

Sipwise NGCP Operations Manual

CE

Sipwise GmbH <support@sipwise.com>

Table of Contents

1. Introduction . . . 1
   1.1. About This Handbook . . . 1
   1.2. What Is the Sipwise C5 CE? . . . 1
   1.3. The Advantages of the Sipwise C5 CE . . . 1
   1.4. Who Is the Sipwise C5 CE For? . . . 2
   1.5. Getting Help . . . 2
2. Architecture . . . 3
   2.1. Platforms . . . 3
   2.2. SIP Signaling and Media Relay . . . 4
   2.3. Redis Database . . . 7
   2.4. Scaling CARRIER Beyond One Hardware Chassis . . . 7
   2.5. Scaling to a Multi-Site Geo-Redundant Setup . . . 8
3. Deployment . . . 12
   3.1. Initial Installation . . . 12
   3.2. Initial System Configuration . . . 16
4. Concept . . . 20
   4.1. Contacts . . . 20
   4.2. Resellers . . . 20
   4.3. SIP Domain . . . 21
   4.4. Contracts . . . 22
   4.5. Customers . . . 22
   4.6. Subscribers . . . 25
   4.7. SIP Peerings . . . 25
5. Kick-Off . . . 27
   5.1. Login to the Administrative Web Panel . . . 27
   5.2. About the Admin Web Interface . . . 28
   5.3. Creating Resellers . . . 32
   5.4. Creating a SIP Domain . . . 33
   5.5. Creating a Customer . . . 34
   5.6. Creating a Subscriber . . . 35
   5.7. Domain Preferences . . . 37
   5.8. Subscriber Preferences . . . 37
   5.9. Creating Peerings . . . 40
   5.10. Configuring Rewrite Rule Sets . . . 49
6. Billing . . . 61
   6.1. Billing Profiles . . . 61
   6.2. Fraud Detection and Locking . . . 66
   6.3. Notes on Billing and Call Rating . . . 67
   6.4. Billing Data Export to Files . . . 68
   6.5. Billing Data Replication to Remote Databases . . . 85
7. Features . . . 93

7.1. Managing System Administrators . . . 93
7.2. Access Control for SIP Calls . . . 96
7.3. Call Forwarding and Call Hunting . . . 105
7.4. Call Forking by Q Value . . . 112
7.5. Local Number Porting . . . 113
7.6. P-Early-Media . . . 123
7.7. Emergency Mapping . . . 125
7.8. Emergency Priorization . . . 129
7.9. Speed Dials . . . 134
7.10. SIP Message Filtering . . . 135
7.11. SIP Trunking with SIPconnect . . . 139
7.12. Trusted Subscribers . . . 144
7.13. Peer Probing . . . 144
7.14. Voicemail System . . . 147
7.15. Configuring Subscriber IVR and VSC Language . . . 154
7.16. Sound Sets . . . 155
7.17. Conference System . . . 163
7.18. Malicious Call Identification (MCID) . . . 167
7.19. Subscriber Profiles . . . 168
7.20. SIP Loop Detection . . . 169
7.21. Invoices and Invoice Templates . . . 169
7.22. Email Reports and Notifications . . . 182
7.23. Calling Line Identification Restriction . . . 187
7.24. Connected Line Identification Restriction . . . 188
7.25. Do Not Disturb . . . 188
7.26. The Vertical Service Code Interface . . . 189
7.27. Call Recording . . . 192
7.28. Media Transcoding and Transrating . . . 202
7.29. Transcription Services . . . 208
7.30. Store Recent Calls and Redial . . . 211
7.31. Time Sets Management . . . 212
7.32. Announcement Before Call Setup . . . 218
7.33. Emulated Ringback Tone . . . 219
7.34. Subscriber Location Mappings . . . 220
7.35. STIR/SHAKEN . . . 224
7.36. Fileshare . . . 229
7.37. Batch Provisioning . . . 235
7.38. Call List Suppressions . . . 257
7.39. Message Body Filtering . . . 259
7.40. Generation of 181 Call Is Being Forwarded . . . 260
8. Configuration Framework . . . 262
8.1. Configuration Templates . . . 262
8.2. config.yml, constants.yml and network.yml Files . . . 267

8.3. ngcpcfg and Its Command Line Options . . . 268
9. Network Configuration . . . 271
9.1. General Structure . . . 271
9.2. Advanced Network Configuration . . . 273
10. Software Upgrade . . . 279
10.1. Release Notes . . . 279
10.2. Overview . . . 279
10.3. Pre-Upgrade Checks . . . 280
10.4. Pre-Upgrade Steps . . . 282
10.5. Upgrading Sipwise C5 CE . . . 283
10.6. Post-Upgrade Steps . . . 284
10.7. Applying the Latest Hotfixes . . . 285
11. Backup, Recovery and Maintenance . . . 287
11.1. Sipwise C5 Backup . . . 287
11.2. Recovery . . . 287
11.3. Reset Database . . . 287
11.4. Accounting Data (CDR) Cleanup . . . 288
12. Security, Performance and Troubleshooting . . . 294
12.1. Sipwise SSH Access to Sipwise C5 . . . 294
12.2. Firewalling . . . 295
12.3. Password Management . . . 303
12.4. UI/API Security . . . 304
12.5. Managing POSIX Users . . . 306
12.6. LDAP Authentication for POSIX Users . . . 306
12.7. Remote 'root' Logins via SSH . . . 307
12.8. 'sudo-io': Logging Input/Output of Commands Run Through 'sudo' . . . 307
12.9. SSL Certificates . . . 308
12.10. Securing Your Sipwise C5 Against SIP Attacks . . . 309
12.11. Topology Hiding . . . 311
12.12. System Requirements and Performance . . . 314
12.13. Troubleshooting . . . 316
12.14. Secure remote 'syslog' logging . . . 318
13. Monitoring and Alerting . . . 319
13.1. Internal Monitoring . . . 319
13.2. Statistics Dashboard . . . 320
14. Licenses . . . 321
15. Customer Self-Care . . . 322
15.1. Customer Self-Care User Interface (CSC UI) . . . 322
15.2. The Customer Self-Care Web Interface . . . 322
15.3. The Voicemail Menu . . . 324
16. REST API . . . 325
16.1. API Workflows for Customer and Subscriber Management . . . 325
16.2. API Performance Considerations . . . 330

Appendix . . . 331
Appendix A: Configuration Overview . . . 331
Appendix B: Basic Call Flows . . . 384
Appendix C: Sipwise C5 Authentication Tokens . . . 389
Appendix D: MariaDB Encryption . . . 393
Appendix E: Disk Partitioning . . . 395
Appendix F: NGCP Internals . . . 398
Appendix G: Kamailio pv_headers Module . . . 407
Appendix H: Extra Configuration Scenarios . . . 416
Appendix I: NGCP CLI Helpers and Tools . . . 418
Appendix J: Handling WebRTC Clients . . . 424
Appendix K: Batch Provisioning Extras . . . 425
Appendix L: Fritzbox Prefix Removal on Sipwise C5 . . . 433
Appendix M: Migration from KeyDB to Valkey . . . 435

Chapter 1. Introduction

1.1. About This Handbook

This handbook describes the architecture and the operational steps to install, operate and modify the Sipwise C5 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 blocking, call forwarding, etc.

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

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

1.2. What Is the Sipwise C5 CE?

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

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

The Sipwise C5 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 Sipwise C5 CE

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

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

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 Sipwise C5 Configuration Framework. This configuration framework makes it possible to change low-level system parameters in a single place, so Sipwise C5 administrators do not need to have any knowledge of dozens of different configuration files from different packages. This provides a bullet-proof way of operating, changing and tweaking an otherwise quite complex system.

Copyright © Sipwise GmbH - All rights reserved

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

1.4. Who Is the Sipwise C5 CE For?

The Sipwise C5 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. It integrates the different building blocks to make them work together in a reasonable way and implements 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 Sipwise C5 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 Sipwise C5 CE, send an email to sales@sipwise.com.

Sipwise also provides training and commercial support for the platform. Additionally, we offer a migration path to the Sipwise C5 PRO or CARRIER appliance, which is the commercial, carrier-grade version of the Sipwise C5 CE. If the user base grows on the Sipwise C5 CE, this will allow operators to migrate seamlessly to a highly available and scalable platform with defined service level agreements, phone support and on-call duty. Please visit www.sipwise.com for more information on commercial offerings.

Copyright © Sipwise GmbH - All rights reserved

Chapter 2. Architecture

2.1. Platforms

2.1.1. CE Platform

The Sipwise C5 CE platform is one single node running all necessary components of the system. The components are outlined in the following figure:

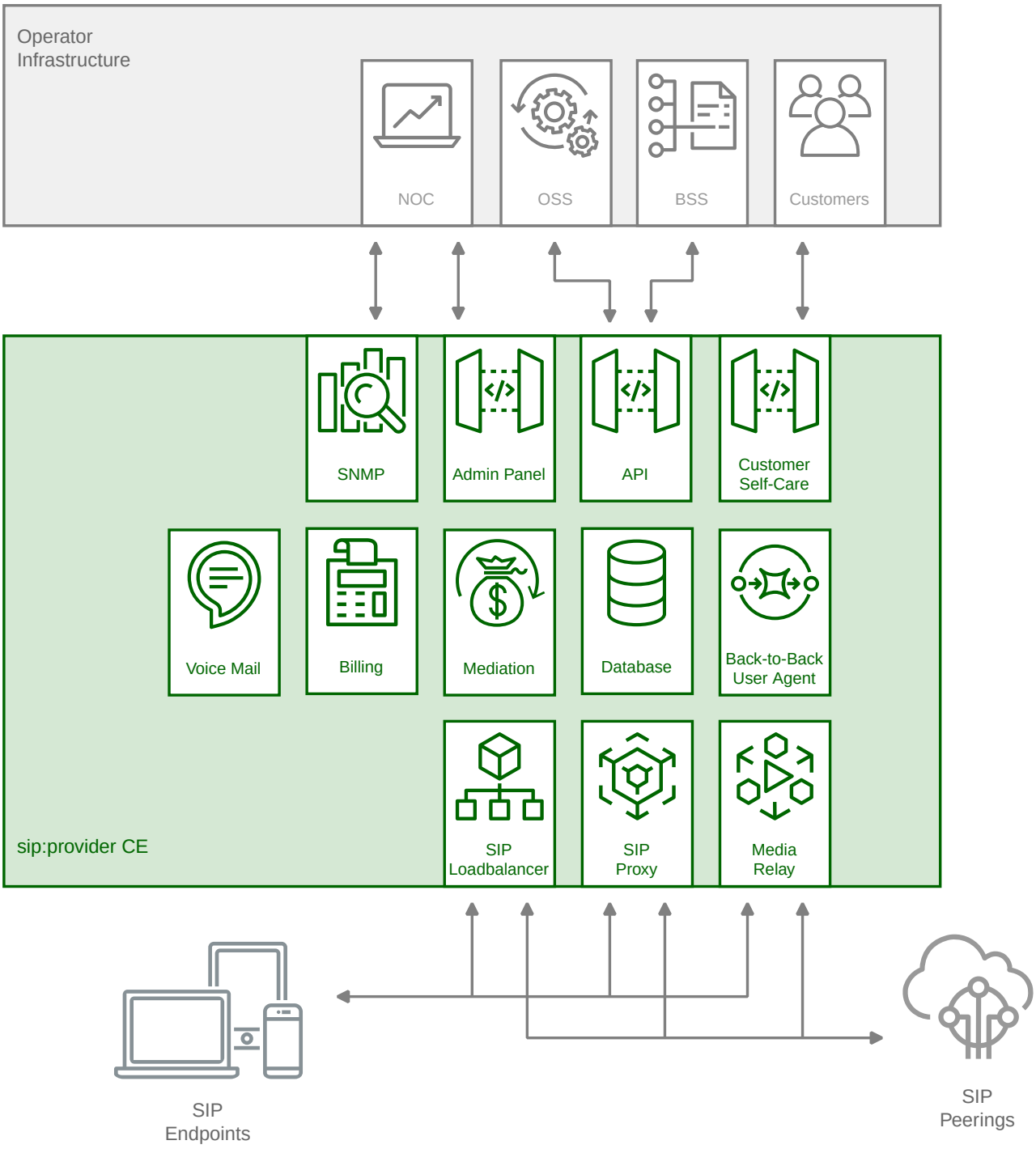

Figure 1. CE Architecture Overview

Copyright © Sipwise GmbH - All rights reserved

The main building blocks of Sipwise C5 are:

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

2.2. 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 Sipwise C5 CE are shown in the following figure:

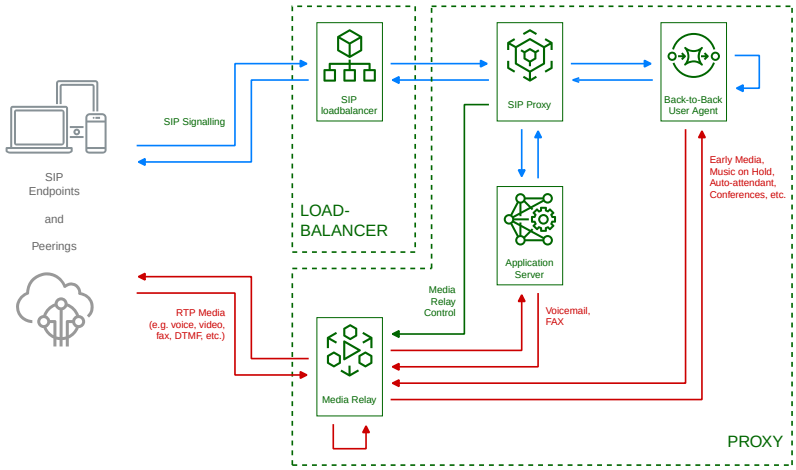

Figure 2. SIP and Media Relay Components

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

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

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

 Copyright © Sipwise GmbH - All rights reserved

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

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

2.2.2. SIP Proxy/Registrar

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

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

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

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

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

TIP

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

Copyright © Sipwise GmbH - All rights reserved

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

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

The software part used for this element is SEMS.

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

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

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

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

Its configuration files reside in `/etc/ngcp-config/templates/etc/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 `ngcp-service start sems`, `ngcp-service stop sems` and `ngcp-service restart sems`. Its status can be queried by executing `ngcp-service status sems` or `ngcp-service summary | grep "sems"`.

2.2.4. 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 does not directly access the file system for user data. The voicemail plugin is a slightly patched version based on Asterisk 16.2.1 to make Asterisk aware of Sipwise C5 internal UUIDs for each subscriber. That way a SIP subscriber can have multiple E164 phone numbers, but all of them terminate in the same voicebox.

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

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

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

 Copyright © Sipwise GmbH - All rights reserved

`service summary | grep "asterisk"`.

2.2.5. Message Routing and Media Relay

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

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

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

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

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

2.3. 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.

2.4. Scaling CARRIER Beyond One Hardware Chassis

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

Copyright © Sipwise GmbH - All rights reserved

2.4.1. Scaling the DB Cluster

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

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

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

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

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

With such setup the central db01 pair can handle all the planned and unexpected DB load without the significant hardware resource usage. DB01a and DB01b can be located in different Geo-locations for High-Availability (low latency link is required for replications).

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

2.4.2. Scaling the Proxy Cluster

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

2.4.3. Scaling the Load Balancers

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

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

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

2.4.4. Scaling the Web Servers

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

2.5. Scaling to a Multi-Site Geo-Redundant Setup

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

 Copyright © Sipwise GmbH - All rights reserved

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

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

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

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

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

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

The final setup looks like:

Copyright © Sipwise GmbH - All rights reserved

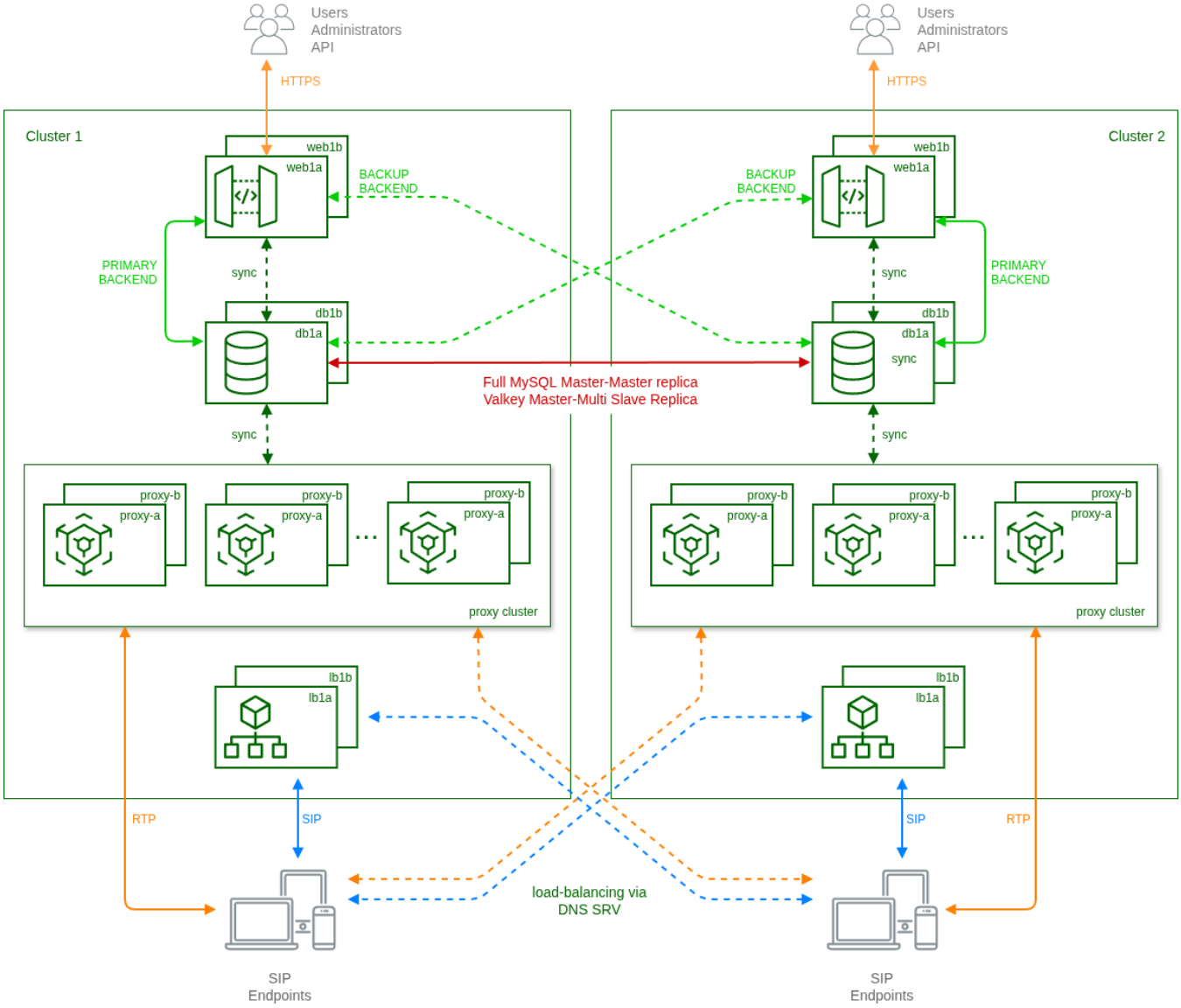

Figure 3. Geo-Redundant Carrier Overview

In particular:

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

 Copyright © Sipwise GmbH - All rights reserved

- ngcpcfg framework enables the operator to manage the whole system from one cluster node.

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

Copyright © Sipwise GmbH - All rights reserved

Chapter 3. Deployment

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

3.1. Initial Installation

3.1.1. Prerequisites

For an initial installation of Sipwise C5 , 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, 24GB HDD

Supported Operating Systems

- Debian 13 (trixie) 64-bit

Internet Connection

- Hardware needs connection to the Internet

IMPORTANT Only **Debian 13 (trixie) 64-bit** is currently supported as a host system for Sipwise C5 .

IMPORTANT It is **HIGHLY** recommended that you use a **dedicated server** (either a physical or a virtual one) for Sipwise C5, because the installation process will wipe out existing MySQL databases and modify several system configurations.

3.1.2. Using Sipwise C5 Install CD (Recommended)

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

IMPORTANT PRO/Carrier can be installed only with a commercial license. Otherwise a warning about lack of access to Debian repository will be displayed.

You can install the current Sipwise C5 CE version mr26.3.1 using install CD image (checksums: sha1, md5).

IMPORTANT The Sipwise C5 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 is strongly recommended to boot the ISO in an environment with empty disks or disks that you do not plan to use for anything else than the newly installed Sipwise C5 system.

TIP When DHCP is available in your infrastructure then you should not have to configure anything, instead choose `DHCP` and press enter. If network configuration still does not work

Copyright © Sipwise GmbH - All rights reserved

> as needed a console based network configuration system will assist you in setting up your network configuration. VLANs are also supported at this stage.

Also, you can use Sipwise C5 install CD to boot the Grml (Debian based live system) rescue system, check RAM using a memory testing tool or install plain Debian system for manual installation using Sipwise C5 installer.

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

Then you need to run the initial configuration:

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

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

3.1.3. Using a Pre-Installed Virtual Machine

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

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 Sipwise C5 Virtual Machine without any hassles.

NOTE The following software must be installed to use Vagrant boxes: VirtualBox v.5.2.26+ and Vagrant v.2.2.3+.

Get your copy of Sipwise C5 by running:

```
vagrant init spce-mr26.3.1
https://deb.sipwise.com/spce/images/mr26.3.1/sip_provider_CE_mr26.3.1_va
grant.box
vagrant up
```

As soon as the machine is up and ready you should have your local copy of Sipwise C5 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

Copyright © Sipwise GmbH - All rights reserved

Use the following command to access the terminal:

```
vagrant ssh
```

or log in to Administrator web-interface at https://127.0.0.1:1443/ (with default user *administrator* and password *administrator*).

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

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

Table 1. Vagrant based VirtualBox VM interfaces:

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

NOTE VM ports smaller than 1024 are mapped to ports 22<vm_port> through NAT, otherwise root on host machine requires to map them. So for example SSH port 22 gets mapped to port 2222, WEB port 443 to 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)
```

 Copyright © Sipwise GmbH - All rights reserved

```
[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/
```

To turn off your Sipwise C5 virtual machine, type:

```
vagrant halt
```

To completely remove Sipwise C5 virtual machine, use:

```
vagrant destroy
vagrant box remove spce-mr26.3.1
```

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
```

- VirtualBox Guest Additions is installed by default but disabled. Enable it to use Vagrant Synced Folders feature. Execute the following commands inside VM:

```
  systemctl start vboxadd-service.service vboxadd.service
  ngcpcfg set /etc/ngcp-config/config.yml
"systemd.custom_preset=['vboxadd-service.service', 'vboxadd.service']"
  ngcpcfg apply "Start VirtualBox Guest Additions on boot"
```

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

Amazon EC2 Image (Deprecated)

Sipwise used to provide AMI (Amazon Machine Images) images in several Amazon EC2 regions for supported Sipwise C5 releases.

NOTE Starting with mr26, AMIs are no longer provided, also AMIs of older releases will disappear.

Copyright © Sipwise GmbH - All rights reserved

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_mr26.3.1_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 Sipwise C5 behind NAT.
- You will need to adjust your timezone and keyboard layout.
- The network configuration is set to DHCP. You will need to change it to the appropriate static configuration.
- As the virtual image is a static file, it will not contain the most updated versions of our software. Please upgrade the system via apt as soon as you boot it for the first time.

3.2. Initial System Configuration

After the configuration you are ready to adjust the system parameters to your needs to make the system work properly.

3.2.1. Network Configuration

If you have only one network card inside your system, its device name is *eth0*, it is 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 --hwaddr=auto
ngcp-network --move-from=lo --move-to=eth0 --type=web_ext --type=sip_ext
--type=rtp_ext --type=ssh_ext --type=web_int
```

 Copyright © Sipwise GmbH - All rights reserved

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 *fdda:5cc1:23:4:0:0:0:1f* on interface *eth0*):

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

TIP Always use a full IPv6 address with 8 octets. Leaving out zero octets (e.g. `FDDA: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.

Check or adjust the network configuration in the */etc/ngcp-config/network.yml* file.

```
editor /etc/ngcp-config/network.yml
```

The following configuration shows NGCP running in the internal 192.168.0.0/24 network behind the NAT:

```
...
  self:
    eth0:
      dns_nameservers:
      - 192.168.0.1
      hwaddr: 11:22:33:44:55:66
      ip: 192.168.0.10
      gateway: 192.168.0.1
      netmask: 255.255.255.0
      type:
      - ssh_ext
      - web_ext
      - web_int
      - sip_ext
      - rtp_ext
    interfaces:
    - lo
    - eth0
    lo:
...
```

Apply the adjusted network configuration, and */etc/network/interfaces* will be regenerated from the new configuration.

```
ngcpcfg apply 'change network configuration'
```

The resulting */etc/network/interfaces* file will look like this:

Copyright © Sipwise GmbH - All rights reserved

```
# File autogenerated by ngcpcfg

# lo ----------------------
auto lo
iface lo inet loopback
# ------------------------------------------

# eth0 ----------------------
auto eth0
iface eth0 inet static
  address 192.168.0.10
  netmask 255.255.255.0
  gateway 192.168.0.1
  dns-nameservers 192.168.0.1
# ------------------------------------------
```

Reboot the server to apply the new network configuration:

```
reboot
```

3.2.2. 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.

3.2.3. Start Securing Your Server

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

```
passwd cdrexport
```

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

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

For the *Administrative Web Panel* located at *https://<your-server-ip>:1443/*, 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 *SettingsAdministrators*

 Copyright © Sipwise GmbH - All rights reserved

and click the *Edit* when hovering over the row.

3.2.4. Configuring the System-Wide Editor

The default editor is set to *nano* on the system. If you prefer a different editor, make sure it is installed and set the default editor via:

```
sudo update-alternatives --config editor
```

TIP if you want to use a specific editor only temporarily, set the *EDITOR* environment variable instead. For example to run *command* with the editor set to *vim*, invoke "*EDITOR=vim command*".

3.2.5. Configuring the Email Server

The Sipwise C5 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:

