFY message, passing the number of new messages in the body. As soon as the voicebox owner dials into his voicebox (e.g. by calling `sip:voicebox@example.org` from his SIP account), another NOTIFY message is sent to his devices, resetting the number of new messages.

**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.

---

## B NGCP configs overview

### B.1 config.yml Overview

`/etc/ngcp-config/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.

#### B.1.1 apps

This section contains parameters for the additional applications that may be activated on sip:provider CE.

```
apps:
  malicious_call: no
```

- `malicious_call`: if set to `yes`, the Malicious Call Identification (MCID) application will be enabled

#### B.1.2 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.

### B.1.3 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.

#### B.1.4 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_subjetc`: Subject for daily backup report.
- `backuptools.mail.send_error`: Send daily backup error report.
- `backuptools.mail.send_log`: Send daily backup log report.
- `backuptools.mysql_backup.enable`: Enable daily mysql backup.
- `backuptools.mysql_backup.exclude_dbs`: exclude mysql databases from backup.



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

### B.1.5 cdrexport

The following is the cdr export section:

```
cdrexport:  
  daily_folder: 'yes'  
  export_failed: 'no'  
  export_incoming: 'no'  
  exportpath: '/home/jail/home/cdrexport'  
  full_names: 'yes'  
  monthly_folder: 'yes'
```

- `cdrexport.daily_folder`:: Set *yes* if you want to create a daily folder for CDRs under the configured path.
- `cdrexport.export_failed`: Export CDR for failed calls.
- `cdrexport.export_incoming`: Export CDR for incoming calls.
- `cdrexport.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.

### B.1.6 checktools

The following is the check tools section:

```
checktools:  
  collcheck:  
    cpuidle: 0.1  
    dfused: 0.9  
    eximaxqueue: 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
kamailio_check_concurrent_calls_enable: 0
kamailio_check_dialog_active_enable: 1
kamailio_check_dialog_early_enable: 1
kamailio_check_dialog_incoming_enable: 1
kamailio_check_dialog_local_enable: 1
kamailio_check_dialog_outgoing_enable: 1
kamailio_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).

### B.1.7 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
  delete_old_cdr_files:
    enabled: 'no'
    max_age_days: 30
    paths:
      -
        max_age_days: ~
        path: '/home/jail/home/*/20[0-9][0-9][0-9][0-9]/[0-9][0-9]'
        remove_empty_directories: 'yes'
        wildcard: 'yes'
      -
        max_age_days: ~
        path: '/home/jail/home/cdrexpport/resellers/*/20[0-9][0-9][0-9][0-9]/[0-9][0-9]'
        remove_empty_directories: 'yes'
        wildcard: 'yes'
      -
        max_age_days: ~
        path: '/home/jail/home/cdrexpport/system/20[0-9][0-9][0-9][0-9]/[0-9][0-9]'
        remove_empty_directories: 'yes'
        wildcard: 'yes'
  sql_batch: 10000
  trash_cleanup_days: 30
```

- `cleanuptools.acc_cleanup_days`: CDR records in `acc` table in `kamailio` database will be deleted after this time
- `cleanuptools.binlog_days`: Time after MySQL binlogs will be deleted.
- `cleanuptools.cdr_archive_months`: How many months worth of records to keep in monthly CDR backup tables, instead of dumping them into archive files and dropping them from database.

- `cleanuptools.cdr_backup_months`: How many months worth of records to keep in the current `cdr` table, instead of moving them into the monthly CDR backup tables.
- `cleanuptools.cdr_backup_retro`: How many months to process for backups, going backwards in time and skipping `cdr_backup_months` months first, and store them in backup tables. Any older record will be left untouched.
- `cleanuptools.delete_old_cdr_files`:
  - `enabled`: Enable (`yes`) or disable (`no`) exported CDR cleanup.
  - `max_age_days`: Gives the expiration time of the exported CDR files in days. There is a general value which may be overridden by a local value provided at a specific path. The local value is valid for the particular path only.
  - `paths`: an array of path definitions
    - \* `path`: a path where CDR files are to be found and deleted; this may contain wildcard characters
    - \* `wildcard`: Enable (`yes`) or disable (`no`) using wildcards in the `path`
    - \* `remove_empty_directories`: Enable (`yes`) or disable (`no`) removing empty directories if those are found in the given `path`
    - \* `max_age_days`: the local expiration time value for files in the particular `path`
- `cleanuptools.sql_batch`: How many records to process within a single SQL statement.
- `cleanuptools.trash_cleanup_days`: Time after CDRs from `acc_trash` and `acc_backup` tables in `kamailio` database will be deleted.

For the description of `cleanuptools` please visit [Cleanuptools Description](#) Section 13.4 section of the handbook.