```
sudoedit /etc/ngcp-config/config.yml  # edit section 'email:' according
to your needs
sudo ngcpcfg apply 'adjust exim4 / MTA configuration'
```

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

3.2.6. 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.

3.2.7. 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.

Copyright © Sipwise GmbH - All rights reserved

Chapter 4. Concept

4.1. Contacts

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

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

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

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

4.2. Resellers

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

Let us consider an example:

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

Copyright © Sipwise GmbH - All rights reserved

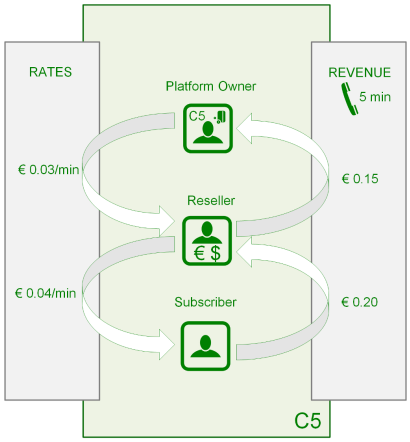

Figure 4. Reseller

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

4.3. SIP Domain

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

4.3.1. Additional SIP Domains

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

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

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

Figure 5. SIP Domain

Copyright © Sipwise GmbH - All rights reserved

4.4. Contracts

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

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

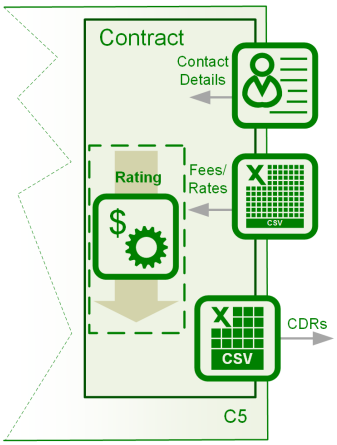

Figure 6. Contract

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

4.5. Customers

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

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

Here are two common examples of the customer model:

4.5.1. Residential and SOHO Customers

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

For a residential customer you usually create one *customer* entity with one *subscriber* under it. A

Copyright © Sipwise GmbH - All rights reserved

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

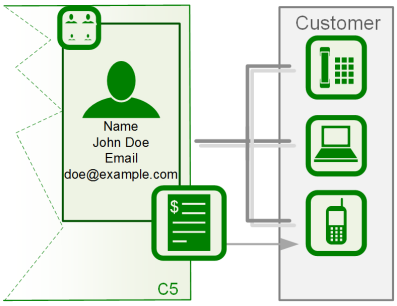

Figure 7. Residential Service Scenario

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

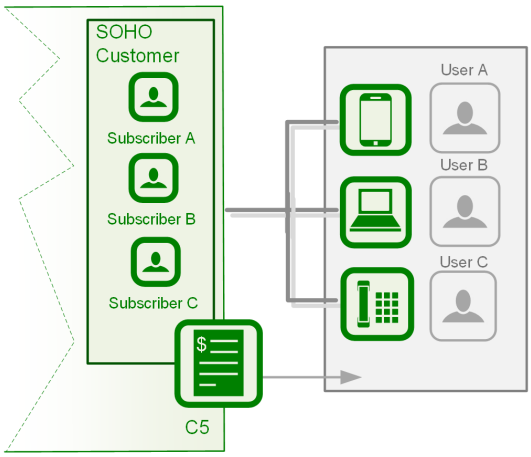

Figure 8. SOHO Scenario

4.5.2. Business Customers with the Cloud PBX Service

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

Copyright © Sipwise GmbH - All rights reserved

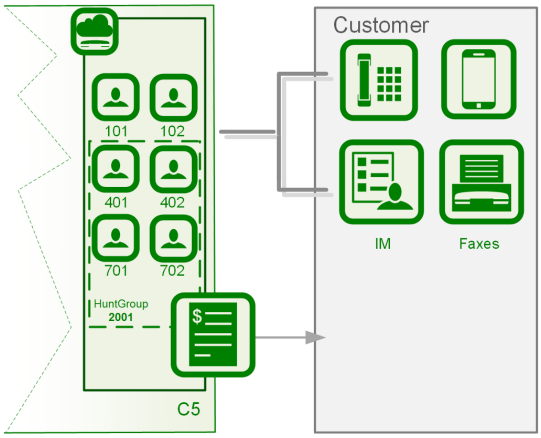

Figure 9. CPBX Service Scenario

4.5.3. SIP Trunking

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

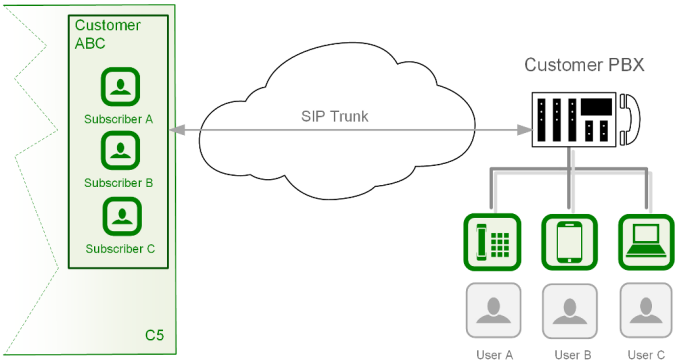

Figure 10. SIP Trunking Service Scenario

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

4.5.4. Mobile Subscribers

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

4.5.5. Prepaid Subscribers Who Use Your Calling Cards

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

Copyright © Sipwise GmbH - All rights reserved

4.6. Subscribers

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

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

Service	Subscriber Type	Purpose	Features
Residential	Regular subscriber	A regular VoIP service	Requires a DID number to receive calls from outside of your network
Enterprise (CloudPBX)	Pilot subscriber	A base number for the enterprise customer; Lists all extra numbers (aliases)	Configures the rest of customer subscribers in its self-care web interface
	Extension	Extra numbers (DIDs, “implicit” extensions) for the enterprise customer	Can be dialed in different ways; The number configuration builds on top of the Pilot subscriber
	PBX Group	Forwards incoming calls to multiple extensions	Ringing policy defines in which order the extensions will ring
SIP Trunk	Digest authentication	Dynamically registers a remote IP PBX device	Handles multiple users behind the IP PBX device
	IP authentication	IP authentication of legacy IP PBX devices incapable of registering with the platform	Might require Trusted Subscriber and Trusted Source configuration

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

4.7. SIP Peerings

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

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

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

Any SIP peering group includes two main technical configurations:

- Peering Servers Represent connections to/from your SIP peering's network. The parameters include an IP address and/or a hostname of the remote part. For outbound calls, this is the destination address where to send calls to and for inbound calls it is an IP authorization of the remote server.

Copyright © Sipwise GmbH - All rights reserved

- Outbound/Inbound Peering Rules Outbound rules define through which SIP peering group a call from a specific subscriber will be sent for termination to a specific destination.

The peering group is associated with a *contract* for reconciliation and billing. Peer invoices can be generated to audit the costs that peering partners are charging the platform operator.

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

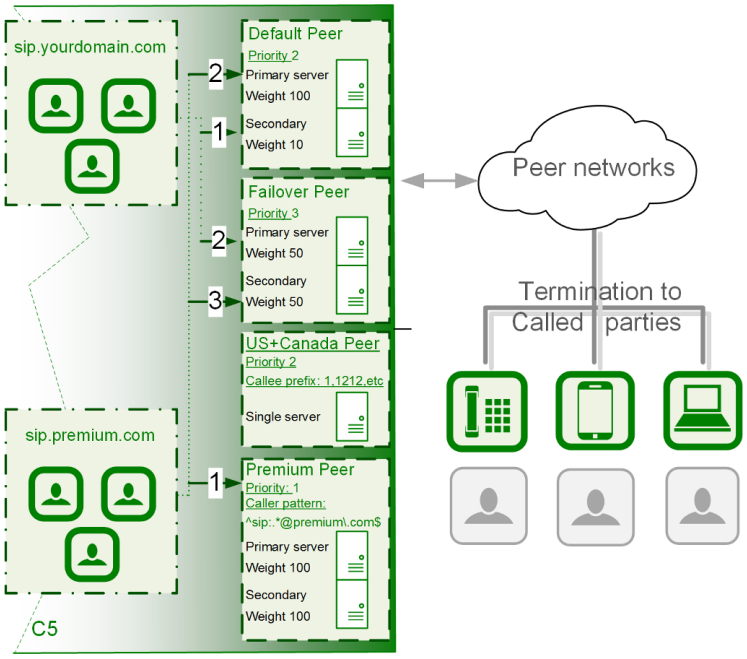

Figure 11. Routing through peering

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

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

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

 Copyright © Sipwise GmbH - All rights reserved

Chapter 5. Kick-Off

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

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

Let us assume you are using the *203.0.113.42* IP address with an associated *sip.yourdomain.com* domain to provision VoIP services. This allows you to provide an easy-to-remember domain name instead of the IP address as the proxy server. Also, your subscribers' URIs will look like *1234567@sip.yourdomain.com*.

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

5.1. Login to the Administrative Web Panel

To reach the *Administrative Web Panel* (also called as *Admin Panel* or simple *AUI*) connect to *https://<ip>:1443/* using a web browser (officially supported browsers are Chrome and Firefox).

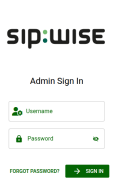

Figure 12. Admin Panel Login Page

Click on the world button at the top right of the screen to change the interface language.

To log in fill in the *Username* and *Password* fields and click on the *Sign In* button. The default web panel user and password are **administrator**, if you have not already changed it in Changing Administrator Password.

In case you have forgotten the password, click on the *Forgot Password?* button. You will be prompted to provide your *Username* and then you will receive an email with the steps to follow to reset the password.

Copyright © Sipwise GmbH - All rights reserved

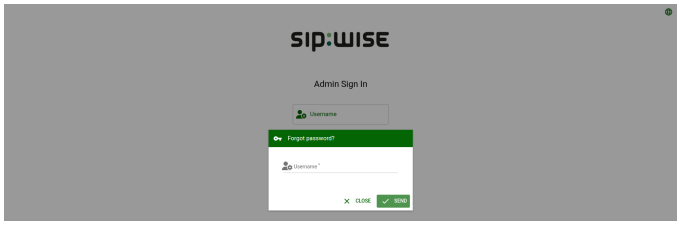

Figure 13. Admin Panel Forgot Password

5.1.1. Login with 2FA

In case your administrator account has the 2FA feature activated, then the login procedure requires an additional step.

The first time a user with 2FA logs in to the admin panel will be asked to register its Sipwise C5 account on an *Authenticator* app (Google Authenticator or similar OTP app). To do that scan the QR-Code on the app, enter the OTP code in the *OTP Code* field and click on *Sign in*.

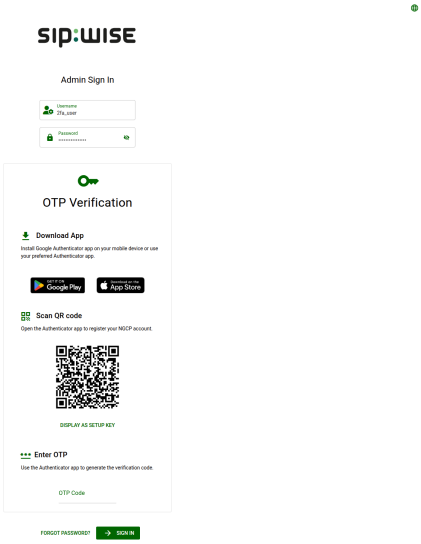

Figure 14. Admin Panel OTP Login Configuration

The next times the user will log in on the *Admin Panel*, he should enter the OTP code in addition to the standard *Username* and *Password*.

In case a user has lost its OTP access, it can make a request to the platform's system administrator to reset its OTP. To reset the OTP access for another user, the system administrator simply has to go to *SettingsAdministrators* page, click the three dots beside the user to administrate and select *Reset OTP*. Finally click on the *Reset* button on the pop-up to confirm the operation.

5.2. About the Admin Web Interface

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

The page is divided in 3 main sections: left sidebar, top bar and central panel.

 Copyright © Sipwise GmbH - All rights reserved

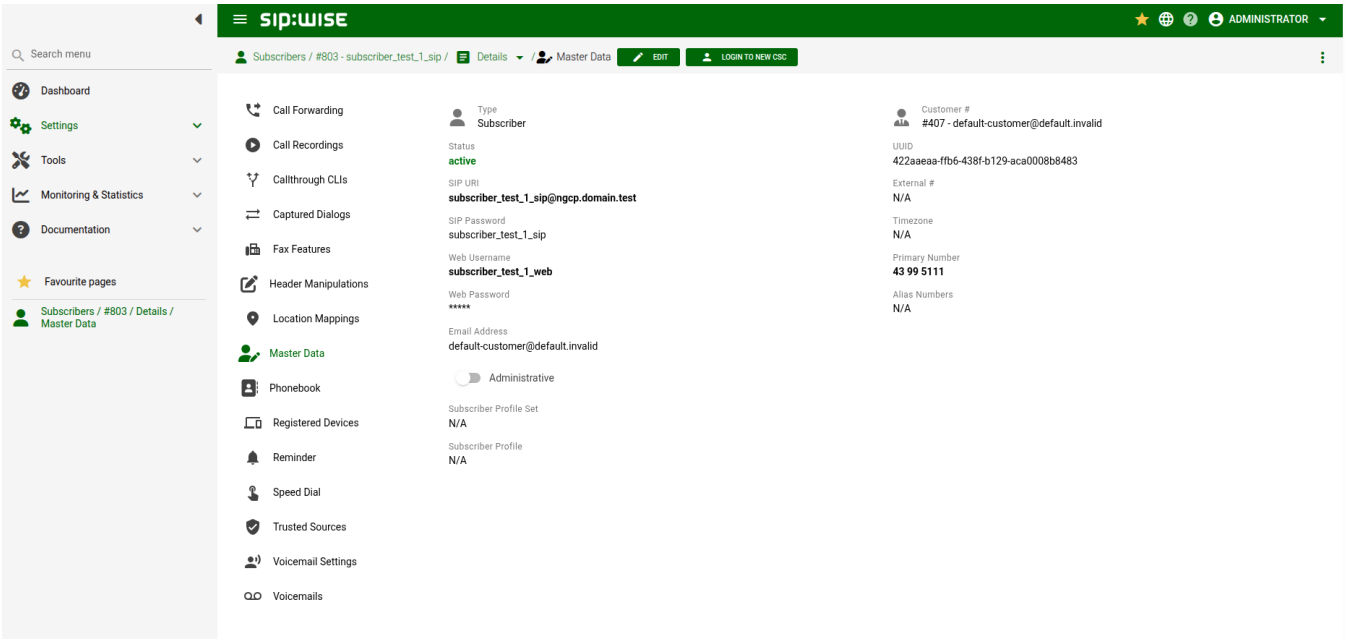

Figure 15. Administrator interface

5.2.1. Top Header Bar

The green bar at the top of the page provides some global utilities like:

- **Star**: add/remove the current page from the favourites list
- **World**: language selection
- **Question mark**: page help (currently supported only for some pages)
- **User icon**: profile management for the logged-in user

5.2.2. Left Sidebar - Main Menu

The primary navigation is anchored by a global sidebar on the left of the screen, which serves as main menu. The bar contains 5 collapsible trees:

- **Dashboard**: provides an overview on the system and its status
- **Settings**: gives access to all the main elements and features
- **Tools**: give access to some additional tools
- **Monitoring & Statistics**: provides a list of monitoring Grafana graphs
- **Documentation**: gives access to the API documentation, the handbook and additional useful links

The search field at the top allows to search for any menu included in the sidebar, while the *Favourite pages* lists all the pages/menu/elements that have been marked as favorite.

The sidebar can be pinned or automatically collapsed pressing the black arrow or the pin icons at the top of the bar.

Copyright © Sipwise GmbH - All rights reserved

5.2.3. Central Panel

The central part of the page is divided in three areas: nav bar, sub-menu and content panel.

Nav Bar

The *Nav Bar* shows a *Breadcrumb* trail, indicating the current view and the hierarchy, allowing a quick way to reach the upper level of the structure.

Beside the *Breadcrumb*, the *Action buttons* related to the selected module/feature are shown. Usual buttons are: *Add*, *Edit*, *Delete*, *Download CSV*, and *Upload CSV*.

At the right side of the bar is located the *Search area*. This area is shown only when the *Content Panel* contains a table. It is composed by two fields:

- **Search By**: a drop-down field that allows to select on which column or element the filter has to be applied.
- **Search Box**: the arbitrary text that has to be searched in the table.

The * symbol can be used as *wildcard* for zero-or-more characters. The * is prepended and appended implicitly to the string entered in *Search* textbox to make filtering easier, for almost all tables.

NOTE The *Call History* table behaves slightly differently when it comes to wildcard usage. The * wildcard needs to be entered explicitly by the user if needed. This is necessary for performance reasons.

Sub-Menu

Directly adjacent to the Main Menu bar is a secondary, *Sub-Menu* dedicated to the specific module/feature being audited. This vertical list allows the operator to toggle between localized functional settings related to the current context.

NOTE The *Sub-Menu* is not always visible, because related to the module/feature currently selected.

Content Panel

The central workspace of the page. It usually contains a table to show already created elements or a form to create/edit new elements.

Tables

When selecting a module in the *Settings* tree, a table that lists the already created elements for that module is shown in the *Content Panel*.

Copyright © Sipwise GmbH - All rights reserved

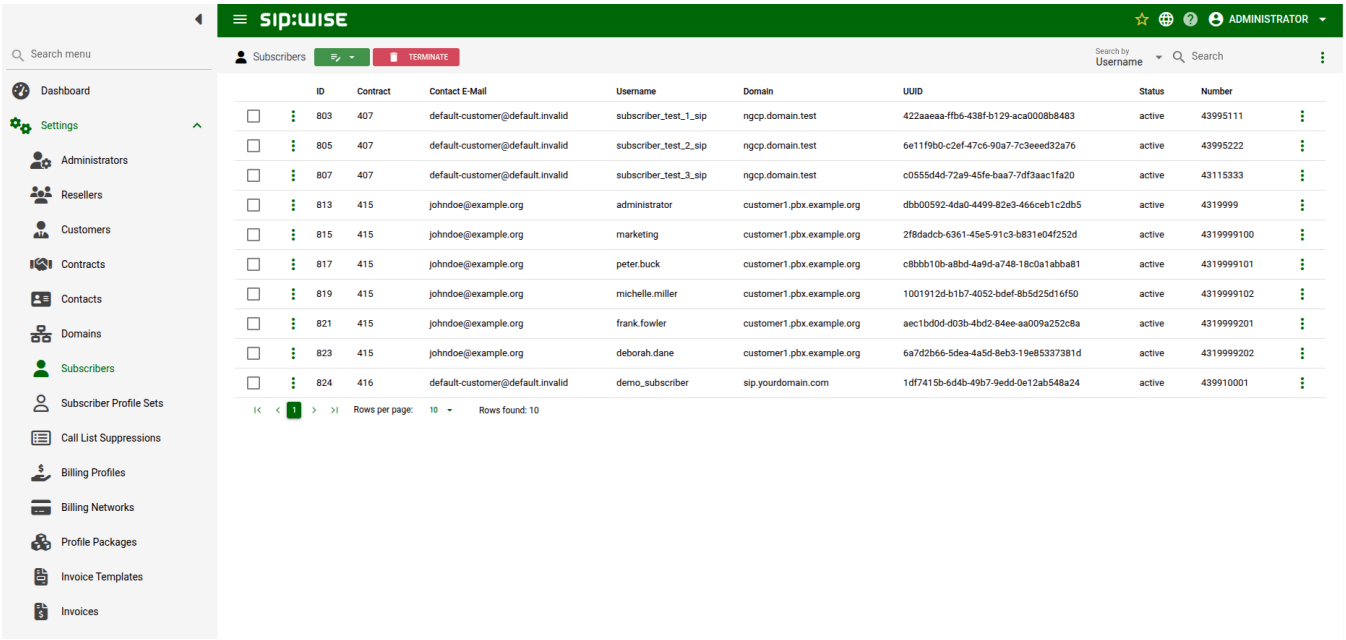

Figure 16. Example of a table

The table usually lists only the first 10 entries, but you can increase the number of elements that are shown and move through the pages using the control buttons below the table.

The columns can be ordered in descending or ascending way.

In some tables, it is possible to inline edit some of the values like text or boolean. The fields that can be edited are easily identifiable: the mouse arrow will turn into a hand when hovering on them.

Clicking on the 3 dots beside each row of the table, gives access to the contextual *Action Menu*. The available actions are usually related to editing, deleting, ordering the current element or to directly reaching some specific configuration of the entry (e.g. *Preferences* of the *Subscribers*).

Most of the actions provided by the *Action Menu* are available also in the *Nav Bar*. To use them, firstly mark the checkbox of one of the table entries and then press the *Edit* or *Delete* buttons of the *Nav Bar*.

NOTE The selection of multiple entries using the checkbox is currently not supported.

Forms

When the *Add* or *Edit* buttons are pressed, the *Content Panel* shows a form with all the fields related to the current selected module/feature.

Copyright © Sipwise GmbH - All rights reserved

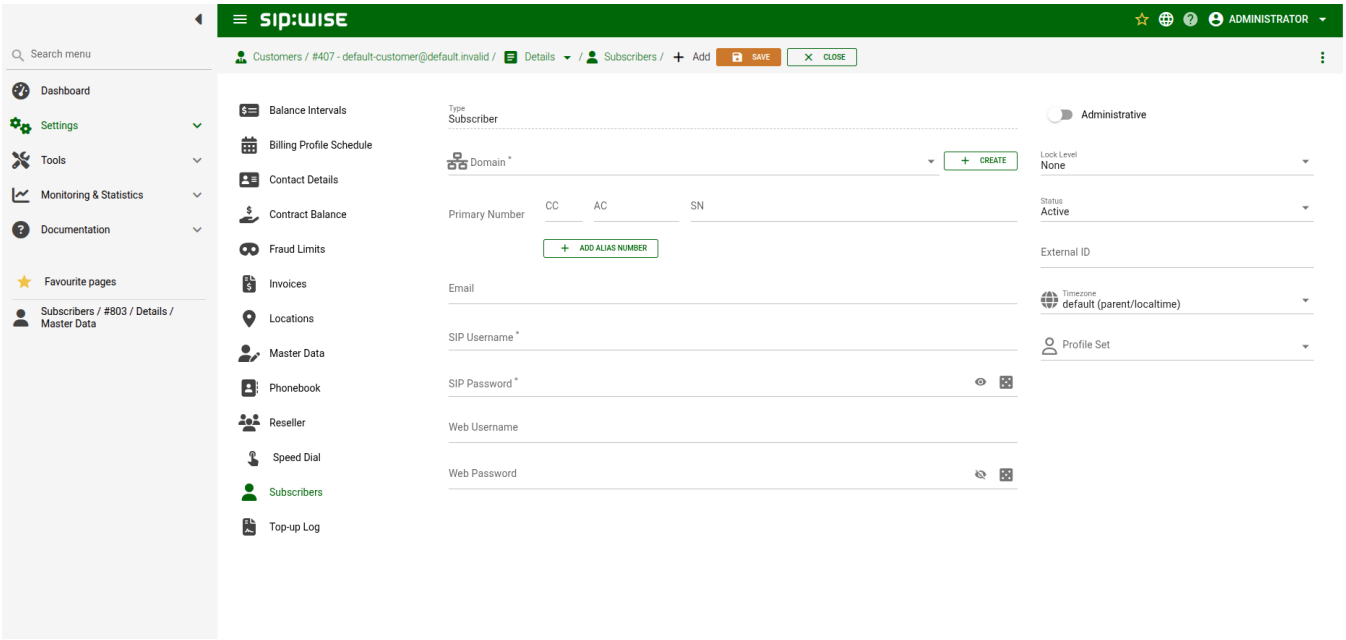

Figure 17. Example of a form

Fill all the necessary fields and then press *Save* to confirm the creation or modification of the element, or press *Close* in case you want to abandon the action. In case of editing of an element, the button *Reset* allows you to remove all the modifications currently done to the fields, reporting the status of the configuration to the previous one.

5.3. Creating Resellers

The resellers allow you to give access to virtual operators and grant limited access to the administrative web interface (the reseller administrator will only see his own customers, domains and billing profiles). The reseller is free to operate under its own brand.

To create a new reseller go to *SettingsResellers* and click on *Add*.

In order to create a reseller, you must select a reseller contract for the correct billing. You will most likely want to create one contract per reseller.

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

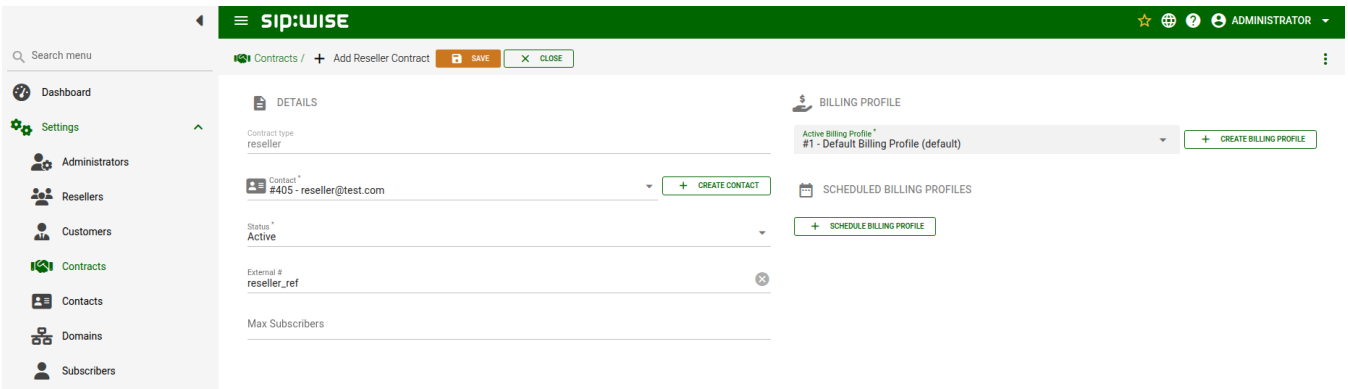

Figure 18. Create Reseller Contract

Copyright © Sipwise GmbH - All rights reserved

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

- **Contract:** select the id of the contract created before
- **Name:** give a name to the resellet
- **Status:** select 'Active'. Other possible status are 'Locked' and 'Terminated'

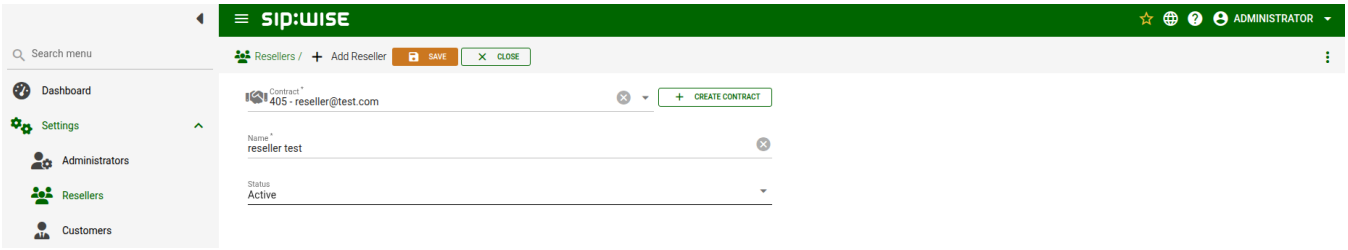

Figure 19. Create New Reseller

Then click *Save* to create the reseller.

NOTE To grant admin interface access to the newly created reseller, see Managing System Administrators chapter.

5.4. Creating a SIP Domain

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

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

To create a SIP domain, follow these steps:

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

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

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

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

   Go to the *SettingsDomains* page and click on *Add* to create a new SIP Domain using the FQDN created above.

Copyright © Sipwise GmbH - All rights reserved

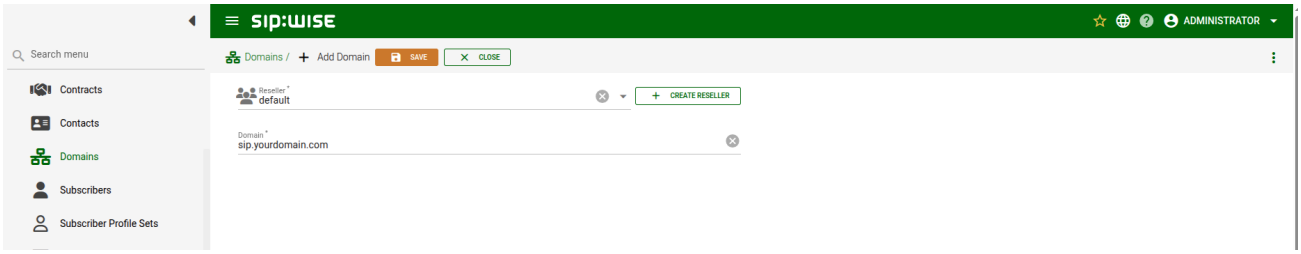

Figure 20. Domain Creation

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

3. Adjust the new SIP domain's preferences, if necessary, clicking on the three dots beside the newly created domain and then selecting *Preferences*.

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

5.5. Creating a Customer

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

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

To create a Customer, go to *SettingsCustomers* and click on *Add*.

Each *Customer* has a *Contact*, a container for the personal and legal information that identifies a private or corporate customer.

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

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

Copyright © Sipwise GmbH - All rights reserved

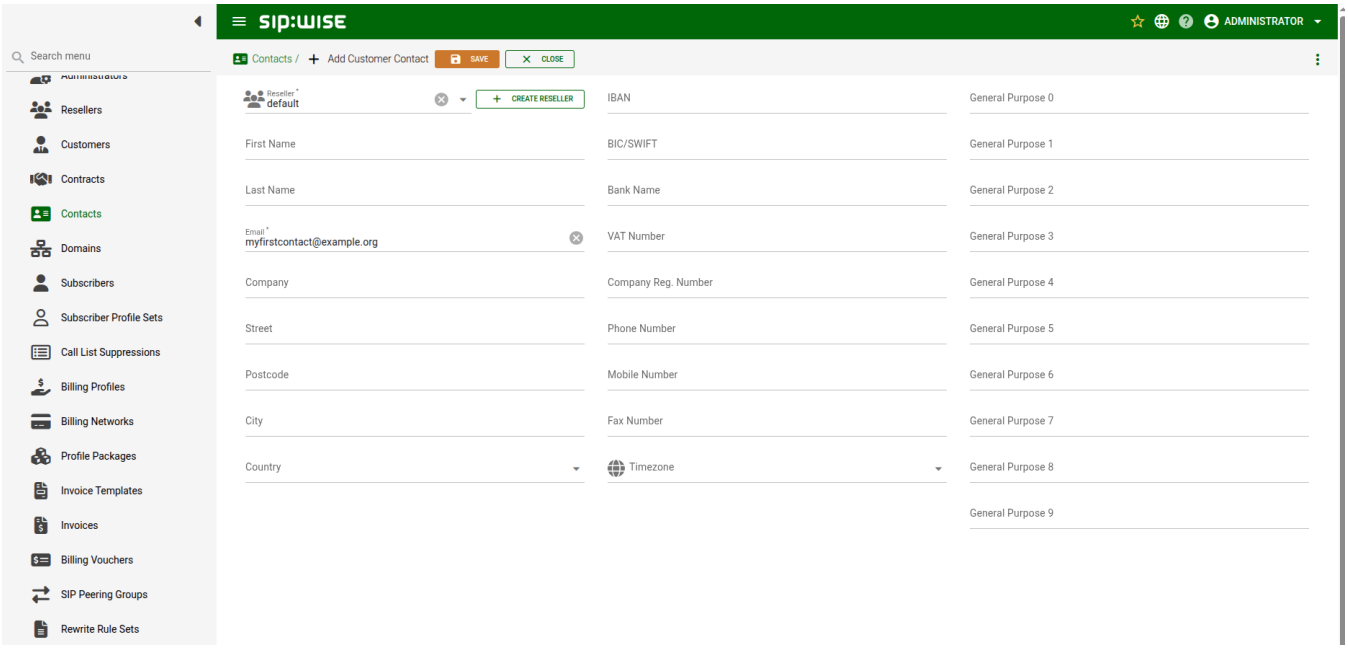

Figure 21. Create Contact

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

You will be redirected back to the *Customer* form. Select the newly created *Contact* and a *Billing Profile*, then press *Save*.

You will now see your first *Customer* in the list. Click on the three dots and select *Details* to make extra configuration if necessary.

5.6. Creating a Subscriber

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

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

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

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

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

Copyright © Sipwise GmbH - All rights reserved

Found response, it falls back to 012345 (the user-part with which the callee is registered).

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

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

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

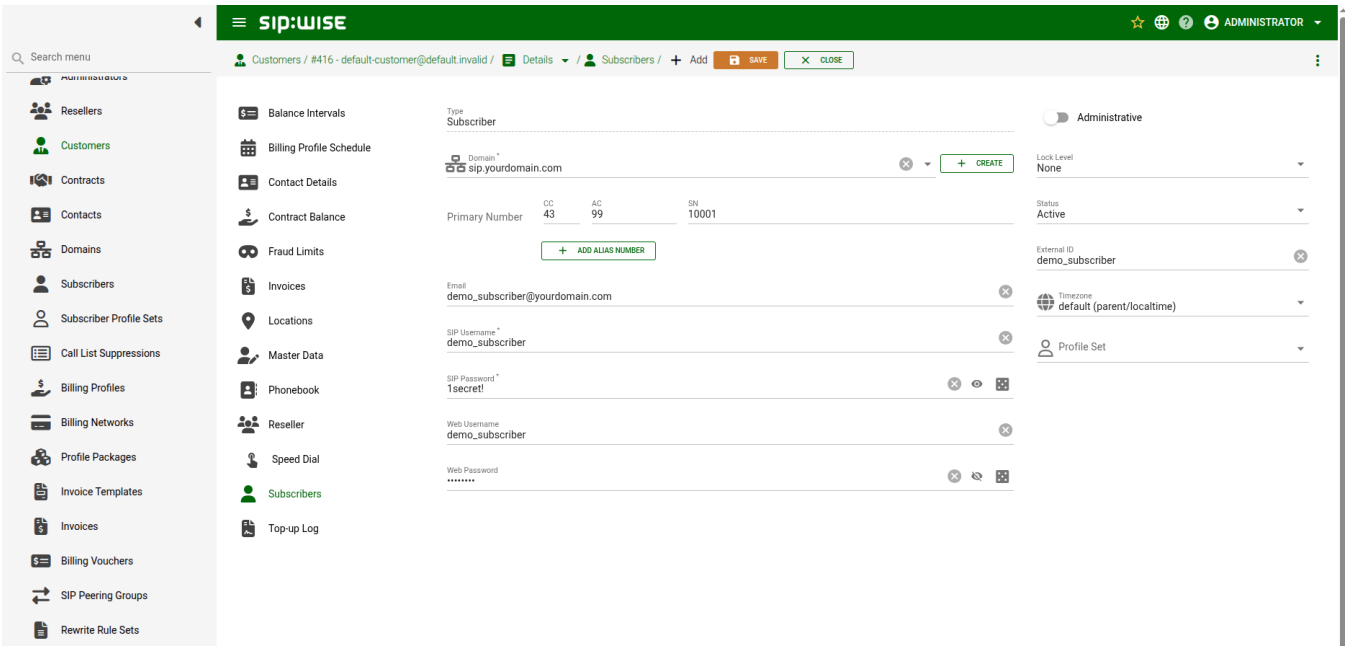

Figure 22. Creating a Subscriber

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 are able to register your subscribers to Sipwise C5 and place calls between these subscribers.

 Copyright © Sipwise GmbH - All rights reserved

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

5.7. Domain Preferences

The *Domain Preferences* are the default settings for *Subscriber Preferences*, so you should set proper values there if you do not want to configure each subscriber separately. You can later override these settings in the *Subscriber Preferences* if particular subscribers need special settings. To configure your *Domain Preferences*, go to *SettingsDomains*, click on the three dots of the domain you need to configure and then select *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 Configuring Rewrite Rule Sets, then assign it to the domain by editing the *rewrite_rule_set* preference.

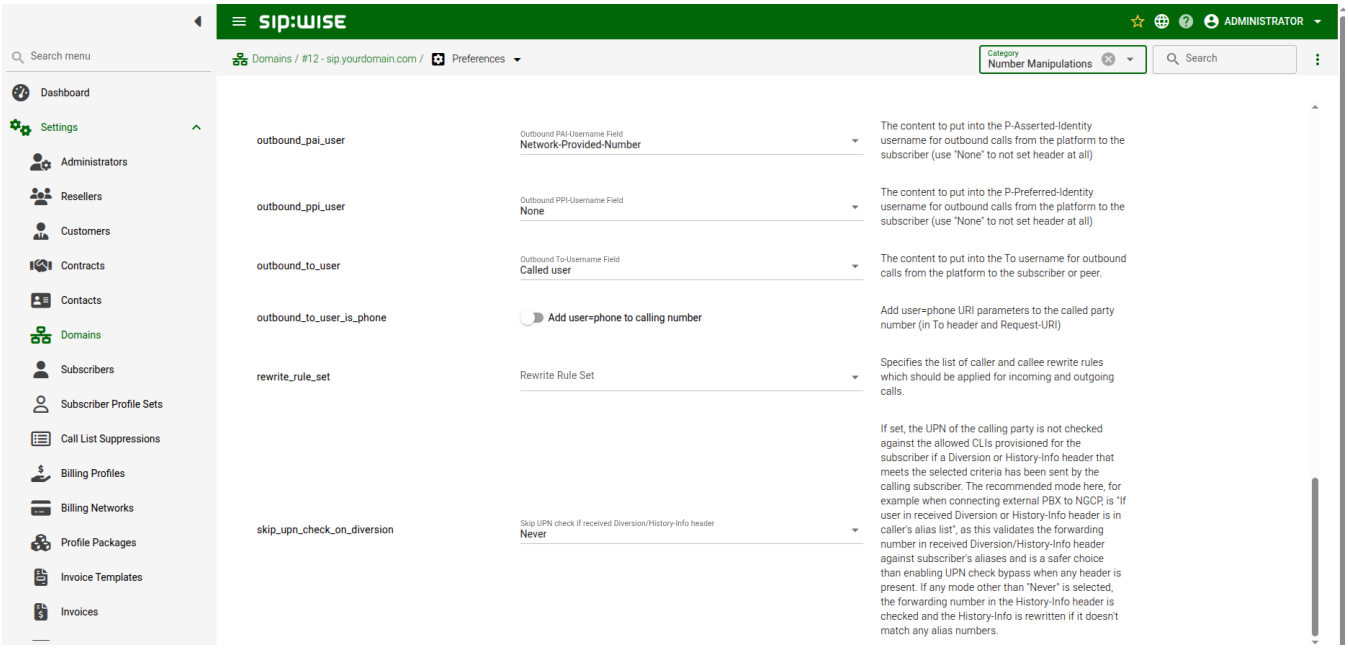

Figure 23. Configure Rewrite Rule Set

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.

5.8. Subscriber Preferences

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

Copyright © Sipwise GmbH - All rights reserved

To configure your *Subscriber*, go to *SettingsSubscribers*, click on the three dots of the subscriber you want to configure, then select *Preferences*.

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 the `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`.

5.8.1. Subscriber Authentication for Outbound Calls

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

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

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

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

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

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

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

 Copyright © Sipwise GmbH - All rights reserved

The *peer_auth_hf_user* preference is optional and can be skipped. It allows you to set a specific username for the Proxy-Authorization header. The *peer_auth_registrar_server* is also optional. In case it is not set, the REGISTER will be sent to the *peer_auth_realm* domain. In case there is a scenario where the realm is different from the registrar server domain/IP, the 'peer_auth_registrar_server' can be set. In this case the REGISTER will be sent there and the 'peer_auth_realm' used only as Digest Realm.

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

5.8.2. Subscriber Authentication for an Outbound Registration

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

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

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

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

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

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

- **peer_auth_register:** *True*

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

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

5.8.3. Subscriber's Preference for Media in Terms of IP Family (IPv4/IPv6)

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

NAT and Media Flow Controlipv46_for_rtpproxy, which can be set to:

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

Copyright © Sipwise GmbH - All rights reserved

- **use domain default** - preference is inherited from the domain values

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

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

5.9. Creating Peerings

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

5.9.1. Creating Peering Groups

Click on *Add* to create a new group.

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

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

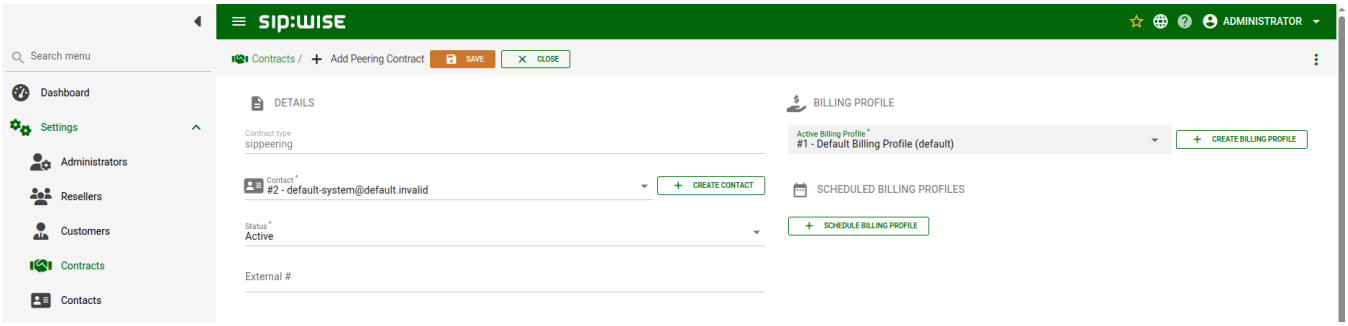

Figure 24. Create Peering Contract

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

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

 Copyright © Sipwise GmbH - All rights reserved

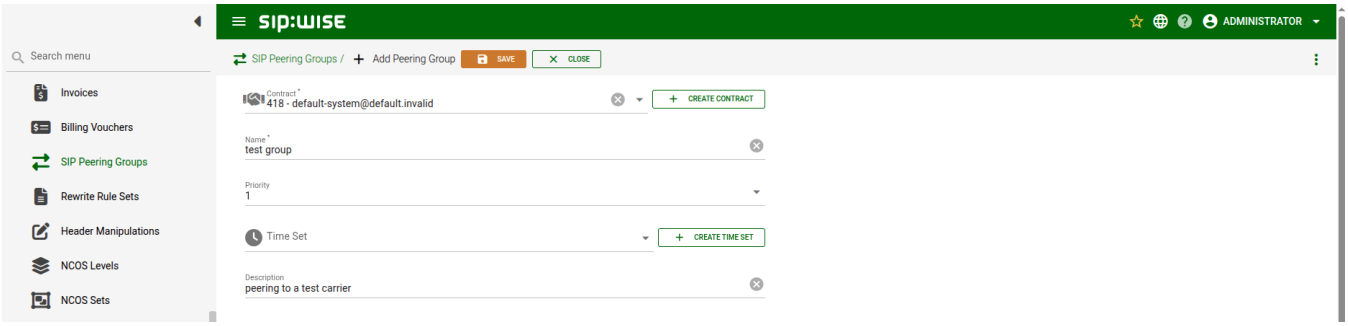

Figure 25. Create New Peering Group

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.

5.9.2. 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 three dots of the newly created *Peering Group* and select *Details. Then, select the Peering Severs tab and click on Add_*.

In this example, we will create a peering server with IP *203.0.113.42* 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
- **Site**: All (default)
- **Enable Probing:** enable it, if remote side supports SIP OPTIONS ping

Copyright © Sipwise GmbH - All rights reserved

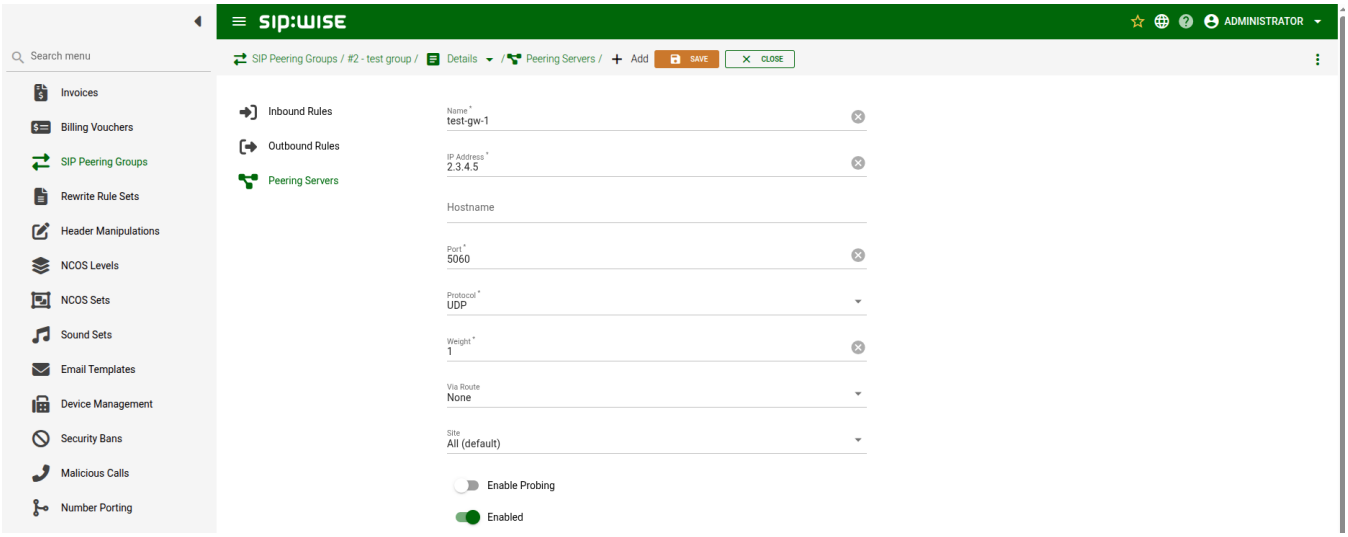

Figure 26. Peering Server Properties

Click *Save* to create the peering server.

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

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

You can force an additional hop (e.g. via an external SBC) towards the peering server by using the *Via Route* option. The available options you can select there are defined in `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.

The *site* parameter will be useful only in case of Multi-Site Geo-Redundant Setup architecture. The parameter allows to control to which site the peer should be attached to. Selecting "All" the peering will be connected to all the sites.

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

NOTE Requests coming from a SIP peering are matched not only by the IP address and a

Copyright © Sipwise GmbH - All rights reserved

transport protocol, but also using the source port of a message. This means, if your SIP peering server created at Sipwise C5 has the 'Port' value set to '5060', then it is expected that messages (requests) coming from this SIP peering, will have the source port '5060'. This however applies only to the UDP transport based connections (TCP and TLS are matched only using an IP address and a transport protocol).

Outbound Peering Rules

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

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

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

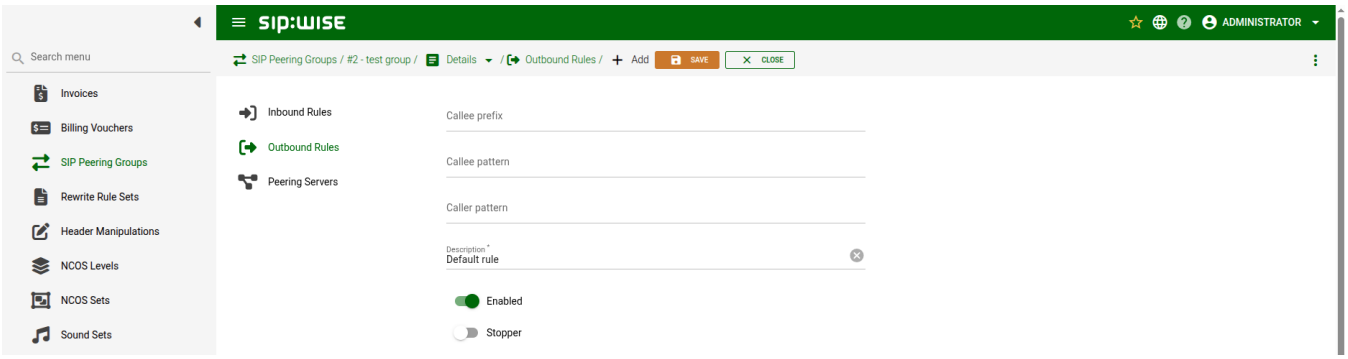

Figure 27. Outbound Peering Rule Properties

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

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

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

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

Inbound Peering Rules

Sipwise C5 supports filtering SIP INVITE requests sent by SIP peers. The system administrator may

Copyright © Sipwise GmbH - All rights reserved

define one or more matching rules for SIP URIs that are present in the headers of SIP INVITE requests, and select which SIP header (or part of the header) must match the pattern declared in the rule.

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

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

In order to create an inbound peering rule you have to select the *_Inbound Rules* tab, then click on the *Add* button.

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

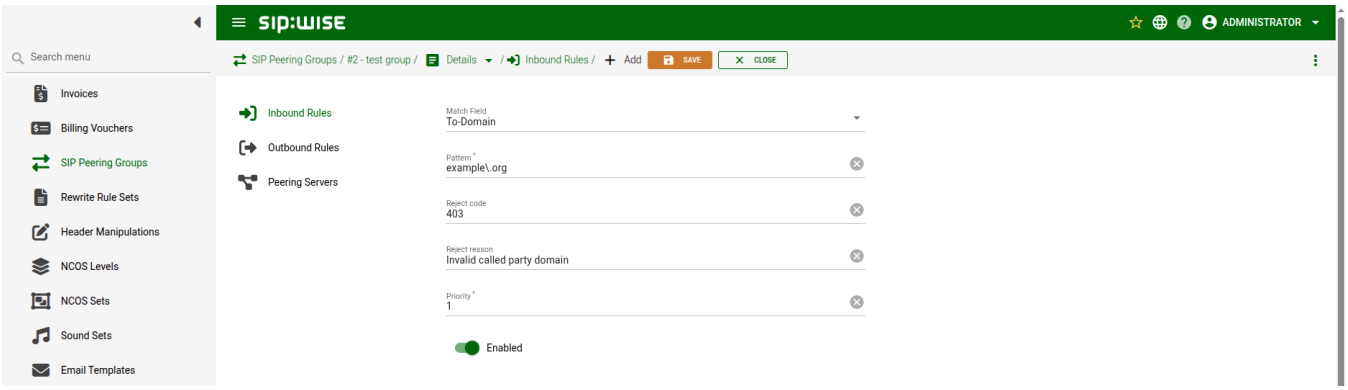

Figure 28. Inbound Peering Rule Properties

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

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

Routing Order Selection

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

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

Copyright © Sipwise GmbH - All rights reserved

Callee's username matches *callee prefix*

Callee's URI matches *callee pattern*

Caller's URI matches *caller pattern*

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

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

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

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

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

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

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

```
#!/usr/bin/perl

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

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

my $iters = 10000;
my @rands;

for (my $i=1; $i <= $iters; $i++) {
    my %elem;
    for (my $j=0; $j < $num_args; $j++) {
```

Copyright © Sipwise GmbH - All rights reserved

```
        my $random = int(rand(2000000000));
        $elem{"$j"} = $ARGV[$j] * $random;
    }
    push(@rands, \%elem);
}