### B.1.8 cluster\_sets

The following is the cluster sets section:

```
cluster_sets:
  default:
    dispatcher_id: 50
  default_set: default
  type: central
```

- `cluster_sets.<label>`: an arbitrary label of the cluster set; in the above example we have `default`
- `cluster_sets.<label>.dispatcher_id`: a unique, numeric value that identifies a particular cluster set
- `cluster_sets.default_set`: selects the default cluster set
- `cluster_sets.type`: the type of cluster set; can be `central` or `distributed`

### B.1.9 database

The following is the database section:

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

- `database.bufferpoolsize`: Innodb\_buffer\_pool\_size value in `/etc/mysql/my.cnf`

### B.1.10 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.

### B.1.11 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)

### B.1.12 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.

### B.1.13 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).

### B.1.14 kamailio

The following is the kamailio section:

```
kamailio:
  lb:
    debug: 'no'
    extra_sockets: ~
    max_forwards: 70
    nattest_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:
    - 0$avp(caller_ac)
    - $avp(caller_cc)$avp(caller_ac)
    - '\+$avp(caller_cc)$avp(caller_ac)'
    - 00$avp(caller_cc)$avp(caller_ac)
  special_numbers:
    - 112
    - 110
    - 118[0-9]{2}
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_lockup: '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 than \* `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 some prefix to emergency numbers.

- `kamailio.proxy.fritzbox.prefixes`: Fritzbox prefixes to check. Ex. `0$avp(caller_ac)`
- `kamailio.proxy.fritzbox.special_numbers`: Specifies Fritzbox special number patterns. They will be checked with the prefixes defined. Ex. `112`, so the performed check will be `sip:0$avp(caller_ac)112@` if prefix is `0$avp(caller_ac)`
- `kamailio.proxy.ignore_auth_realm`: Ignore SIP authentication realm.
- `kamailio.proxy.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.

### B.1.15 mediator

The following is the mediator section:

```
mediator:
  interval: 10
```

- mediator.interval: Running interval of mediator.

### B.1.16 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

### B.1.17 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.

### B.1.18 ossbss

The following is the ossbss section:

```
ossbss:
  apache:
    port: 2443
    proxyluport: 1080
  restapi:
    sslcertfile: '/etc/ngcp-panel/api_ssl/api_ca.crt'
    sslcertkeyfile: '/etc/ngcp-panel/api_ssl/api_ca.key'
```

```
serveradmin: support@sipwise.com
servername: "\"myserver\""
ssl_enable: 'yes'
sslcertfile: '/etc/ngcp-config/ssl/myserver.crt'
sslcertkeyfile: '/etc/ngcp-config/ssl/myserver.key'
frontend: 'no'
htpasswd:
-
  pass: '{SHA}w4zj3mxbmynIQ1jsUEjSkN2z2pk='
  user: ngcpssoap
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).

### B.1.19 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.

### B.1.20 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.

### B.1.21 pushd

The following is the pushd section:

```
pushd:
  apns:
    certificate: '/etc/ngcp-config/ssl/PushChatCert.pem'
    enable: yes
    endpoint: gateway.push.apple.com
    feedback_endpoint: feedback.push.apple.com
    feedback_interval: 3600
    key: '/etc/ngcp-config/ssl/PushChatKey.pem'
    socket_timeout: 0
  enable: yes
  gcm:
    enable: yes
    key: 'google_api_key_here'
    priority:
      call: high
      groupchat: normal
      invite: normal
      message: normal
  one_device_per_subscriber: no
  port: 45060
  processes: 4
  ssl: yes
  sslcertfile: /etc/ngcp-config/ssl/CAsigned.crt
  sslcertkeyfile: /etc/ngcp-config/ssl/CAsigned.key
  unique_device_ids: no
```

- `pushd.enable`: Enable/Disable the Push Notification feature.
- `pushd.apns.certificate`: Specify the Apple certificate for push notification https requests from the NGCP to an endpoint.
- `pushd.apns.enable`: Enable/Disable Apple push notification.
- `pushd.apns.key`: Specify the Apple key for push notification https requests from the NGCP to an endpoint.
- `pushd.gcm.enable`: Enable/Disable Google push notification.
- `pushd.gcm.key`: Specify the Google key for push notification https requests from the NGCP to an endpoint.
- `pushd.ssl`: The security protocol the NGCP uses for https requests from the app in the push notification process.
- `pushd.sslcertfile`: The trusted certificate file purchased from a CA
- `pushd.sslcertkeyfile`: The key file that purchased from a CA
- `pushd.unique_device_ids`: Allows a subscriber to register the app and have the push notification enabled on more than one mobile device.

### B.1.22 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.

### B.1.23 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.

### B.1.24 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.

### B.1.25 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.

### B.1.26 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.

### B.1.27 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).

### B.1.28 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`).

### B.1.29 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 outbound calls.
- `sems.sbc.session_timer.enable`: Enable/Disable session timers (deprecated, use the web interface configuration).
- `sems.vsc.*`: Define here the VSC codes.

### B.1.30 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.

### B.1.31 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`).

### B.1.32 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)'
```

```
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*.

### B.1.33 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.

## B.2 constants.yml Overview

`/etc/ngcp-config/constants.yml` is one of the main configuration files that contains important (static) configuration parameters, like NGCP system-user data.



### Caution

NGCP platform administrator should not change content of `constants.yml` file unless absolutely necessary. Please contact Sipwise Support before changing any of the parameters within the `constants.yml` file!

## B.3 network.yml Overview

`/etc/ngcp-config/network.yml` is one of the main configuration files that contains network-related configuration parameters, like IP addresses and roles of the node(s) in sip:provider CE system.

The next example shows a part of the `network.yml` configuration file. Explanation of all the configuration parameters is provided in [Network Configuration](#) Section 11 section of the handbook.

### Sample host configuration for sip:provider CE

```

self:
  dbnode: '1'

```

```
eth0:
  ip: 10.0.2.15
  netmask: 255.255.255.0
  type:
    - web_ext
    - web_int
    - ssh_ext
eth1:
  ip: 10.15.20.143
  netmask: 255.255.255.0
  type:
    - ssh_ext
    - web_ext
    - web_int
    - sip_ext
    - rtp_ext
    - mon_ext
interfaces:
  - lo
  - eth0
  - eth1
lo:
  cluster_sets:
    - default
  ip: 127.0.0.1
  netmask: 255.255.255.0
  shared_ip: []
  shared_v6ip: []
  type:
    - sip_int
    - ha_int
    - aux_ext
    - ssh_ext
    - api_int
  v6ip: ':::1'
role:
  - proxy
  - lb
  - mgmt
  - rtp
  - db
```

## C RTC:engine

### C.1 Overview

WebRTC is an open project providing browsers and mobile applications with Real-Time Communications (RTC) capabilities. The RTC:engine protocol is a light weight messaging and signaling protocol for WebSocket clients. Technically it is a WebSocket sub protocol. It consists of JSON messages that are used to initiate and control call dialogs, send chat messages, join and control conferences and share files. It is similar to well known signaling protocols like SIP, but much simpler. It does not care about the underlying network protocols, like SIP does.

### C.2 RTC:engine enabling

The RTC:engine is not activated by default and needs a few steps to setup.

#### C.2.1 Enabling services via CLI

First you have to enable it first on your server via CLI. Connect with SSH on your server, open `/etc/ngcp-config/config.yml` with your editor of choice and change the following properties:

```
fileshare:
  enable: yes

rtcengine:
  conference:
    relay:
      app_id: bormuth
      url: http://xms.sipwise.com:81
  call:
    relay:
      app_id: bormuth
      url: http://xms.sipwise.com:81
  enable: yes
  expose_provisioning_api: yes

www_admin:
  http_csc:
    servername: '$IP_OF_VM'
```

Save the config.yml file and run `$ ngcpcfg apply enable rtcengine`. After the script ran, check the status of all services via `$ monit summary` or `$ monit status`.



## C.2.2 Enabling via Panel for resellers and subscribers

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\_rtproxy**: Always with rtproxy 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.

## C.2.3 Create RTC:engine session

### C.2.3.1 Create sessions

#### Request:

```
curl -i -X POST --insecure --user SUBSCRIBER_ID:SUBSCRIBER_PW -H 'Content-Type: application ←
  /json' --data-binary '{} ' https://IP_OF_VM/api/rtcsessions/
```

**Response Header:**

```
Location: /api/rtcsessions/7
```

**C.2.3.2 Receive sessions****Request:**

```
curl -i -X GET --insecure --user SUBSCRIBER_ID:SUBSCRIBER_PW -H 'Content-Type: application/ ↵  
json' https://IP_OF_VM/api/rtcsessions/{ID_FROM_LAST_REQUEST_HEADER}
```

**Response Header:**

```
{  
  ...  
  "rtc_app_name" : "default_default_app",  
  "rtc_browser_token" : "22fz8e51-ad6e-481e-a389-15c58c3fe5ac",  
  "rtc_network_tag" : "",  
  "subscriber_id" : "263"  
}
```

---

**Tip**

Use `rtc_browser_token` in your `cdk.Client`.

---

**C.3 RTC:engine protocol details****C.3.1 Terminology****C.3.1.1 Connector**

There are two kinds of connectors. The front and the back connectors. The only front connector is the `BrowserConnector`. It has access to all `WebSocket` connections and is responsible for delivering RCT:engine protocol messages to the `WebSocket` clients, and for forwarding messages from the `WebSocket` clients to the router.

Currently there are four back connectors (`SipConnector`, `XmppConnector`, `WebrtcConnector`, `ConferenceConnector`). Every back connector implements a certain communication use case.