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

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

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

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

```
# lcr_weight_test.pl 1 2

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

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

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

Least Cost Routing (LCR) Configuration

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

 Copyright © Sipwise GmbH - All rights reserved

cheapest SIP peers saving your termination cost.

To enable LCR routing, do the following:

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

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

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

   Callee's username matches *callee prefix*

   Callee's URI matches *callee pattern*

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

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

Let us consider a short example. There are two peering groups (PG1 and PG2) that can deliver calls to New York (e.g. 12121234567) and they have the following rates:

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

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

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

5.9.3. Authenticating and Registering Against Peering Servers

Copyright © Sipwise GmbH - All rights reserved

Proxy-Authentication for Outbound Calls

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

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

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

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

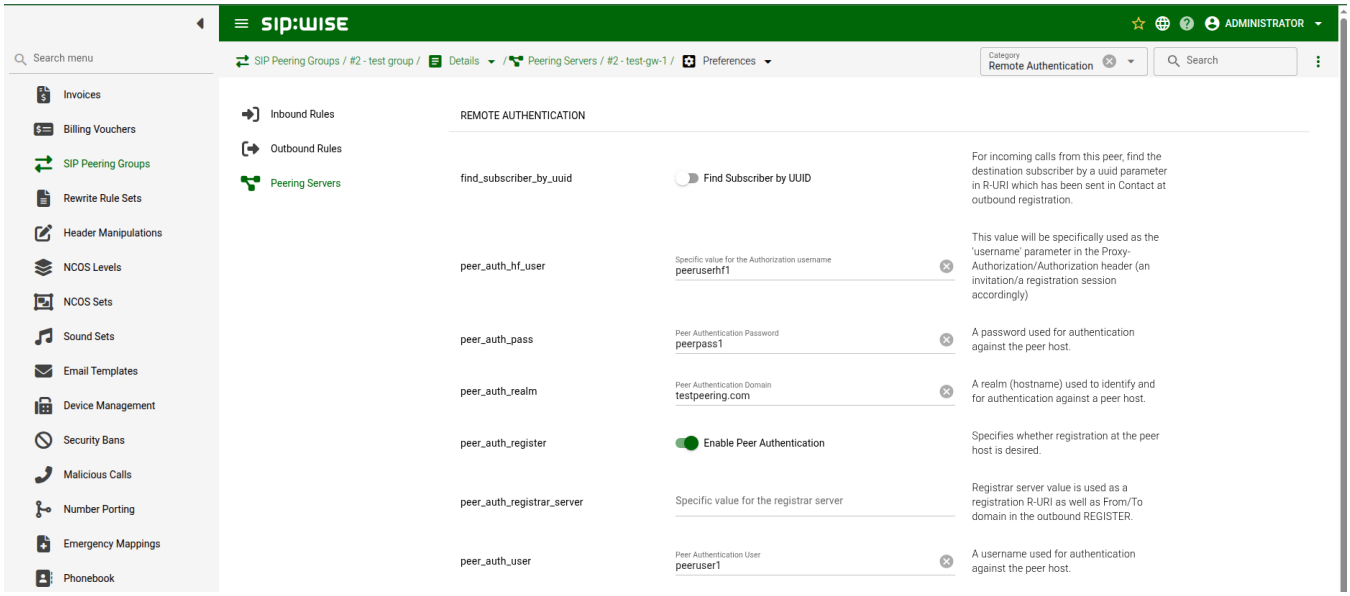

Figure 29. Set Authentication Credentials For Peer

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

TIP If **peer_auth_realm** is set, the system may overwrite the Request-URI with the *peer_auth_realm* value of the peer when sending the call to that peer or *peer_auth_realm* value of the subscriber when sending a call to the subscriber. If you need the

 Copyright © Sipwise GmbH - All rights reserved

> replacement, you should set `set_ruri_to_peer_auth_realm: 'yes'` in `config.yml`.

Registering at a Peering Server

It is simple as that, you need to tick a check-box *peer_auth_register*, and if previously all needed authentication data has been provided (user, password and realm) the registration process will start right away.

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

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

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

5.10. 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:

Copyright © Sipwise GmbH - All rights reserved

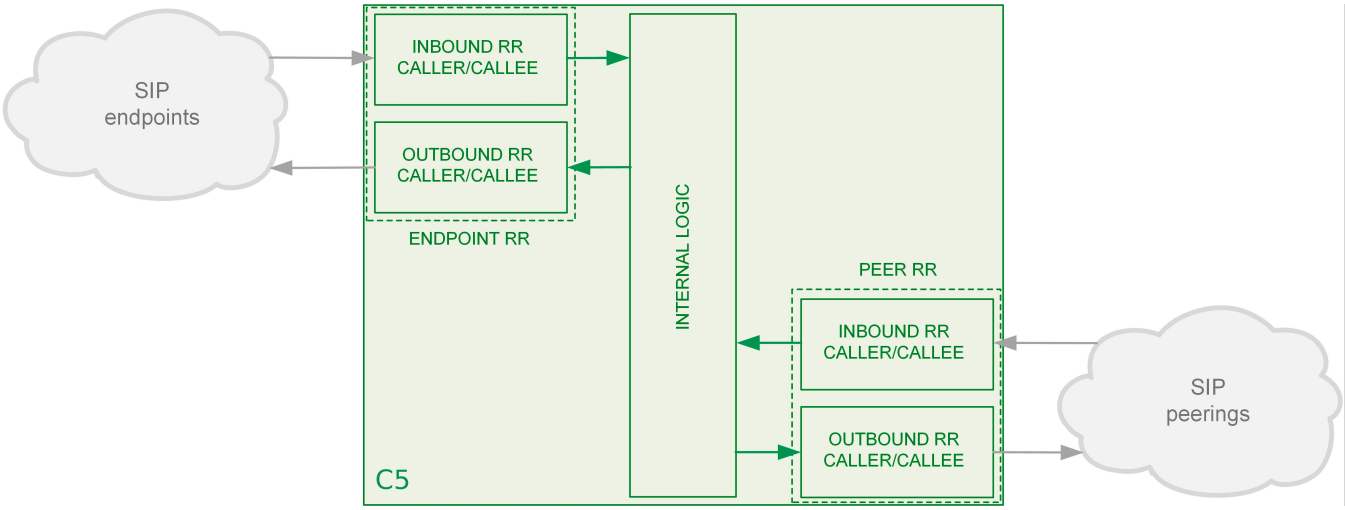

Figure 30. How Rewrite Rules are used

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

 Copyright © Sipwise GmbH - All rights reserved

Click *Add* and fill in the form accordingly.

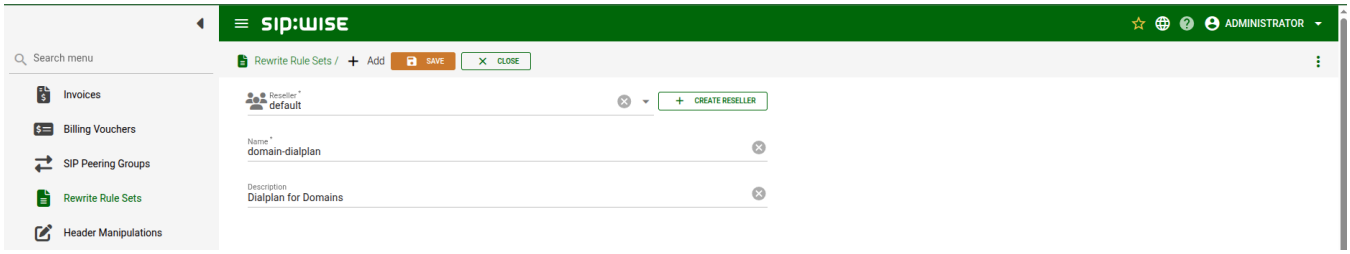

Figure 31. Save New Rewrite Rule Set

Press the *Save* button to create the set.

To view the *Rewrite Rules* within a set, click on the three dots and then select *Rules*. 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.

5.10.1. 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 will 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:

Copyright © Sipwise GmbH - All rights reserved

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

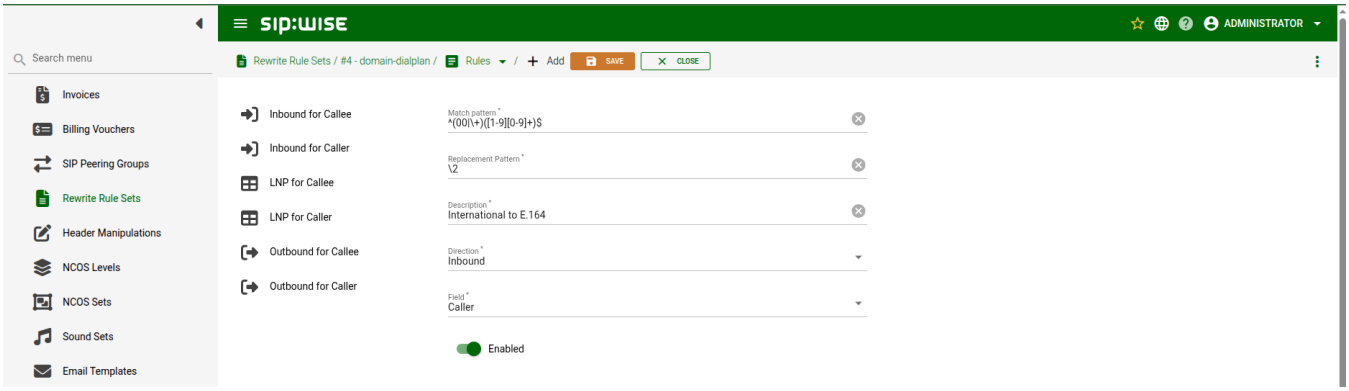

Figure 32. Save New Rewrite Rule

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 cannot use them when a call comes from a SIP peer (the variables will be empty in this case).

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

5.10.2. 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:

 Copyright © Sipwise GmbH - All rights reserved

- 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.

5.10.3. 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 is not already the case).

Copyright © Sipwise GmbH - All rights reserved

5.10.4. 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 the 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 an 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`

5.10.5. Emergency Number Handling

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

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

Simple Emergency Number Handling Overview

The overview of emergency call processing is as follows:

Copyright © Sipwise GmbH - All rights reserved

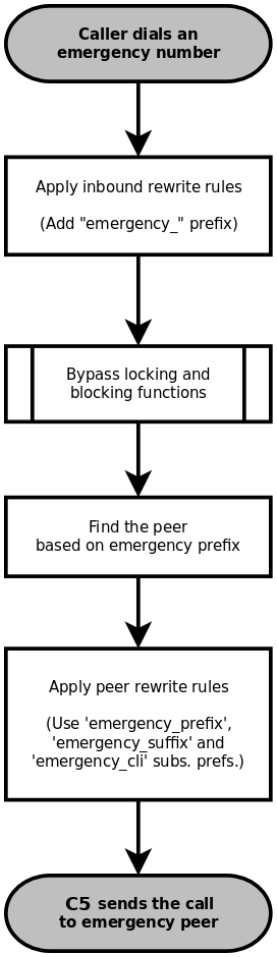

Figure 33. Simple Emergency Call Handling

Configuring Emergency Numbers is also done via Rewrite Rules.

Tagging Inbound Emergency Calls

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

Tag an Emergency Call

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

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

Copyright © Sipwise GmbH - All rights reserved

Normalize Emergency Calls for Peers

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

Normalize Emergency Call for Peer

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

5.10.6. Emergency Geo-Location Formats

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

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

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

PIDF-LO Format

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

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

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

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

- `timestamp`

 Copyright © Sipwise GmbH - All rights reserved

- retention-expiry

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

An example of XML document with the mandatory elements:

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

Copyright © Sipwise GmbH - All rights reserved

cirpack Format

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

5.10.7. Assigning Rewrite Rule Sets to Domains and Subscribers

Once you have finished defining 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 the default set for all subscribers. To do so, go to *SettingsDomains*, click on the three dots and select *Preferences* on the domain you want the set to assign to. Edit the rewriting rule and select the Rewrite Rule Set created before.

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

5.10.8. Creating Dialplans for Peering Servers

For each peering server, you can use one of the Rewrite Rule Sets that was created previously as explained in Configuring Rewrite Rule Sets (keep in mind that special variables `${caller_ac}` and `${caller_cc}` cannot 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 do not 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 putting them from national into E.164 format.

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

5.10.9. Call Routing Verification

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

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

- What is considered during the test:

  subscriber preferences: cli and allowed_clis

  domain / subscriber / peer rewrite rules

- What is not taken into account during the test:

  other subscriber or peer preferences

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

 Copyright © Sipwise GmbH - All rights reserved

You can access the utility following the path on the Admin web interface: *ToolsCall 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:

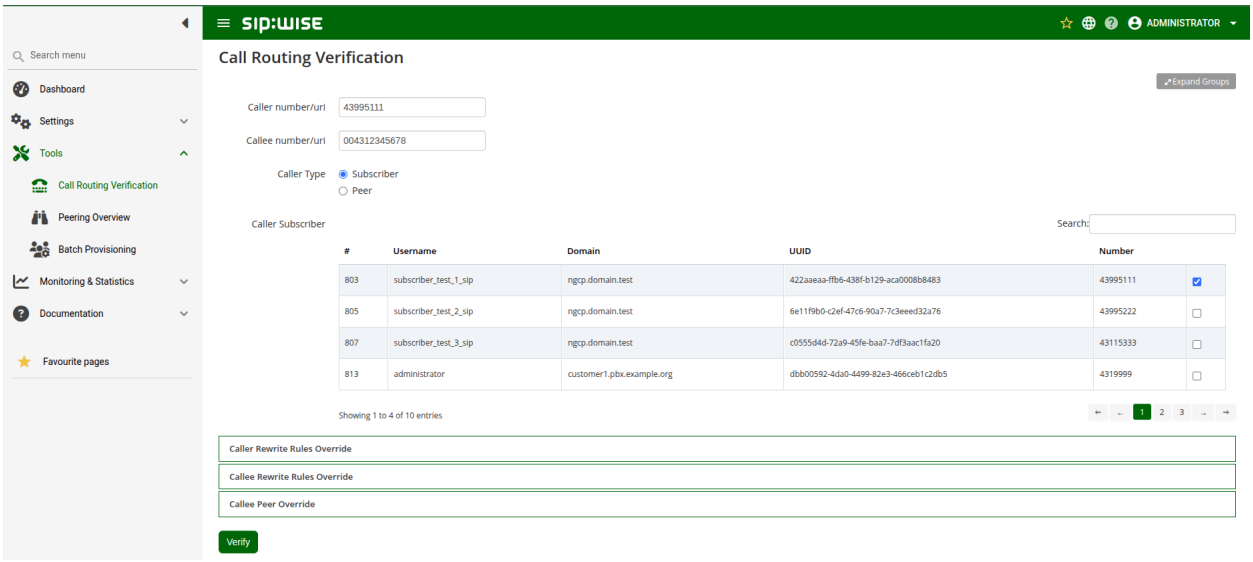

Figure 34. Call Routing Verif. - Only Numbers - Input

   Result:

Copyright © Sipwise GmbH - All rights reserved

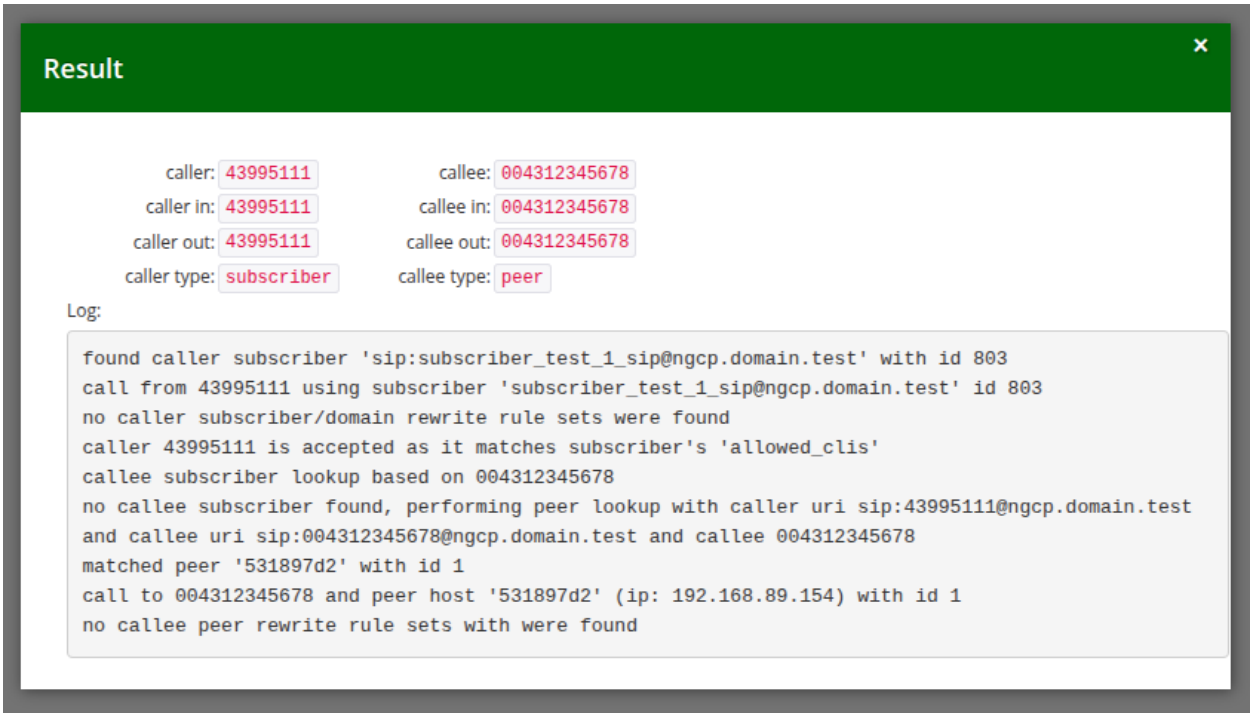

Figure 35. Call Routing Verif. - Only Numbers - Result

 Copyright © Sipwise GmbH - All rights reserved

Chapter 6. Billing

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

6.1. Billing Profiles

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

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

6.1.1. Creating Billing Profiles

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

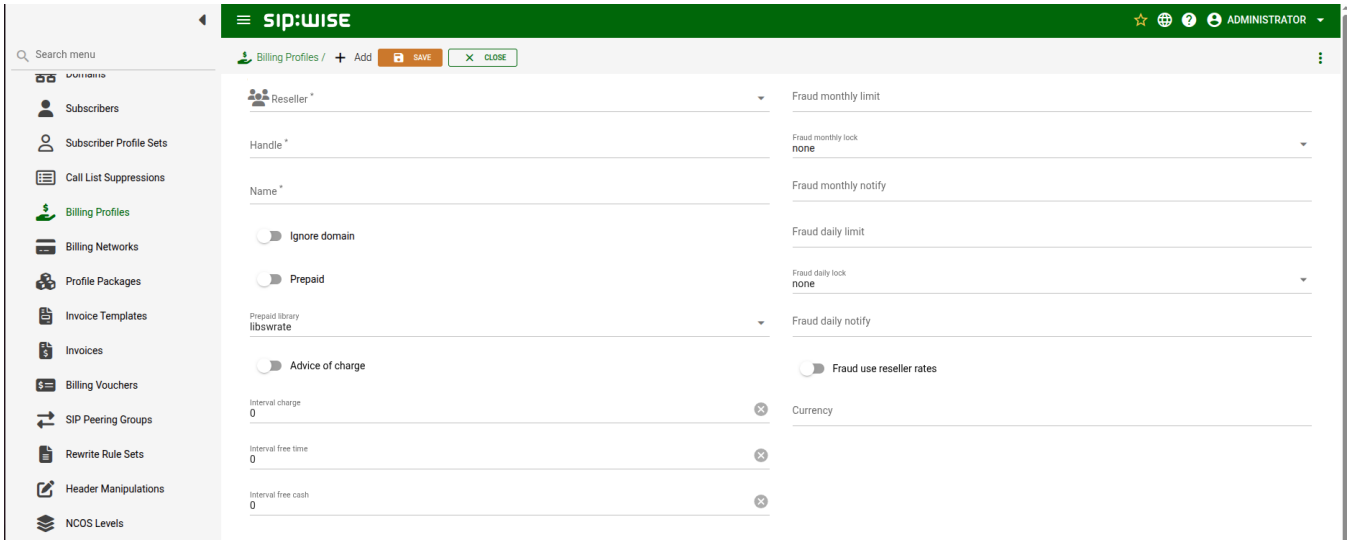

Figure 36. Create Billing Profile

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.

Copyright © Sipwise GmbH - All rights reserved

- **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.
- **Fraud use reseller rates**: correlate the reseller billing profile and the respective subscriber billing profile in terms of fraud handling.
- **Currency**: The currency symbol for your currency. Any UTF-8 character may be used and will be printed in web interfaces.

6.1.2. Creating Billing Fees

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

To set up billing fees, click on the three dots of the billing profile you want to configure and the select *Fees*. Billing fees may be uploaded using a configurable CSV file format, or entered directly via the web interface by clicking *Add*. To configure the CSV field order for the file upload, rearrange the entries in the `www_admin.fees_csv.element_order.array` in `config.yml` file and execute the command `ngcpcfg apply 'changed fees element order'`. The following is an example of working CSV file to upload (pay attention to double quotes):

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

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

Copyright © Sipwise GmbH - All rights reserved

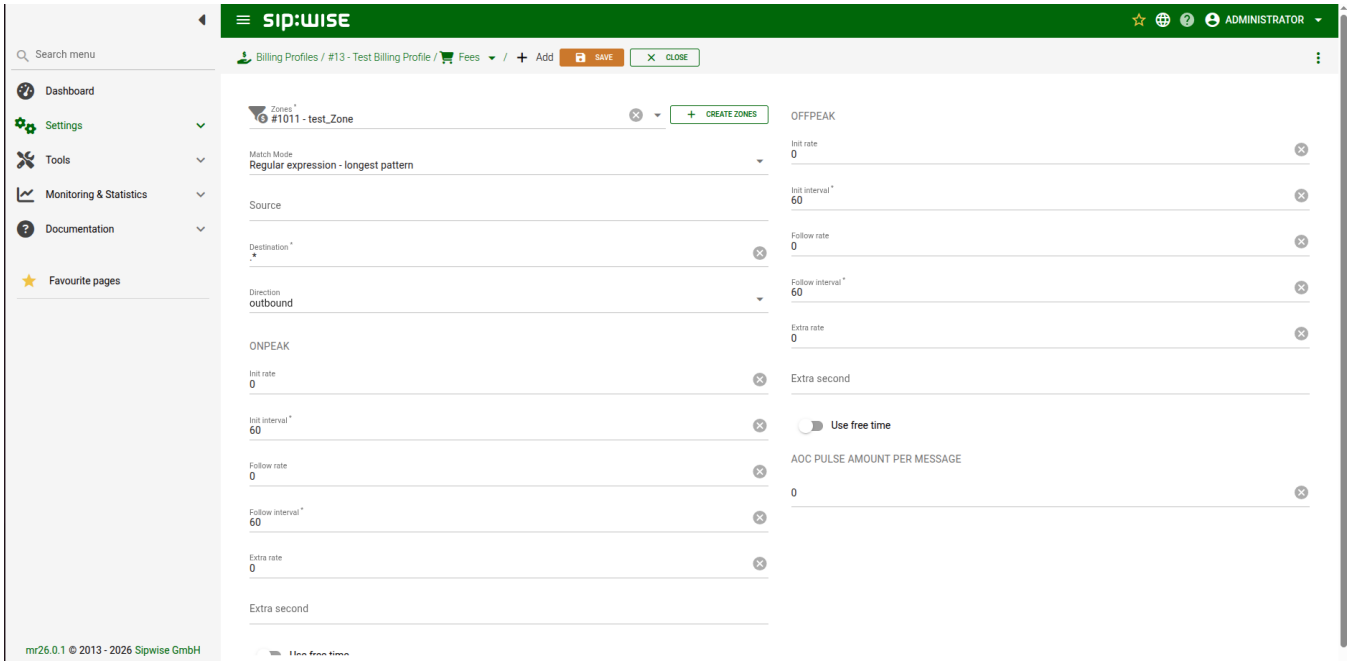

Figure 37. Create Billing Fee

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

- **Zones**: A zone for a group of fees. May be used to group fees for simplified display, e.g. on invoices. (e.g. *foreign zone 1*)
- **Match Mode**: The mode for matching a fee's source and destination patterns against a CDR's source fields (the caller given by `<source_cli>@<source_domain>` or `<source_cli>` only) and destination fields (the callee given by `<destination_user_in>@<destination_domain>` or `<destination_user_in>` only). Each of the currently supported modes below provides different flexibility and speed:
  1. Exact string (destination): The destination string has to match the destination from the CDR exactly. Fastest, O(log(#fees)). In csv files, this match mode is specified by `exact_destination`.
  2. Prefix string: The fee's source/destination represent strings which both the source/destination from the CDR have to start with. The fee with the longest destination prefix is picked. If there are multiple, the one with the longest source prefix is picked. In contrast to regular-expression based match modes, this algorithm uses database index lookups instead of SQL `REGEXP` table scans. The performance boundary is O(length(cdr src) * length(cdr dest) * log(#fees)), hence this will be the preferred mode for tens of thousands of fees in place or high throughput (LCR, rating peer-to-peer calls). In csv files, this match mode is specified by `prefix`.
  3. Regular expression - longest match: The fee's source/destination patterns represent PCREs which both have to match the source/destination from the CDR. The fee with the longest match within the destination string is picked. If there are multiple, the one with the longest match within the source string is picked. In csv files, this match mode is specified by `regex_longest_match`.
  4. Regular expression - longest pattern: The fee's source/destination represent PCREs which both have to match the source/destination from the CDR. The fee with the longest (most distinctive) destination pattern is picked. If there are multiple, the one with the longest (most distinctive) source pattern is picked. In csv files, this match mode is specified by `regex_longest_pattern`.

Copyright © Sipwise GmbH - All rights reserved

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

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

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

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

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

IMPORTANT There are several internal services (vsc, conference, voicebox) which will need a specific destination entry with a domain-based destination. If you do not 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*.
- **Onpeak extra rate**: The rate to charge if the call time exceeds *extra second* in cent (of whatever currency, represented as a floating point number). Applicable to calls started during onpeak hours.
- **Onpeak use free time**: Specifies whether free time minutes may be used when calling this destination during onpeak hours. Specified in the file upload as 0, n[o], f[alse] and 1, y[es], t[rue] respectively.

 Copyright © Sipwise GmbH - All rights reserved

- **Onpeak extra second**: If defined, an extra rate will be charged at the given second of call time for post-paid calls. Applicable to calls started during onpeak hours.
- **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.
- **Offpeak extra rate**: See onpeak extra rate. Applicable to calls started during offpeak hours.
- **Offpeak extra second**: See onpeak extra second. Applicable to calls started during offpeak hours.
- **Offpeak use free time**: Specifies whether free time minutes may be used when calling this destination during off-peak. Specified in the file upload as 0, n[o], f[alse] and 1, y[es], t[rue] respectively.

6.1.3. Creating Off-Peak Times

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

To set off-peak times for a weekday, click on the + button or select one entry in the weekly calendar. For each period you need to provide:

- **Day of the week**: day of the week of the off-peak period.
- **Start time**: timestamp in the form of *hh:mm:ss* specifying a time of day for the start of the off-peak period.
- **End time**: timestamp in the form of *hh:mm:ss* specifying a time of day for the end of the off-peak period.

Click on *Add time slot* to add more off-peak time periods corresponding to the same day of the week.

Copyright © Sipwise GmbH - All rights reserved

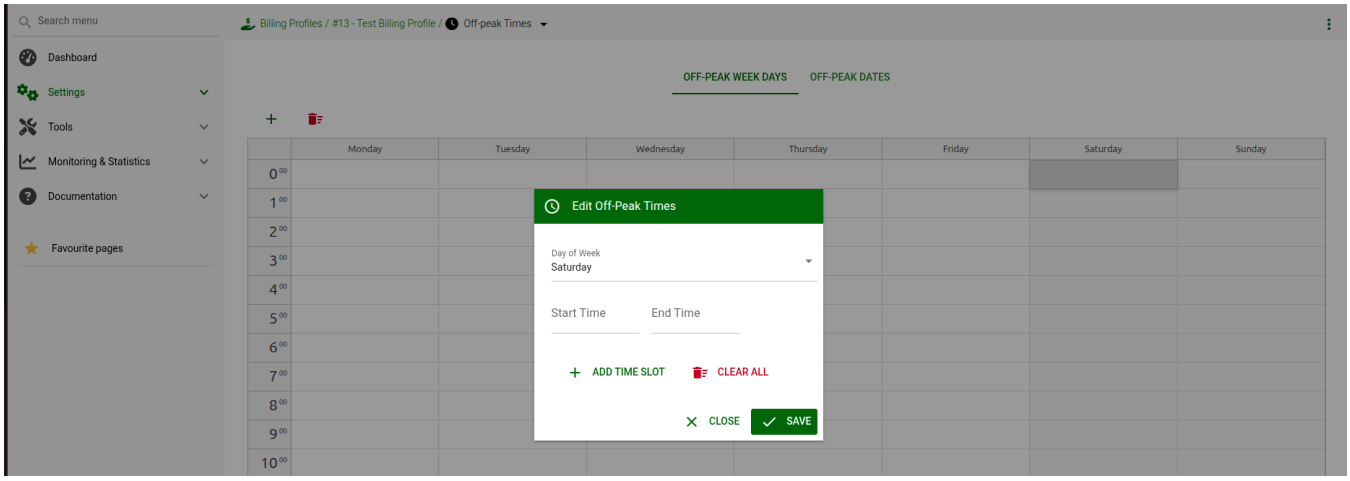

Figure 38. Create Weekday Offpeak

To specify off-peak ranges based on calendar dates, click on the + button. Enter a datetime value in the form of *MM-DD-YYYY hh:mm:ss* into the *Start Date* input field and *End Date* input field to define a range for the off-peak period.

Click on *Add data range* to add more off-peak ranges period in one time.

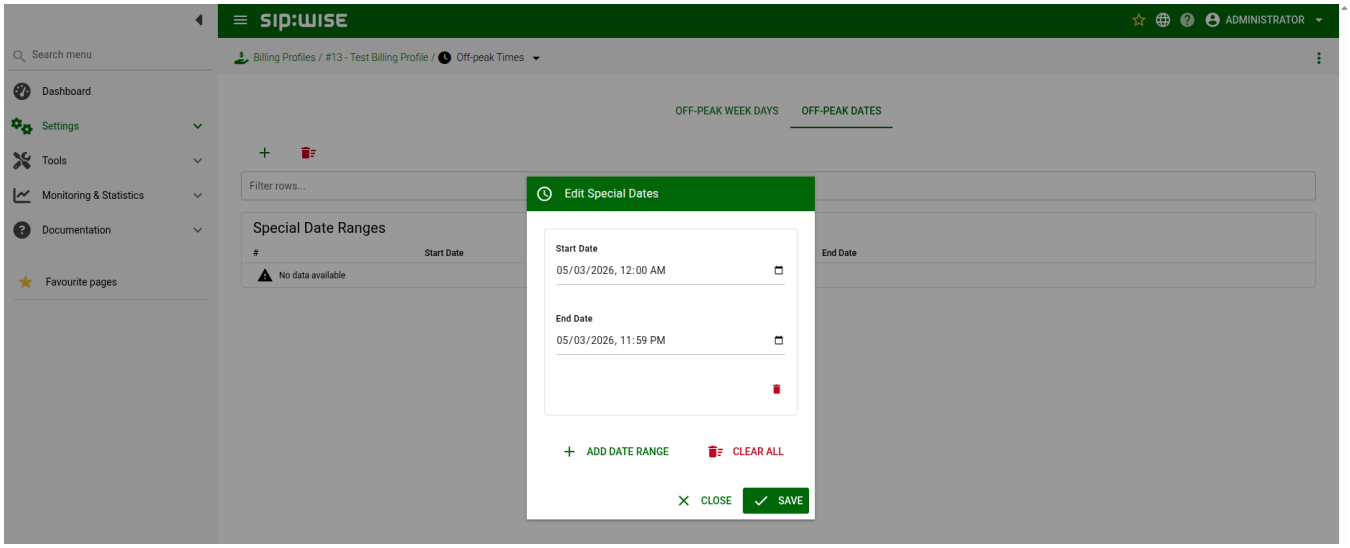

Figure 39. Create Special Offpeak

6.2. Fraud Detection and Locking

The Sipwise C5 supports a fraud detection feature, which is designed to detect accounts causing unusually high customer costs, and then to perform one of several actions upon those accounts. This feature can be enabled and configured through two sets of billing profile options described in Creating Billing Profiles, namely the monthly (*fraud monthly limit*, *fraud monthly lock* and *fraud monthly notify*) and daily limits (*fraud daily limit*, *fraud daily lock* and *fraud daily notify*). Either monthly/daily limits or both of them can be active at the same time.

As soon as call costs of a CDR are determined by rate-o-mat, database tables for bookkeeping daily and monthly sums are updated. Fraud lock levels will be applied instantly in case of a *fraud event* - that is when either the *fraud monthly limit* or *fraud daily limit* got exceeded. If **fraud lock** is set to anything

 Copyright © Sipwise GmbH - All rights reserved

other than *none*, it will lock the account's subscribers accordingly (e.g. if **fraud lock** is set to *outgoing*, the account will be locked for all outgoing calls).

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

6.2.1. 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**: subscriber cannot place any calls, except calls to emergency destinations
- **incoming and outgoing**: subscriber cannot place and receive any calls, except calls to emergency destinations
- **global**: same restrictions as at **incoming and outgoing** level, additionally subscribers are not allowed to access the Customer Self Care (CSC) interface
- **ported**: only automatic call forwarding, due to number porting, is allowed

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

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

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

6.3. Notes on Billing and Call Rating

6.3.1. Cash Balance with Post-Paid Billing Profile

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

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

In case the account has a positive cash balance, the cost of the call will be deducted from that balance

Copyright © Sipwise GmbH - All rights reserved

and not considered as additional cost of that particular call for the customer.

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

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

6.4. Billing Data Export to Files

Legacy billing data export is done using CSV (*comma separated values*) files which are generated by a *cron* job and may be downloaded from the platform using the *cdrexport* user which has been created during the installation.

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

6.4.1. Glossary of Terms

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

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

6.4.2. 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
```

 Copyright © Sipwise GmbH - All rights reserved

```
number><suffix>
```

The definition of the specific fields is as follows:

Table 2. 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
```

6.4.3. File Format

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

File Header Format

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

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

The definition of the specific fields is as follows:

Table 3. 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:

Copyright © Sipwise GmbH - All rights reserved

```
007,0738
```

File Body Format for Call Detail Records (CDR)

The body of a CDR consists of a minimum of zero and a default maximum of 5000 lines. The platform operator can configure the maximum number of lines kept in a file by updating the `cdrexport.max_rows_per_file` parameter in `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 `config.yml` file, `cdrexport.admin_export_fields` parameter.

Table 4. 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 Sipwise C5 subscriber.)
`SOURCE_SUBSCRIBER_ID`	1-11	uint	Internal ID of calling party subscriber. Value is `0` if calling party is external.
`SOURCE_EXTERNAL_CONTRACT_ID`	0-255	string	External, arbitrary ID of calling party customer. (A string value shown as "External ID" property of an Sipwise C5 customer/peer.)
`SOURCE_ACCOUNT_ID`	1-11	uint	Internal ID of calling party customer.
`SOURCE_USER`	0-255	string	SIP username of calling party.
`SOURCE_DOMAIN`	0-255	string	SIP domain of calling party.
`SOURCE_CLI`	0-64	string	CLI of calling party in E.164 format.
`SOURCE_CLIR`	1	uint	`1` for calls with CLIR, `0` otherwise.
`SOURCE_IP`	0-64	string	IP Address of the calling party.

Copyright © Sipwise GmbH - All rights reserved

Body Element	Length	Type	Description
DESTINATION_USER_ID	36	string	Internal UUID of called party subscriber. Value is 0 if called party is external.
DESTINATION_PROVIDER_ID	0-255	string	Internal ID of the contract of called party provider (i.e. reseller or peer).
DESTINATION_EXTERNAL_SUBSCRIBER_ID	0-255	string	External, arbitrary ID of called party subscriber. (A string value shown as "External ID" property of an Sipwise C5 subscriber.)
DESTINATION_SUBSCRIBER_ID	1-11	uint	Internal ID of called party subscriber. Value is 0 if calling party is external.
DESTINATION_EXTERNAL_CONTRACT_ID	0-255	string	External, arbitrary ID of called party customer. (A string value shown as "External ID" property of an Sipwise C5 customer/peer.)
DESTINATION_ACCOUNT_ID	1-11	uint	Internal ID of called party customer.
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.

Copyright © Sipwise GmbH - All rights reserved

Body Element	Length	Type	Description
CALL_TYPE	3-4	string	The type of the call - one of: call: normal call cfu: call forward unconditional cfb: call forward busy cft: call forward timeout cfna: call forward not available cfs: call forward for SMS cfr: call forward on response cfo: call forward on overflow
CALL_STATUS	2-8	string	The final call status - one of: ok: successful call busy: called party busy noanswer: no answer from called party cancel: cancel from caller offline called party offline timeout: no reply from called party other: unspecified, see CALL_CODE field for details
CALL_CODE	3	string	The final SIP status code.
INIT_TIME	23	timestamp	Timestamp of call initiation (SIP 'INVITE' received from calling party). Includes date, time with milliseconds (3 decimals).
START_TIME	23	timestamp	Timestamp of call establishment (final SIP response received from called party). Includes date, time with milliseconds (3 decimals).
DURATION	4-13	fixed precision (3 decimals)	Length of call (calculated from START_TIME) including milliseconds (3 decimals).

Copyright © Sipwise GmbH - All rights reserved

Body Element	Length	Type	Description
END_TIME	23	timestamp	START_TIME plus DURATION.
INIT_TIME_TRUNCATED	19	timestamp	INIT_TIME without milliseconds.
START_TIME_TRUNCATED	19	timestamp	START_TIME without milliseconds.
END_TIME_TRUNCATED	19	timestamp	END_TIME without milliseconds.
TIMEZONE	100	string	The name of the local subscriber's active timezone, defined by the contact of the subscriber/contract/reseller. The caller's timezone is used for outgoing calls. For calls from external subscribers, the callee's timezone is used. System timezone (defined in config.yml) is used for transit calls.
INIT_TIME_LOCALIZED	23	timestamp	INIT_TIME, converted to TIMEZONE.
START_TIME_LOCALIZED	23	timestamp	START_TIME, converted to TIMEZONE.
END_TIME_LOCALIZED	23	timestamp	END_TIME, converted to TIMEZONE.
INIT_TIME_LOCALIZED_TRUNCATED	19	timestamp	INIT_TIME_LOCALIZED without milliseconds.
START_TIME_LOCALIZED_TRUNCATED	19	timestamp	START_TIME_LOCALIZED without milliseconds.
END_TIME_LOCALIZED_TRUNCATED	19	timestamp	END_TIME_LOCALIZED without milliseconds.
CALL_ID	0-255	string	The SIP Call-ID.
RATING_STATUS	2-7	string	The internal rating status of the CDR - one of: unrated: not rated ok: successfully rated failed: error while rating Currently always ok or unrated, depending on whether rating is enabled or not.
RATED_AT	0-19	datetime	Time of rating, including date and time (with seconds precision). Empty if CDR is not rated.

Copyright © Sipwise GmbH - All rights reserved

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

Copyright © Sipwise GmbH - All rights reserved

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

Copyright © Sipwise GmbH - All rights reserved

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

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

```
'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',
```


Copyright © Sipwise GmbH - All rights reserved

```
'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 `config.yml` file, `cdrexport.reseller_export_fields` parameter.

Multiple independent export configurations can be declared in `config.yml`, representing a separate CDR *stream* each.

```
cdrexport:
  my_stream1:
    admin_export_fields:
    ...
  other_stream2:
    admin_export_fields:
    ...
  ...
```

The streams will be processed *one after the other* by the exporter job.

Extra Fields That Can Be Exported to CDRs

Supplementary Data

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

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

Table 5. 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.

Copyright © Sipwise GmbH - All rights reserved

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

Account Balance Details (Prepaid Calls)

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

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

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

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

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

 Copyright © Sipwise GmbH - All rights reserved

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.

Transformation of Values

Each field allows to define a *perl* function to *transform* the values obtained from the database records. This can e.g. be used to manipulate numbers, as shown in the example below.

```
cdrexport:
  my_stream1:
    admin_export_fields:
    ...
    - name: header_Diversion
      enable: yes
      # Example: transform the subscriber number provided by the
Diversion header
      transformation: "sub {

  my ($value,$context) = @_;
  my ($stmt,$sth);

  my $number = $value;
  my $cdr_id = $context->{out}->[1];

  unless ($number) {
    unless ($context->{cdr}) {
      $stmt = 'select * from accounting.cdr where id = ?';
      $sth = $context->{dbh}->prepare($stmt);
      $sth->execute($cdr_id) or die 'failed to load cdr';
      $context->{cdr} = $sth->fetchrow_hashref();
      die 'failed to load cdr' unless $context->{cdr};
      $sth->finish();
    }
  }

  $number = $context->{cdr}->{source_cli} unless $number;
  $number = $context->{cdr}->{source_user} unless $number;

  if ($number =~ /^\\+?43/) {
    $number =~ s/^\\+?43/00/;
  } elsif ($number =~ /^[1-9]/) {
    $number = '0' . $number;
  }

  return $number;
```

Copyright © Sipwise GmbH - All rights reserved

```
}"
    ...
```

Distinguish Between On-net and Off-net Calls CDRs

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

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

File Body Format for Event Detail Records (EDR)

The body of an EDR consists of a minimum of zero and a maximum of 5000 lines. The platform operator can configure the maximum number of lines kept in a file by updating the `eventexport.max_rows_per_file` parameter in `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 `config.yml` file, `eventexport.admin_export_fields` parameter.

Table 6. 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: start_profile: A subscriber profile has been newly assigned to a subscriber. end_profile: A subscriber profile has been removed from a subscriber. update_profile: A subscriber profile has been changed for a subscriber. start_huntgroup: A subscriber has been provisioned as PBX / hunting group. end_huntgroup: A subscriber has been deprovisioned as PBX / hunting group. start_ivr: A subscriber has a new call-forward to Auto-Attendant. end_ivr: A subscriber has removed a call-forward to Auto-Attendant.

Copyright © Sipwise GmbH - All rights reserved

Body Element	Length	Type	Description
CONTRACT_EXTERNAL_ID	0-255	string	The external ID of the customer. (A string value shown as "External ID" property of an Sipwise C5 customer.)
COMPANY	0-127	string	The company name of the customer's contact.
SUBSCRIBER_EXTERNAL_ID	0-255	string	The external ID of the subscriber. (A string value shown as "External ID" property of an Sipwise C5 subscriber.) *PLEASE NOTE: This field is empty in case of* `start_huntgroup` *and* `end_huntgroup` *events.*
PRIMARY_NUMBER	0-64	string	The VoIP number of the subscriber with the highest ID (DID or primary number).
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.

Copyright © Sipwise GmbH - All rights reserved

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

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

```
"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 `config.yml` file, `eventexport.reseller_export_fields` parameter.

Extra Fields That Can Be Exported to EDRs

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

Copyright © Sipwise GmbH - All rights reserved

Table 7. Supplementary data in EDR fields

Body Element	Length	Type	Description
SUBSCRIBER_PROFILE_SET_NAME	0-255	string	The subscriber's profile set name.
PILOT_SUBSCRIBER_PROFILE_SET_NAME	0-255	string	The profile set name of the subscriber's pilot subscriber.
PILOT_SUBSCRIBER_PROFILE_NAME	0-255	string	The profile name of the subscriber's pilot subscriber.
FIRST_NON_PRIMARY_ALIAS_USERNAME_BEFORE	0-255	string	The subscriber's non-primary alias with lowest ID, before number updates during the operation.
FIRST_NON_PRIMARY_ALIAS_USERNAME_AFTER	0-255	string	The subscriber's non-primary alias with lowest ID, after number updates during the operation.
PILOT_FIRST_NON_PRIMARY_ALIAS_USERNAME_BEFORE	0-255	string	The non-primary alias with lowest ID of the subscriber's pilot subscriber, before number updates during the operation.
PILOT_FIRST_NON_PRIMARY_ALIAS_USERNAME_AFTER	0-255	string	The non-primary alias with lowest ID of the subscriber's pilot subscriber, after number updates during the operation.
NON_PRIMARY_ALIAS_USERNAME	0-255	string	The non-primary alias of a subscriber affected by an `update_profile`, `start_profile` or `end_profile` event to track number changes.
PRIMARY_ALIAS_USERNAME_BEFORE	0-255	string	The subscriber's primary alias, before number updates during the operation.
PRIMARY_ALIAS_USERNAME_AFTER	0-255	string	The subscriber's primary alias, after number updates during the operation.
PILOT_PRIMARY_ALIAS_USERNAME_BEFORE	0-255	string	The primary alias of the subscriber's pilot subscriber, before number updates during the operation.
PILOT_PRIMARY_ALIAS_USERNAME_AFTER	0-255	string	The primary alias of the subscriber's pilot subscriber, after number updates during the operation.
FIRST_NON_PRIMARY_ALIAS_USERNAME_BEFORE_AFTER	0-255	string	Equals FIRST_NON_PRIMARY_ALIAS_USERNAME_BEFORE, if the value is not NULL, otherwise it is the same as FIRST_NON_PRIMARY_ALIAS_USERNAME_AFTER.
PILOT_FIRST_NON_PRIMARY_ALIAS_USERNAME_BEFORE_AFTER	0-255	string	Equals PILOT_FIRST_NON_PRIMARY_ALIAS_USERNAME_BEFORE, if the value is not NULL, otherwise it is the same as PILOT_FIRST_NON_PRIMARY_ALIAS_USERNAME_AFTER.

Copyright © Sipwise GmbH - All rights reserved

File Trailer Format

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

```
<md5 sum>
```

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

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

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

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

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

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

6.4.4. File Transfer

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

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

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

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

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

Copyright © Sipwise GmbH - All rights reserved

6.5. Billing Data Replication to Remote Databases

Sipwise C5 PRO/Carrier provide the `ngcp-cdr-exporterd` system service that replicates CDR data to one or more remote Databases in *real-time*:

- PostgreSQL
- Microsoft SQL
- MySQL/MariaDB
- Oracle Database
- SQLite

It uses a 2PC (2-Phase-Commit) approach when transferring CDR data from the accounting.cdr table (source) to the target table (remote), known from wire transfer transactions. This ensures that records cannot get duplicated or lost during the transfer process, no matter if suspended, interrupted or restarted etc.

Each export configuration in the `cdr_exporterd` section in `config.yml` represents an independent CDR *stream*:

```
bulk_processor:
  cdr_exporterd:
    enable: yes
    streams:
      my_stream1:
        type: postgresql
        ...
      other_stream2:
        type: oracle
        ...
      new_stream3:
        type: mssql
        ...
      stream4:
        type: postgres
        enabled: false
        ...
      ...
```

Each stream is processed by the service *simultaneously*, to keep the latency minimal.

There is no distinction between reseller/system data; it supports more flexible filtering and transforming mechanisms instead.

6.5.1. Replication Pipeline

The processing follows a map-reduce scheme based on the *bulk-processor* ETL framework. It is a pipeline that builds and processes CDR object graphs. Each stage can be configured in detail:

1. fetch source CDR to export

Copyright © Sipwise GmbH - All rights reserved

2. populate the CDR object by fetching related data
3. expand (split) the CDR object
4. reduce (skip) CDR objects
5. map CDR object fields to output table columns

Stage 1: Fetch Source CDR

An optimized SQL query is generated that defines what records from the accounting.cdr table to consider for the replication. `export_conditions` represent a list of conjunction terms that are required to match in order to include a record. Each term is specified by:

- `column`: SQL table column name
- `operand`: comparison operator
- `compare_value`: rhs value

Tables of related records referenced in the conditions may require to specify an `export_join` which is made up of:

- `alias`: SQL join alias
- `key_column`: table 1 column name
- `reference_column`: table 2 column name

```
bulk_processor:
  cdr_exporterd:
    enable: yes
    streams:
      my_stream1:
        ...
        export_join_alias: cdrtable
        export_joins:
          - table: accounting.cdr
            alias: x
            key_column: cdrtable.id
            reference_column: x.id
        export_conditions:
          - column: x.update_time
            operator: >
            value: '2025-06-25 00:00:00'
        ...
```

Stage 2: Populate Relations of the CDR Object

The CDR's related data such as source/destination subscriber or contract records form a *tree* structure. Each relation is addressed by a path notation, which uses the `.` symbol to denote the tree depth, e.g. `cdr.source_user.provisioning_voip_subscriber`. Available relations to choose from are hard-wired, with data sources not limited to just related records from NGCP MariaDB tables:

- Redis records, i.e. subscriber location data

Copyright © Sipwise GmbH - All rights reserved

- REST-APIs (NGCP or remote services)
- Data from message queues (Kafka, Redis PubSub, ...)

A relation in the `relations` list will be loaded from the data source when enabled. It can be conditionally skipped or transformed if needed:

- `load`: *Javascript* or *perl* function to define whether to load the relation value (tree)
- `filter`: *Javascript* or *perl* function to filter collection values
- `transform`: *Javascript* or *perl* function to modify the relation value (tree)

All relations with data required in the output table have to be retrieved, but to minimize load and latency it is recommended to disable those that are not needed. Below example gives an overview of relations available for the CDR object scheme:

```
bulk_processor:
  cdr_exporterd:
    enable: yes
    streams:
      my_stream1:
        ...
        relations:
          cdr.source_user:
            load: !!perl/code |
              {
                my ($cdr) = @_;
                info("this function should return 1 to load the cdr.source_user graph");
                #return 0 if $cdr->{source_user_id} eq '0';
                return 1;
              }
          cdr.source_user.domain: 1
          cdr.source_user.primary_number: 1
          cdr.source_user.provisioning_voip_subscriber: 1
          cdr.source_user.provisioning_voip_subscriber.voip_dbaliases: 1
          cdr.source_user.provisioning_voip_subscriber.voip_usr_preferences: 1
          cdr.source_user.provisioning_voip_subscriber.voip_usr_preferences.attribute: 1
          cdr.source_user.provisioning_voip_subscriber.voip_usr_preferences.allowed_ips: 1
          cdr.source_user.provisioning_voip_subscriber.voip_usr_preferences.ncos: 1
          cdr.source_user.provisioning_voip_subscriber.voip_usr_preferences.cf_mapping: 1
          cdr.source_user.provisioning_voip_subscriber.voip_usr_preferences.cf_mapping.destinations:
1
          cdr.source_user.provisioning_voip_subscriber.voicemail_users: 1
          #cdr.source_user.provisioning_voip_subscriber.voicemail_users.voicemail_spool: 1
          cdr.source_user.provisioning_voip_subscriber.voip_fax_preferences: 1
          cdr.source_user.provisioning_voip_subscriber.voip_fax_destinations:
            transform: !!perl/code |
              {
                my ($fax_destinations,$prov_subscriber) = @_;
                info("this function converts each fax_destination record to a string");
                return [
                  map { $_->{destination} . ' (' . $_->{filetype} . ')'; } @$fax_destinations
                ];
              }
          cdr.source_user.provisioning_voip_subscriber.profile: 1
          cdr.source_user.provisioning_voip_subscriber.voip_domain: 1
          cdr.source_user.provisioning_voip_subscriber.registrations: 1
          cdr.source_user.provisioning_voip_subscriber.registrations._entries: 1
          cdr.source_provider: 1
          cdr.source_provider.contact: 1
          cdr.source_provider.voip_peer_groups: 1
          cdr.source_provider.resellers: 1
          cdr.source_account: 1
```

Copyright © Sipwise GmbH - All rights reserved

```
          cdr.source_account.product: 1
          cdr.source_account.voip_subscribers:
            filter: |
              function(billSubscriber, sourceAccount) {
                if (billSubscriber.status == 'terminated') {
                  debug("skipping terminated source_account subscriber " + billSubscriber.username);
                  return 0;
                }
                return 1;
              }
          cdr.source_provider.contact: 1
          #cdr.source_provider.voip_peer_groups: 0
          #cdr.source_provider.resellers: 0
          cdr.destination_user: 1
          cdr.destination_provider: 1
          cdr.destination_account: 1
          cdr.source_carrier_billing_fee: 1
          cdr.source_reseller_billing_fee: 1
          cdr.source_customer_billing_fee: 1
          cdr.source_carrier_billing_zone: 1
          cdr.source_reseller_billing_zone: 1
          cdr.source_customer_billing_zone: 1
          cdr.destination_carrier_billing_fee: 1
          cdr.destination_reseller_billing_fee: 1
          cdr.destination_customer_billing_fee: 1
          cdr.destination_carrier_billing_zone: 1
          cdr.destination_reseller_billing_zone: 1
          cdr.destination_customer_billing_zone: 1
          cdr.cdr_groups: 1
          cdr.cdr_groups.cdrs: 1
          cdr.cdr_groups.cdrs.source_user: 1
          cdr.cdr_tags: 1
          cdr.cdr_tags.tag: 1
          cdr.cdr_tags.provider: 1
          cdr.cdr_tags.direction: 1
          cdr.cdr_mos: 1
          cdr.cdr_status: 1
          cdr.cdr_status.status: 1
          cdr.cdr_relations: 1
          cdr.cdr_relations.relation: 1
          cdr.cdr_relations.provider: 1
          cdr.cdr_relations.direction: 1
          cdr.cdr_presentity: 1
          cdr.cdr_cash_balances: 1
          cdr.cdr_cash_balances.contract_balance: 1
          cdr.cdr_cash_balances.provider: 1
          cdr.cdr_cash_balances.direction: 1
          cdr.cdr_time_balances.contract_balance: 1
          cdr.cdr_time_balances.provider: 1
          cdr.cdr_time_balances.direction: 1
          ...
        ...
```

Stage 3: Expand/Split the CDR Object

Complex exporter logic requirements can involve modifying the number of exported records, which can be implemented with the expand/reduce stages. e.g. when exporting CDRs of special call scenarios, it can be a desired to produce 2 or more records from the originating CDR. This can be done with a function provided by the `expand_cdrs` element.