**C.3.1.2 Router**

The router is very simple stateless message broker, that is responsible for delivering the messages to the right connector. To decide where to send the message, the router takes a look at the recipient address (to) and forwards the message to the specified connector.

### C.3.1.3 User

### C.3.1.4 App

An app is a scope for a certain RTC:engine integration. Every user can have multiple apps. And an app contains sessions.

### C.3.1.5 Network

A network is a user wide configuration, that maps a custom network name (tag) to a certain back connector. Additionally it can also store network specific configurations. And any account that is related to a certain network, will merge its custom configs with the network configs, and send its messages to the specified connector.

### C.3.1.6 Session

### C.3.1.7 Account

An account represents the credentials for a specific network. Usually it consists of an identifier like a SIP uri (sip:user@domain.tld) and an access token or rather a password.

### C.3.1.8 Browser SDK

The Browser SDK is an abstraction layer on top of the RTC:engine protocol. It is served as bundled javascript library, and provides convenient components and methods for all use cases.

## C.3.2 Messages

A typical message created by the browser sdk contains the following fields:

```
{
  "method": "module.action",
  "from": "connector:id",
  "to": "connector:id",
  "session": "session",
  "body": {
    ...
  }
}
```

### C.3.2.1 Fields

### C.3.2.2 method

It is separated in two parts. The first part is the module. It is a delegation key to separate concerns in the code. The second part is the action, which represents a specific method in a module.

### C.3.2.3 from

It represents the current sender of a message. For example the user creates a new call via the browser sdk, the message would look like this:

```
{
  "method": "call.start",
  "from": "",
  "to": "webrtc:b2bua1",
  "session": "session1",
  "body": {
    ...
  }
}
```

The content of the field is completely irrelevant, because the BrowserConnector will overwrite this field. The reason is to avoid user manipulation.

```
{
  "method": "call.start",
  "from": "browser:ws1",
  "to": "webrtc:b2bua1",
  "session": "session1",
  "body": {
    ...
  }
}
```

### C.3.2.4 to

In general this field represents the recipient of a message. The recipients address consists of two parts. First part is the prefix that targets the connector. Second part is the identifier of the recipient.

### C.3.2.5 session

If you provisioned with the RTCEngine, you get a session and its token property. The browser SDK adds this token to every message.

### C.3.2.6 body

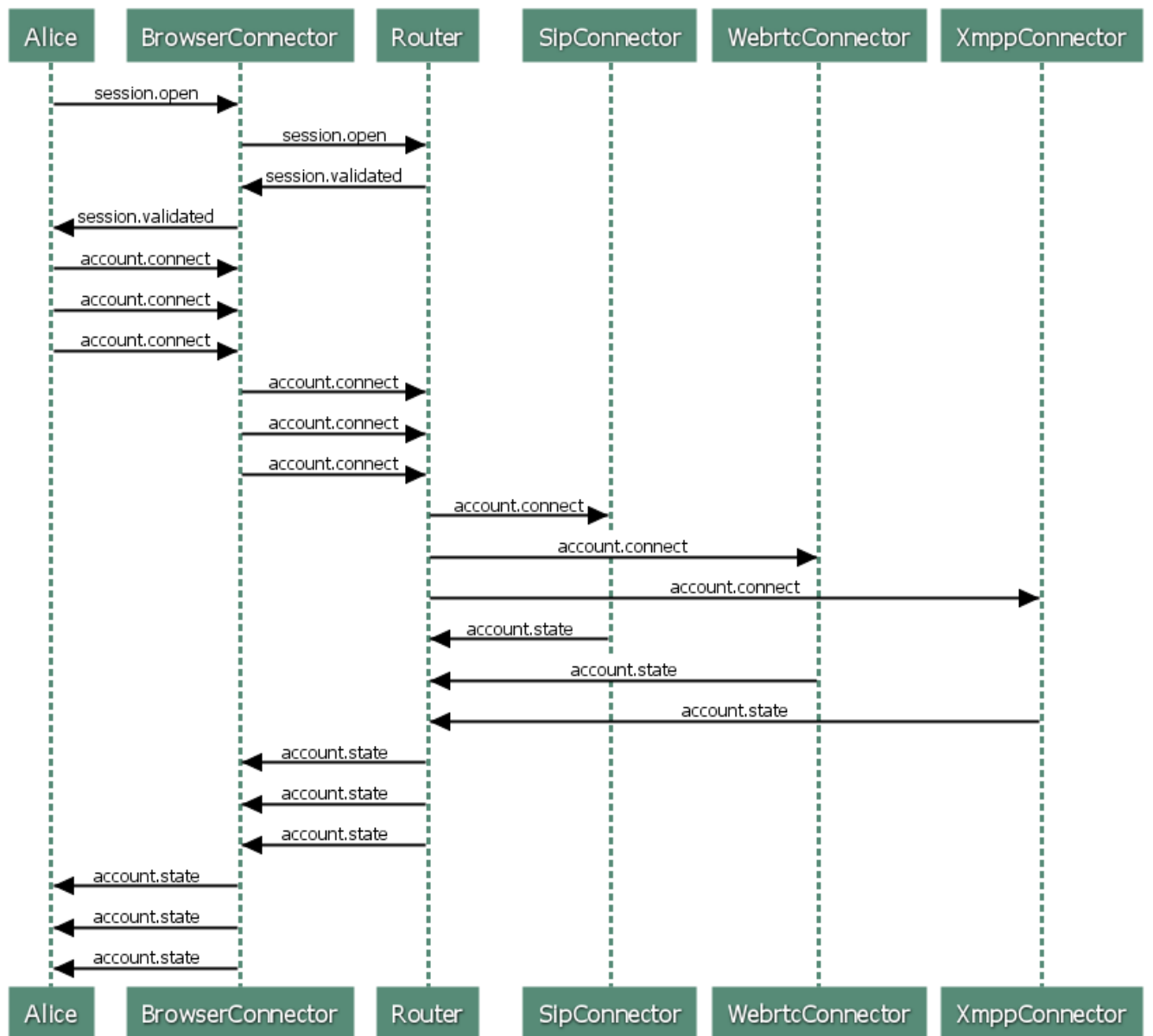
The body contains the payload of the message. Every message type has its own body schema.