```
bulk_processor:
  cdr_exporterd:
    enable: yes
```


Copyright © Sipwise GmbH - All rights reserved

```
    streams:
      my_stream1:
        ...
        expand_cdrs: !!perl/code |
          sub {
            my ($arg) = @_;
            if (stash_get('expandcdrsDone')) { return; } else { stash_set('expandcdrsDone',1); }
            my $test = { x => 'x', y => 'y' };
            stash_set('test',$test);
            if (to_json($test,0,1) eq to_json(stash_get('test'),0,1)) {
              info("stash test ok");
            } else {
              warn("stash test failed");
            }
            info("this is a perl snippet (coderef from yaml)");
            my ($x,$y) = (2, 3);
            info($x . " + " . $y . " = " . ($x + $y));
            info("symbol table: " . join(',', get_env()));
            info("we are processing cdr id " . $cdr->{id});
            info("dump cdr variable: " . to_json($cdr,1));
            info("dump arg variable: " . to_json($arg));
            #info("dump cdrs variable: " . to_json($cdrs));
            #info("dump stream variable: " . to_json($stream));
            info("some db query result - cdr count = " .
              $db->db_get_value("select count(1) from accounting.cdr"));
            return ($cdr,$cdr); #export every cdr twice ..
          }
        ...
```

The example above demonstrates general features of the scripting environment, including:

- stash for storing variables across iterations (`stash_set`, `stash_get`)
- logging (`info`, `warn`, …)
- serialisation (`to_json`, …)
- database queries (`db_get_value`, …)
- context argument and variables (`$cdr`/`cdr`)

The same `expand_cdrs` function example using *Javascript* language:

```
bulk_processor:
  cdr_exporterd:
    enable: yes
    streams:
      my_stream1:
        ...
        expand_cdrs: |
          function(arg) {
            if (stash_get('expandcdrsDone')) { return; } else { stash_set('expandcdrsDone',true); }
            var test = { x: 'x', y: 'y' };
            stash_set('test',test);
            if (to_json(test,0,1) == to_json(stash_get('test'),0,1)) {
              info("stash test ok");
            } else {
              warn("stash test failed");
            }
            info("this is a javascript snippet (string)");
            var x = 2, y = 3;
            info(x + " + " + y + " = " + (x + y));
            info("symbol table: " + get_env());
```

Copyright © Sipwise GmbH - All rights reserved

```
            info("we are processing cdr id " + cdr.id);
            info("dump cdr variable: " + to_json(cdr,1));
            info("dump arg variable: " + to_json(arg));
            //info("dump cdrs variable: " + to_json(cdrs));
            //info("dump stream variable: " + to_json(stream));
            info("select count(1) from accounting.cdr: " +
              db_get_value("select count(1) from accounting.cdr"));
            return [ cdr, cdr ]; // e.g. export every cdr twice ..
          }
        ...
```

If omitted, the CDR object will be passed as-is by default, which is equivalent to the `expand_cdrs` function below:

```
bulk_processor:
  cdr_exporterd:
    enable: yes
    streams:
      my_stream1:
        ...
        expand_cdrs: |
          function(arg) {
            return cdr;
          }
        ...
```

Stage 4: Reduce/Skip CDR Objects

The reduce stage allows to skip creating records. This can be required to e.g. collapse CDRs from call legs or correlated CDRs (`cdr_group`) of special call scenarios. To skip exporting a CDR safely across batches and threads, the export_status needs to be set to a value other than `unexported`. This can be done with a function provided by the `cdrs_export_status` element. The example below is notional, it will skip every other pair of CDRs:

```
bulk_processor:
  cdr_exporterd:
    enable: yes
    streams:
      my_stream1:
        ...
        cdrs_export_status: |
          function(record) {
            return [[ record.id, "ok" ], [ record.id+2, "ok" ]];
          }
        ...
```

If omitted, the status of exported CDRs will be set to `ok` by default.

Copyright © Sipwise GmbH - All rights reserved

Stage 5: Map CDR Object to Output

The final stage defines which CDR tree elements to map to the output table. This projection can be configured with the `table_fields` list. Each element defines mapping parameters:

- `path`: path notation of the CDR tree element. An element of a collection value is denoted by the element index in `[]` brackets, e.g. `provisioning_voip_subscriber.voip_usr_preferences.cfu[0].cf_mapping.destinations[99].destination`
- `colname`: column name in the output table of the remote database
- `transform`: *Javascript* or *perl* function to convert the value (tree)
- `sep`: convert a collection value to string using the given separator
- `field`: convert objects in the collection value to string using the given object field

If the output table in the remote database is not present yet, it will be created automatically with columns of SQL `TEXT` data type. Below `table_fields` list presumes the relations loaded for the CDR object from the stage 2 example.

```
bulk_processor:
  cdr_exporterd:
    enable: yes
    streams:
      my_stream1:
        ...
        table_fields:
          - path: rating_status
            colname: column1
            transform: !!perl/code |
              sub {
                my ($v) = @_;
                info("count " . $db->db_get_value("select count(1) from accounting.cdr"));
                #info($context,"test");
                use Data::Dumper;
                #info($context,Dumper($context));
                #info($context,join(",",keys %main::));
                return uc($v); # convert to uppercase
              }
          - path: update_time
            colname: column2
          - path: id
            colname: column3
          - path: contract.id
            transform: !!perl/code |
              {
                my ($id,$bill_subs) = @_;
                return $id; # pass through ...
              }
          - path: primary_number.cc
          - path: primary_number.ac
          - path: primary_number.sn
          - path: provisioning_voip_subscriber.voicemail_users[0].attach
          - path: provisioning_voip_subscriber.voicemail_users[0].delete
          - path: provisioning_voip_subscriber.voicemail_users[0].email
          - path: provisioning_voip_subscriber.voicemail_users[0].password
          - path: provisioning_voip_subscriber.voip_usr_preferences.allowed_clis
            sep: ','
            field: 'value'
          - path: provisioning_voip_subscriber.voip_usr_preferences.allowed_ips_grp[0].allowed_ips
            sep: ','
            field: 'ipnet'
```

Copyright © Sipwise GmbH - All rights reserved