### C.3.3 Account

Mainly an account consists of credentials (identifier, accessToken), that are needed to authenticate against the related network. Its lifecycle is bound to the lifecycle of the related session.

After RTC:engine received session.open, it responds a session.validated message. This message contains all provisioned accounts in its property "body.accounts".

#### C.3.3.1 Flow



www.websequencediagrams.com

### C.3.3.2 Messages

#### C.3.3.3 account.connect

RTC:engine needs one message per account. The message should contain the id of the account. The id is the object key in the accounts object from the [session.validated](../session/index.md) message.

```
{
  "from": "",
  "to": "...:...",
  "method": "account.connect",
  "session": "...",
  "body": {
    "id": "..."
  }
}
```

#### C.3.3.4 account.state

This message gives state information about the authentication and registration process of the related network and the corresponding connector. For example, if the related connector is the SipConnector, it creates a new SIP B2BUA in background, and notify the browser if any state change happens.

```
{
  "from": "...:...",
  "to": "browser:...",
  "method": "account.state",
  "session": "...",
  "body": {
    "id": "...",
    "reason": "...",
    "state": "..."
  }
}
```

#### C.3.3.5 State reasons

- OK
- CONNECTING
- DISCONNECTING
- SERVICE\_UNAVAILABLE
- SERVICE\_ERROR
- BAD\_CONFIGURATION

- WRONG\_CREDENTIALS
- CONNECTOR\_UNAVAILABLE
- CONNECTOR\_BUSY
- CONNECTOR\_ERROR
- ACCOUNT\_NOT\_FOUND

#### **C.3.3.6 States**

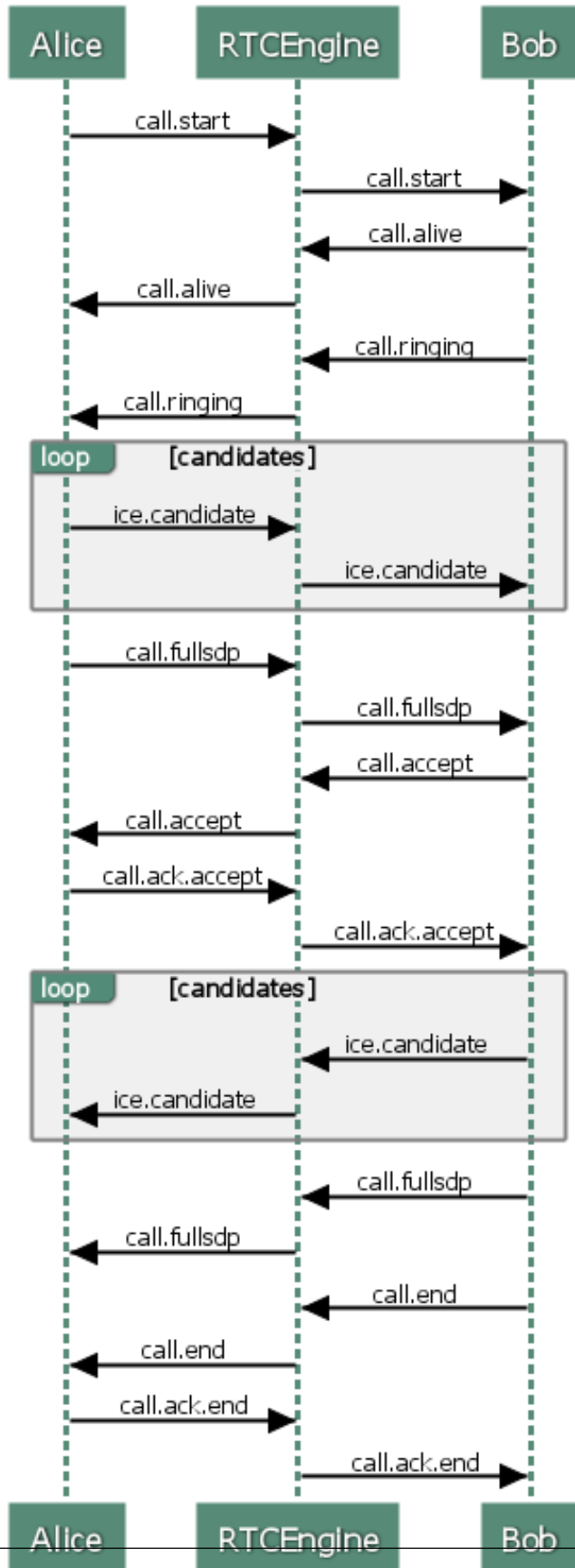
- CONNECTED
- DISCONNECTED





### C.3.4 Call

#### C.3.4.1 Flow



### C.3.4.2 call.start

The caller sends this message to the RTC:engine to initiate a new call dialog.

```
{
  "from": "local",
  "to": ["...:..."],
  "method": "call.start",
  "session": "...",
  "body": {
    "id": "...",
    "gcid": "...",
    "account": "...",
    "replace": true|false,
    "trickle": true|false,
    "target": "...",
    "sdp": "..."
  }
}
```

### C.3.4.3 Body properties

#### C.3.4.4 id

The id is a UUID version 4 that identifies the call dialog in the system. But caller and callee never have the same.

#### C.3.4.5 gcid

Whereas the gcid is a system wide and end-to-end consistent call identifier. It is necessary to track the entire call dialog.

#### C.3.4.6 account

It contains the callers account id. [(See accounts)](../account/index.md)

#### C.3.4.7 replace

This property is not used yet. It should support a call handover scenario.

#### C.3.4.8 trickle

If is set to true, the callee expects ice candidates, before the full sdp delivered by the caller, to accelerate the negotiation process.

### C.3.4.9 target

It's the URI (sip:user@domain.tld) of the callee.

### C.3.4.10 sdp

The sdp property contains a very early state of the browsers media machine. It contains no ice candidates so far.

### C.3.4.11 call.alive

After the callee received the "call.start" message, it responds with a "call.alive" to the RTC:engine, immediately.

```
{
  "from": "...",
  "to": "...",
  "method": "call.alive",
  "session": "...",
  "body": {
    "id": "...",
    "gcid": "..."
  }
}
```

### C.3.4.12 call.ringing

After the callee received the "call.start" message, it responds with a "call.ringing" to the RTC:engine, immediately.

```
{
  "from": "...",
  "to": "...",
  "method": "call.ringing",
  "session": "...",
  "body": {
    "id": "...",
    "gcid": "...",
    "account": null
  }
}
```

### C.3.4.13 call.accept

The callee sends this message after accepting the call explicitly.

```
{
  "from": "...",
  "to": "...",
```

```
"method": "call.accept",
"session": "...",
"body": {
  "id": "...",
  "gcid": "...",
  "account": null,
  "trickle": true|false,
  "sdp": "..."
}
}
```

#### C.3.4.14 call.ack.accept

Caller sends this message after it received the "call.accept" message from the callee.

```
{
  "from": "...",
  "to": "...",
  "method": "call.ack.accept",
  "session": "...",
  "body": {
    "id": "...",
    "gcid": "..."
  }
}
```

#### C.3.4.15 call.candidate

Both, caller and callee send ice candidates immediately after initiating respectively accepting the call.

```
{
  "from": "...",
  "to": "...",
  "method": "call.candidate",
  "session": "...",
  "body": {
    "id": "...",
    "gcid": "...",
    "candidate": {
      "payload": "...",
      "type": "WEBRTC_LEGACY"
    }
  }
}
```

### C.3.4.16 call.fullsdp

Both, caller and callee send this message after the ice gathering finished and all candidates are available.

```
{
  "from": "...",
  "to": "...",
  "method": "call.fullsdp",
  "session": "...",
  "body": {
    "id": "...",
    "gcid": "...",
    "sdp": "..."
  }
}
```

### C.3.4.17 call.change....

All messages, that begin with "call.change", are important for renegotiation and glare handling.