```
          - path: provisioning_voip_subscriber.voip_usr_preferences.block_out_list
            sep: ','
            field: 'value'
          - path: provisioning_voip_subscriber.voip_usr_preferences.block_out_mode[0].value
          - path: provisioning_voip_subscriber.voip_usr_preferences.block_in_list
            sep: ','
            field: 'value'
          - path: provisioning_voip_subscriber.voip_usr_preferences.block_in_mode[0].value
          - path: provisioning_voip_subscriber.voip_usr_preferences.adm_block_in_list
            sep: ','
            field: 'value'
          - path: provisioning_voip_subscriber.voip_usr_preferences.adm_block_in_mode[0].value
          - path: provisioning_voip_subscriber.voip_usr_preferences.adm_block_out_list
            sep: ','
            field: 'value'
          - path: provisioning_voip_subscriber.voip_usr_preferences.adm_block_out_mode[0].value
          - path: provisioning_voip_subscriber.voip_usr_preferences.ncos_id[0].ncos.level
          - path: provisioning_voip_subscriber.voip_usr_preferences.adm_ncos_id[0].ncos.level
          - path: provisioning_voip_subscriber.voip_usr_preferences.cfb[0].cf_mapping.destinations
            sep: ','
            field: 'destination'
          - path: provisioning_voip_subscriber.voip_usr_preferences.cfna[0].cf_mapping.destinations
            sep: ','
            field: 'destination'
          - path: provisioning_voip_subscriber.voip_usr_preferences.cfo[0].cf_mapping.destinations
            sep: ','
            field: 'destination'
          - path: provisioning_voip_subscriber.voip_usr_preferences.cfr[0].cf_mapping.destinations
            sep: ','
            field: 'destination'
          - path: provisioning_voip_subscriber.voip_usr_preferences.cfs[0].cf_mapping.destinations
            sep: ','
            field: 'destination'
          - path: provisioning_voip_subscriber.voip_usr_preferences.cft[0].cf_mapping.destinations
            sep: ','
            field: 'destination'
          - path: provisioning_voip_subscriber.voip_usr_preferences.cfu[0].cf_mapping.destinations
            sep: ','
            field: 'destination'
          - path: provisioning_voip_subscriber.voip_fax_preferences.active
          - path: provisioning_voip_subscriber.voip_fax_preferences.ecm
          - path: provisioning_voip_subscriber.voip_fax_preferences.name
          - path: provisioning_voip_subscriber.voip_fax_preferences.t38
          - path: provisioning_voip_subscriber.voip_fax_destinations
            sep: ','
          - path:
provisioning_voip_subscriber.voip_usr_preferences.force_inbound_calls_to_peer[0].value
          - path: provisioning_voip_subscriber.voip_usr_preferences.lnp_for_local_sub[0].value
          ...
        ...
```


Copyright © Sipwise GmbH - All rights reserved

Chapter 7. Features

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

7.1. Managing System Administrators

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

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

There is a single administrator (username: "administrator"), whose account is enabled by default and who belongs to the *default reseller*. This user is the *superuser* of Sipwise C5 administrative web interface (the so-called "admin panel"), and he has the right to modify administrators of other *Resellers* as well.

7.1.1. 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*.

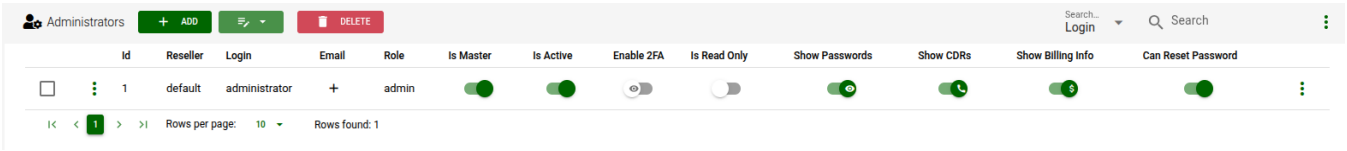

Figure 40. List of System Administrators

You have 2 options:

- If you'd like to **create** a new administrator user press *ADD* button.
- If you'd like to **edit** an existing administrator user press the three dots beside an existing administrator and then press the *Edit* button.

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

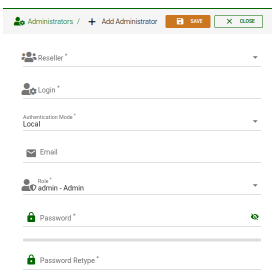

Figure 41. Generic System Administrator Attributes

Copyright © Sipwise GmbH - All rights reserved

- *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 authenticating REST API requests
- *Authentication Mode*: currently available options are:

  local: Authenticate by matching login name and password stored in the local NGCP database

  LDAP: Authenticate login name (converted to a cn) and provided password against a (remote) LDAP service
- *Email*: the email of the administrator user, used for resetting password.
- *Role*: role associated to the administrator user. The six available access roles are summarized in the next section of the handbook

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

7.1.2. Administrator's Roles

Six different types of roles are available for administrators. The table below describes each of them and provides a correspondence with the old 'role flags'.

Table 8. System Administrator's Roles

Role	Old Access Right	Description
System	`Is system`	The user is allowed to create, delete or modify Lawful Intercept Admins which are otherwise invisible to other types of administrators.
Admin	`Is superuser`	The user is allowed to modify data on Reseller level and, if mixed with the "is_master" access right it is also able to modify administrators of other resellers. There should be only 1 user on Sipwise C5 with the combination of "admin" role and "is_master" privilege.
Reseller		The user is allowed to create, delete or modify Customers, Subscribers and any other element in the system only within a Reseller the user belongs to. It cannot create other administrator unless it has "is_master" access right.
Customer Care Admin	`Is ccare, Is superuser`	The user is allowed to create, delete or modify Customers and Subscribers across all Resellers. The user can access relevant information required to create or modify Customers or Subscribers, such as Domains, Billing Profiles, Email Templates, but the user cannot access the entries (e.g.: see the detailed info about a Billing Profile), nor modify them.

Copyright © Sipwise GmbH - All rights reserved

Role	Old Access Right	Description
Customer Care	`Is ccare`	The user is allowed to create, delete or modify Customers and Subscribers only within a Reseller the user belongs to. The user can access to relevant information required to create or modify Customers or Subscribers, such as Domains, Billing Profiles, Email Templates, but the user cannot access the entries (e.g.: see the detailed info about a Billing Profile), nor modify them.
Lawful Intercept	`Lawful intercept`	Visible only for System administrators. It is the only role with access to the interceptions REST API (that is: `/api/interceptions`). Administrators with this flag do not have access to anything but the administrators in UI and API and `/api/interceptions` in the API. NOTE: This means that besides enabling LI in `config.yml` configuration file one also needs to enable the API via the LI privilege of an administrator user, so that Sipwise C5 can really provide LI service.

7.1.3. Access Rights of Administrators

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

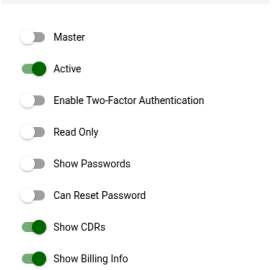

Figure 42. Access Rights of System Administrators

Table 9. Access Rights of System Administrators

Label in admin list	Access Right	Description
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 log in on the web panel or authenticate himself on REST API; otherwise user authentication will fail.
Enable Two-Factor Authentication	`Enable 2fa`	It enables mandatory 2FA for the administrator user.

Copyright © Sipwise GmbH - All rights reserved

Label in admin list	Access Right	Description
Read Only	`Read only`	The user will only be able to list various data but is not allowed to modify anything. * For the **web interface** this means that *Create…* and *Edit* buttons will be hidden or disabled. * For the **REST API** this means that only `GET, HEAD, OPTIONS` HTTP request methods are accepted, and Sipwise C5 will reject those targeting data modification: PUT, PATCH, POST, DELETE.
Show Passwords	`Show passwords`	The user sees subscriber passwords (in plain text) on the web interface. NOTE: Admin panel user passwords and subscriber web passwords are stored in an unreadable way (cryptographic hash digest) in the database, while subscriber SIP passwords are stored in plain text. The latter happens on purpose, e.g. to make subscriber data migration possible.
Can Reset Password	`Can Reset Password`	The user is allowed to request a password reset.
Show CDRs	`Call data`	This privilege has an effect on 2 items that will be displayed on admin panel of NGCP, when *Subscriber Details* is selected: 1. PBX *Groups* list, 2. *Captured Dialogs* list
Show Billing Info	`Billing data`	Some REST API resources that are related to billing are disabled: HTTP requests on `/api/vouchers`, `/api/topupcash` and `/api/topupvoucher` resources are rejected.

7.2. 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 (Network Class of Service) Levels*, where the admin predefines rules for outbound calls, which are grouped in certain levels. The subscriber can then choose the level, or the admin can restrict a subscriber to a certain level. Also Sipwise C5 offers some options to restrict the IP addresses that subscriber is allowed to use the service from. The following sections describe these features in detail.

7.2.1. 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.

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

Copyright © Sipwise GmbH - All rights reserved

Block Modes

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

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

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

Block Lists

The list contents are controlled by the User Preferences *block_in_list*, *block_out_list* and their administrative counterparts.

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 below 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.

In the call forward scenario both the original caller number as well as the forwarder number are checked against the *block_in_list*.

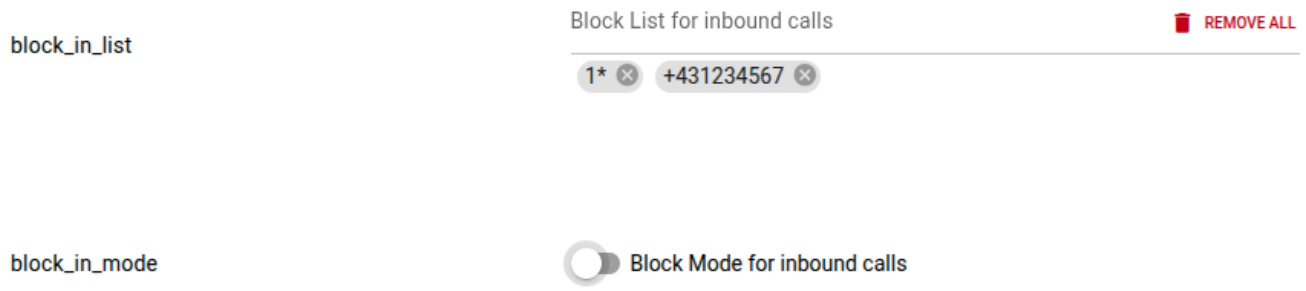

Figure 43. Outgoing Block List

Block Anonymous Numbers

For incoming call, the User Preference *block_in_clir* and *adm_block_in_clir* controls whether or not to reject incoming calls with number suppression (either "[Aa]nonymous" in the display- or user-part of the From-URI or a header *Privacy: id* is set). This flag is independent from the Block Mode.

7.2.2. NCOS (Network Class of Service) Levels

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

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

Copyright © Sipwise GmbH - All rights reserved

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

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

NCOS levels can also have a Timeset associated which will enable the NCOS level check only within the time window defined. By default, in case the day time is outside the defined time window, the NCOS level check is skipped. Use the flag: "Timeset Invert" to match the timeset in the opposite way: skip the NCOS level check in case the day time is inside the defined time window.

The *Expose to Customer* flag allows the customer's administrator subscribers to see and set the NCOS level in the CSC portal. This allows them to autonomously configure the NCOS level for all the subscribers belonging to the same customer.

ATTENTION: when the *Expose to Customer* flag of an NCOS level is switched from *set* to *unset*, all assigned NCOS preferences for customers and subscribers that contain the NCOS Level are removed. That means that those customers and subscribers will not have any NCOS level assigned other than the administrative ones (adm_ncos and adm_cf_ncos).

Creating NCOS Levels

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

Select a reseller, enter a level name, select the mode, add a description:

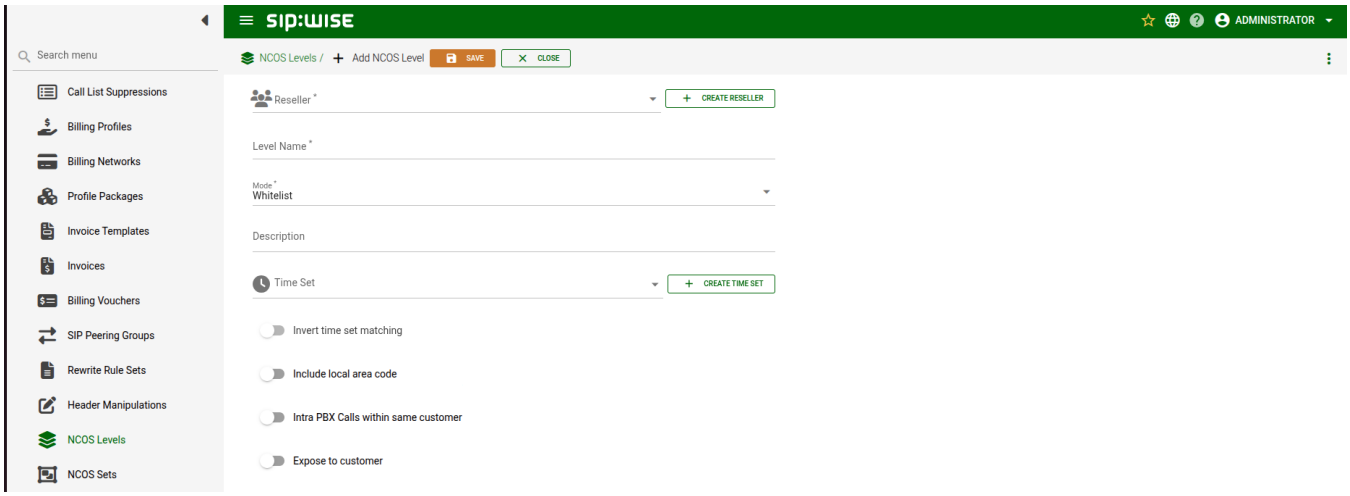

Figure 44. Create NCOS Levels

If necessary, assign a timeset then click the *Save* button.

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

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

 Copyright © Sipwise GmbH - All rights reserved

caller himself.

Creating Rules per NCOS Level

To define the rules within the newly created NCOS Level, click on the three dots beside its table row and select *Details*.

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

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

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

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

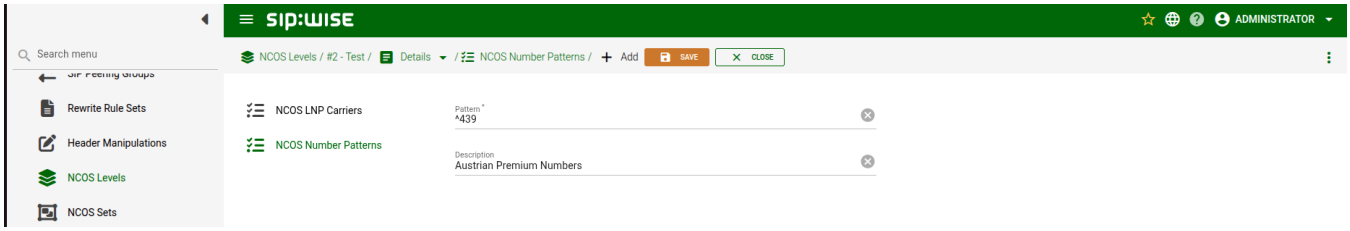

Figure 45. Create NCOS Number Pattern

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.

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

An example of *NCOS LNP Carrier* definition:

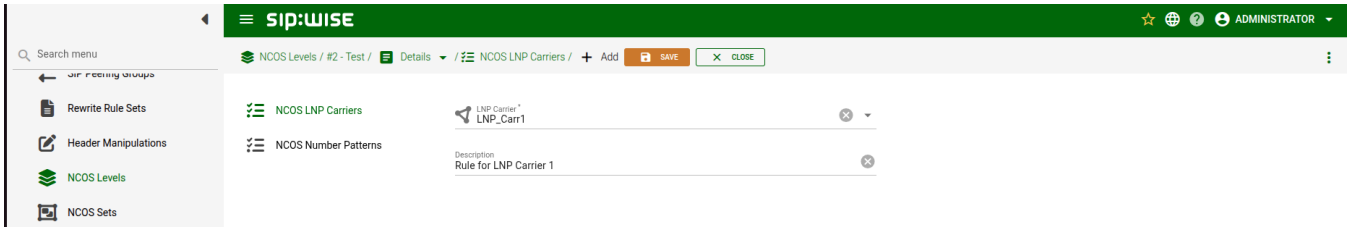

Figure 46. Create NCOS LNP Carrier

In the above example we created a rule that blocks calls to "LNP_Carr1" carrier, supposing we use

Copyright © Sipwise GmbH - All rights reserved

blacklist mode of the NCOS Level.

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.

You can create any combination of ***NCOS Number Patterns*** and ***NCOS LNP Carriers*** inside a NCOS Level. The below table gives an overview about the system behaviour in all the possible cases:

Table 10. NCOS combinations

TYPE	NCOS	NCOS_LNP	RESULT
Whitelist	empty table	empty table	Blocked
Whitelist	empty table	no match	Blocked
Whitelist	empty table	match	Allowed
Whitelist	no match	empty table	Blocked
Whitelist	no match	no match	Blocked
Whitelist	no match	match	Blocked
Whitelist	match	empty table	Allowed
Whitelist	match	no match	Blocked
Whitelist	match	match	Allowed
Blacklist	empty table	empty table	Allowed
Blacklist	empty table	no match	Allowed
Blacklist	empty table	match	Blocked
Blacklist	no match	empty table	Allowed
Blacklist	no match	no match	Allowed
Blacklist	no match	match	Blocked
Blacklist	match	empty table	Blocked
Blacklist	match	no match	Blocked
Blacklist	match	match	Blocked

The parameter kamailio.proxy.lnp.strictly_check_ncos contained in '/etc/ngcp-config/config.yml' specify whether the NCOS LNP should be evaluated even if the LNP lookup was not previously executed (because not required by the inbound/outbound call) or if it did not return any occurrence. If set to *yes*, a whitelist NCOS will fail if the LNP lookup does not return any match. The parameter has no impact on blacklist NCOS.

Assigning NCOS Levels to Subscribers/Domains

Once you have defined your NCOS Levels, you can assign them to local subscribers. To do so, navigate to *SettingsSubscribers*, search for the subscriber you want to edit, click on the three dots and press the *Preferences* button. There, you can assign the ncos level either the *ncos* or *adm_ncos* setting in the *Call*

 Copyright © Sipwise GmbH - All rights reserved

Blockings section.

You can assign the NCOS level to all subscribers within a particular domain. To do so, navigate to *SettingsDomains*, select the domain you want to edit, click on the three dots and press the *Preferences* button. There, you can assign the ncos level either *ncos* or *admin_ncos* in the *Call Blockings* section.

Note: if both domain and subscriber have same NCOS preference set (either *ncos* or *adm_ncos*, or both) the subscriber's preference is used. This is done so that you can override the domain-global setting on the subscriber level.

Assigning NCOS Level for Forwarded Calls to Subscribers/Domains

In some countries there are regulatory requirements that prohibit subscribers from forwarding their numbers to special numbers like emergency, police etc. While Sipwise C5 does not deny provisioning Call Forward to these numbers, the administrator can prevent the incoming calls from being actually forwarded to numbers defined in the NCOS list: select the appropriate NCOS level in the domain's or subscriber's preference ***adm_cf_ncos***. This NCOS will apply only to the Call Forward from the subscribers and not to the normal outgoing calls from them.

NCOS Exceptions

Using the subscriber's preference ***adm_ncos_exceptions*** an administrator can configure exceptions per subscriber to assigned NCOS Levels. In case a called number matches one of the *adm_ncos_exceptions* preference entries, then all the NCOS Level checks (administrative or not) are skipped and the call will be accepted.

7.2.3. NCOS Sets

NCOS Sets provides the possibility to create a Set of *NCOS Levels*. Multiple *NCOS Levels*, with different rules and modes, can be grouped together in a *NCOS Set* which can be assigned to a local subscriber or a local domain. This is suitable in case you need to group different levels with different modes (*whitelist* and *blacklist*) and build more efficient NCOS rules. In order for a call to pass the *NCOS Sets* check, it must fulfill all the *NCOS Levels* that belong to the Set, otherwise, the call will be rejected.

Creating NCOS Set

To create an NCOS Set, go to SettingsNCOS Sets and press the Add button.

Mandatory parameters are reseller, name and description.

The *Expose to Customer* flag allows the customer's administrator subscribers to see and set the NCOS Set in the CSC portal. This allows them to autonomously configure the NCOS Set for all the subscribers belonging to the same customer.

ATTENTION: when the *Expose to Customer* flag of an NCOS Set is switched from set to unset, all assigned NCOS preferences for customers and subscribers that contain the NCOS Set are removed. That means that those customers and subscribers will not have any NCOS Set assigned other than the administrative ones (adm_ncos_set and adm_cf_ncos_set).

Copyright © Sipwise GmbH - All rights reserved

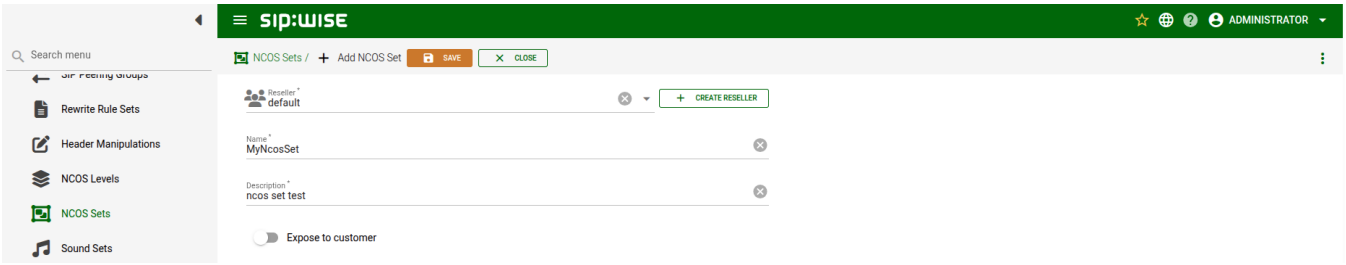

Figure 47. Creating NCOS Set

Add Levels to NCOS Set

To add NCOS levels to the NCOS Set, click on the three dots and then to *Levels*

There are two sections in the *Levels* page, the first section shows the *Selected NCOS Levels* and the second one the list of all available NCOS Levels.

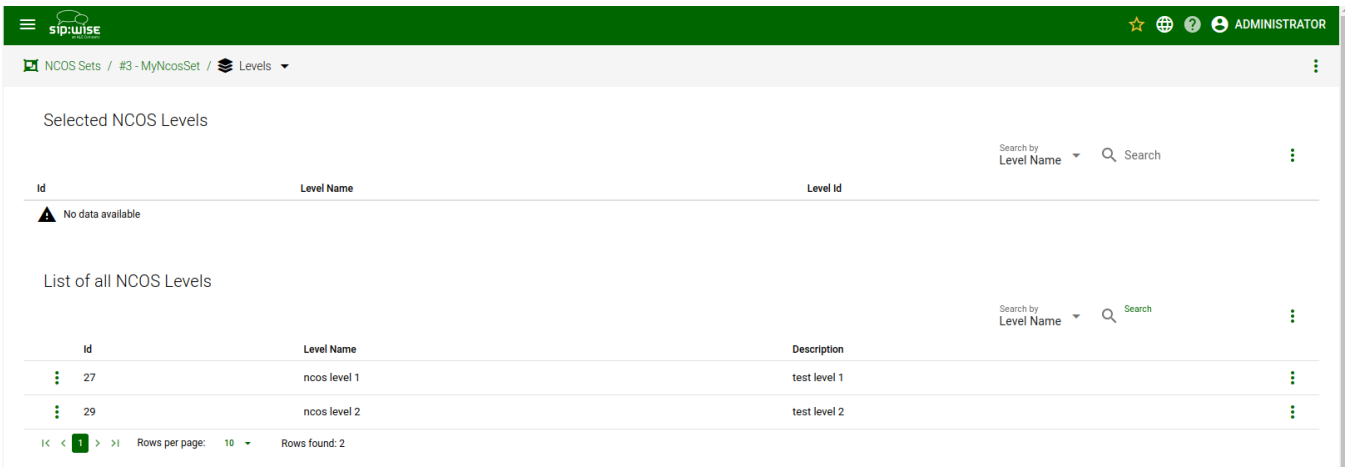

Figure 48. Add Levels to NCOS Set

In order to add a Level to the set, click on the three dots of a level and select *Add*. The selected level will be added to the set, but it will not be removed from the available list, because the same level can be assigned to different sets.

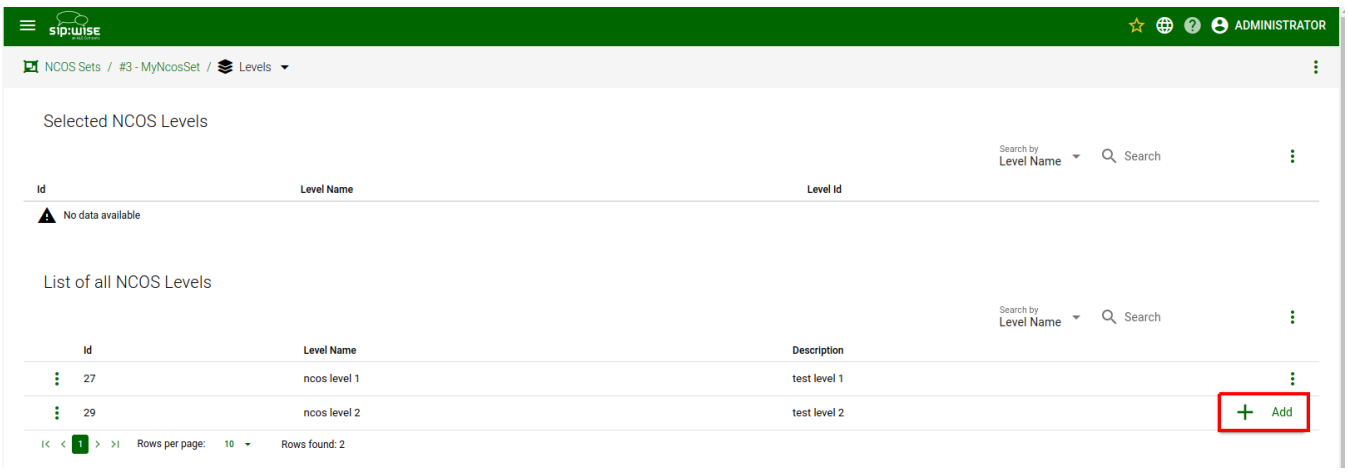

Figure 49. Add Levels to NCOS Set

Copyright © Sipwise GmbH - All rights reserved

Assigning NCOS Sets to Subscribers/Domains

Once you have defined your NCOS Sets, you can assign them to local subscribers or domains. To do so, navigate to *SettingsSubscribers*, search for the subscriber you want to edit, press the three dots and then the *Preferences* button. There, you can assign the ncos set either to *ncos_set* or to *adm_ncos_set* setting in the *Call Blockings* section.

Assigning NCOS Set for Forwarded Calls to Subscribers/Domains

In some countries there are regulatory requirements that prohibit subscribers from forwarding their numbers to special numbers like emergency, police etc. While Sipwise C5 does not deny provisioning Call Forward to these numbers, the administrator can prevent the incoming calls from being actually forwarded to numbers defined in the NCOS list: select the appropriate NCOS set in the domain's or subscriber's preference *adm_cf_ncos_set*. This NCOS will apply only to the Call Forward from the subscribers and not to the normal outgoing calls from them.

NCOS Set Exceptions

The subscriber's preference "adm_ncos_exceptions", mentioned before, is valid also for the NCOS Sets. In case a called number matches one of the 'adm_ncos_exceptions' preference entries, then all the NCOS Set checks (administrative or not) are skipped and the call will be accepted.

7.2.4. IP Address Restriction

The Sipwise C5 provides subscriber and domain preference *allowed_ips* to restrict the IP addresses that a particular subscriber or any subscribers within the respective domain is allowed to use the service from. If the REGISTER or INVITE request comes from an IP address that is not in the allowed list, Sipwise C5 will reject it with a 403 message. Also a voice message can be played when the call attempt is rejected (if configured).

By default, *allowed_ips* is an empty list which means that subscriber is not restricted. If you want to configure a restriction, navigate to *SettingsSubscribersPreferences* or *SettingsDomainsPreferences*, and search for the *allowed_ips* preference in the *Access Restrictions* section. You can enter multiple allowed IP addresses or IP address ranges one after another.

7.2.5. CLI-Based Access Control

The Sipwise C5 provides subscriber preference *upn_block_list* to restrict the CLI that subscriber is allowed to use the service from. If the INVITE request comes with a CLI that is not in the allowed list, Sipwise C5 will reject it with a 403 message. Also a voice message can be played when the call attempt is rejected (if configured).

The restriction is applied to User-Provided Number (UPN) which is obtained from the configurable source based on the setting of *inbound_upn* preference in the *Number Manipulation* section in the Domain and/or User preferences, after it has been rewritten with Inbound Rewrite Rules for Caller.

In case the *inbound_upn* preference is set to the "From Display-Name" the UPN value can be alphanumeric so the access control supports the alphanumeric (caller name) matching as well. If the incoming message does not have the Display-Name, though, the UPN value will be taken from the From-Username.

The *inbound_upn* preference has a slightly different meaning if *kamailio.proxy.multiple_headers_for_valid_upn* is set to yes inside config.yml.

Copyright © Sipwise GmbH - All rights reserved

In this particular case the *inbound_upn* preference only defines which Sip Header must be checked first and, in case of failure, the check is repeated evaluating the username part of the following headers:

- P-Preferred-Identity
- P-Asserted-Identity
- From

In case of positive result, that value will be used as valid UPN and the subsequent headers will be not evaluated, otherwise if all additional checks fail, restrictions are applied as described above.

By default, *upn_block_list* is an empty list which means that subscriber is not restricted. If you want to configure a restriction, navigate to *SettingsSubscribers*, search for the subscriber you want to edit, press *Details* and then *Preferences* and press *Edit* for the *upn_block_list* preference in the *Call Blockings* section to define the list entries.

In block list entries, you can provide shell patterns like * and `[]`. The CLI-based block list can either be *whitelist* or *blacklist*.

- The *blacklist* mode indicates to **allow everything except the entries in this list**. This is the default mode of operation and is effective when the preference *upn_block_mode* is unset.
- The *whitelist* mode indicates to **reject anything except the entries in this list**. In order to switch to this mode, set the preference *upn_block_mode* (it is a toggle between whitelist/blacklist).

If separate preference *upn_block_clir* is enabled, outgoing anonymous calls from this user will be dropped.

If the caller's UPN is allowed it is also checked according to *allowed_clis* preference as usual and can be rewritten according to *allowed_clis_reject_policy* for correct calling number presentation on outgoing calls. This step happens after Access Control.

7.2.6. Call Limit Control

There is a set of preferences that limits calls to and from subscribers. The option *concurrent_max_total* defines the maximum number of concurrent calls (incoming and outgoing) for a subscriber, while the option *concurrent_max_out_total* limits only subscriber's outbound concurrent calls and the option *concurrent_max_in_total* only subscriber's inbound concurrent calls.

Preferences *concurrent_max*, *concurrent_max_out*, and *concurrent_max_in* have the same effect, excluding calls to voicemail, application server and intra-PBX calls.

It is also possible to limit the number of concurrent calls of a subscriber compared to the number of calls made or received by all subscribers within the same customer (account). The options *concurrent_max_per_account*, *concurrent_max_out_per_account*, *concurrent_max_in_per_account* permit to apply this limit. To better understand how they work, suppose we have two subscribers A and B, owned by the same customer. If we set *concurrent_max_per_account*=2 on B preferences and A is placing two calls, then B cannot receive or place new calls at the same time. For instance, an administrator may define this restriction to some non-manager subscribers, in which *concurrent_max_per_account*=2. Hence, they will be able to make a maximum of two calls if there are no other calls in place within the customer. On manager subscribers, the limit can be defined differently, or even not set at all. In the last case, calls will be always allowed.

When *concurrent_max_total* limit is reached, announcement set on *max_calls_in* is played to those who

 Copyright © Sipwise GmbH - All rights reserved

try to call that subscriber. The same announcement is played for *concurrent_max*, *concurrent_max_per_account*, *concurrent_max_in_total*, *concurrent_max_in*, *concurrent_max_in_per_account*. When *concurrent_max_out_total* limit is reached, announcement set on *max_calls_out* is played. The same announcement is played for *concurrent_max_out* or *concurrent_max_out_per_account*.

Options *concurrent_max*, *concurrent_max_out* and *concurrent_max_in* are configurable on peers as well.

Furthermore, options *concurrent_max*, *concurrent_max_out*, *concurrent_max_in* and their *_total* version (*concurrent_max_total* and so on), are configurable on customer and reseller as well. In this case the limits are applied considering the number of calls of all the subscribers belonging to that account/reseller.

7.3. Call Forwarding and Call Hunting

The Sipwise C5 provides the capabilities for normal *Call Forwarding* (deflecting a call for a local subscriber to another party immediately or based on other event) and *Serial Call Hunting* (sequentially executing a group of deflection targets until one of them succeeds). Targets can be stacked, which means if a target is also a local subscriber, it can have another Call Forward or hunt group which is executed accordingly.

7.3.1. Call Forward Types

Currently 7 different types of Call Forward are available in Sipwise C5:

- **Call Forward Unconditional (CFU)**: The Call Forward is always executed, completely disregarding the subscriber state.
- **Call Forward Busy (CFB)**: The Call Forward is executed when the subscriber returns a busy state.
- **Call Forward Timeout (CFT)**: The Call Forward is executed when no answer is received from the subscriber before the timeout expiration. Timeout is configurable in *ringtimeout* subscriber preference.
- **Call Forward Unavailable (CFNA)**: The Call Forward is executed when the subscriber has no endpoint registered.

  CAUTION The Call Forward Unavailable is also executed if the callee responds with *480 Temporarily Unavailable*, which may be the case when a subscriber's endpoint (e.g. an IP-PBX) is registered but the callee user is not available.

- **Call Forward SMS (CFS)**: The SMS forward is always executed, completely disregarding the subscriber state. SMS service has to be enabled, see the SMS (Short Message Service) subchapter for a detailed description on how to activate it.
- **Call Forward on Response (CFR)**: The Call Forward is executed only for particular reply codes received back from the destination endpoint. The list of the reply codes and the activation mode can be configured in *rerouting_codes* and *rerouting_mode* subscriber's preferences. Example: suppose that *rerouting_codes* is set to 503, *rerouting_mode* to whitelist and the CFR is configured. If that subscriber receives a call and the device replies back with the code 503, then the call will be re-routed to the destination configured in the CFR. For all the other reply codes the CFR will be NOT executed.
- **Call Forward on Overflow (CFO)**: The Call Forward is executed when the new incoming call for the subscriber exceeds the limits configured in *concurrent_max_in_total*, *concurrent_max_in* or

Copyright © Sipwise GmbH - All rights reserved

concurrent_max_in_per_account subscriber's preferences. If none of the preferences is set then the CFO will be NOT executed.

7.3.2. Setting a Call Forward

Go to *SettingsSubscribersselect the desired subscriberDetails* and then select the *Call Forwarding* tab.

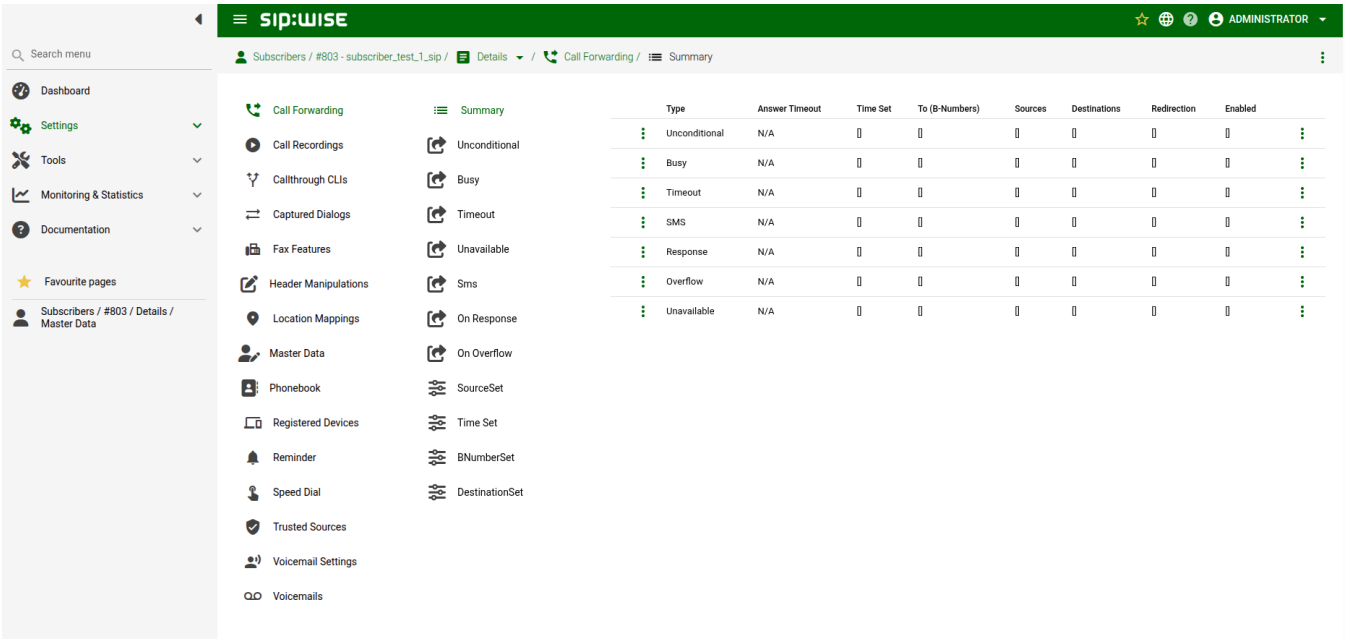

Figure 50. Subscriber's Call Forward page

The table in the page shows all the Call Forward entries that are configured for the selected subscriber with their corresponding parameters.

Beside Call Forwarding to a single destination, Sipwise C5 offers the possibility to activate Call Forwarding:

- to multiple destinations
- only during a pre-defined time set
- only for specific callers
- only for specific callee

To achieve this you can configure the following settings:

- **Destination Set**: it is a list of destinations where the call will be routed to, one after another, according to the order of their assigned priorities. See the Destination Sets subchapter for a detailed description.
- **Time Set**: it is a time period definition, i.e. when the Call Forwarding has to be active. See the Time Sets subchapter for a detailed description.
- **Source Set**: it is a list of number patterns that will be matched against the calling party number; if the calling number matches the Call Forwarding will be executed. See the Source Sets subchapter for a detailed description.
- **B-Number Set**: it is a list of number patterns that will be matched against the called party number; if

 Copyright © Sipwise GmbH - All rights reserved

the callee number matches the Call Forwarding will be executed. See the B-Number Sets subchapter for a detailed description.

To configure a basic Call Forward, it is at least necessary to create a *Destination Set*. Once this is done, it can be used as destination for a Call Forward.

7.3.3. Configuring Destination Sets

Click on the *Destination Set* tab to see the list of the already created sets. To create a new *Destination Set* click on the *Add* button.

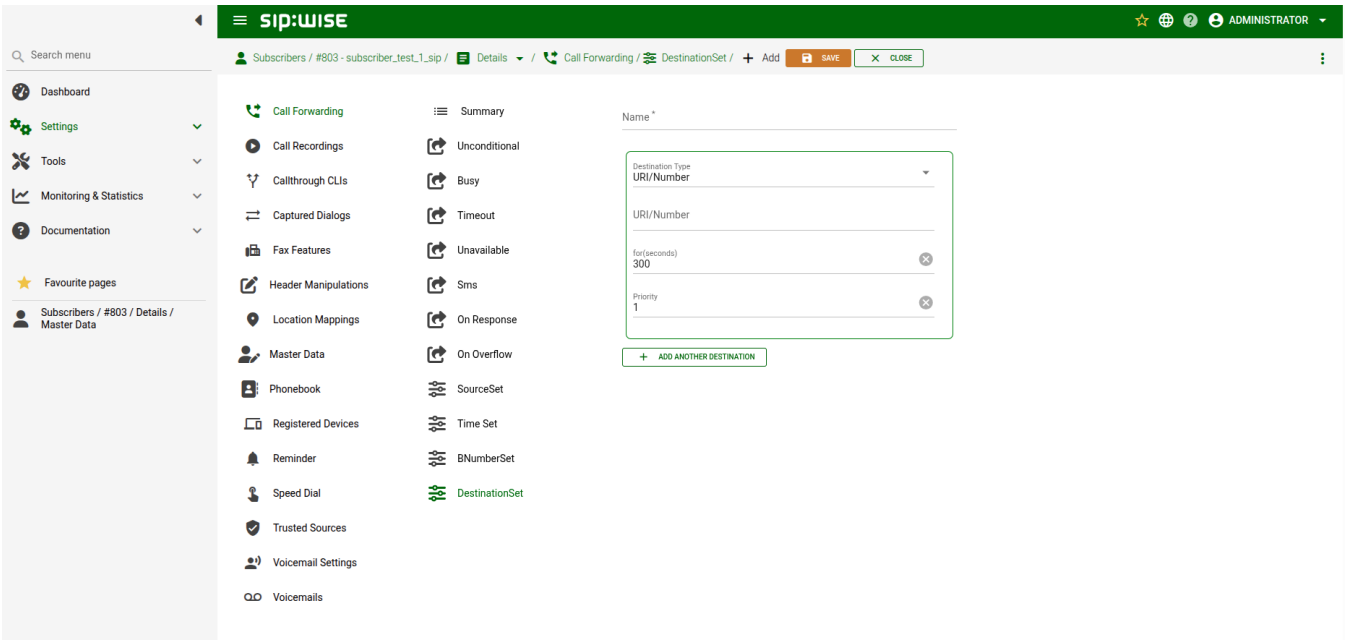

Figure 51. Create CF Destination Set

You need to fill the following fields:

- **Name**: Name to assign to the *Destination Set*
- **Destination Type**: Type of destination where the call has to be forwarded.
- **URI/Number**: The destination to forward calls to. This option is only shown in case the destination type is set to *URI/Number*. Specify a valid SIP URI string or a plain number.
- **for (seconds)**: Sets the ringing time after which the call is forwarded to the next number on the list (if configured).
- **Custom Announcement**: Custom announcement to be played back to the caller. This option is only shown in case the destination type is set to *Custom Announcement*. The audio file to play are created in the *Sound Sets* and must have the same name of the selected entry.
- **Priority**: Priority assigned to this destination. It is important to define the destinations execution order in case they are more than one.

You can add more destinations to the *Destination Set* by pressing the *Add Another Destination_* button. When you have finished adding all destinations, press the *Save* button.

Copyright © Sipwise GmbH - All rights reserved

Destinations Types

Sipwise C5 allows to configure different types of destination:

- **URI/Number**: Calls are forwarded to a specific SIP URI string or plain number. It is commonly used to forward the call to another subscriber or to a number reachable through peering connections. If this type is selected, the *URI/Number* field has to be filled.
- **Voicemail**: Calls are forwarded to the Voicemail Application Server of the current subscriber, where the caller can leave a message.
- **Conference**: Calls are forwarded to the conference room. The subscriber is the host of the conference.
- **Custom Announcement**: Calls are forwarded to a custom announcement that is played back to the caller. If this type is selected, the *Custom Announcement* field has to be filled.

7.3.4. Configuring Time Sets

Click on the *Time Set* tab to see the list of the already created sets. To create a new *Time Set* click on the *Add* button.

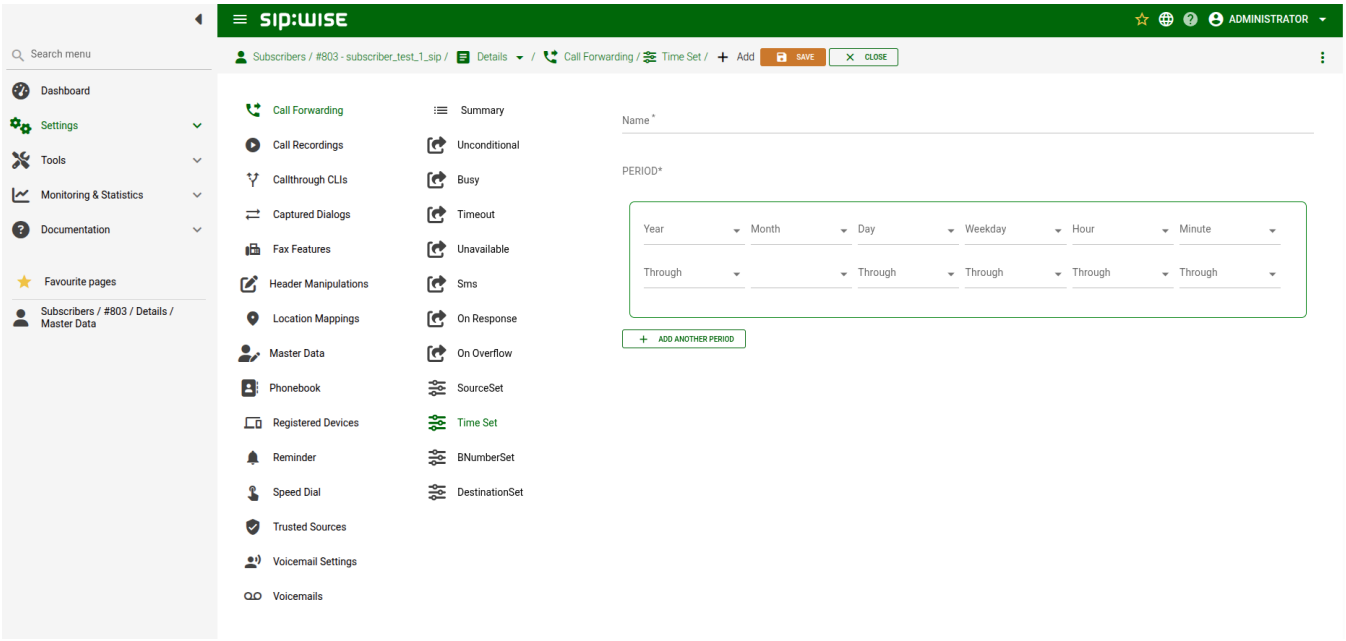

Figure 52. Create CF Time Set

You need to fill the following fields:

- **Name**: Name to assign to the *Time Set*
- **Periods**: Period in which the Call Forward will be executed.

The setup of the *Period* works as the following. If you only set the top line setting of a date field, then it is valid for only this setting (for example the full year if you set only the *Year* value). If you provide the bottom setting as well, it defines a period that starts from the value of the first line and ends with the value of the second line (for example a period valid for March, April and May, if you set March in the first line and May in the second one).

A more concrete example: if a Call Forward is set with the top row hour 10 and minute 20, bottom row

 Copyright © Sipwise GmbH - All rights reserved

hour 12 and minute 30, then the Call Forward 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).

You can add more periods to the *Time Set* by pressing the *Add Another Period_* button. When you have finished adding all periods, press the *Save* button.

7.3.5. Configuring Source Sets

Click on the *Source Set* tab to see the list of the already created sets. To create a new *Source Set* click on the *Add* button.

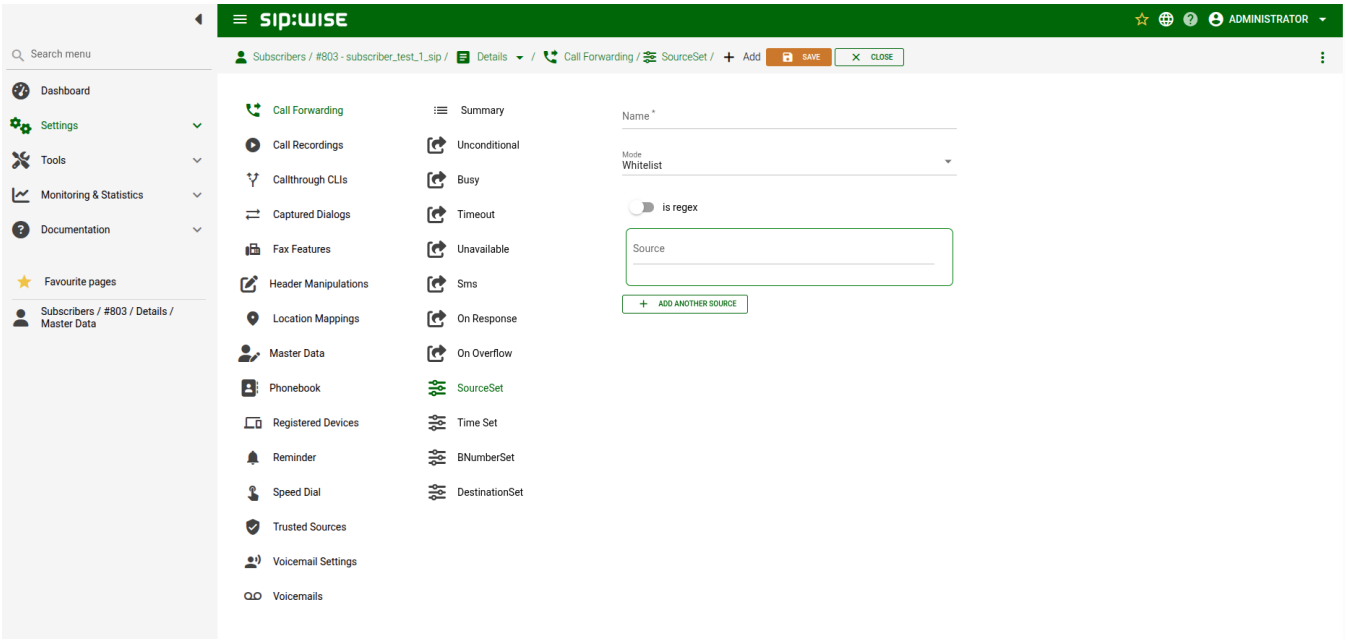

Figure 53. Create CF Source Set

You need to fill the following fields:

- **Name**: Name to assign to the *Source Set*
- **Mode**: Indicates if the match has to be done using a whitelist or blacklist logic.
- **Is regex**: Flag to indicate if the *Source* field has to be considered a perl compatible regular expression.
- **Source**: The value to be compared with the call source number.

The *Source* value could be:

- A simple phone number in E.164 format.
- A pattern, in order to define a range of numbers. You can use * (matches a string of 0 to any

Copyright © Sipwise GmbH - All rights reserved

number of characters), `?` (matches any single character), `[abc]` (matches a single character that is part of the explicitly listed set: a, b or c) and `[0-9]` (matches a single character that falls in the range 0 to 9) as wildcards, as usual in shell patterns. Examples:

`431*` (all numbers from Vienna / Austria)

`49176[0-5]77*` (German numbers containing fixed digits and a variable digit in 0-5 range in position 6)

`43130120??` (numbers from Vienna with fixed prefix and 2 digits variable at the end)

- A perl compatible regular expression (only if *is_regex* is set). Capturing groups can be formed using parentheses and referenced in the *Destination Set* via `\\1`, `\\2`,...
- The constant string "anonymous" that indicates a suppressed calling number (CLIR).

You can add more sources to the *Source Set* by pressing the *Add Another Source_* button. When you have finished adding all sources, press the *Save* button.

7.3.6. Configuring B-Number Sets

Click on the *B-Number Set* tab to see the list of the already created sets. To create a new *B-Number Set* click on the *Add* button.

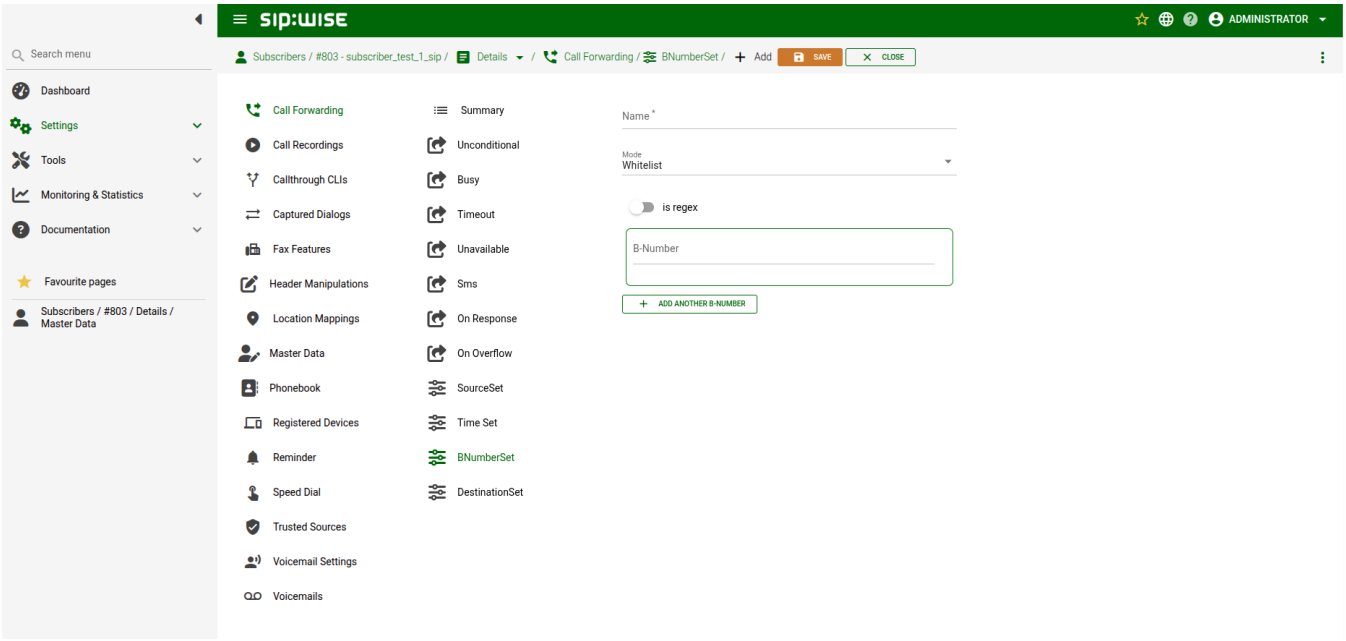

Figure 54. Create CF B-Number Set

You need to fill the following fields:

- **Name**: Name to assign to the *B-Number Set*
- **Mode**: Indicates if the match has to be done using a whitelist or blacklist logic.
- **Is regex**: Flag to indicate if the *B-Number* field has to be considered a perl compatible regular expression.
- **B-Number**: The value to be compared with the original destination number.

The *_B-Number* value could be:

 Copyright © Sipwise GmbH - All rights reserved

- A simple phone number in E.164 format.
- A pattern, in order to define a range of numbers. You can use `*` (matches a string of 0 to any number of characters), `?` (matches any single character), `[abc]` (matches a single character that is part of the explicitly listed set: a, b or c) and `[0-9]` (matches a single character that falls in the range 0 to 9) as wildcards, as usual in shell patterns. Examples:

  `431*` (all numbers from Vienna / Austria)

  `49176[0-5]77*` (German numbers containing fixed digits and a variable digit in 0-5 range in position 6)

  `43130120??` (numbers from Vienna with fixed prefix and 2 digits variable at the end)
- A perl compatible regular expression (only if *is_regex* is set). Capturing groups can be formed using parentheses and referenced in the *Destination Set* via `\\1`, `\\2`,...

You can add more B-Numbers to the *B-Number Set* by pressing the *Add Another B-Number_* button. When you have finished adding all B-Numbers, press the *Save* button.

7.3.7. Finalizing the Call Forward Definition

The final step of defining a Call Forward settings is the creation of the Call Forward entry itself. To proceed click on the tab corresponding to the Call Forward type you want to configure, for example *Unconditional* to create an Unconditional Call Forward.

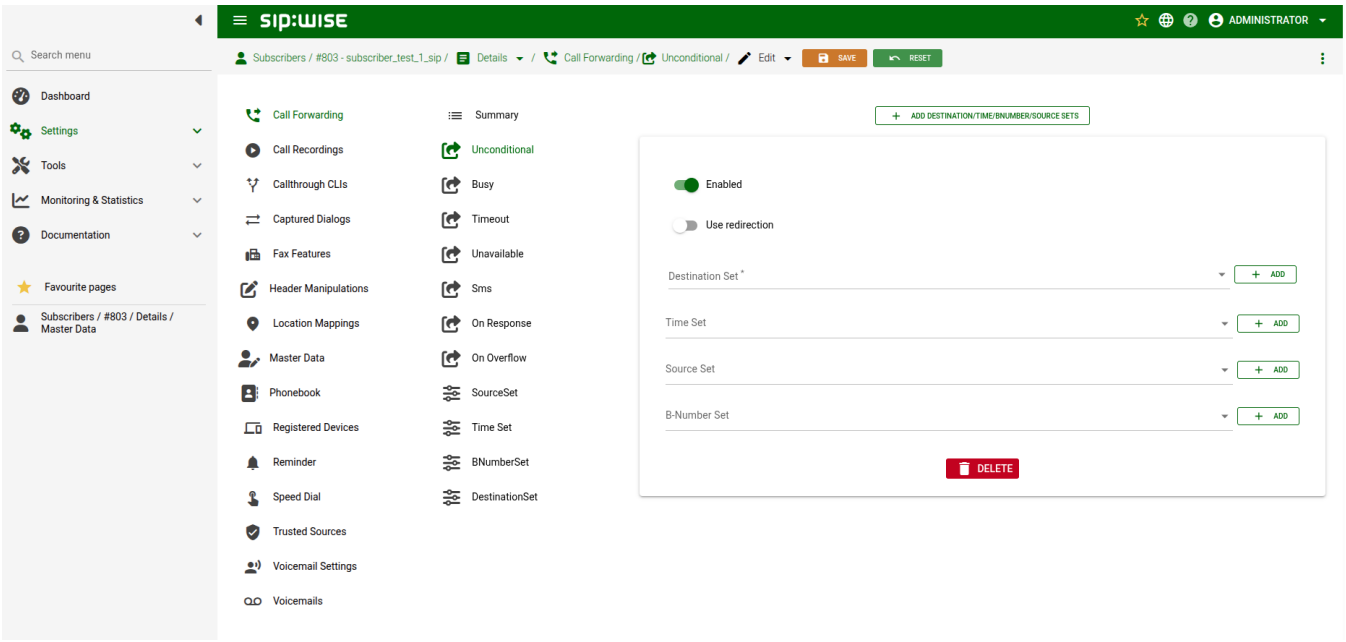

Figure 55. Create the Call Forward

The parameters to set are:

- **Enabled**: Defines whether the Call Forward rule is being used or not.
- **Use redirection**: If enabled and only for calls coming from PSTN, the callforward is performed as a redirection and Sipwise C5 will reply with a "302 Moved Temporarily" message, the new destination is set in the Contact header.
- **Destination Set**: Select an existing Destination Set from the list

Copyright © Sipwise GmbH - All rights reserved

- **Time Set** (optional): Select an existing Time Set from the list
- **Source Set** (optional): Select an existing Source Set from the list
- **B-Number Set** (optional): Select an existing B-Number Set from the list

If one of the endpoints of the Call Forwards parameters is not created yet, you can switch to the creation of the element pressing the corresponding *Add* button. Attention because doing any unsaved progress will be lost.

You can add more Call Forwards of the same type pressing the *Add Destination/Time/B-Number/Source Set* button. The Call forwards will be executed in the order of appearance in the UI.

When you have finished adding all Call Forwards and setting all the parameters, press the *Save* button.

7.4. Call Forking by Q Value

The Sipwise C5 platform allows you to register multiple devices under the same subscriber.

By default, you can register up to 5 devices, but this limit can be adjusted using the *kamailioproxy max_registrations_per_subscriber* setting in the *config.yml* file.

If you want to disable call forking, an alternative is to set *kamailioproxylast_registration_only* to "yes." This restricts each subscriber to a single registered device. This behavior is similar to *kamailioproxy max_registrations_per_subscriber*, with the key difference being that it removes all previous registrations and retains only the most recent contact per AoR.

If a customer registers multiple devices, Sipwise C5 – once it receives a call for that user – sends the call to all the registered devices, in parallel. All the devices will ring at the same time. This is called Parallel Forking, and this is the default behavior. The Sipwise C5 can also do the so-called Serial Forking, which means let ring one device first, then after a timeout let ring the next device, and so on and so forth. The Serial Forking feature can be activated setting subscriber/domain preference *serial_forking_by_q_value*.

7.4.1. The Q Value

Serial Forking is based on SIP Contact's parameter called 'Q value', which is a priority number, set by the clients during their Registration. The q value is a floating point number in a range 0 to 1.0 specified as a parameter in the Contact header field.

In case the client does not set the q value, Sipwise C5 set a default value of 'q=-1' in the database.

Q value can be also specified during the creation of a subscriber's permanent registration (*Details Registered DevicesCreate Permanent Registration*).

The Sipwise C5 can apply two different type of algorithm to the q values in order to achieve two different types of serial forking called *standard* and *probability*.

7.4.2. The Standard Method

This method uses the q values as a pure priority index. The higher the q value number, the more priority that device has. Contacts with q value 1.0 have maximum priority, so such contacts will be always tried first in serial forking. Contacts with q value 0 have the lowest priority and they will be tried after all other

 Copyright © Sipwise GmbH - All rights reserved

contacts with higher priority.

NOTE In case two or more contacts have the same q value, then they are tried in parallel. This allows to create a very flexible mix of Serial and Parallel forking.

This method can be activated setting *Standard* in subscriber/domain preference *serial_forking_by_q_value*.

7.4.3. The Probability Method

This method uses the q values as the weight of the contact. The higher the q value number, the more probability that the device has to ring first. Equals q values means equals probability to be tried. Contacts with q values equals to 0 or lower are not considered by the ordering algorithm, but added at the end of the list as backup option if all other contacts fail.

NOTE Differently from the *standard* method there is no possibility to have parallel forking. This algorithm can be useful to load-balance the calls in case of endpoints in ACTIVE-ACTIVE configuration.

This method can be activated setting *Probability* in subscriber/domain preference *serial_forking_by_q_value*.

7.4.4. Advanced Configurations

If a subscriber with Serial Forking enabled receives a call, Sipwise C5 calls the registered devices one after the other. The forking is stopped only in the following cases:

- there are no more devices to try to contact
- one of the ringing devices answers the call
- one of the ringing devices replies with the SIP code 600, 603, 604, 606 or one of the response codes defined in *stop_forking_code_lists* subscriber/domain preference.
- a Call Forward on Timeout is set and the ringtimeout is reached.

Sipwise C5 allows you also to define how long each single device has to ring during a Serial Forking call. To do that, set the subscriber/domain preference *contact_ringtimeout* to the desired value.

NOTE In case both *contact_ringtimeout* and Call Forward on Timeout are configured, CFT timeout has higher priority. To clarify this concept, please take a look at the following examples: Case 1: CFT timeout lower than the total ringtimeout of the contacts. For example: CFT timeout = 100, contact_ringtimeout = 40 and 3 devices registered: 1st device will ring for 40 seconds, 2nd device will ring for other 40 seconds, 3rd device will ring only for 20 seconds because of the CFT. Case 2: CFT timeout higher than the total ringtimeout of the contacts. For example: CFT timeout = 100, contact_ringtimeout = 40 and 2 devices registered: 1st device will ring for 40 seconds, 2nd device will ring till reaching the CFT timeout (60 seconds in this case).

7.5. Local Number Porting

The Sipwise C5 platform comes with two ways of accomplishing local number porting (LNP):

- one is populating the integrated LNP database with porting data,

Copyright © Sipwise GmbH - All rights reserved

- the other is accessing external LNP databases via the Sipwise LNP daemon using the LNP API.

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

7.5.1. Local LNP Database

The local LNP database provides the possibility to define LNP Carriers (the owners of certain ported numbers or number blocks) and their corresponding LNP Numbers belonging to those carriers. It can be configured on the admin panel in *SettingsNumber Porting* or via the API. The LNP configuration can be populated individually or via CSV import/export both on the panel and the API.

NOTE For huge LNP datasets, there is a CLI importer that also supports an efficient way of applying day-to-day deltas: LNP Dataset Importer.

LNP Carriers

LNP Carriers are defined by an arbitrary *Name* for proper identification (e.g. *British Telecom*) and contain a *Prefix* which can be used as routing prefix in LNP Rewrite Rules and subsequently in Peering Rules to route calls to the proper carriers. The LNP prefix is written to CDRs to identify the selected carrier for post processing and analytics purposes of CDRs. LNP Carrier entries also have an *Authoritative* flag indicating that the numbers in this block belong to the carrier operating Sipwise C5 . This is useful to define your own number blocks, and in case of calls to those numbers reject the calls if the numbers are not assigned to local subscribers (otherwise they would be routed to a peer, which might cause call loops). Finally the *Skip Rewrite* flag skips executing of LNP Rewrite Rules if no number manipulation is desired for an LNP carrier.

LNP Numbers

LNP Carriers contain one or more LNP Numbers. Those LNP Numbers are defined by a *Number* entry in E164 format (<*cc*><*ac*><*sn*>) used to match a number against the LNP database. Number matching is performed on a longest match, so you can define number blocks without specifying the full subscriber number (e.g. a called party number *431999123* is going to match an entry *431999* in the LNP Numbers).

For an LNP Numbers entry, an optional *Routing Number* can be defined. This is useful to translate e.g. premium 900 or toll-free 800 numbers to actual routing numbers. If a Routing Number is defined, the called party number is implicitly replaced by the Routing Number and the call processing is continued with the latter. For external billing purposes, the optional *Type* tag of a matched LNP number is recorded in CDRs.

An optional *Start Date* and *End Date* makes it possible to schedule porting work-flows up-front by populating the LNP database with certain dates, and the entries are only going to become active with those dates. Empty values for start indicate a start date in the past, while empty values for end indicate an end time in the future during processing of a call, allowing to define infinite date ranges. As intervals can overlap, the LNP number record with a start time closest to the current time is selected.

Enabling Local LNP Support

Sipwise C5 currently supports different types of local number porting: All Call Query (ACQ), Onward Routing (OR) and Query on Release (QoR).

In order to activate Local LNP lookup during call routing, the feature must be activated in `/etc/ngcp-config/config.yml`. Set these parameters:

 Copyright © Sipwise GmbH - All rights reserved

- `kamailio→proxy→lnp→enable` : *yes*
- `kamailio→proxy→lnp→type` : *local / local_qor*

All Call Query (ACQ) and Onward Routing (OR) methods can be covered by the *type* 'local' and they will be referred as 'local' methods from now onwards. For Query on Release (QoR) the 'local_qor' type has to be set up and it will be referred as QoR in the rest of the document.

NOTE Query on Release (QoR) LNP type is available for PRO and CARRIER products only.

Local LNP Routing Procedure

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 to find the LNP prefix of the calling peer, which is then stored as *source_lnp_prefix* in the CDR, together with the selected LNP number's *type* tag (*source_lnp_type*). If the LNP lookup does not return a result (e.g. the calling party number is not populated in the local LNP database), but the peer preference *default_lnp_prefix* is set for the originating peer, then the value of this preference is stored in *source_lnp_prefix* of the CDR.

Next, an LNP lookup is performed using the normalized called party number. If no number is found (using a longest match), no further manipulation is performed.

If an LNP number entry is found, and the *Routing Number* is set, the called party number is replaced by the routing number. Also, if the *Authoritative* flag is set in the corresponding LNP Carrier, and the called party number is not assigned to a local subscriber, the call is rejected. This ensures that numbers allocated to the system but not assigned to subscribers are dropped instead of routed to a peer.

IMPORTANT If the system is serving a local subscriber with only the routing number assigned (but not e.g. the premium number mapping to this routing number), the subscriber will not be found and the call will either be rejected if the called party premium number is within an authoritative carrier, or the call will be routed to a peer. This is due to the fact that the subscriber lookup is performed with the dialled number, but not the routing number fetched during LNP. So make sure to assign e.g. the premium number to the local subscriber (optionally in addition to the routing number if necessary using alias numbers) and do not use the LNP routing number mechanism for number mapping to local subscribers.

Next, if the LNP carrier does not have the *Skip Rewriting* option set, the *LNP Rewrite Rules for Callee* are engaged. The rewrite rule set used is the one assigned to the originating peer or subscriber/domain via the *rewrite_rule_set* preference. The variables available in the match and replace part are, beside the standard variables for rewrite rules:

- `${callee_lnp_prefix}`: The prefix stored in the LNP Carrier
- `${callee_lnp_basenumber}`: The actual number entry causing the match (may be shorter than

Copyright © Sipwise GmbH - All rights reserved

the called party number due to longest match)

Typically, you would create a rewrite rule to prefix the called party number with the *callee_lnp_prefix* by matching `^([0-9]+)$` and replacing it by `${callee_lnp_prefix}\1`.

Once the LNP processing is completed, the system checks for further preferences to finalize the number manipulation. If the terminating local subscriber or peer has the preference *lnp_add_npdi* set, the Request URI user-part is suffixed with `;npdi`. Next, if the preference *lnp_to_rn* is set, the Request URI user-part is suffixed with `;rn=LNP_ROUTING_NUMBER`, where *LNP_ROUTING_NUMBER* is the *Routing Number* stored for the number entry in the LNP database, and the originally called number is kept in place. For example, if *lnp_to_rn* is set and the number *1800123* is called, and this number has a routing number *1555123* in the LNP database, the resulting Request-URI is `sip:1800123;rn=1555123@example.org`.

Finally, the *destination_lnp_prefix* in the CDR table is populated either by the prefix defined in the Carrier of the LNP database if a match was found, or by the *default_lnp_prefix* preference of the destination peer or subscriber/domain.

Blocking Calls Using LNP Data

The Sipwise C5 provides means to allow or block calls towards ported numbers that are hosted by particular LNP carriers. Please visit Creating Rules per NCOS Level in the handbook to learn how this can be achieved.

Transit Calls Using LNP

If a call originated from a peer and the peer preference *force_outbound_calls_to_peer* is set to *force_nonlocal_lnp* (the *if callee is not local and is ported* selection in the panel), the call is routed back to a peer selected via the peering rules.

This ensures that if a number once belonged to your system and is ported out, but other carriers are still sending calls to you (e.g. selecting you as an anchor network), the affected calls can be routed to the carrier the number got ported to.

Upload and Download LNP Entries

The LNP database can be downloaded as CSV, and in the same format can be uploaded to the system. On upload, you can decide whether to drop existing data prior to applying the data from the CSV.

The CSV file format contains the fields in the following order:

Table 11. LNP CSV Format

Name	Description
Carrier Name	The *Name* in the LNP Carriers table (string, e.g. *My Carrier*)
Carrier Prefix	The *Prefix* in the LNP Carriers table (string, e.g. *DD55*)
Number	The *Number* in the LNP Numbers table (E164 number, e.g. *1800666*)

Copyright © Sipwise GmbH - All rights reserved

Name	Description
Routing Number	The *Routing Number* in the LNP Numbers table (E164 number or empty, e.g. *1555666*)
Start	The *Start* in the LNP Numbers table (YYYY-MM-DD or empty, e.g. *2016-01-01*)
End	The *End* in the LNP Numbers table (YYYY-MM-DD or empty, e.g. *2016-12-30*)
Authoritative	The *Authoritative* flag in the LNP Carriers table (0 or 1)
Skip Rewrite	The *Skip Rewrite* flag in the LNP Carriers table (0 or 1)
Type	The *Type* tag in the LNP Numbers table (alphanumeric string, e.g. *mobile*)

LNP Dataset Importer

Large LNP datasets can be imported from CLI:

```
/usr/sbin/ngcp-lnp-importer
```

The importer uses a dedicated SQLite database file to identify the delta of LNP dataset files provided between (daily) runs, which is populated and processed in 3 separate stages:

1. import LNP dataset text file to SQLite database and identify delta
2. process *ADDED* delta: new LNP number (and carrier) records are created in the NGCP database tables
3. process *DELETED* delta: removed LNP number (and carrier) records are deleted in the NGCP database tables

The implementation is based on a multithread streaming processor design. In contrast to panel CSV upload/download, there will be a constant memory consumption when reading a LNP dataset file (regardless of its size).

NOTE LNP dataset files are line-delimited (utf-8) text files with tens of millions of lines. A line will correspond to a LNP number or range of LNP numbers. The importer configuration allows to specify any custom file format implementation.

The importer is based on the *bulk-processor* ETL framework, which allows to configure database connectivity and working directories on your own. To start with, copy `/usr/share/perl5/NGCP/BulkProcessor/Projects/ETL/Lnp/config.cfg` to `/var/myworkspace/lnp/config.cfg` and adjust it accordingly:

```
##general settings:
working_path = /var/myworkspace/lnp
cpucount = 4
enablemultithreading = 1
```

Copyright © Sipwise GmbH - All rights reserved

```
##load credentials from constants.yml (requires root privileges):
constants_yml = /etc/ngcp-config/constants.yml

##gearman/service listener config:
#jobservers = 127.0.0.1:4730

##NGCP MySQL connectivity - "accounting" db:
#accounting_host = db01
#accounting_port = 3306
#accounting_databasename = accounting
#accounting_username = root
#accounting_password =

##NGCP MySQL connectivity - "billing" db:
#billing_host = db01
#billing_port = 3306
#billing_databasename = billing
#billing_username = root
#billing_password =

##NGCP MySQL connectivity - "provisioning" db:
#provisioning_host = db01
#provisioning_port = 3306
#provisioning_databasename = provisioning
#provisioning_username = root
#provisioning_password =

##NGCP MySQL connectivity - "kamailio" db:
#kamailio_host = db01
#kamailio_port = 3306
#kamailio_databasename = kamailio
#kamailio_username = root
#kamailio_password =

##NGCP MySQL connectivity - default db for distributed transactions (XA)
to connect to:
#xa_host = db01
#xa_port = 3306
#xa_databasename = ngcp
#xa_username = root
#xa_password =

##NGCP REST-API connectivity:
#ngcprestapi_uri = https://127.0.0.1:1443
#ngcprestapi_username = administrator
#ngcprestapi_password = administrator
#ngcprestapi_realm = api_admin_http

##sending email:
emailenable = 0
erroremailrecipient =
warnemailrecipient =
```


Copyright © Sipwise GmbH - All rights reserved

```
completionemailrecipient = jdoe@company.com
doneemailrecipient =

##logging:
#fileloglevel = DEBUG
fileloglevel = INFO
screenloglevel = INFO
emailloglevel = OFF
```

NOTE Default values apply if not specified explicitly. When you i.e. omit explicit connection details in `config.cfg`, credentials will be read from `/etc/ngcp-config/constants.yml`. This however requires to run the importer as unix user with elevated privileges.

Next, copy the importer configuration `/usr/share/perl5/NGCP/BulkProcessor/Projects/ETL/Lnp/settings.yml` to `/var/myworkspace/lnp/settings.yml` and adjust it to your needs:

```
##CLI option to make changes to NGCP database permanent, or "dry run"
only:
#dry=0

##CLI option to abort run on errors:
#skip_errors=0

##LNP dataset file name (if not specified as CLI arg):
#lnp_filename: /var/myworkspace/lnp/input/large_lnp_dataset.txt
##line number (1-based) to start reading from:
lnp_rownum_start: 2

##SQLite database file to identify delta between runs -
/var/myworkspace/lnp/db/lnp.db
sqlite_db_file: lnp

##use multiple threads for loading lnp_filename to lnp.db:
import_multithreading: 1
##limit thread count:
lnp_import_numofthreads: 2
##insert multiple LNP numbers per transaction:
lnp_import_single_row_txn: 0
ignore_lnp_unique: 0

##use multiple threads for adding LNP number records to NGCP database:
create_lnp_multithreading: 1
##limit thread count:
create_lnp_numofthreads: 1
##insert multiple LNP numbers per transaction:
create_lnp_blocksize: 1000000
lnp_numbers_multi_insert: 1
lnp_numbers_single_row_txn: 0
ignore_lnp_numbers_unique: 0
```

Copyright © Sipwise GmbH - All rights reserved

```
##use multiple threads for removing LNP number records from NGCP
database:
delete_lnp_multithreading: 1
##limit thread count:
delete_lnp_numofthreads: 1
##delete multiple LNP numbers per transaction:
delete_lnp_blocksize: 1000
lnp_numbers_batch_delete: 1

##file format implementation
expand_numbers: !!perl/code |
    {
      my ($context,$row) = @_;
      my ($carrier_name,
        $carrier_prefix,
        $number,
        $routing_number,
        $start,
        $end,
        $authoritative,
        $skip_rewrite,
        $type) = split(/,/,$row);

      my @lnp_numbers = ();
      push(@lnp_numbers,[
        $carrier_name,
        $carrier_prefix,
        $number,
        $routing_number,
        (length($start) ? $start : undef),
        (length($end) ? $end : undef),
        $authoritative,
        $skip_rewrite,
        $type]) if length($row);
      return \@lnp_numbers;
    }
```

By default, the LNP dataset file is considered as CSV with the known fields:

- carrier_name
- carrier_prefix
- number
- routing_number
- start
- end
- authoritative
- skip_rewrite

Copyright © Sipwise GmbH - All rights reserved

- type

Any custom file format implementation can be provided in `settings.yml`:

```
##the custom file format implementation is provided as a perl sub.
##the line string is passed as input argument, to be be processed/split
as desired.
##the sub is supposed to return an array of LNP number records for NGCP,
##each of which is a array of the LNP number fields:
##[
##  'carrier_name',
##  'carrier_prefix',
##  'number',
##  'routing_number',
##  'start',
##  'end',
##  'authoritative',
##  'skip_rewrite',
##  'type',
##]
##these will also be the expected comma-separated fields,
##if no custom format is specified.
##below shows an example of a real-world file format implementation with
##number range expansion:
expand_numbers: !!perl/code |
    {
      my ($context,$row) = @_;
      if ($row =~ /^Linha\d+=([^=]*)$/i) {
        my (
          $DonorID,
          $HolderID,
          $TypeOfNumber,
          $PABXMainTelephoneNumber,
          $FirstTelephoneNumber,
          $LastTelephoneNumber,
          $PresentNRN,
          $DateTimeFrom
        ) = split(/,/,$1);

        my @lnp_numbers = ();
        #020,012,0,212879000,212879000,212879999,D012001,2016-08-31
14:20:18
        unless ($PresentNRN =~ /^D012[0-9]{3,3}$/) { #custom number
range notation
          foreach my $number ($FirstTelephoneNumber ..
$LastTelephoneNumber) {
            push(@lnp_numbers,[
              $PresentNRN, #'carrier_name',
              $PresentNRN, #'carrier_prefix',
              $number, #'number',
              undef, #'routing_number',
```

Copyright © Sipwise GmbH - All rights reserved