### C.3.4.18 call.change.lock.reset

### C.3.4.19 call.change.lock

### C.3.4.20 call.change.lock.ok

### C.3.4.21 call.change.offer

### C.3.4.22 call.change.answer

### C.3.4.23 call.dtmf

Only works if the connector of the related account supports DTMF messages.

```
{
  "from": "...",
  "to": "...",
  "method": "call.dtmf",
  "session": "...",
  "body": {
    "id": "...",
    "gcid": "...",
    "dtmf": "...",
    "account": null
  }
}
```

#### C.3.4.24 call.end

Both, caller and callee can send this message. It forces the counter part to end and destroy the call.

```
{
  "from": "...",
  "to": "...",
  "method": "call.end",
  "session": "...",
  "body": {
    "id": "...",
    "gcid": "...",
    "reason": "..."
  }
}
```

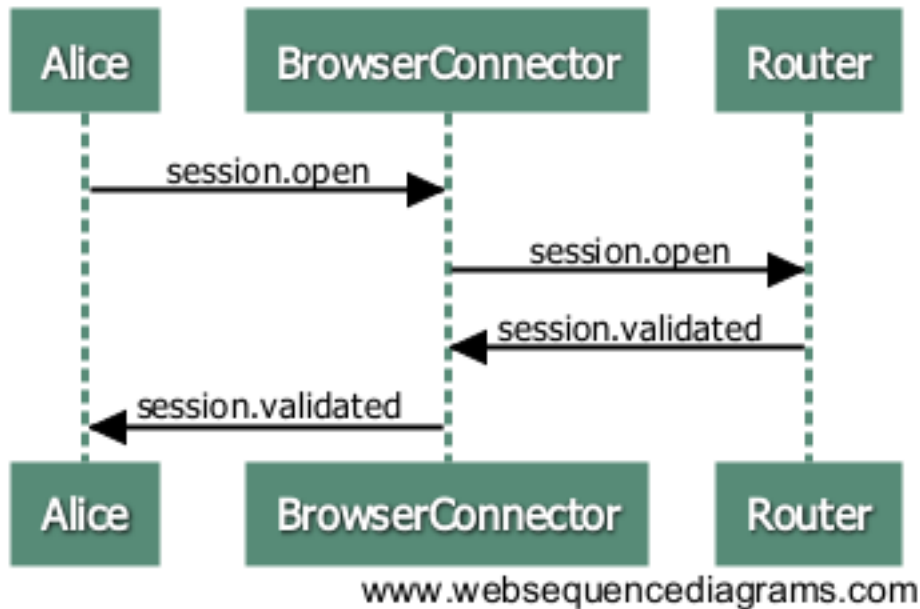
#### C.3.4.25 call.ack.end

The counter part, that receives the "call.end" message, sends the "call.ack.end" message.

```
{
  "from": "...",
  "to": "...",
  "method": "call.ack.end",
  "session": "...",
  "body": {
    "id": "...",
    "gcid": "...",
    "account": null
  }
}
```

### C.3.5 Session

#### C.3.5.1 Flow



#### C.3.5.2 Messages

##### C.3.5.3 `session.open`

```

{
  "method": "session.open",
  "from": "",
  "to": "",
  "session": "session1",
  "body": {
    "credentials": {
      "userSession": "session1"
    }
  }
}

```

##### C.3.5.4 `session.validated`

This message is the response to `session.open`. If the session property is a valid session, you get a response where the result property is true. In addition you get the account information to connect to the networks.

```

{
  "method": "session.validated",
  "from": "core",

```

```
"to": "browser:wsl",
"session": "session1"
"body": {
  "result": true,
  "accounts": {
    "account1": {
      "identifier": "sip:account1@foo.bar"
      "target": "sip-connector:b2bua-account1",
      "network": {
        "tag": "sip-network"
      }
    }
  }
},
}
```

If something went wrong, result is set to false and an error reason appears.

```
{
  "method": "session.validated",
  "from": "core",
  "to": "browser:wsl",
  "session": "session1"
  "body": {
    "result": false,
    "reason": {
      "type": "invalidToken",
      "message": "Your token is not a valid user session token!"
    }
  }
},
}
```

### C.3.5.5 Reason types

- invalidToken
- tokenExpired
- missingCredentials



## D NGCP Internals

This chapter documents internals of the sip:provider CE that should not be usually needed, but might be helpful to understand the overall system.

### D.1 Pending reboot marker

The sip:provider CE has the ability to mark a pending reboot for any server, using the file `/var/run/reboot-required`. As soon as the file exists, several components will report about a pending reboot to the end-user. The following components report about a pending reboot right now: `ngcp-status`, `ngcpcfg status`, `motd`, `ngcp-upgrade`. Also, `ngcp-upgrade` will NOT allow proceeding with an upgrade if it notices a pending reboot. It might affect `rtengine` dkms module building if there is a pending reboot requested by a newly installed kernel, etc.

### D.2 Redis id constants

The list of current sip:provider CE Redis DB IDs:

Service	Redis DB N:	central	local	Release	Ticket	Description
<b>sems</b>	redis_db:	-	0	mr3.7.1+	-	HA switchover
<b>rtengine</b>	redis_db:	-	1	mr3.7.1+	-	HA switchover
<b>proxy</b>	redis_db:	2	-	mr3.7.1+	-	Counter of hunting groups
<b>proxy</b>	redis_db:	3	-	mr3.7.1+	-	Concurrent dialog counters
<b>proxy</b>	redis_db:	-	4	mr3.7.1+	-	List of keys of the central counters
<b>prosody</b>	redis_db:	5	-	mr3.7.1+	-	XMPP cluster
<b>sems PBX</b>	redis_db:	-	6	mr3.7.1+	-	HA switchover
<b>sems</b>	redis_db:	7	-	mr4.1.1+	MT#12707	Sems malicious_call app
<b>captagent</b>	redis_db:	-	8	mr4.1.1+	MT#15427	Captagent internal data
<b>monitoring</b>	redis_db:	9	-	mr4.3+	MT#31	SNMP agent monitoring data
<b>proxy</b>	redis_db:	10	-	mr4.3+	MT#16079	SIP Loop detection

### D.2.1 Redis monitoring keys

The redis monitoring database contains a cache of several current monitoring values. These values are stored in namespaced hashes:

node:<nodename>	Cluster node information.
fsys:<nodename>:<fsysname>	Mounted filesystems information.
proc:<nodename>:<procname>	Monitored processes information.
mysql:<nodename>	MySQL database information.

To access all *fsys* and *proc* hashes there are two sets that list them:

fsys-list:<nodename>	Set of mounted filesystems.
proc-list:<nodenam>	Set of monitored processes.

The *node* hashes contain the following keys:

hb_proc_state	Cluster node heartbeat process state (boolean: stopped/running).
hb_host_state	Cluster node host state (boolean: up/down).
hb_node_state	Cluster node HA state (ngcp-check_active -p).
num_cpus	Total number of CPUs on cluster node.

The *fsys* hashes contain the following keys:

name	The mounted filesystem name (such as /).
size	The filesystem total size in bytes.
used	The filesystem used size in bytes.

The *proc* hashes contain the following keys:

name	The process name.
proc_status	The process status.
monit_status	The monit status.
pid	The process ID.
ppid	The process parent ID.
children	The number of children.
uptime	The process uptime.
cpu_percent	The CPU usage in percent for this process.
cpu_percent_total	The CPU usage in percent for the process group.
memory	The memory in bytes for this process.
memory_total	The memory in bytes for the process group.
memory_percent	The memory in percent for this process.
memory_percent_total	The memory in percent for the process group.

data_collected	The timestamp when the data was collected.
----------------	--

The *mysql* hashes contain the following keys:

last_io_errno	Last IO error number.
last_io_error	Last IO error description.
last_sql_errno	Last SQL error number.
last_sql_error	Last SQL error description.
seconds_behind_master	Delay in seconds since last db replication.
slave_io_running	Status of slave IO thread.
slave_sql_running	Status of slave SQL thread.

### D.3 Enum preferences

All tables are in database "provisioning".

So called "enum preferences" allow a fixed set of possible values, an enumeration, for preferences. Following the differences between other preferences are described.

Setting the attribute "data\_type" of table "voip\_preferences" to "enum" marks a preferences as an enum. The list of possible options is stored in table "voip\_preferences\_enum".

voip\_preferences\_enum is:

id

boring pkey

preference\_id

Reference to table voip\_preferences.

label

A label to be displayed in frontends.

value

Value that will be written to voip\_[usr|dom|peer]\_preferences.value if it is NOT NULL. Will not be written if it IS NULL. This can be used to implement a "default value" for a preference that is visible in frontends as such (will be listed first if nothing is actually selected), but will not be written to voip\_[usr|dom|peer]\_preferences.value. Usually forcing a domain or peer default. Should also be named clearly (eg. \_\_\_"use domain default"\_\_\_). (Note: Therefore will also not be written to any kamailio table.)

usr\_pref

dom\_pref

peer\_pref

Flag if this is to be used for [usr|dom|peer] preferences.

default\_val

Flag indicating if this should be used as a default value when creating new entities or introducing new enum preferences (both done via triggers). (Note: For this to work, value must also be set.)

#### Relevant triggers:

enum\_update

Propagates changes of voip\_preferences\_enum.value to voip\_[usr|dom|peer]\_preferences.value

enum\_set\_default

Will create entries for default values when adding a new enum preference. The default value is the tuple from voip\_preferences\_enum WHERE default\_val=1 AND value NOT NULL.

trigger voip\_dom\_crepl\_trig

trigger voip\_phost\_crepl\_trig

trigger voip\_sub\_crepl\_trig

These three triggers will set possible default values (same condition as for enum\_set\_default) when creating new subscribers/domains/peers.

Find a usage example in a section in *db-schema/db\_scripts/diff/9086.up*.