```
              $DateTimeFrom, #'start',
              undef, #'end',
              undef, #'authoritative',
              undef, #'skip_rewrite',
              undef, #'type',
            ]);
        }
      }
      return \@lnp_numbers;
    }

    return [];
  }
```

The importer stages can be invoked using the `--task` options:

Load LNP Dataset File:

```
sudo ngcp-lnp-importer \
  --task=load_file
  --config=/var/myworkspace/lnp/config.cfg
  --settings=/var/myworkspace/lnp/settings.yml
  --file=/var/myworkspace/lnp/input/large_lnp_dataset.txt
```

Purge All LNP Records in NGCP:

```
sudo ngcp-lnp-importer \
  --task=purge_lnp
  --config=/var/myworkspace/lnp/config.cfg
  --settings=/var/myworkspace/lnp/settings.yml
```

Create *ADDED* Delta LNP Records in NGCP:

```
sudo ngcp-lnp-importer \
  --task=create_lnp
  --config=/var/myworkspace/lnp/config.cfg
  --settings=/var/myworkspace/lnp/settings.yml
```

Delete *DELETED* Delta LNP Records in NGCP:

```
sudo ngcp-lnp-importer \
  --task=delete_lnp
  --config=/var/myworkspace/lnp/config.cfg
  --settings=/var/myworkspace/lnp/settings.yml
```


Copyright © Sipwise GmbH - All rights reserved

Load LNP Dataset File, Create and Delete LNP Records in NGCP *in One Go*:

```
sudo ngcp-lnp-importer \
  --task=load_file
  --task=create_lnp
  --task=delete_lnp
  --config=/var/myworkspace/lnp/config.cfg
  --settings=/var/myworkspace/lnp/settings.yml
  --file=/var/myworkspace/lnp/input/large_lnp_dataset.txt
```

NOTE The `--dry` option switch will simulate manipulating records of the NGCP database, without making permanent changes. This allows to test if e.g. unique constraints are violated by the dataset.

Reset SQLite Database:

```
sudo ngcp-lnp-importer \
  --task=cleanup
  --config=/var/myworkspace/lnp/config.cfg
  --settings=/var/myworkspace/lnp/settings.yml
```

IMPORTANT The SQLite database holds the delta information. The NGCP LNP records should be empty/purged before any initial import (without delta).

NOTE The SQLite *.db* database file can be easily copied to save/restore delta information.

Local LNP Returned Values

If a match in the local LNP table is found corresponding LNP Carrier code will be stored in CDR data.

Additionally two dedicated headers can be added to the outgoing SIP message:

- `P-NGCP-LNP-Number`: The returned LNP number, if any
- `P-NGCP-LNP-Status`: The LNP query return code (200 if successful, 404 if no entry found)

This feature is not enabled by default, but can be activated with the following parameters:

- `kamailio→proxy→lnp→add_reply_headers→enable` : *no*
- `kamailio→proxy→lnp→add_reply_headers→number` : *P-NGCP-LNP-Number*
- `kamailio→proxy→lnp→add_reply_headers→status` : *P-NGCP-LNP-Status*

7.6. P-Early-Media

The Sipwise C5 platform supports P-Early-Media Sip Header according to RFC5009.

7.6.1. Implementation

The Sipwise C5 platform tries to find a P-Early-Media SIP header on every INVITE message coming

Copyright © Sipwise GmbH - All rights reserved

from peers or subscribers. If the header is found, it tries to parse the content, in particular it looks for the following parameters:

- sendonly
- recvonly
- inactive
- sendrecv
- supported
- gated

Any other parameter found will be silently discarded.

The first four parameters are also known as *directional parameters* and they will influence how the Sipwise C5 platform will behave when a service, potentially producing early media, is involved during the call. The P-Early-media header will be also transparently sent to the b leg of the call and to the callee which must honour the contents of the header before playing early media.

A typical example is when a subscriber with an announce preference activated is called, under normal conditions the early media with the announce is played before connecting to the callee. The early media however will not be played if the caller is not willing to receive the early audio stream (*inactive* and *sendonly* directional parameters set in P-Early-Media).

There are cases when Sipwise C5 produces an early media to alert the user of an error (typical examples are when the callee is offline or PSTN termination is not available). In these situations, if the caller is not willing to receive the early audio stream, Sipwise C5 will just return a SIP error without playing the early media announce.

The presence of the *supported* parameter forces the Sipwise C5 platform to send back P-Early-Media in 18X replies even if the callee user agent does not include it. This will help the caller user agent to understand that we are going to send early media and an audio channel must be allocated and opened. The Sipwise C5 platform will add the missing P-Early-Media back from the callee towards the caller, according to the following schema:

- if caller sends P-Early-Media: ***sendonly*** Sipwise C5 will reply with P-Early-Media: ***recvonly***
- if caller sends P-Early-Media: ***recvonly*** Sipwise C5 will reply with P-Early-Media: ***sendonly***
- if caller sends P-Early-Media: ***sendrecv*** Sipwise C5 will reply with P-Early-Media: ***sendrecv***
- if caller sends P-Early-Media: ***inactive*** Sipwise C5 will reply with P-Early-Media: ***inactive***

The *gated* parameter is recognized by Sipwise C5 but it does not affect its mode of operation. This parameter, if present, is just passed transparently to the callee.

7.6.2. Configuration

It is possible to alter how Sipwise C5 manages P-Early-Media just by changing the following parameters inside the *config.yml* file:

- kamailio.proxy.p_early_media.default_auth: this parameter will instruct how Sipwise C5 behaves when a P-Early-Media without directional parameters is received. The default behaviour will permit early media streams on both directions (sendrecv).

 Copyright © Sipwise GmbH - All rights reserved

- kamailio.proxy.p_early_media.play_on_missing: this parameter will drive the Sipwise C5 behaviour when the caller sends an INVITE message without P-Early-Media header. If set to yes (default), Sipwise C5 will be allowed to play early media for its own services. If set to no Sipwise C5 will skip early media since it is assuming the caller is not willing to receive early media.

7.7. Emergency Mapping

As opposed to the Simple Emergency Number Handling solution, Sipwise C5 supports an advanced emergency call handling method, called *emergency mapping*. The main idea is: instead of obtaining a statically assigned emergency prefix / suffix from subscriber preferences, Sipwise C5 retrieves an emergency routing prefix and suffix 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:

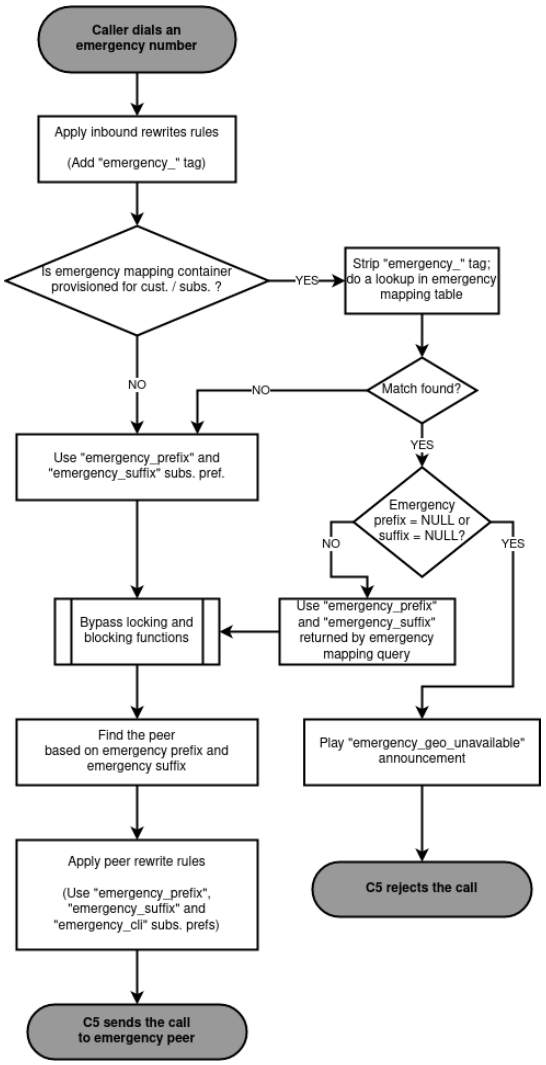

Figure 56. Emergency Call Handling with Mapping

7.7.1. Emergency Mapping Description

Emergency numbers per geographic location are mapped to different routing prefixes not derivable from an area code or the emergency number itself. In some cases the suffix is also needed to be added

Copyright © Sipwise GmbH - All rights reserved

to the emergency number. This is why a **global emergency mapping table** related to resellers is introduced, allowing to map emergency numbers to their geographically dependent routing numbers.

The geographic location is referenced by a location ID, which has to be populated by a north-bound provisioning system. No towns, areas or similar location data is stored on Sipwise C5 platform. The locations are called *Emergency Containers* on NGCP.

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

NOTE Sipwise C5 performs an exact match on the emergency number in the emergency routing table.

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

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

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

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

7.7.2. Emergency Mapping Configuration

The administrative web panel of Sipwise C5 provides the configuration interface for emergency mapping. Please navigate to *Settings* *Emergency Mapping* menu item first, in order to start configuring the mapping.

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

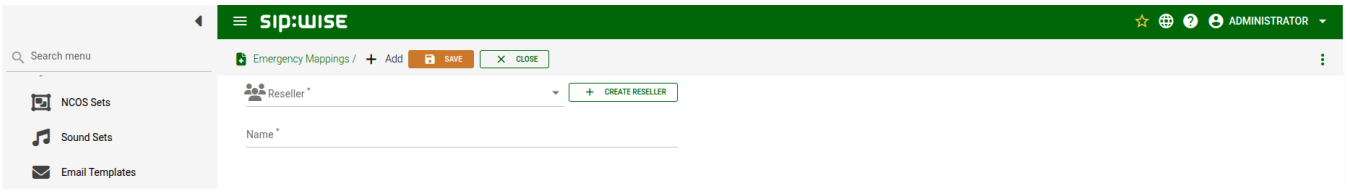

Figure 57. Creating an Emergency Container

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.

 Copyright © Sipwise GmbH - All rights reserved

As the second step of emergency mapping provisioning, the *Emergency Mapping* entries must be created. Press the three dots beside a container and select *Mappings*, then click on *_Add* to start this step. An example is shown here:

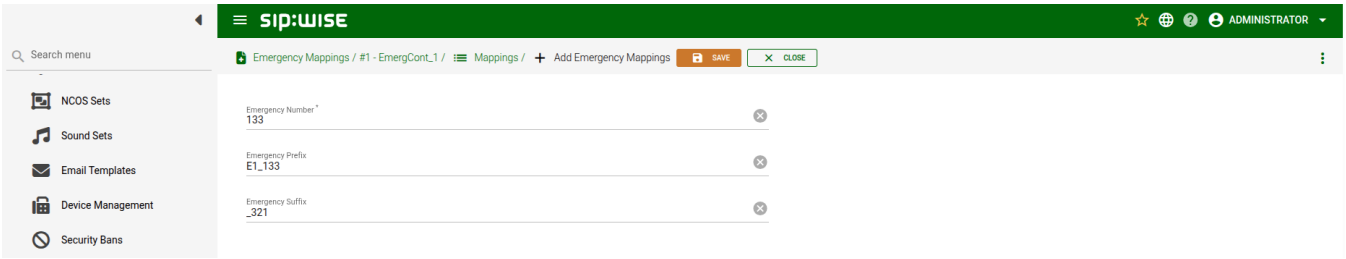

Figure 58. Creating an Emergency Mapping Entry

The following parameters must be set:

- Emergency Number: the emergency number that subscribers will dial
- Emergency Prefix: the routing prefix that belongs to the particular emergency service within the selected location
- Emergency Suffix: the routing suffix that belongs to the particular emergency service within the selected location

The emergency number mapping is now defined. As the next step, the platform administrator has to assign the emergency containers to *Customers / Domains* or *Subscribers*. We will take an example with a *Customer*: select the customer, then navigate to *Preferences Number Manipulations*. In order to assign a container, select the corresponding one from the drop-down list of the *emergency_mapping_container* preference.

Rewrite Rules for Emergency Mapping

Once emergency containers and emergency mapping entries are defined, Sipwise C5 administrator has to ensure that the proper number manipulation takes place, before initiating any emergency call towards peers.

IMPORTANT Please do not forget to define the rewrite rules for peers—particularly: *Outbound Rewrite Rules for Callee*—as described in Normalize Emergency Calls for Peers section of the handbook.

Emergency Calls Not Allowed

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

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

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

Copyright © Sipwise GmbH - All rights reserved

Bulk Upload or Download of Emergency Mapping Entries

The Sipwise C5 offers the possibility to upload / download emergency mapping entries in form of CSV files. This operation is available for each reseller, and is very useful if a reseller has many mapping entries.

Downloading Emergency Mapping List

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_,_321
EmergCont_1,144,E1_144_,_321
EmergCont_2,133,E2_133_,_321
```

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
- Emergency Suffix

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:

- `Upload mapping`: the CSV file must be selected after pressing the *Choose File* button
- `Reseller`: selected from the list
- `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

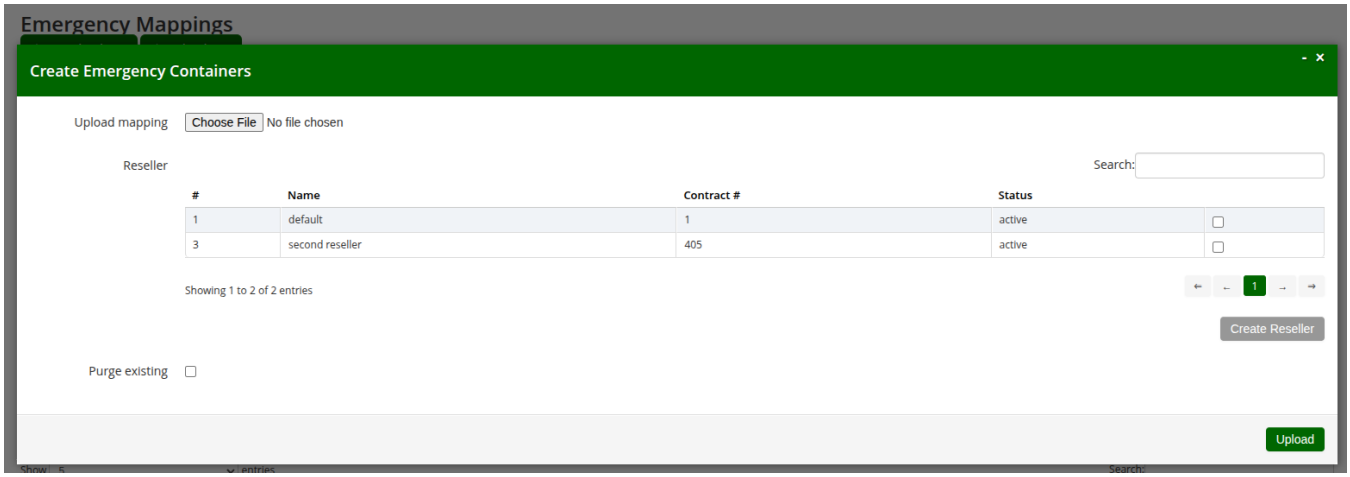

Figure 59. Uploading Emergency Mapping Data

Copyright © Sipwise GmbH - All rights reserved

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

7.8. Emergency Priorization

The Sipwise C5 can potentially host *privileged subscribers* that offer emergency or at least prioritized services (civil defence, police etc.). In case of an emergency, the platform has to be freed from any SIP flows (calls, registrations, presence events etc.) which do not involve those privileged subscribers.

Such an exceptional condition is called ***emergency mode*** and it can be activated for all domains on the system, or only for selected domains.

Once emergency mode is activated, Sipwise C5 will immediately apply the following restrictions on new SIP requests or existing calls:

- Any SIP requests (calls, registrations etc.) from subscribers within the affected domains, who are not marked as privileged, are rejected.
- Any calls from peers not targeting privileged subscribers are rejected.
- Any active calls which do not have a privileged subscriber involved are terminated.

Calls from non-privileged subscribers to emergency numbers are still allowed.

7.8.1. Call-Flow with Emergency Mode Enabled

Typical call-flows of emergency mode will be shown in this section of the handbook. We have the following assumptions:

- Emergency priorization has been enabled on system-level
- There is a domain for which the emergency mode has been activated
- There is a privileged subscriber in that domain
- A generic peering connection has been configured for non-emergency calls
- A dedicated peering connection has been configured for emergency calls

The examples do not show details of SIP messages, but rather give a high-level overview of the call-flows.

1. A **non-privileged** subscriber makes a call **to another non-privileged subscriber**. Result: the call will be **rejected**.

Copyright © Sipwise GmbH - All rights reserved

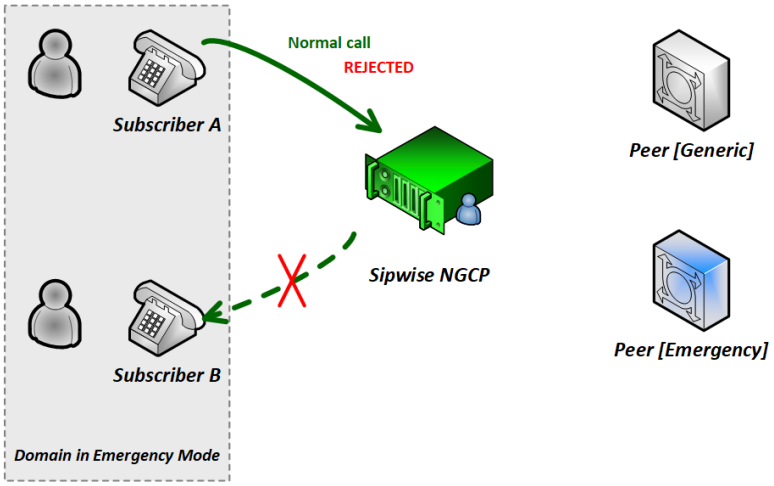

Figure 60. Call-flow in Emergency Mode 1. (Std to Std)

2. A **non-privileged** subscriber makes a call **to an external subscriber (via peer)**. Result: the call will be **rejected**.

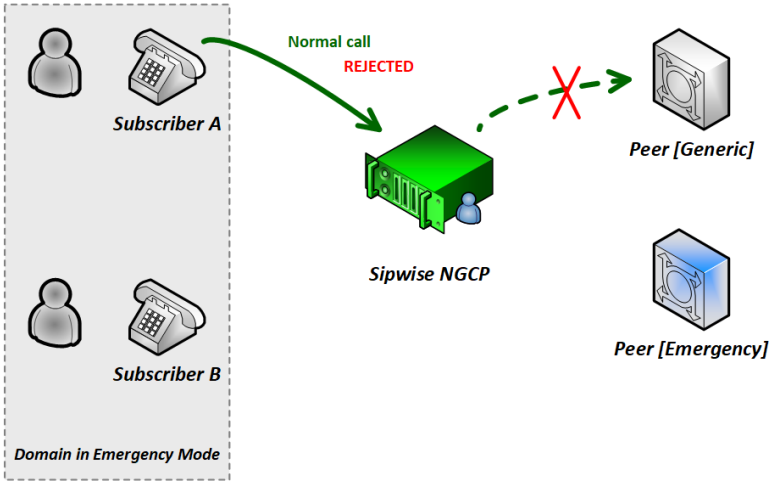

Figure 61. Call-flow in Emergency Mode 2. (Std to Peer)

3. A **non-privileged** subscriber makes a call **to a privileged subscriber**. Result: the call will be **accepted**.

Copyright © Sipwise GmbH - All rights reserved

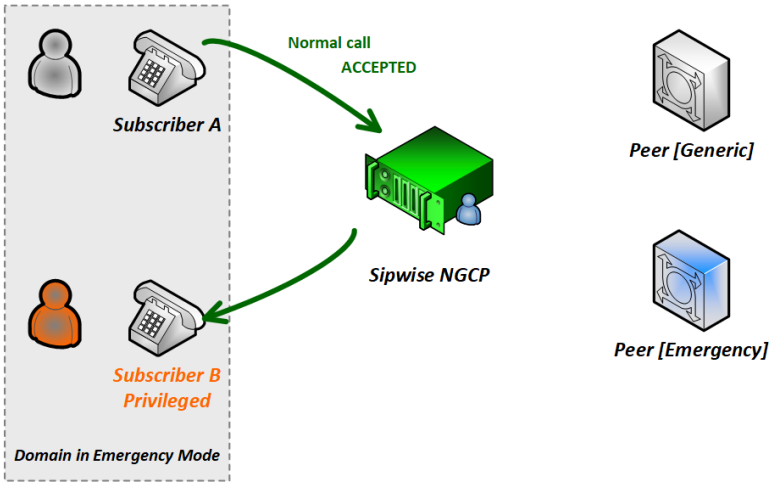

Figure 62. Call-flow in Emergency Mode 3. (Std to Priv)

4. A **non-privileged** subscriber makes a call **to an emergency number**. Result: the call will be **accepted**.

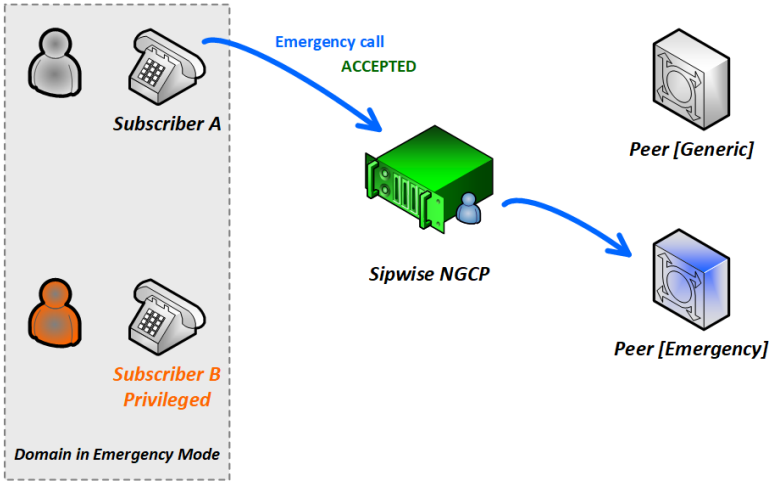

Figure 63. Call-flow in Emergency Mode 4. (Std to Emerg)

5. A **privileged** subscriber makes a call **to a non-privileged subscriber**. Result: the call will be **accepted**.

Copyright © Sipwise GmbH - All rights reserved

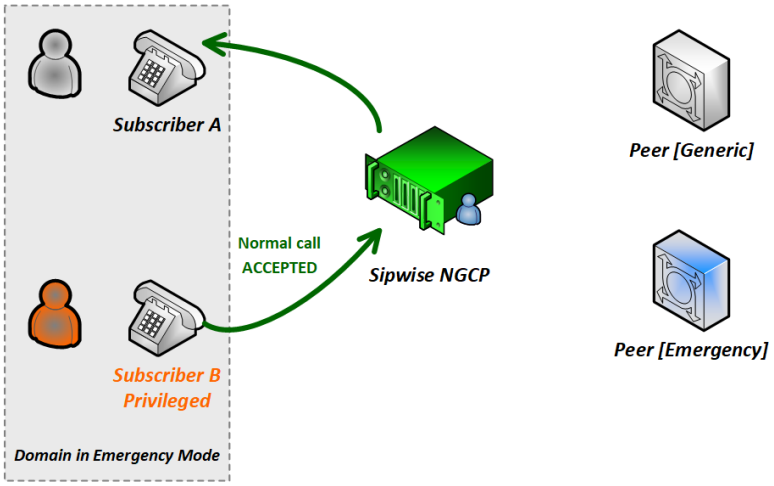

Figure 64. Call-flow in Emergency Mode 5. (Priv to Std)

6. A **privileged** subscriber makes a call **to an external subscriber (via peer)**. Result: the call will be **accepted**.

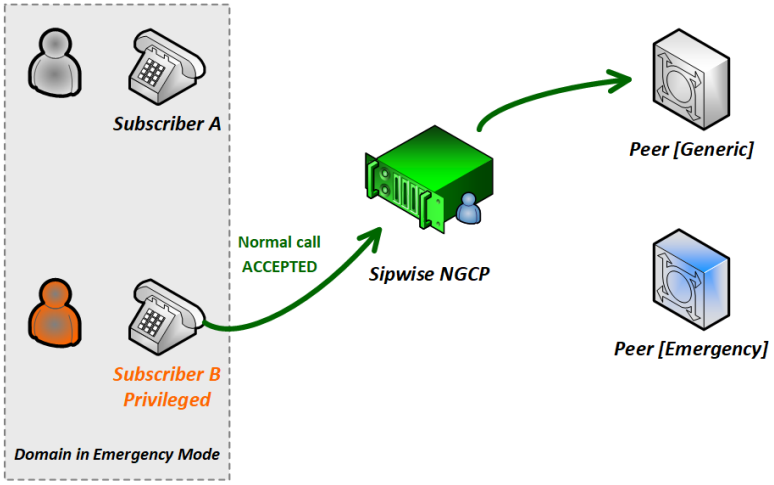

Figure 65. Call-flow in Emergency Mode 6. (Priv To Peer)

7.8.2. Configuration of Emergency Mode

The platform operator has to perform 2 steps of configuration so that the emergency mode can be activated. After the configuration is completed it is necessary to explicitly activate emergency mode, which can be accomplished as described in Activating Emergency Mode later.

1. System-level Configuration

The emergency priorization function must be enabled for the whole system, otherwise emergency mode cannot be activated. The platform operator has to set `kamailio.proxy.emergency_priorization.enabled` configuration parameter value to `"yes"` in the main configuration file `/etc/ngcp-config/config.yml`. Afterwards changes have to be applied in the usual way, with the command: `ngcpcfg apply "Enabled emergency priorization"`

In order to learn about other parameters related to emergency priorization please refer to kamailio part of the handbook.

 Copyright © Sipwise GmbH - All rights reserved

2. Subscriber-level Configuration

The platform operator (or any administrator user) has the capability to declare a subscriber privileged, so that the subscriber can initiate and receive calls when emergency mode has been activated on the NGCP. In order to do that the administrator has to navigate to *Settings* *Subscribers* *select the subscriber* *Preferences* *Internals* *emergency_priorization* on the **administrative web interface**, tick the preference.

The same privilege can be added via the **REST API** for a subscriber: a HTTP PUT/PATCH request must be sent on `/api/subscriberpreferences/id` resource and the `emergency_priorization` property must be set to `"true"`.

7.8.3. Activating Emergency Mode

The platform operator can activate emergency mode for a single or multiple domains in 3 different ways:

- via the administrative web interface
- via the REST API
- via a command-line tool

IMPORTANT	The interruption of ongoing calls is only possible with the command-line tool! Activating emergency mode for domains via the web interface or REST API will only affect upcoming calls.

1. Activate emergency mode via web interface: this way of activation is more appropriate if only a single (or just a few) domain is affected. Please navigate to *Settings* *Domains* *select a domain* *Preferences* *Internals* *emergency_mode_enabled* and tick the preference.

2. Activate emergency mode via REST API: this way of activation is more appropriate if only a single (or just a few) domain is affected.

For that purpose a HTTP PUT/PATCH request must be sent on `/api/domainpreferences/id` resource and the `emergency_mode_enabled` property must be set to `"true"`.

3. Activate emergency mode using a command-line tool: Sipwise C5 provides a built-in script that may be used to enable/disable emergency mode for some particular or all domains.

- Enable emergency mode:

```
> ngcp-emergency-mode enable <all|[domain1 domain2 ...]>
```

- Disable emergency mode:

```
> ngcp-emergency-mode disable <all|[domain1 domain2 ...]>
```

- Query the status of emergency mode:

```
> ngcp-emergency-mode status <all|[domain1 domain2 ...]>
```

Copyright © Sipwise GmbH - All rights reserved

7.9. Speed Dials

Speed dials are short codes, prefixed by an '*' (asterisk), that map a destination number. A list of speed dials can be configured on subscribers or customers level. In both cases, when a subscriber wants to call the destination associated to the speed dial code, they could simply dial *X or *XXX (in case of customer's speed dial).

7.9.1. Subscriber's Speed Dials

Subscriber's speed dials allow the user to configure a maximum of ten codes (based on 1-digit). To configure the subscriber's speed dials go to *SettingsSubscribers select a subscriberDetailsSpeed Dial* and click on *Edit*. The required parameters are:

- **Slot**: choose the slot to assign to the speed dial. The list of pre-defined codes goes from *0 till *9
- **Destination**: the number or uri to use as speed dial destination

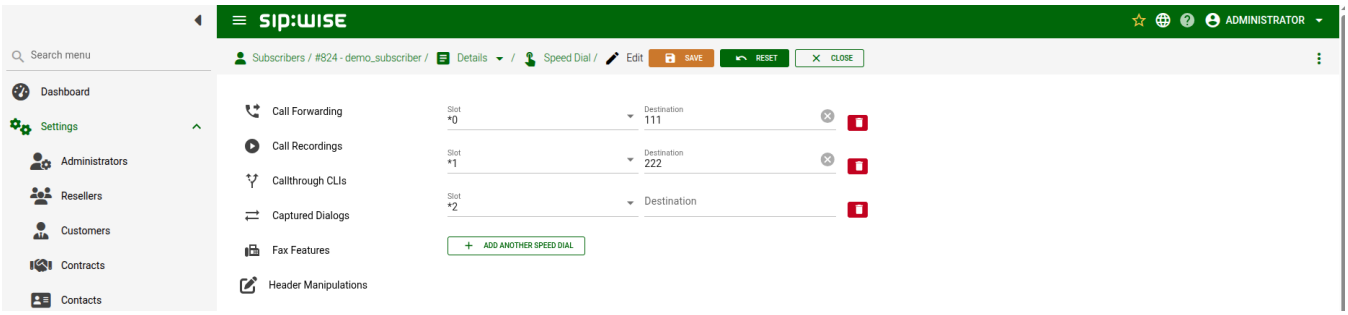

Figure 66. Set Subscriber's Speed dials

7.9.2. Customer's Speed Dials

A similar configuration can be done on customer level too. Customer's Speed dials allow to configure a maximum of one thousand codes (based on 3-digits), from *000 till *999. Each of the customer's speed dials will be visible and usable by all the subscribers belonging to that customer. To configure the customer's speed dials go to *SettingsCustomers select a customerDetailsSpeed Dial* and click on *Edit*. The required parameters are:

- **Slot**: choose the slot to assign to the speed dial. The list of pre-defined codes goes from *000 till *999
- **Destination**: the number or uri to use as speed dial destination

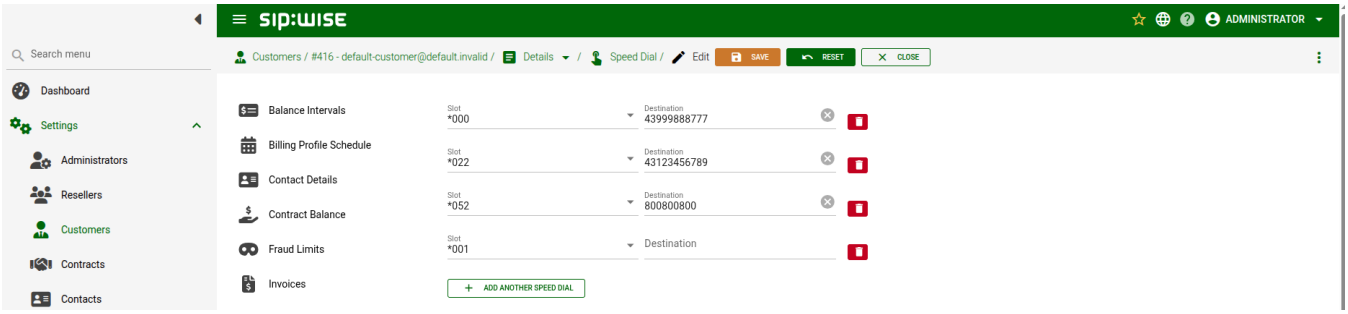

Figure 67. Set Customer's Speed dials

Copyright © Sipwise GmbH - All rights reserved

7.10. SIP Message Filtering

7.10.1. Header Filtering

Adding additional SIP headers to the initial INVITEs relayed to the callee (second leg) is possible by creating a patchtt file for the following template:

`/etc/ngcp-config/templates/etc/ngcp-sems/etc/ngcp.sbcprofile.conf.tt2`.

The following section can be changed:

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

By default the system will remove from the second leg all the SIP headers which are not in the above list. If you want to keep some additional/custom SIP headers, coming from the first leg, into the second leg you need to add them at the end of the *header_list=* list. After that, as usual, you need to apply the changes. In this way the system will keep your headers in the INVITE sent to the destination subscriber/peer.

WARNING Refrain from making any adjustments to the list if you are not familiar with ALL the required procedures to do so.

7.10.2. Codec Filtering

At times, you may want to filter specific audio codecs from the SDP payload. For example, to prevent subscribers from using certain codecs or to require the use of a particular one. To achieve that you need to change the `/etc/ngcp-config/config.yml`, in the following section:

```
sdp_filter:
      codecs: PCMA,PCMU,telephone-event
      enable: yes
      mode: whitelist
```

In the example above, the system is removing all the audio CODECS from the initial INVITE except G711 a-law and μ-law 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
```

Copyright © Sipwise GmbH - All rights reserved

```
      mode: blacklist
```

In this way the G729 and G722 will be removed from the SDP payload. In order to apply the changes, run

```
ngcpcfg apply 'Enable CODEC filtering'
```

WARNING Codec filtering feature applies to SDP attributes and it does not remove a whole SDP media session in case all the related attributes match a filter rule. For example, this SDP video media session:

```
m=video 9078 RTP/AVPF 96 97
a=rtpmap:96 VP8/90000
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=42801F
```

in combination with:

```
sdp_filter:
      codecs: PCMA,PCMU,telephone-event
      enable: yes
      mode: whitelist
```

will result in no filtering at all because both codecs id 96 97 are not whitelisted and an empty media session is not allowed.

7.10.3. Codec Filtering with User Preferences

Codec filtering management is also possible using the following preferences:

- codecs_list
- codecs_filter
- codecs_id_filter
- codecs_id_list

These preferences offer an alternative way to filter codecs without updating the configuration file and restarting the `kamailio-proxy` service. Since they can be applied at domain/peer/subscriber preferences level, they feature a more granular management compared when configuration is set via config.yml file, which is system wide.

The "codecs_list" preference allows you to enter a list of codecs names, valid names are G722, PCMU, PCMA, speex, GSM, G723, DVI4, L16, QCELP, CN, MPA, G728, DVI4, G729, AMR, AMR*WB, opus, telephone-event, CelB, JPEG, H261, H263, H263*1998, MPV, MP2T, nv, vp8, vp9, h264.

Once the list is filled, you have to specify if the codecs are going to be blacklisted or whitelisted by `kamailio-proxy`.This is done through the `codecs_filter` preference.

 Copyright © Sipwise GmbH - All rights reserved

By default, the codecs filter preferences are cleared (blacklisted) causing all codecs listed inside the codecs lists preferences to be filtered out. Differently, if the codecs filters are set, only the ones inside the codecs list will be retained causing the others to be filtered out.

Examples:

- Example 1
- codecs_list: "G729"
- codecs_filter: unset (false)
- SDP IN:

```
m=audio 6000 RTP/AVP 8 0 18 101
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:18 G729/8000
a=rtpmap:101 telephone-event/8000
```

- SDP OUT:

```
m=audio 30000 RTP/AVP 8 0 101
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
```

- Codec G729 has been filtered out because blacklisted in codecs_list.
- Example 2
- codecs_list: "G729"
- codecs_filter: set (true)
- SDP IN:

```
m=audio 6000 RTP/AVP 8 0 18 101
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:18 G729/8000
a=rtpmap:101 telephone-event/8000
```

- SDP OUT:

```
m=audio 30108 RTP/AVP 18
a=rtpmap:18 G729/8000
```

- Codec G729 is retained because whitelisted, PCMA and PCMU have been filtered out because they are not whitelisted.

Copyright © Sipwise GmbH - All rights reserved

7.10.4. Enable History and Diversion Headers

It may be useful and mandatory - specially with NGN interconnection - to enable SIP History header and/or Diversion header for outbound requests to a peer or even for on-net calls. In order to do so, you should enable the following preferences in Domain's and Peer's Preferences:

- Domain's Preferences: *inbound_uprn* = **Forwarder's NPN**
- Peer's Preferences: *outbound_history_info* = **UPRN**
- Peer's Preferences: *outbound_diversion* = **UPRN**
- Domain's Preferences: *outbound_history_info* = **UPRN** (if you want to allow History Header for on-net call as well)
- Domain's Preferences: *outbound_diversion* = **UPRN** (if you want to allow Diversion Header for on-net call as well)

7.10.5. User Agent Filtering

It could be useful to filter the received REGISTER and INVITE requests based on the User-Agent header's value. For example, if you want to force your subscribers to use certain type/model of devices. To do that system wide you need to change the `/etc/ngcp-config/config.yml`, in the following section:

```
kamailio:
  lb:
    block_useragents:
      action: reject
      block_absent: no
      block_empty: no
      enable: yes
      mode: whitelist
      ua_patterns:
      - Yealink.*
```

In the example above, the system allows all the requests, which have the User-Agent header beginning with 'Yealink' value. All other UACs will be rejected with the *403 Forbidden* message. To silently drop the received message, it is possible to specify the *drop* action instead of the default *reject*.

Another example is the `blacklist` mode:

```
kamailio:
  lb:
    block_useragents:
      action: drop
      block_absent: no
      block_empty: no
      enable: yes
      mode: blacklist
      ua_patterns:
      - friendly-scanner
```

 Copyright © Sipwise GmbH - All rights reserved

In the example above the Sipwise C5 drops all requests, which have the User-Agent header equal to 'friendly-scanner'. Since there is the *drop* action, these messages will be dropped silently, without providing a response to the sender.

Another example with *block_empty:* and *block_absent:* options set to 'yes':

```
kamailio:
  lb:
    block_useragents:
      action: reject
      block_absent: yes
      block_empty: yes
      enable: yes
      mode: whitelist
      ua_patterns: []
```

In the example above the Sipwise C5 rejects all requests, which have the User-Agent header absent or the User-Agent header with no value. Such requests will be rejected with a *403 Forbidden* message. It is possible to set only one of the options (*block_empty: / block_absent:*) to 'yes'. As well as it is possible to keep both of them enabled.

In order to apply the changes, run the following:

```
ngcpcfg apply 'Enable User-Agent filtering'
```

IMPORTANT Please note that implementing these changes will require a restart of the LB component. It is advisable to schedule this only during non-working hours.

Regardless of the system-wide configuration (UA filtering enabled globally or not), it is possible to define a specific User-Agent filtering for each Domain or Subscriber. In order to do so, you should configure the following fields in Domain's or Subscriber's Preferences section:

- ua_filter_list: Contains wildcard list of allowed or denied SIP User-Agents matched against the User-Agent header.
- ua_filter_mode: Specifies the operational mode of the SIP User-Agent Filter List: Blacklist or Whitelist.
- ua_reject_missing: Rejects any request if no User-Agent header is given.

In case of rejection the response with code `kamailio.proxy.early_rejects.block_admin.announce_code` and reason `kamailio.proxy.early_rejects.block_admin.announce_reason` will be sent back to the subscriber.

7.11. SIP Trunking with SIPconnect

7.11.1. User Provisioning

For the purpose of external SIP-PBX interconnect with the Sipwise C5, the platform admin should

Copyright © Sipwise GmbH - All rights reserved

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 the leading plus.

To configure the Subscriber, go to *SettingsSubscribers*, click the three dotted icon and click *Preferences* on the row of your subscriber.

You should look into the *Number Manipulations* and *Access Restrictions* sections in particular, which control the calling and called number presentation.

7.11.2. Inbound Calls Routing

Enable preference *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).

Fallback Behaviours

The Sipwise C5 can enhance pbx integration providing four different extended dialing modes that can be selected using *extended_dialing_mode* preference:

- Strict number matching: dialing arbitrary extension behind subscriber number is not allowed
- Extended matching, send dialed number with extension: the system will locate the subscriber by longest matching prefix, only the whole dialled number (base number + extension) will be written inside the Request URI user part.
- Extended matching, send base matching number: the system will locate the subscriber by longest matching prefix, only the base number or subscriber username will be written inside the Request URI user part.
- Extended matching, send primary number with extension: the system will locate the subscriber by longest matching prefix, the new Request URI user part will be equal to the primary number with appended the extension (calculated using the exceeding part of the callee).

Fallback procedure is active by default and usually applies to extended matching or e164_to_ruri modes. If the callee UAC replies with a 404 not found error, the fallback feature triggers the generation of a new INVITE. In this case the new Request URI will be set accordingly to the *extended_dialing_mode* and *e164_to_ruri* values.

The whole 404 fallback procedure can, however, be disabled activating the *no_404_fallback* option under subscriber's or domain's preferences.

7.11.3. Number Manipulations

The following sections describe the recommended configuration for correct call routing and CLI presentation according to the SIPConnect 2.0 recommendation.

Rewrite Rules

The SIP PBX by default inherits the domain dialplan which usually has rewrite rules applied to normal Class 5 subscribers with inbound rewrite rules normalizing the dialed number to the E.164 standard. If

 Copyright © Sipwise GmbH - All rights reserved

most users of this domain are Class 5 subscribers the dialplan may supply calling number in national format - see Configuring Rewrite Rule Sets. While the SIP-PBX trunk configuration can be sometimes amended it is a good idea in sense of SIPconnect recommendation to send only the global E.164 numbers.

Moreover, in mixed environments with the Sipwise C5 Cloud PBX sharing the same domain with SIP trunking (SIP-PBX) customers, the subscribers may have different rewrite rules sets assigned to them. The difference is caused by the fact that the dialplan for Cloud PBX is fundamentally different from the dialplan for SIP trunks due to extension dialing, where the Cloud PBX subscribers use the break-out code 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`

Copyright © Sipwise GmbH - All rights reserved

- Field: `Callee`

Assign the aforementioned Rewrite Rule Set to the SIP-PBX subscribers.

WARNING Outbound Rewrite Rules for Callee shall NOT be applied to the calls to normal SIP UAs like IP phones since the number with plus does not correspond to their SIP username.

User Parameter

The following configuration is needed for your platform to populate the From and To headers and Request-URI of the INVITE request with "user=phone" parameter as per RFC 3261 Section 19.1.1 (if the user part of the URI contains telephone number formatted as a telephone-subscriber).

- Domain's Preferences: *outbound_from_user_is_phone* = Y
- Domain's Preferences: *outbound_to_user_is_phone* = Y

Forwarding Number

The following is our common configuration that covers the calling number presentation in a variety of use-cases, including the incoming calls, on-net calls and Call Forward by the platform:

- Domain's Preferences: *inbound_uprn* = **Forwarder's NPN**
- Domain's Preferences: *outbound_from_user* = **UPRN (if set) or User-Provided Number**
- Domain's Preferences: *outbound_pai_user* = **UPRN (if set) or Network-Provided Number**
- Domain's Preferences: *outbound_history_info* = **UPRN** (if the called user expects History-Info header)
- Domain's Preferences: *outbound_diversion* = **UPRN** (if the called user expects Diversion header)
- Domain's Preferences: *outbound_to_user* = **Original (Forwarding) called user** if the callee expects the number of the subscriber forwarding the call, otherwise leave default.

The above parameters can be tuned to operator specifics as required. You can override these settings in the Subscriber Preferences if particular subscribers need special settings.

TIP On outgoing call from SIP-PBX subscriber the Network-Provided Number (NPN) is set to the *cli* preference prefilled with main E.164 number. In order to have the full alias number as NPN on outgoing call set preference *extension_in_npn* = Y.

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): the Sipwise C5 platform extends the History-Info header received from the PBX by adding another level of indexing according to the specification RFC 7044.

Allowed CLIs

- For correct calling number presentation on outgoing calls, you should include the pattern matching

 Copyright © Sipwise GmbH - All rights reserved

all the alias numbers of SIP-PBX or each individual alias number under the *allowed_clis* preference.

- If the signalling calling number (usually taken from From user-part, see *inbound_upn* preferences) does not match the *allowed_clis* pattern, the *user_cli* or *cli* preference (Network-Provided Number) will be used for calling number presentation.

7.11.4. Registration

SIP-PBX can use either Static or Registration Mode. While SIPConnect 2.0 support for TLS is a MUST, one should note that using TLS for signaling does not require the use of the SIP URIs scheme. SIP URIs scheme is obsolete for this purpose.

Static Mode

While SIPConnect 2.0 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, three dotted icon, *DetailsRegistered DevicesAdd* and put the address of the SIP-PBX in the following format: sip:username@ipaddress:5060 where username=username portion of SIP URI and ipaddress = IP address of the device.

Registration Mode

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

IMPORTANT The use of RFC 6140 style "bulk number registration" is discouraged. The SIP-PBX should register one AOR with email-style SIP URI. The Sipwise C5 will take care of routing the aliases to the AOR with *e164_to_ruri* preference.

Trusted Sources

If a SIP-PBX cannot perform the digest authentication, you can authenticate it by its source IP address in the Sipwise C5. To configure the IP-based authentication, go to *SettingsSubscribersDetailsTrusted Sources* and specify the *Protocol* and the *IP address* of the SIP-PBX.

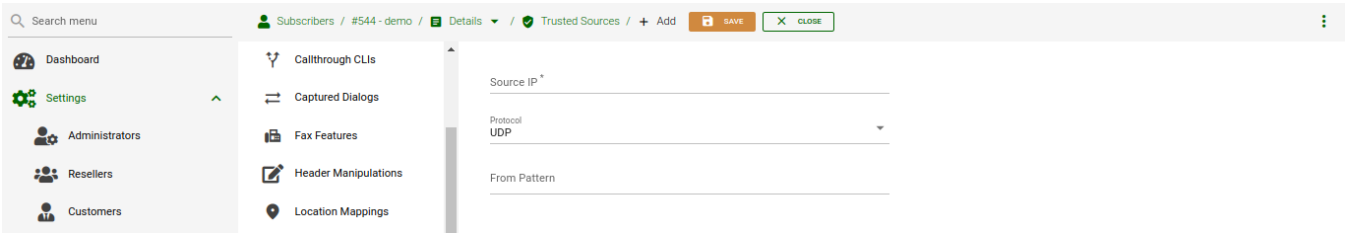

Figure 68. Add Trusted Sources

To authenticate multiple subscribers from the same IP address, use the *From Pattern* field to distinguish these subscribers, it requires a regular expression that matches the From URI.

The syntax for regular expression is POSIX Basic Regular Expressions (BRE) with GNU extensions. In BRE, the characters `+`, `?`, `|`, `{`, `}`, `(` and `)` are literal unless they are escaped with a backslash, e.g. use `\+` for "one or more" and `\|` for alternation.

When this feature is configured for a subscriber, the Sipwise C5 authenticates all calls that arrive from

Copyright © Sipwise GmbH - All rights reserved

the specified IP address without challenging them.

IMPORTANT If the same IP address and the From Pattern field are mistakenly specified as trusted for different subscribers, the Sipwise C5 will not know which subscriber to charge for the call and will randomly select one.

7.12. Trusted Subscribers

In some cases, when you have a device that cannot authenticate itself against Sipwise C5, you may need to create a *Trusted Subscriber*. Trusted Subscribers use IP-based authentication and they have a Permanent SIP Registration URI in order to receive messages from the Sipwise C5.

In order to make a regular subscriber trusted, perform the following extra steps:

- Create a permanent registration via *SettingsSubscribersDetails Registered Devices*.
- Add the IP address of the device as Trusted Source in your subscriber: SettingsSubscribersDetailsTrusted Sources__.

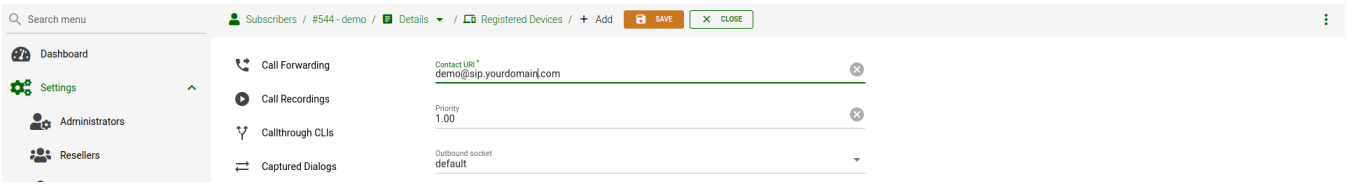

Figure 69. Add Subscriber Registered Devices

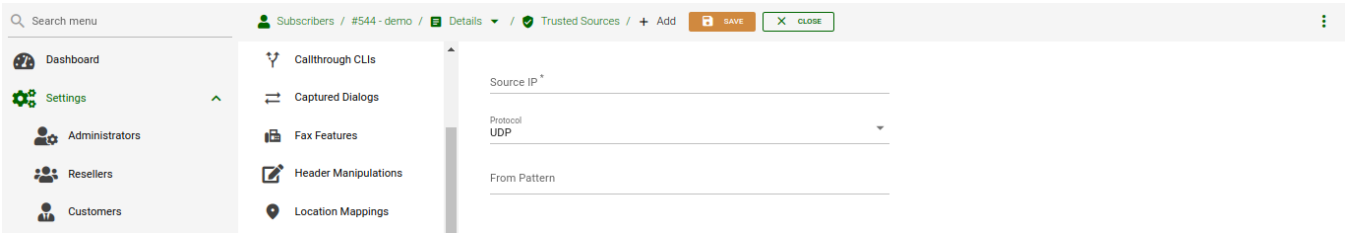

Figure 70. Add Trusted Sources

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

7.13. Peer Probing

The basic way of selecting the appropriate peering server, where an outbound call can be routed to, has already been described in Routing Order Selection of the handbook.

7.13.1. Introduction to Peer Probing Feature

The Sipwise C5 provides a web admin panel and API capabilities to configure peering servers in order to terminate calls to non-local subscribers. Those peering servers may become *temporarily unavailable* due to overloading or networking issues. The Sipwise C5 will fail over to another peering server (matching the corresponding peering rules) after a timeout configured at system level (see the `b2b.sbc.outbound_timeout` configuration parameter; 6 sec by default), if no provisional response (a

 Copyright © Sipwise GmbH - All rights reserved

response with a code in the range of 100 to 199) is received for the outbound INVITE request.

Even if this timer is set much lower, like 3 sec, the call setup time is increased significantly. This is even more true if multiple peering servers fail at the same time, which will sum up the individual timeouts, finally *causing call setup times to reach the order of tens of seconds*.

To optimize the call setup time in such scenarios, a new feature is implemented to *continuously probe peering servers* via SIP messages, and mark them as unavailable on timeout or when receiving unexpected response codes. Appropriate SIP response codes from the peering servers will mark them as available again.

Peering servers *marked as unavailable* are then *skipped during call routing* in the peering selection process, which significantly shortens the call setup times if peering servers fail.

7.13.2. Configuration of Peer Probing

The system administrator has to configure the peer probing feature in 2 steps:

1. System-level configuration enables the peer probing feature in general on the Sipwise C5 and determines the operational parameters, such as timeouts, the SIP method used for probing requests, etc.
2. Peering server configuration will add / remove a peering server to the list of probed endpoints.

System-Level Configuration

The parameters of peer probing are found in the main system configuration file `/etc/ngcp-config/config.yml`. You can see the complete list of configuration parameters in kamailio of the handbook, while the most significant ones are discussed here.

Enabling peer probing system-wide happens through the `kamailio.proxy.peer_probe.enable` parameter. If it is set to 'yes' (which is the default value), the Sipwise C5 will consider probing of individual peering servers based on their settings.

Timeout of a single probing request can be defined through `kamailio.proxy.peer_probe.timeout` parameter. This is a value interpreted as seconds while Sipwise C5 will wait for a SIP response from the peering server. Default is 5 seconds.

The **probing interval** can be set through the `kamailio.proxy.peer_probe.interval` parameter. This is the time period in seconds that determines how often a probing request is sent to the peering servers. Default is 10 seconds.

The **SIP method** used for probing requests can be defined through `kamailio.proxy.peer_probe.method` parameter. Allowed values are: OPTIONS (default) and INFO.

TIP The system administrator, in most of the cases, will not need to modify the default configuration values other than that of timeout and interval.

If no available peering server is found, the call is rejected with the response code and reason configured in `kamailio.proxy.early_rejects.peering_unavailable.announce_code` and `kamailio.proxy.early_rejects.peering_unavailable.announce_reason`. If a sound file is configured within the *system sound set* assigned to the calling party, an announcement is played as early media before the rejection.

Copyright © Sipwise GmbH - All rights reserved

Individual Peering Server Configuration

When the peer probing feature is enabled on system-level, it is possible to add each individual peering server to the list of probed endpoints. To enable the probing of a peering server via the admin web interface:

1. Open the properties panel of a peering server: *Peerings select a peering group Details select a peering server Edit*
2. Tick the checkbox *Enable Probing*
3. *Save* changes

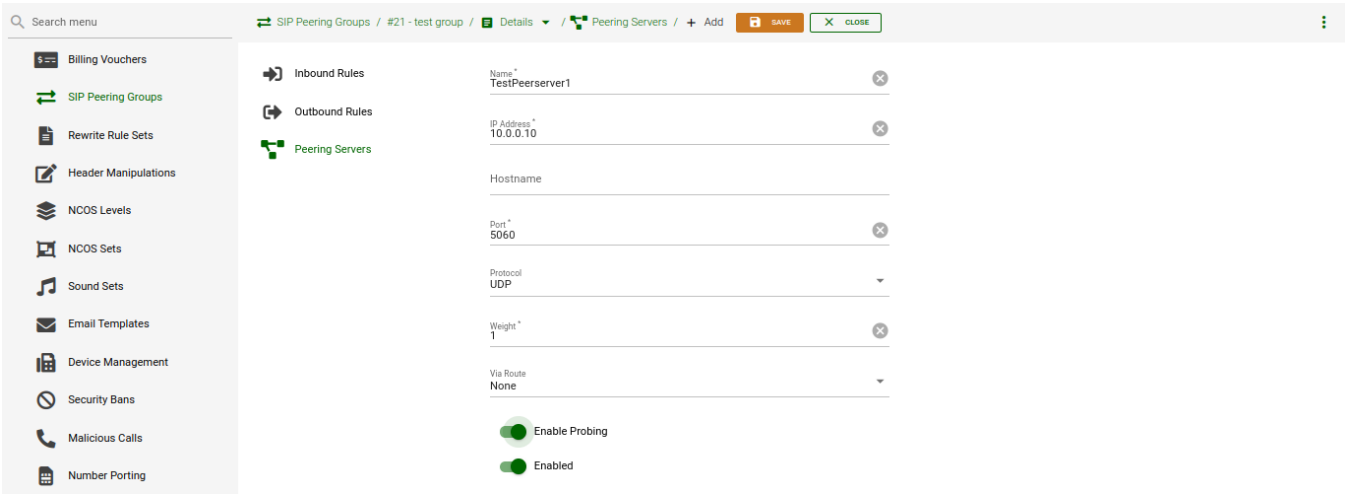

Figure 71. Enable Probing of Peering Server

7.13.3. Monitoring of Peer Probing

Peering server states, such as "reachable" / "unreachable", are continuously stored in a time-series database (Prometheus compatible database) by the Sipwise C5 Proxy nodes. It is possible to **graphically represent the state of peering servers** on NGCP's admin web interface, just like other system variables (like CPU and memory usage, number of registered subscribers, etc.). However this is not yet available by default.

7.13.4. Further Details for Advanced Users

TIP This subchapter of the handbook is targeted on advanced system operators and Sipwise engineers. Reading this section is not essential for effectively managing the peer probing feature of the NGCP.

Behaviour of Kamailio Proxy Instances

Each *kamailio-proxy* instance on the proxy nodes performs the probing individually for performance reasons. Each proxy holds its result in its cache to avoid central storage and replication of the probing results.

Each peering server is cross-checked against the hash table filled during outbound probing requests and is skipped by call routing logic, if a match is found.

On start or restart of the *kamailio-proxy* instance, the probing will start after the first interval, and NOT

 Copyright © Sipwise GmbH - All rights reserved

immediately after start. In the first probing interval the proxy will always try to send call traffic to peering servers until the first probing round is finished, and will only then start to skip unavailable peering servers.

Database Changes

A new DB column: `provisioning.voip_peer_hosts.probe` with type TINYINT(1) (boolean) is added to the DB schema.

A peer status change will populate the `kamailio.dispatcher` table, inserting the SIP URI in format `sip:$ip:$port;transport=$transport` in dispatcher group 100, which defines the probing group for peering servers.

Also the `kamailio.dispatcher.attrs` column is populated with a parameter `peerid=$id`. This ID is used during probing to load the peer preferences: `outbound_socket` and `lbrtp_set`, that are required to properly route the probing request.

7.14. Voicemail System

7.14.1. Accessing the IVR Menu

There are three ways to access the voicebox, so that the subscriber can manage their voicebox via the IVR:

1. 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.
2. Call the URI `voiceboxpass@yourdomain` from any number, causing the system to prompt for a mailbox and the PIN.
3. Call the URI `voiceboxpin@yourdomain` where only the password/PIN will be requested. The PIN can be set in the *SettingsSubscribersDetailsVoicemail Settings* page.

Mapping Numbers and Codes to IVR Access

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

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

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

External IVR Access

When reaching `voiceboxpass`, the subscriber is prompted for their 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,

Copyright © Sipwise GmbH - All rights reserved

for example `4912345` if they got assigned a German number.

While reaching for `voiceboxpin`, the subscriber is only prompted for the PIN.

You can globally configure a rewrite rule in `config.yml` using `asterisk.voicemail.normalize_match` and `asterisk.voicemail.normalize_replace`, allowing you to customize the format a subscriber can enter, e.g. having `^0([1-9][0-9]+)$` as match part and `49$1` as replace part to accept German national format.

7.14.2. 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

 Copyright © Sipwise GmbH - All rights reserved

'2' Listen to it

'3' Rerecord it

'2' Record your busy message

'1' accept it

'2' Listen to it

'3' Rerecord it

'3' Record your name

'1' accept it

'2' Listen to it

'3' Rerecord it

'4' Record your temporary greetings

'1' accept it / or re-record if one already exist

'2' Listen to it / or delete if one already exist

'3' Rerecord it

'5' Change your password

'*' To return to the main menu

- '*' Help
- '#' Exit

7.14.3. Type of Messages

A message/greeting is a short message that plays before the caller is allowed to record a message. The message is intended to let the caller know that you are not able to answer their call. It can also be used to convey other information like when you will be available, other methods to contact you, or other options that the caller can use to receive assistance.

The IVR menu has three types of greetings.

Unavailable Message

The standard voice mail greeting is the "unavailable" greeting. This is used if you do not answer the phone and so the call is directed to your voice mailbox.

- You can record a custom unavailable greeting.
- If you have not recorded your unavailable greeting but have recorded your name, the system will play a generic message like: "`Recorded name` is unavailable."
- If you have not recorded your unavailable greeting, the phone system will play a generic message like: "`Digits-of-number-dialed` is unavailable".

Busy Message

If you wish, you can record a custom greeting used when someone calls you and you are currently on the phone. This is called your "Busy" greeting.

Copyright © Sipwise GmbH - All rights reserved

- You can record a custom busy greeting.
- If you have not recorded your busy greeting but have recorded your name, the phone system will play a generic message: "`Recorded name` is busy."
- If you have not recorded your busy greeting and have not recorded your name (see below), the phone system will play a generic message: "`Digits-of-number-dialed` is busy."

Temporary Greeting

You can also record a temporary greeting. If it exists, a temporary greeting will always be played instead of your "busy" or "unavailable" greetings. This could be used, for example, if you are going on vacation or will be out of the office for a while and want to inform people not to expect a return call anytime soon. Using a temporary greeting avoids having to change your normal unavailable greeting when you leave and when you come back.

7.14.4. Folders

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

The Default Folder List

- 0 - New Messages
- 1 - Old Messages
- 2 - Work Messages
- 3 - Family Messages
- 4 - Friends Messages

When a caller leaves a message for you, the system will put the message into the "New Messages" folder. If you listen to the message, but do not delete the message or save the message to a different folder, it will automatically move the message to the "Old Messages" folder. When you first log into your mailbox, the Voicemail System will make the "New Messages" folder the current folder if you have any new messages. If you do not have any new messages it will make the "Old Messages" folder the current folder.

7.14.5. Voicemail Languages Configuration

To add a new language or to change the pronunciation for an existing one, ensure that **mode=new** is defined in */etc/ngcp-config/templates/etc/asterisk/say.conf.tt2*. Adjust the configuration in the same file using the manual in the beginning. Remember not to edit directly the *.tt2* file, but create a new *.customtt.tt2* as described in the Configuration Framework chapter. Then, as usual, make the new configuration active.

7.14.6. Flowcharts with Voice Prompts

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

 Copyright © Sipwise GmbH - All rights reserved

Listening to New Messages

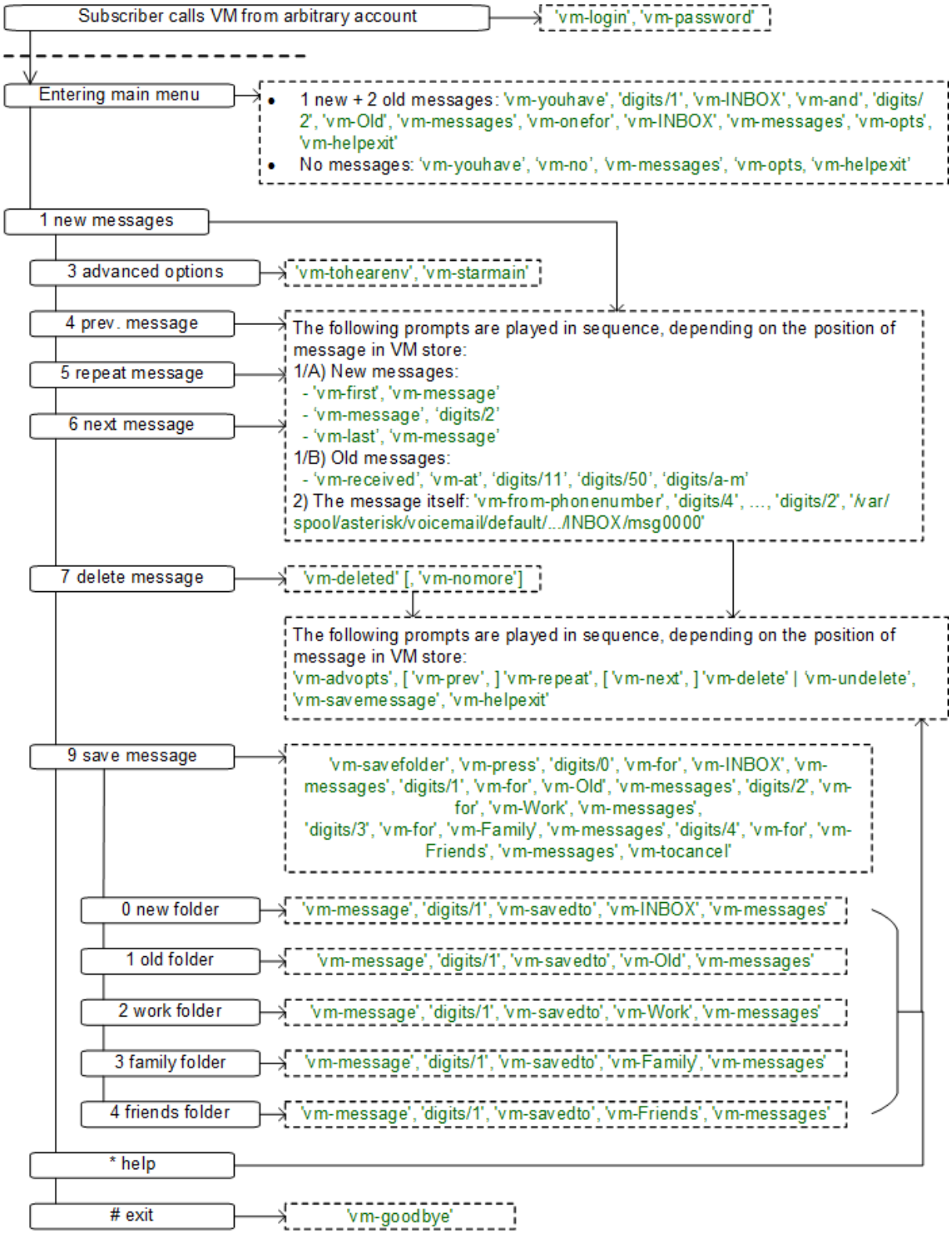

Figure 72. Flowchart of Listening to New Messages

Changing Voicemail Folders

Copyright © Sipwise GmbH - All rights reserved

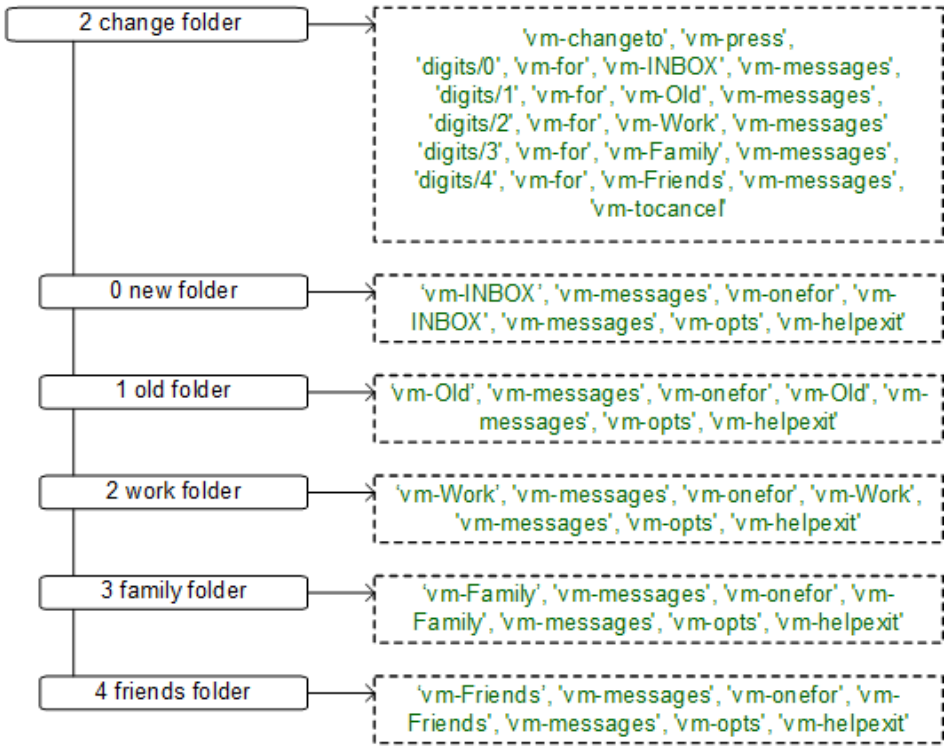

Figure 73. Flowchart of Changing Voicemail Folders

Mailbox Options

Copyright © Sipwise GmbH - All rights reserved

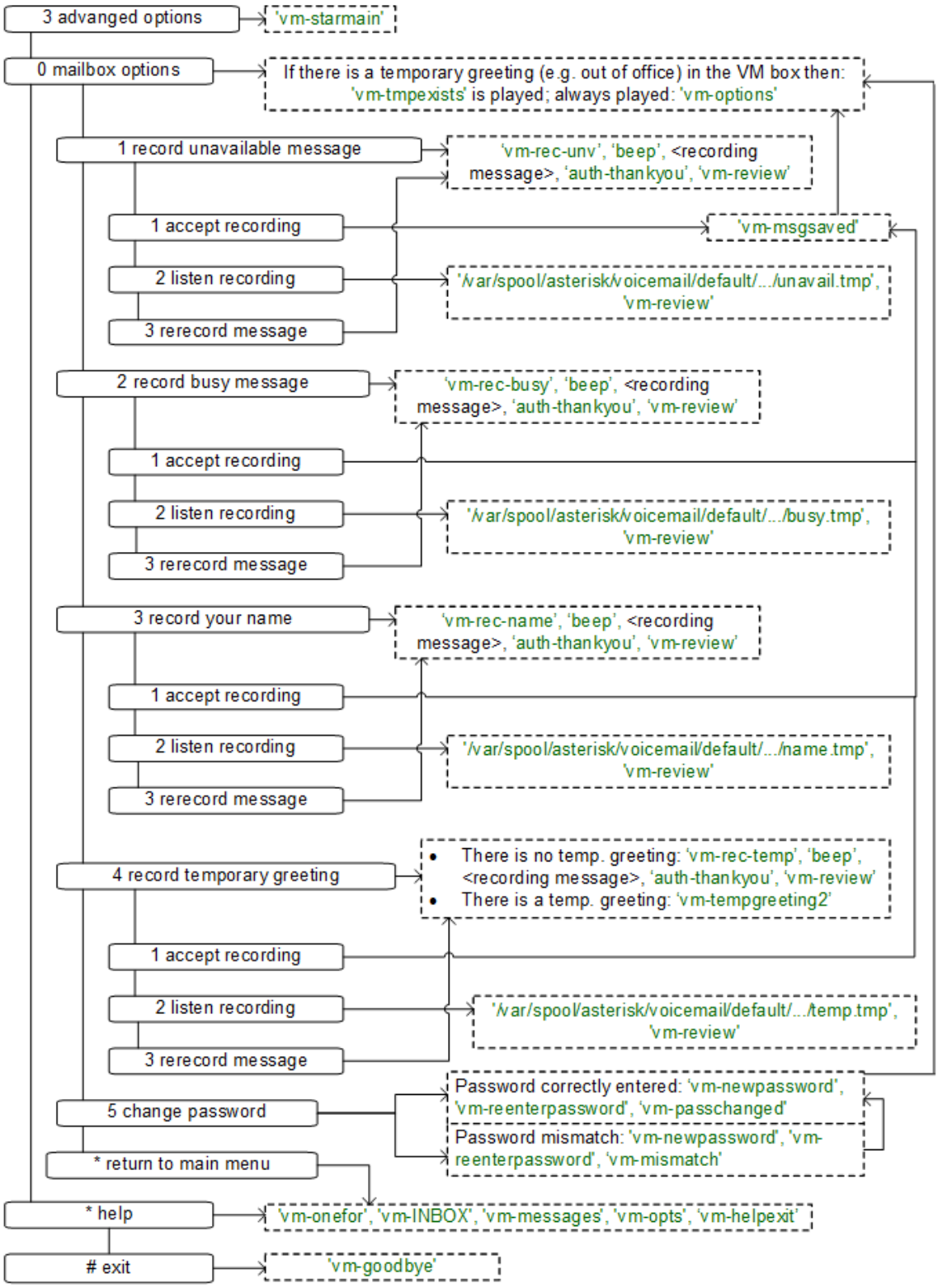

Figure 74. Flowchart of Changing Mailbox Options

Leaving a Message

Copyright © Sipwise GmbH - All rights reserved

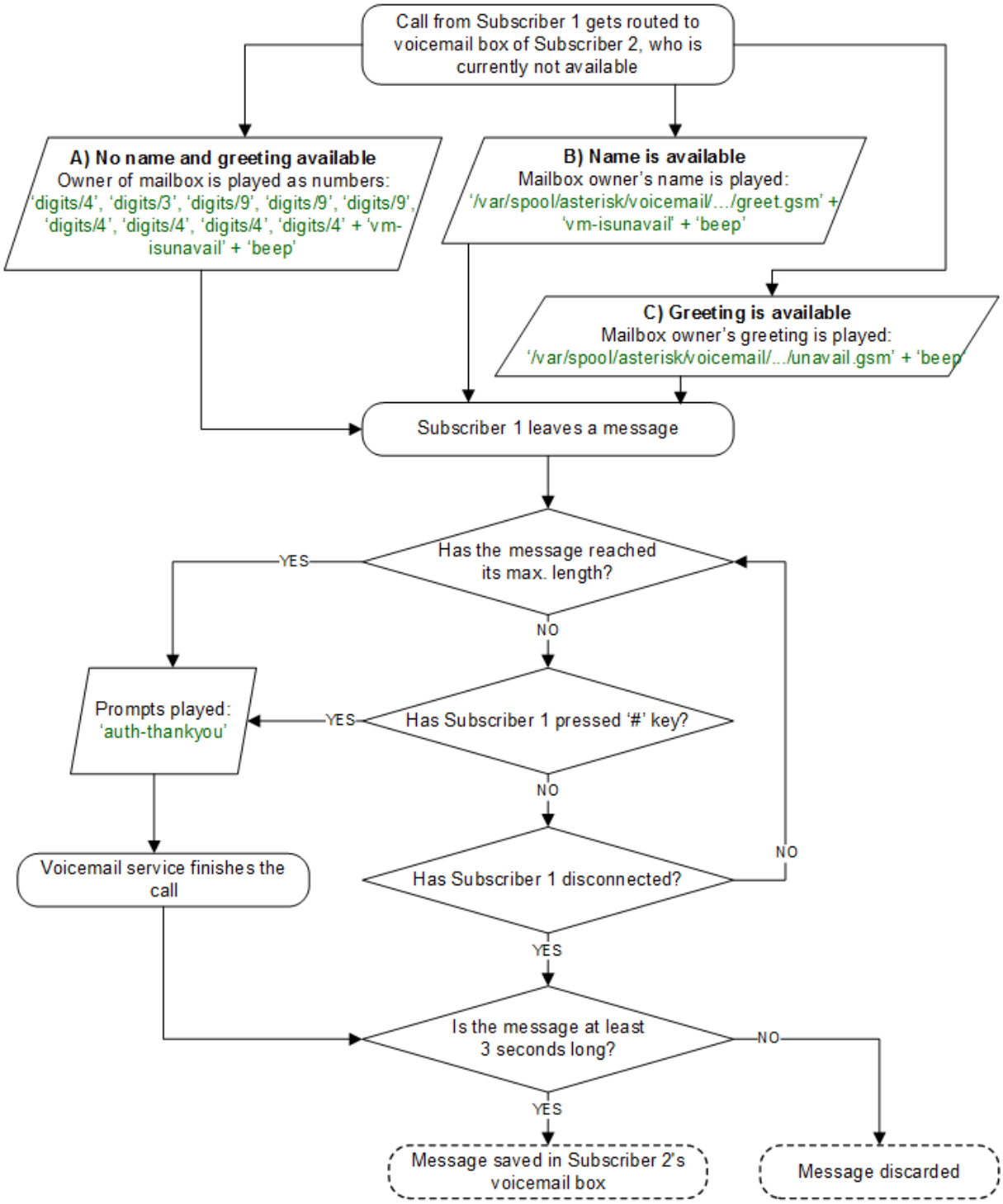

Figure 75. Flowchart of Leaving a Voice Message

7.15. Configuring Subscriber IVR and VSC Language

The language for the Voicemail system IVR or Vertical Service Codes (VSC) IVRs may be set using the subscriber or domain preference *language*.

 Copyright © Sipwise GmbH - All rights reserved

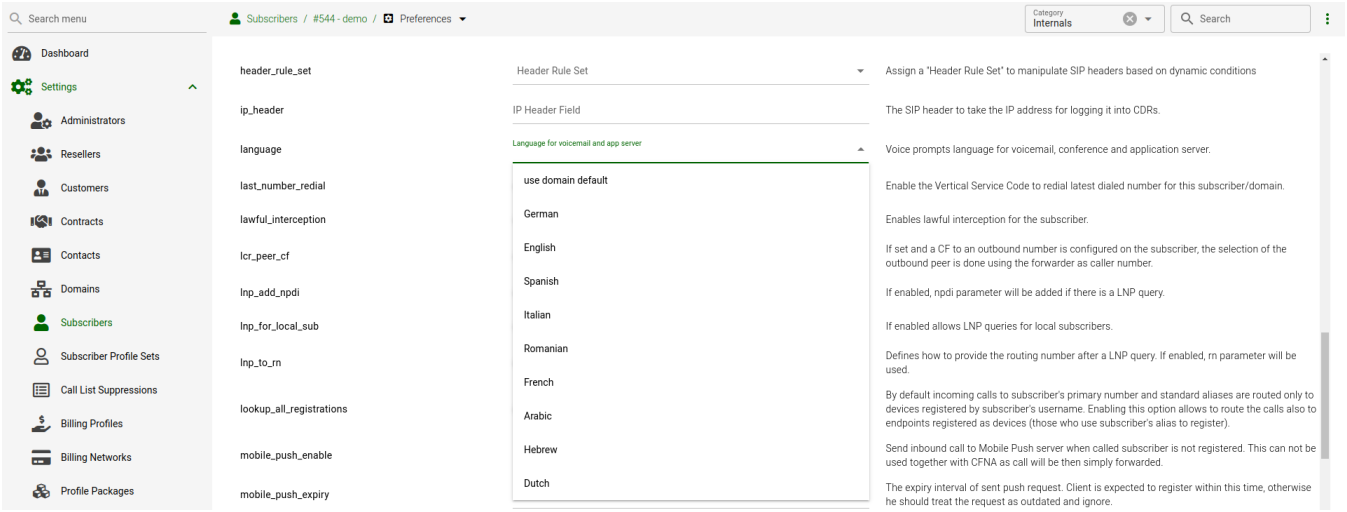

Figure 76. Set Subscriber IVR Language

The Sipwise C5 offers pre-installed prompts for voicemail in English, Spanish, French and Italian, to name a few, and pre-installed prompts for 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 the Sipwise C5 Panel and uploaded by the administrator in their language of choice.

7.16. Sound Sets

The Sipwise C5 provides the administrator with ability to upload the voice prompts such as conference prompts or call error announcements on the *Sound Sets page*.

There is a preference *sound_set* in the *NAT and Media Flow Control* section on Domain and Subscriber levels to link subscribers to the sound set that they should hear. Additionally Sound Sets can be configured on Peer level in order to play a dedicated early reject prompt when an incoming call does not match any local subscriber.

Sound Sets can be defined in *SettingsSound Sets*. To create a new Sound Set, click *Add*.

Copyright © Sipwise GmbH - All rights reserved

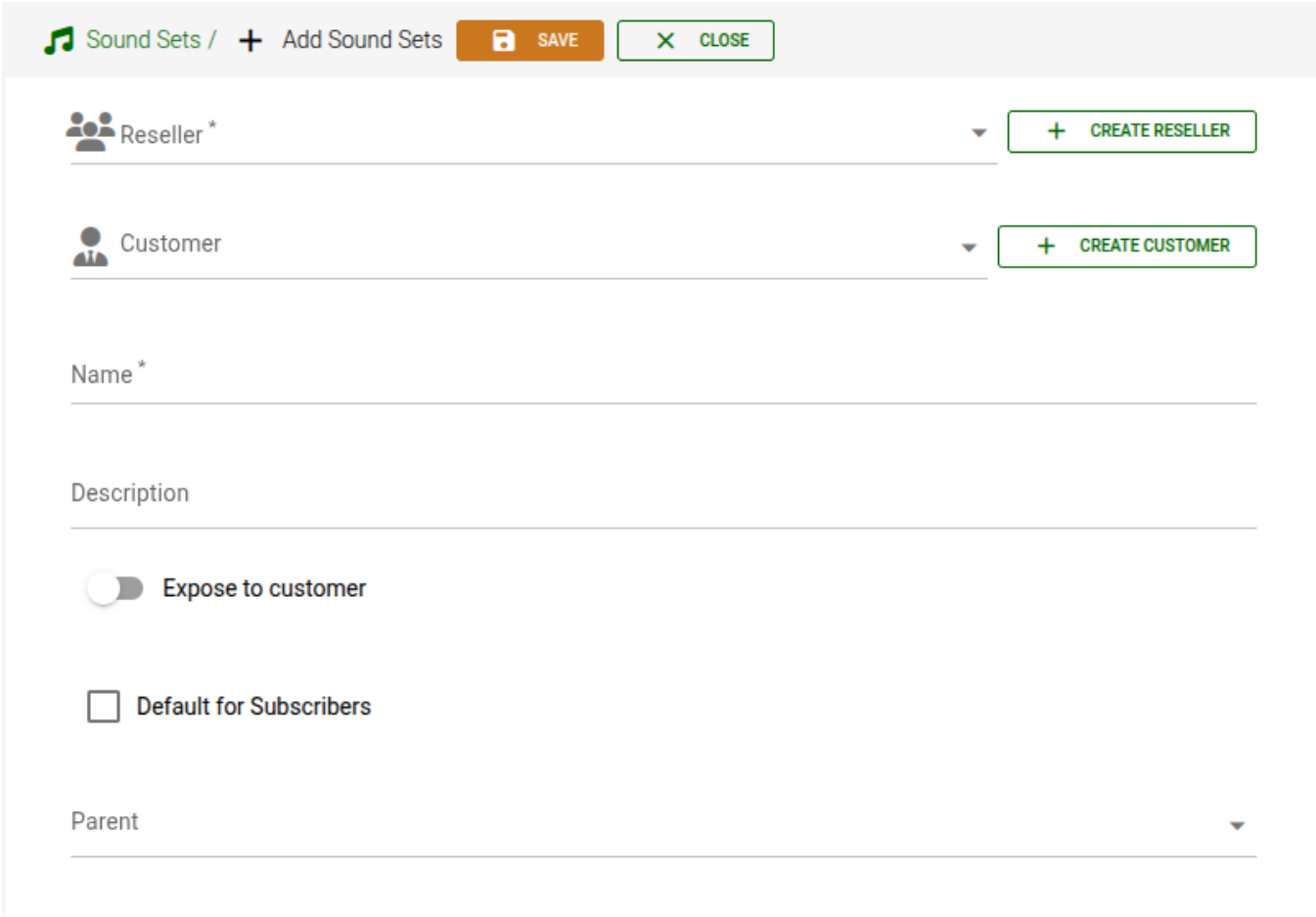

Figure 77. Sound Sets

NOTE As soon as the first audio files (for the sound sets) are uploaded, they will be stored using the SQL backend and will not be available in the filesystem (as usual files). But, as soon as these files are first-time played for a caller, they will be stored (cached) in the '/ngcp-data/cache' directory. This is done in order to reduce the time needed to retrieve them (on a routine basis). Anyway, this data will always be stored in the SQL backend as well for redundancy reasons. So your Sipwise C5 will always be able to retrieve them from SQL and cache them again if needed.

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

7.16.1. Sound_Set and Contract_Sound_Set Usage

Sound_Set and *Contract_Sound_Set* are used for different purposes:

- *Sound_Set* general soundset that can be assigned by system administrators at domain or subscriber level.
- *Contract_Sound_Set* is a customer-specific sound set. It can be assigned only to a PBX customer. It can be managed also by subscriber administrators through the CSC or API.

In cases where both types of sound set are assigned to a subscriber, the *Contract_Sound_Set* takes precedence.

 Copyright © Sipwise GmbH - All rights reserved

7.16.2. Parent and Sound Set Inheritance

Every *Sound_Set* can have a parent *Sound_Set*. If the parent is configured, all the announcements not defined in the current *Sound_Set* will be inherited from the parent, if any. The parent *Sound_Set* can also have a *Sound_Set* assigned as a parent, producing the "chained" behavior. The Sipwise C5 has an internal check to prevent the creation of parent inheritance loops. The *Sound_Set* can be used as a parent from a *Contract_Sound_Set* and vice versa.

If a *Sound_Set* has the *Expose to Customer* flag enabled, it will be visible for subscriber administrators in the CSC panel (and through API), but only in the read-only mode. This allows the subscriber administrators to:

- listen to the currently configured announcements without the possibility to change them
- create a new *Contract_Sound_Set* using the exposed *Sound_Set* as the parent

When a *Sound_Set* has a parent assigned, there is a new column *Use parent* shown on the UI that enables a choice to use this file (even if empty) or use the parent one (even if not empty). The following table describes the system behavior for different combinations of the *Parent*, *Use Parent* and file presence settings.

Table 12. Sound Sets Inheritance

SoundSet1 (no parent)	SoundSet2 (parent SoundSet1)	SoundSet3 (parent SoundSet2)	Announcement Played
file present	file present	file present	From Soundset3
file present	file present	file present + use parent	From Soundset2
file present	file present + use parent	file present + use parent	From Soundset1
no file	file present + use parent	file present + use parent	No announcement
file present	file present	no file + use parent	From Soundset2
file present	no file + use parent	no file + use parent	From Soundset1
file present	file present	no file	No announcement

When a parent *Sound_Set* is deleted, the system removes the parent configuration from all *Sound_Set* that used it as a parent. The same happens in case a *Sound_Set* is parent of a *Contract_Sound_Set* and the *Expose to Customer* flag is disabled afterwards.

7.16.3. Configuring Early Reject Sound Sets

The call error announcements are grouped under *Early Rejects* section. Click the three dots beside a soundset and select "files". Then unfold the *Early Rejects* section and click *Select File* next to the sound handles (Names) that you want to use. Choose a WAV file from your file system, and tick the Loopplay setting if you want to play the file in a loop instead of only once. Click Save to upload the file.

Copyright © Sipwise GmbH - All rights reserved

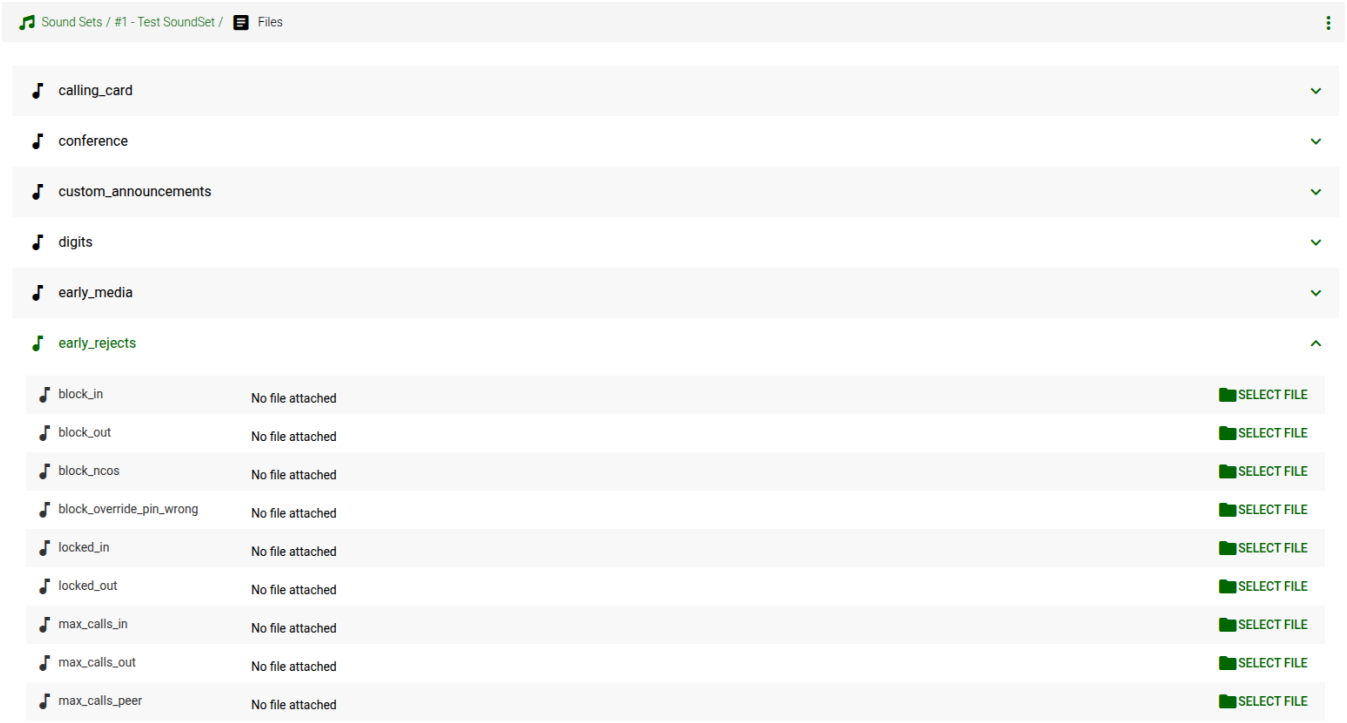

Figure 78. Early Reject Sounds

The call error announcements are played to the user in early media hence the name "Early Reject". If you do not provide the sound files for any handles they will not be used and the Sipwise C5 will fallback to sending the error response code back to the user.

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

Table 13. Early Reject Announcements

Handle	Description	Message played
`block_in`	This is what the calling party hears when a call is made from a number that is blocked by the incoming block list (*adm_block_in_list*, *block_in_list* customer/subscriber preferences)	Your call is blocked by the number you are trying to reach.
`block_out`	This is what the calling party hears when a call is made to a number that is blocked by the outgoing block list (*adm_block_out_list*, *block_out_list* customer/subscriber preferences)	Your call to the number you are trying to reach is blocked.

 Copyright © Sipwise GmbH - All rights reserved

Handle	Description	Message played
block_ncos	This is what the calling party hears when a call is made to a number that is blocked by the NCOS level assigned to the subscriber or domain (the NCOS level chosen in *ncos* and *adm_ncos* preferences). *PLEASE NOTE:* It is not possible to configure the status code and text.	Your call to the number you are trying to reach is not permitted.
block_override_pin_wrong	Announcement played to calling party if it used wrong PIN code to override the outgoing user block list or the NCOS level for this call (the PIN set by *block_out_override_pin* and *adm_block_out_override_pin* preferences)	The PIN code you have entered is not correct.
callee_busy	Announcement played on incoming call to the subscriber which is currently busy (486 response from the UAS)	The number you are trying to reach is currently busy. Please try again later.
callee_offline	Announcement played on incoming call to the subscriber which is currently not registered	The number you are trying to reach is currently not available. Please try again later.
callee_tmp_unavailable	Announcement played on incoming call to the subscriber which is currently unavailable (408, other 4xx or no response code or 30x with malformed contact)	The number you are trying to reach is currently not available. Please try again later.
callee_unknown	Announcement that is played on call to unknown or invalid number (not associated with any of our subscribers/hunt groups)	The number you are trying to reach is not in use.
callee_dnd	Announcement that is played on call to a subscriber that has the DND preference set	The number you are trying to reach is currently not accepting calls. Please try again later.
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.

Copyright © Sipwise GmbH - All rights reserved

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. *PLEASE NOTE:* The configuration entry for this case in /etc/ngcp-config/config.yml file is emergency_invalid.	The emergency number you have dialed is not available in your region.
emergency_unsupported	Announcement played when emergency destination is dialed but the emergency calls are administratively prohibited for this user or domain (*reject_emergency* preference is enabled)	You are not allowed to place emergency calls from this line. Please use a different phone.
error_please_try_later	Announcement played when there was an error during call processing. *PLEASE NOTE:* This announcement may be configured in the sound set in voucher_recharge section.	An error has occurred. Please try again later.
invalid_speeddial	This is what the calling party hears when it calls an empty speed-dial slot	The speed dial slot you are trying to use is not available.
locked_in	Announcement played on incoming call to a subscriber that is locked for incoming calls	The number you are trying to reach is currently not permitted to receive calls.
locked_out	Announcement played on outgoing call to a subscriber that is locked for outgoing calls	You are currently not allowed to place outbound calls.
max_calls_in	Announcement played on incoming call to a subscriber who has exceeded the *concurrent_max* or *concurrent_max_in* limit or whose customer has exceeded the *concurrent_max_per_account* or *concurrent_max_in_per_account* limit	The number you are trying to reach is currently busy. Please try again later.
max_calls_out	Announcement played on outgoing call to a subscriber who has exceeded the *concurrent_max* (total limit) or *concurrent_max_out* (limit on number of outbound calls) or whose customer has exceeded the *concurrent_max_per_account* or *concurrent_max_out_per_account* limit	All outgoing lines are currently in use. Please try again later.

Copyright © Sipwise GmbH - All rights reserved

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

There are some early reject scenarios when either **no voice announcement is played, or a fixed announcement is played**. In either case a SIP error status message is sent from Sipwise C5 to the calling party. It is possible to configure the exact status code and text for such cases in the /etc/ngcp-config/config.yml file, in kamailio.proxy.early_rejects section. The below table gives an overview of those early reject cases.

Table 14. 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)

Copyright © Sipwise GmbH - All rights reserved

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

By default the Early Reject sounds are being played using early media, so after the 183 Session Progress message. Using the subscriber's preference *announce_conn_type* this behavior can be changed.

If *announce_conn_type* is set to "Early" (default value), the behavior does not change. If *announce_conn_type* is set to "Established" the prompt will be played with an established media (200 OK).

This can help such scenario where 183 cannot be properly handled by the caller side.

Furthermore, the change of this preference will affect the behavior of all announcement based applications for this subscriber, such as:

- **Office-Hours:** Office-Hours
- **Pre-Recording Announcement:** Pre-Recording Announcement
- **Early-Reject Announcement:** Configuring Early Reject Sound Sets
- **Any other type of announcements**

7.16.4. Play an Announcement on Behalf of Callee Server Failure in Case of Outbound Calls

The Sipwise C5 makes it possible to play an announcement on behalf of callee server failure in case of outbound calls. This feature can be activated on Subscribers and on Peers. For example: if subscriber A calls subscriber B, but subscriber B refuses the call with code 404 without providing any

 Copyright © Sipwise GmbH - All rights reserved

announcement, the Sipwise C5 can be configured to play a customized announcement to subscriber A on behalf of subscriber B.

To activate this feature, first create a system *Sound Set*, or use an already existing one, and then assign it to the callee subscriber. Upload in the *Sound Set* one or more announcements. Once the *Sound Set* is configured, the subscriber's preference *announce_error_codes_enable* must be enabled under *Subscriber Preferences NAT and Media Flow Control* menu. Last step is to list in the subscriber's preference *announce_error_codes_list* the announcements that will be played to the caller in case a particular error code is returned back from the callee. Each entry of the list has to be a string composed in the following way: <error_code>;<announcement_name>, where error_code is the SIP return code and announcement_name is name of the announcement taken from the sound_set list. Returning to the example above, to play *callee_unknown* message in case of *404* returned from the callee, the entry '404;callee_unknown' has to be added in *announce_error_codes_list* preference.

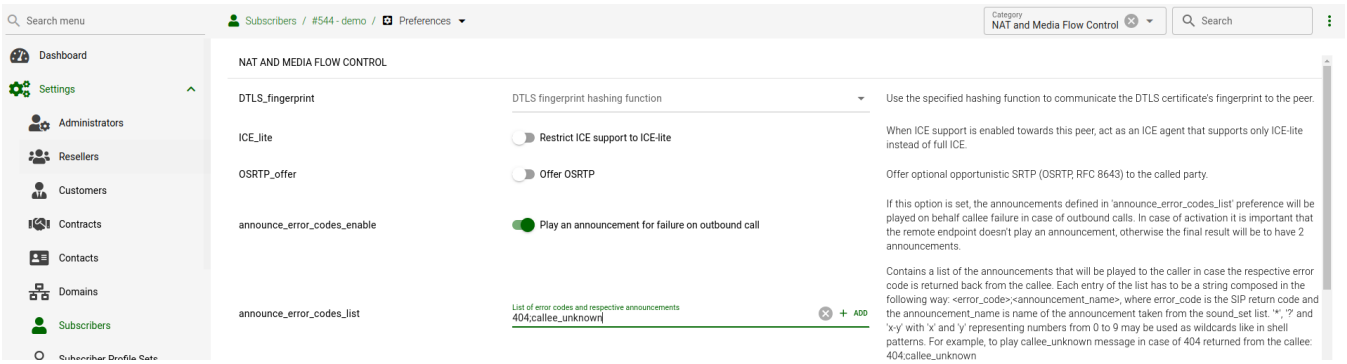

Figure 79. Set Announcement for failure on outbound call

The same feature is available for peer as well.

IMPORTANT

In case *announce_error_codes_enable* is enabled, it is important that the remote endpoint does not play any announcement for error codes listed in *announce_error_codes_list* otherwise the final result will be to have two announcements: one generated by the remote endpoint and one generated by the Sipwise C5.

7.17. Conference System

The Sipwise C5 provides the simple pin-protected conferencing service built using the SEMS DSM scripting language.

7.17.1. Configuring Call Forward to Conference

Go to your *Subscriber Details*, then *Call Forwarding* and configure a Call Forward of the type you want selecting *Conference* as destination. See the Call Forwarding and Call Hunting chapter to have more details on how to configure a call forward.

Copyright © Sipwise GmbH - All rights reserved

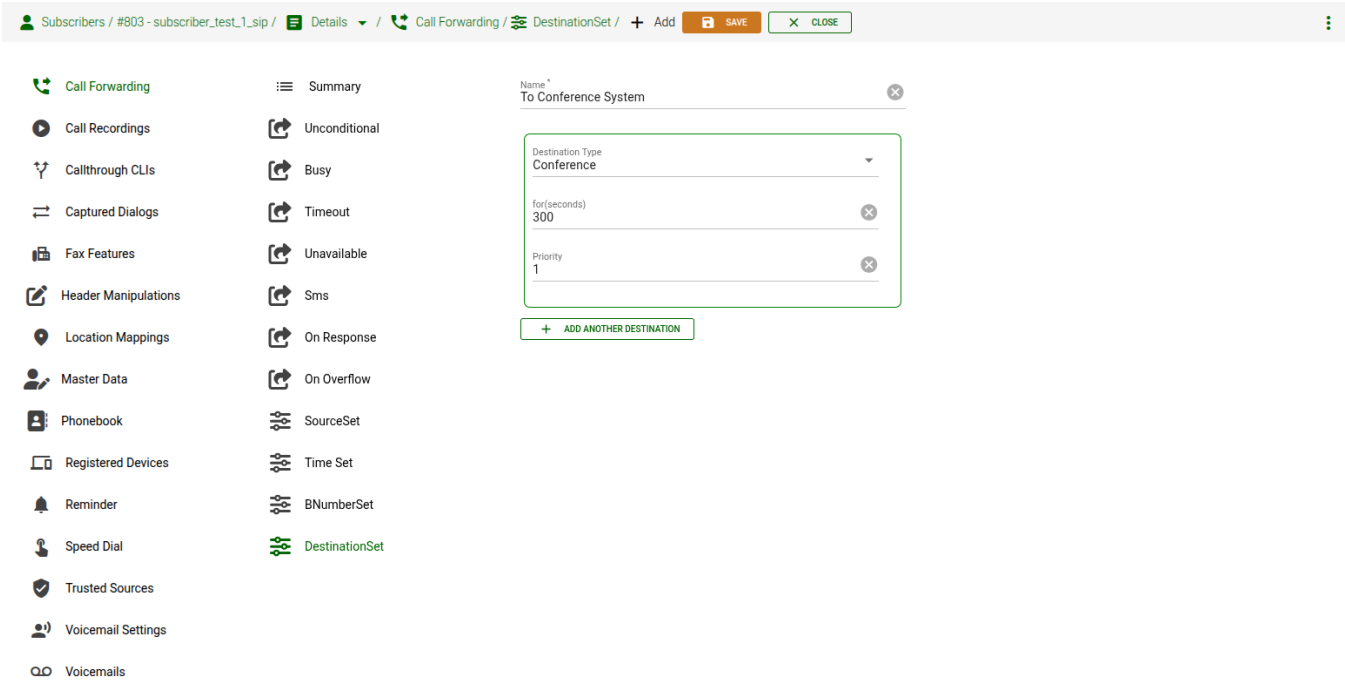

Figure 80. Set Call Forward to Conference

The timeout defines for how long this destination should be tried to ring.

7.17.2. Configuring Conference Sound Sets

The announcements played in a conference can be customised using Sound Sets. To create a new Sound Set or edit an existing one, see the Sound Sets chapter. In particular, the files to upload are the ones contained in the *conference* group. Upload the following files:

Table 15. 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.

 Copyright © Sipwise GmbH - All rights reserved

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

Then set the preference *sound_set* on the Domain or Subscriber level in order to assign the Sound Set you have just created to the subscriber.

7.17.3. 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 Preferences Internals conference_pin* field.

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

7.17.4. Conference Flowchart with Voice Prompts

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

Conference Flowchart with PIN Validation

Copyright © Sipwise GmbH - All rights reserved

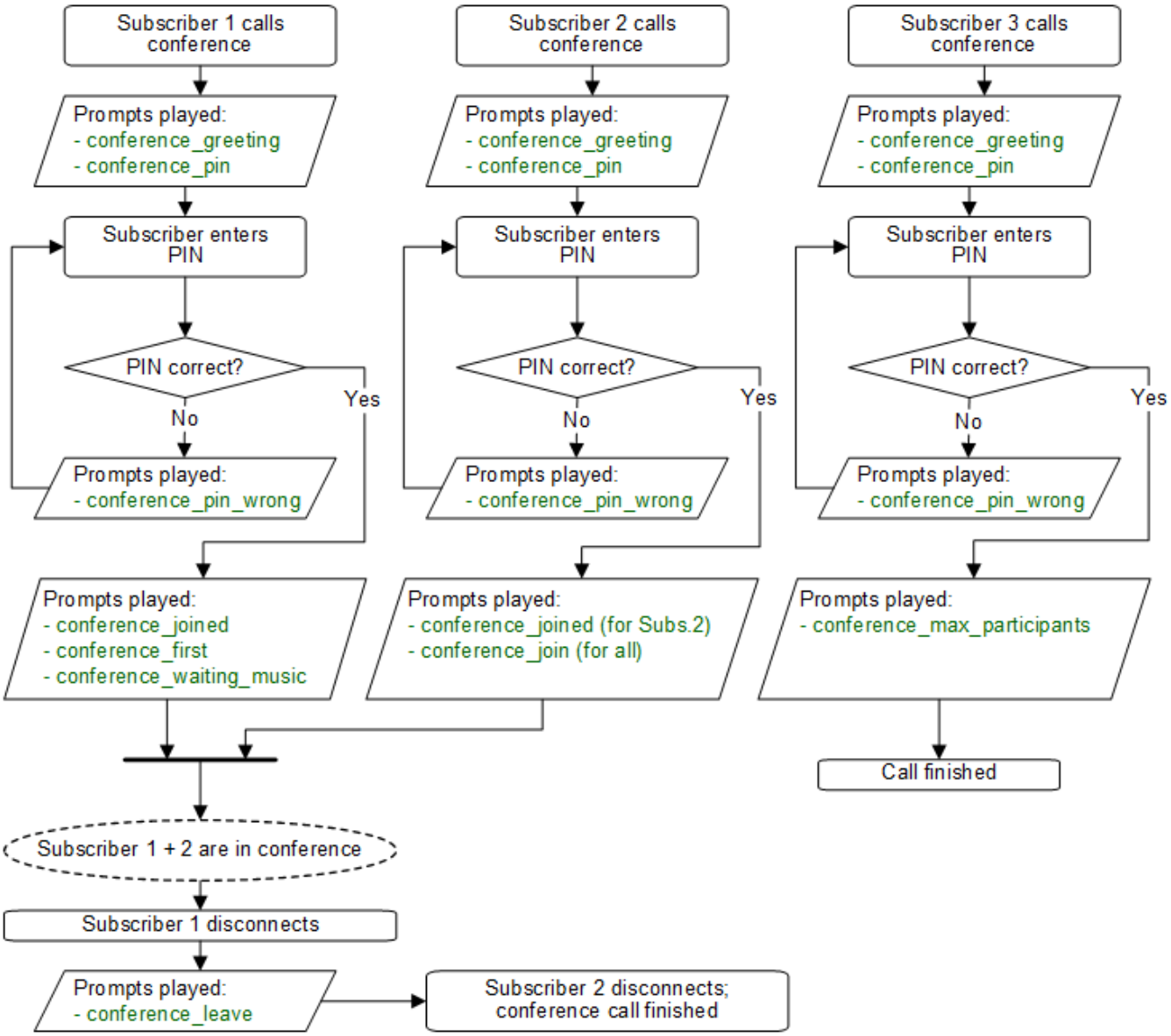

Figure 81. Flowchart of Conference with PIN Validation

Conference Flowchart Without PIN

Copyright © Sipwise GmbH - All rights reserved

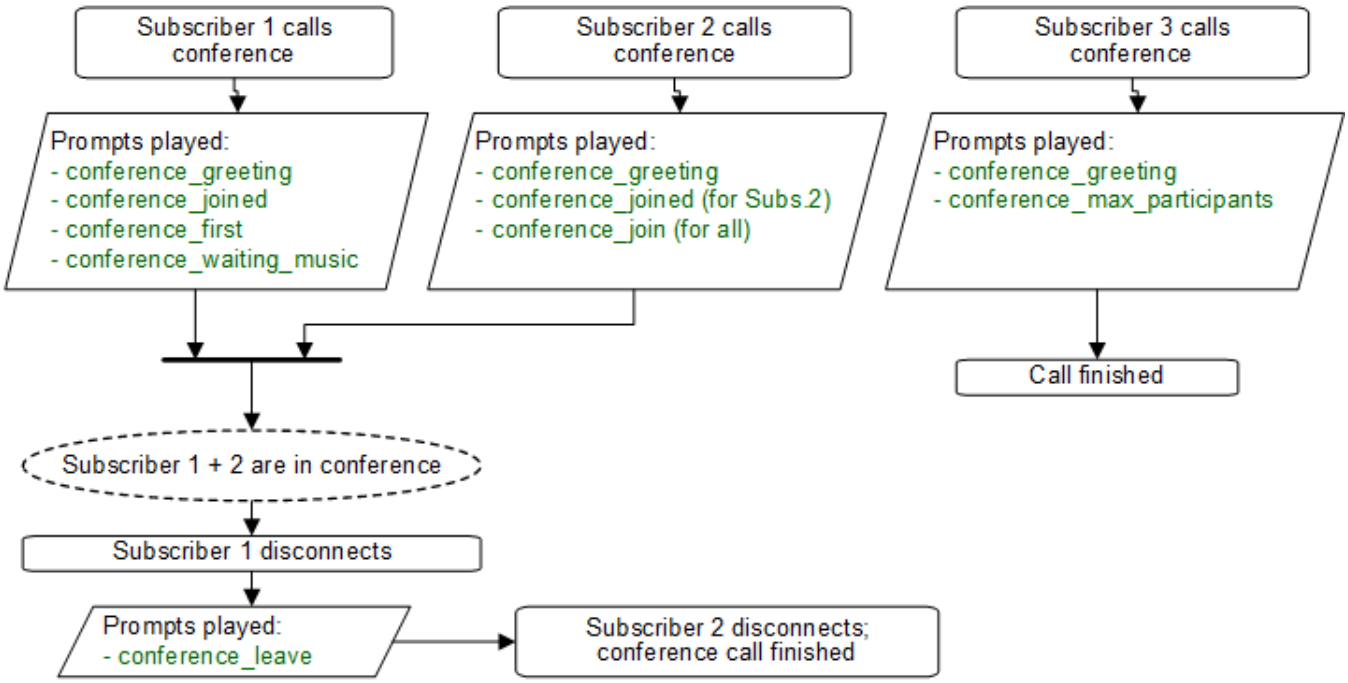

Figure 82. Flowchart of Conference without PIN

7.18. Malicious Call Identification (MCID)

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

7.18.1. 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 *SettingsSound Sets* either use your already existing *Sound Set* or create a new *Sound Set* and then assign it to your domain or subscribers. In the *Sound Set* there is a fileset *malicious_call_identificationmailicious_call_report* for that purpose.

Once the *Sound Set* is created, the Subscriber's Preferences *Malicious Call Identification* must be enabled under *SubscribersPreferencesApplicationsmalicious_call_identification*. 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, run `ngcpcfg apply "Enabling MCID"` to recreate the templates and automatically restart dependent services.

7.18.2. Usage

For a subscriber to report a malicious call, they can either call *malicious_call* or call their custom number assigned for that purpose. Please note that you can only report your last received call. You will hear the media reply from the *Sound Set* you have previously configured.

Copyright © Sipwise GmbH - All rights reserved

To check reported malicious calls as the platform operator open *SettingsMalicious Calls* tab where you will see a list of registered calls. You can selectively delete records from the list or alternatively you can manage the reported calls by using the REST API.

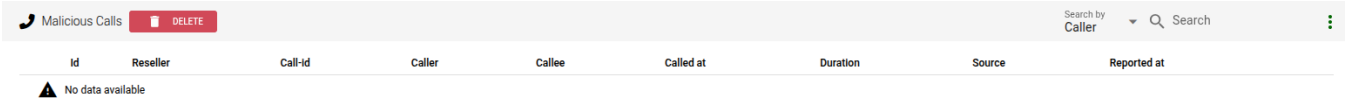

Figure 83. Malicious Calls

7.19. Subscriber Profiles

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

7.19.1. 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 they are permitted to do so).

Profile Sets can be defined in *SettingsSubscriber Profiles*. To create a new Profile Set, click *Add*.

You need to provide a reseller, a name and a description.

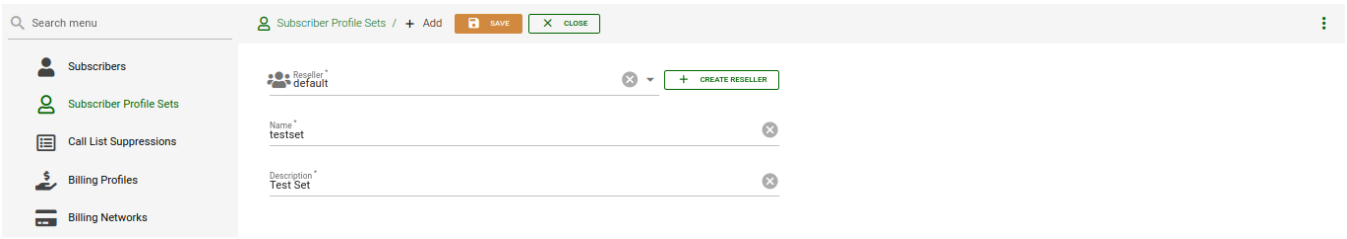

Figure 84. Save Subscriber Profile Sets

To create Profiles within a Profile Set, click on three dots beside a Profile Set and click the *Profiles* button.

Profiles within a Profile Set can be created by clicking the *Add* button.

Copyright © Sipwise GmbH - All rights reserved

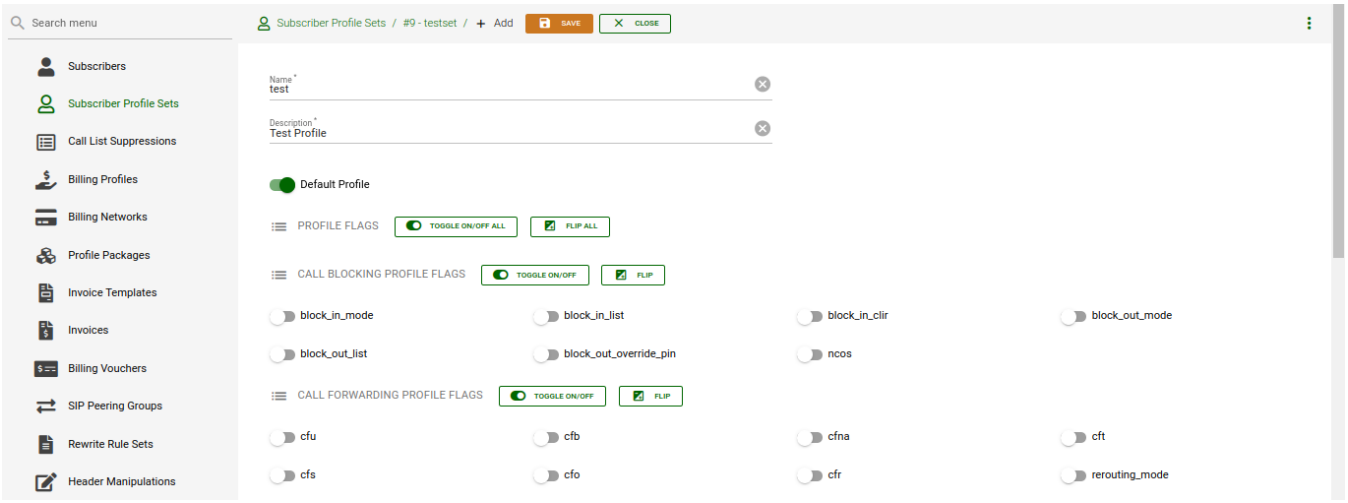

Figure 85. Set Subscriber Profile Flags

You need to provide a name and a description.

Checking the *Default Profile* option causes this profile to get assigned automatically to all subscribers, who have the profile set assigned. All the other subsequent flags define the user preferences which should be made available to the subscriber in the personal CSC.

When the platform administrator selects *Preferences* of the Subscriber Profile they will get the list of preferences that can be set by default for all subscribers who have this profile assigned. For example, if the system administrator configures the **ncos** (NCOS level) preference in the profile, this NCOS will be assigned by default to all the subscribers who use that profile.

NOTE Profile Preferences take precedence over the Domain preferences, but Profile Preferences hold a lower priority compared to the Subscribers Preferences.

NOTE The Subscribers Profiles can be created prior creating any subscriber.

7.20. SIP Loop Detection

In order to detect a SIP loop (incoming call as a response for a call request) the Sipwise C5 checks the combination of *SIP URI*, *To* and *From* headers.

This check can be enabled in `config.yml` by setting `kamailio.proxy.loop_detection.enable: yes`. The system tolerates `kamailio.proxy.loop_detection.max` loops within `kamailio.proxy.loop_detection.expire` seconds. Higher occurrence of loops will be reported with a SIP 482 "Loop Detected" error message.

7.21. Invoices and Invoice Templates

Content and vision of the invoices are customizable by Invoice Templates.

NOTE The Sipwise C5 generates invoices in pdf format.

7.21.1. Invoices Management

Invoices can be requested for generation, searched, downloaded and deleted on the administrative

Copyright © Sipwise GmbH - All rights reserved

web interface. Navigate to *Settings* *Invoices* menu and you get a list of all invoices currently stored in the database.

TIP The system operator or a third party application can also generate, list, retrieve and delete invoices via the REST API. Please read further details in Invoice Management via REST API.

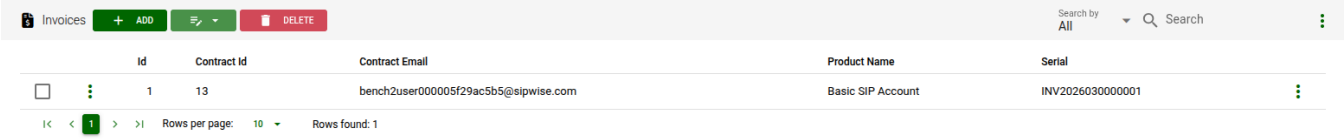

Figure 86. Invoice Management Interface

To request invoice generation for the particular customer/reseller/peer and period, press the "Add" button. On the invoice creation form, fill in the following parameters:

- **Template**: A prepared invoice template can be selected for the invoice generation.
- **Contract**: The owner of the billing account, recipient of the invoice.

  Customer Invoice: Customer Contract

  Reseller Invoice: Reseller Contract

  Peer Invoice: Peering Group Contract
- **Invoice period**: The billing period. This can only be specified as one calendar month. Calls with start time between the first and last second of the period will be considered for the invoice

All form fields are mandatory.



Figure 87. Invoice Management Interface

Generated invoice can be downloaded as pdf file. To do that click on the three dots beside the newly created invoice, then press the "Download" button

 Copyright © Sipwise GmbH - All rights reserved

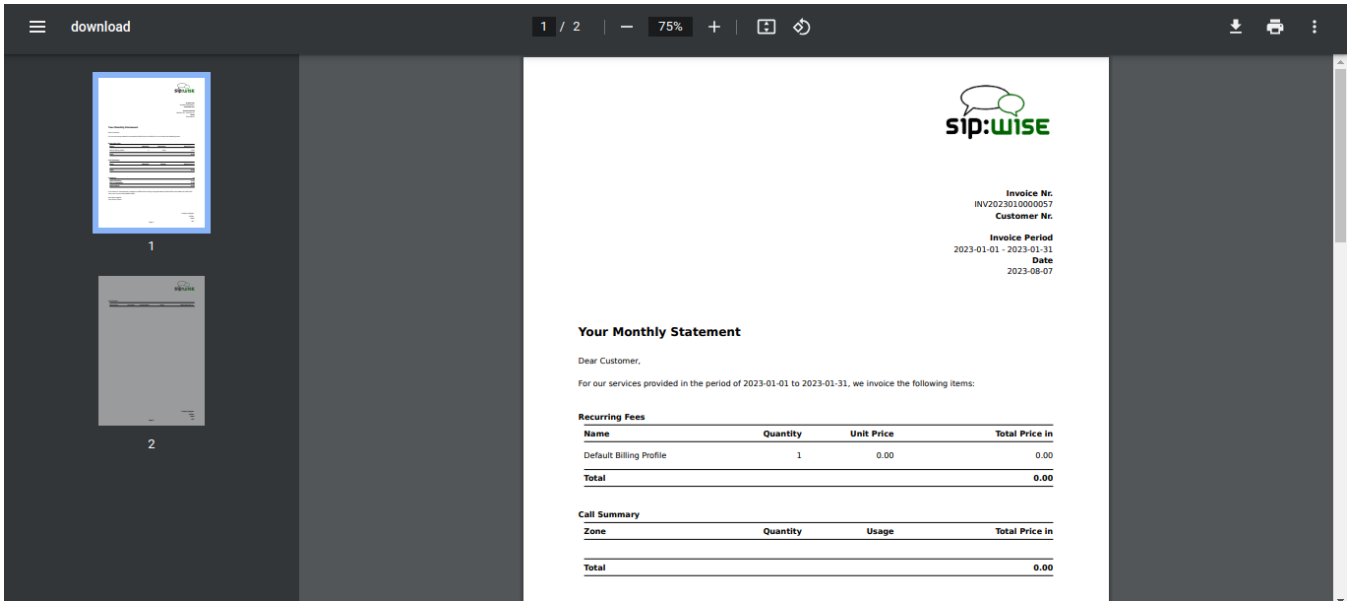

Figure 88. Invoice PDF File

7.21.2. Invoice Management via REST API

Besides managing invoices on the admin web interface of NGCP, the system administrator (or a third party system) has the opportunity to request generation and retrieval of invoices via the *REST API*.

The subsequent sections describe the available operations for invoice management with API requests in details. All operations work on the *Invoices* resource and use the `/api/invoices` base path. The authentication method is username/password in the examples given below, however it is recommended to use a TLS client certificate for authentication on the REST API.

NOTE The full API documentation is always available at the location: https://<IP_of_NGCP_web_panel>:1443/api

Generate a New Invoice

The following example shows a CURL command that will request generation of an invoice:

- for contract with ID "79"
- for the time period of August 2023
- based on the invoice template with ID "1"

```
curl -i -X POST -H 'Connection: close' -H 'Content-Type:
application/json' \
  --user adminuser:adminpwd -k 'https://127.0.0.1:1443/api/invoices/'
\
  --data-binary '{ "customer_id" : "79", "template_id" : "1",
\
  "period_start": "2023-08-01 00:00:00", "period_end": "2022-08-31
23:59:59"  }'
```

Please note that in this operation the `/api/invoices` path (the *invoices* collection) was used and a

Copyright © Sipwise GmbH - All rights reserved

POST request on it to create a new invoice item.

In case of a **successful operation**, the Sipwise C5 will reply with **201 Created** HTTP status and send the ID of the invoice in *Location* header. In our example the new invoice item may be directly referred as `/api/invoices/3` (ID = 3).

```
HTTP/1.1 201 Created
Server: nginx
Date: Tue, 12 Sep 2023 13:38:40 GMT
Content-Length: 0
Connection: close
Location: /api/invoices/3
Set-Cookie: ngcp_panel_session=d5e4a8dd003fd7cac646653a6b5aefa703cf3e66;
path=/; expires=Tue, 12-Sep-2023 14:38:38 GMT; HttpOnly
X-Catalyst: 5.90114
Strict-Transport-Security: max-age=15768000
```

In case of a **failed operation**, e.g. when an invoicing period that is invalid for the customer is requested, the Sipwise C5 will reply with **422 Unprocessable Entity** or **500 Internal Server Error** HTTP status.

Download Invoice Data

You can download properties / data of a specific invoice by selecting the item by its ID, using an HTTP *GET* request.

```
curl -i -X GET -H 'Connection: close' --user adminuser:adminpwd -k \
'https://127.0.0.1:1443/api/invoices/3'
```

The above request will return a JSON data structure containing invoice properties:

```
HTTP/1.1 200 OK
Server: nginx
Date: Tue, 12 Sep 2023 12:13:04 GMT
Content-Type: application/hal+json;
profile="http://purl.org/sipwise/ngcp-api/"; charset=utf-8
Content-Length: 759
Connection: close
Link: </api/invoices/>; rel=collection
Link: <http://purl.org/sipwise/ngcp-api/>; rel=profile
Link: </api/invoices/3>; rel="item self"
Link: </api/invoices/3>; rel="item http://purl.org/sipwise/ngcp-
api/#rel-invoices"
Link: </api/customers/79>; rel="item http://purl.org/sipwise/ngcp-
api/#rel-customers"
Set-Cookie: ngcp_panel_session=219feccbee4fa936defd1ee511c84efe7b5a6d6a;
path=/; expires=Wed, 12-Sep-2023 13:13:03 GMT; HttpOnly
Strict-Transport-Security: max-age=15768000

{
```


Copyright © Sipwise GmbH - All rights reserved

```
   "_links" : {
      "collection" : {
         "href" : "/api/invoices/"
      },
      "curies" : {
         "href" : "http://purl.org/sipwise/ngcp-api/#rel-{rel}",
         "name" : "ngcp",
         "templated" : true
      },
      "ngcp:customers" : {
         "href" : "/api/customers/79"
      },
      "ngcp:invoices" : {
         "href" : "/api/invoices/3"
      },
      "profile" : {
         "href" : "http://purl.org/sipwise/ngcp-api/"
      },
      "self" : {
         "href" : "/api/invoices/3"
      }
   },
   "amount_net" : 0,
   "amount_total" : 0,
   "amount_vat" : 0,
   "id" : 3,
   "period_end" : "2023-08-31T23:59:59+00:00",
   "period_start" : "2023-08-01T00:00:00+00:00",
   "sent_date" : null,
   "serial" : "INV202308000003"
}
```


It is also possible to query the complete *invoices* collection and use a filter (e.g. invoicing period, contract ID, etc.) to get the desired invoice item. In the example below we request all available invoices that belong to the customer with ID "79".

```
curl -i -X GET -H 'Connection: close' --user adminuser:adminpwd -k \
'https://127.0.0.1:1443/api/invoices/?customer_id=79'
```

The returned dataset is now slightly different because it is represented as an array of items, although in our example the array consist of only 1 item:


```
{
   "_embedded" : {
      "ngcp:invoices" : [
         {
            "_links" : {
               "collection" : {
                  "href" : "/api/invoices/"
```


Copyright © Sipwise GmbH - All rights reserved

```
                },
                "curies" : {
                   "href" : "http://purl.org/sipwise/ngcp-api/#rel-
{rel}",
                   "name" : "ngcp",
                   "templated" : true
                },
                "ngcp:customers" : {
                   "href" : "/api/customers/79"
                },
                "ngcp:invoices" : {
                   "href" : "/api/invoices/3"
                },
                "profile" : {
                   "href" : "http://purl.org/sipwise/ngcp-api/"
                },
                "self" : {
                   "href" : "/api/invoices/3"
                }
             },
             "amount_net" : 0,
             "amount_total" : 0,
             "amount_vat" : 0,
             "id" : 3,
             "period_end" : "2023-08-31T23:59:59+00:00",
             "period_start" : "2023-08-01T00:00:00+00:00",
             "sent_date" : null,
             "serial" : "INV2023080000003"
          }
       ]
    },
    "_links" : {
       "curies" : {
          "href" : "http://purl.org/sipwise/ngcp-api/#rel-{rel}",
          "name" : "ngcp",
          "templated" : true
       },
       "ngcp:invoices" : {
          "href" : "/api/invoices/3"
       },
       "profile" : {
          "href" : "http://purl.org/sipwise/ngcp-api/"
       },
       "self" : {
          "href" : "/api/invoices/?page=1&rows=10"
       }
    },
    "total_count" : 1
}
```


 Copyright © Sipwise GmbH - All rights reserved

Download Invoice as PDF File

You can download a specific invoice as a PDF file in the following way:

- selecting the item by its ID (as in our example, but you can also use a filter and query the complete *invoices* collection)
- using an HTTP *GET* request
- adding **"Accept: application/pdf" header** to the request

```
curl -X GET -H 'Connection: close' -H 'Accept: application/pdf' \
  --user adminuser:adminpwd -k 'https://127.0.0.1:1443/api/invoices/3' >
result.pdf
```

Please note that in the example above we **do not add the "-i" option** that would also include the headers of the HTTP response in the output file. The output of the CURL command, i.e. the PDF file, is saved as "result.pdf" locally.

Delete an Invoice

In order to delete an invoice item you have to send a *DELETE* request on the specific item:

```
curl -i -X DELETE -H 'Connection: close' --user adminuser:adminpwd -k \
'https://127.0.0.1:1443/api/invoices/3'
```

In case of successful deletion Sipwise C5 should send HTTP status `204 No Content` as a response:

```
HTTP/1.1 204 No Content
Server: nginx
Date: Tue, 12 Sep 2023 13:42:42 GMT
Connection: close
Set-Cookie: ngcp_panel_session=10b66a6baf25a09739c2bb2377c70ecceee78387;
path=/; expires=Wed, 12-Sep-2023 14:42:42 GMT; HttpOnly
X-Catalyst: 5.90114
Strict-Transport-Security: max-age=15768000
```

7.21.3. Invoice Templates

Invoice template defines the structure and look of the generated invoices. The Sipwise C5 allows to create several invoice templates. Multiple invoice templates can be used to create invoices for the different contract types and different languages.

For the deprecated batch invoice generator, each contract was supposed to be associated to one of the existing invoice templates. The encouraged approach is to use the REST API to trigger invoice creation per contract.

NOTE Call cost sums from CDRs are used for invoices, which allows to restrict by call direction and use rated call costs according to the carrier/reseller/customer billing fees.

Copyright © Sipwise GmbH - All rights reserved

IMPORTANT Generating invoices requires CDRs for the billing period, so ensure the CDR retention period is set accordingly in the cleanup/archiving configuration.

Invoice Templates Management

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

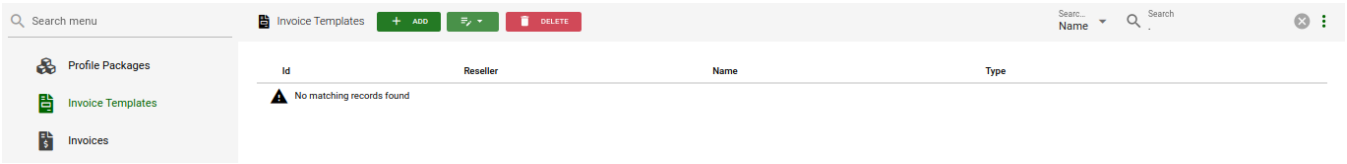

Figure 89. Invoice Templates

Invoice template creation is separated on two steps:

- Create a new invoice template with meta information.
- Visually edit content (template itself) of the invoice template.

To start a new invoice template, press the "Add" button.

On the invoice template meta information form, the following parameters can be specified:

- **Reseller**: The reseller who owns this invoice template. For customer/DID invoices, this is mandatory so each reseller has their individual set of templates for their end customers. Templates for peer and reseller invoices are maintained by the platform operator, so the reseller has to be empty for those categories.
- **Name**: A unique invoice template name to differentiate invoice templates if there are some.
- **Type**: Currently the Sipwise C5 supports only SVG format of the invoice templates.
- **Call Direction**: Information of the calls that are going to included in the invoice. Whether it is outgoing only, incoming only or both.
- **Category**: This defines the invoice type:

  Customer: The intrinsic invoice type that resellers use to charge end customers. Invoices using a template of the *Customer* category will require a customer contract. They present call costs rated according to the billing fees of the source/destination customer contract.

  Reseller: The invoice type that the platform operator uses to charge resellers. Invoices using a template of the *Reseller* category will require a reseller contract. They present call costs rated according to the billing fees of the source/destination reseller contract.

  Peer: The invoice the platform operator uses to charge peering partners or verify their costs. Invoices using a template of the *Peer* category will require a peering group contract. They present call costs rated according to the billing fees of the source/destination peering group contract.

  DID: A special invoice type that resellers use to charge end customers with detailed cost breakdown per subscriber - DID (Direct Inward Dialing) extensions respectively. Invoices using a template of the *DID* category will require a customer contract. They present call costs rated according to the billing fees of the source/destination customer contract.

 Copyright © Sipwise GmbH - All rights reserved

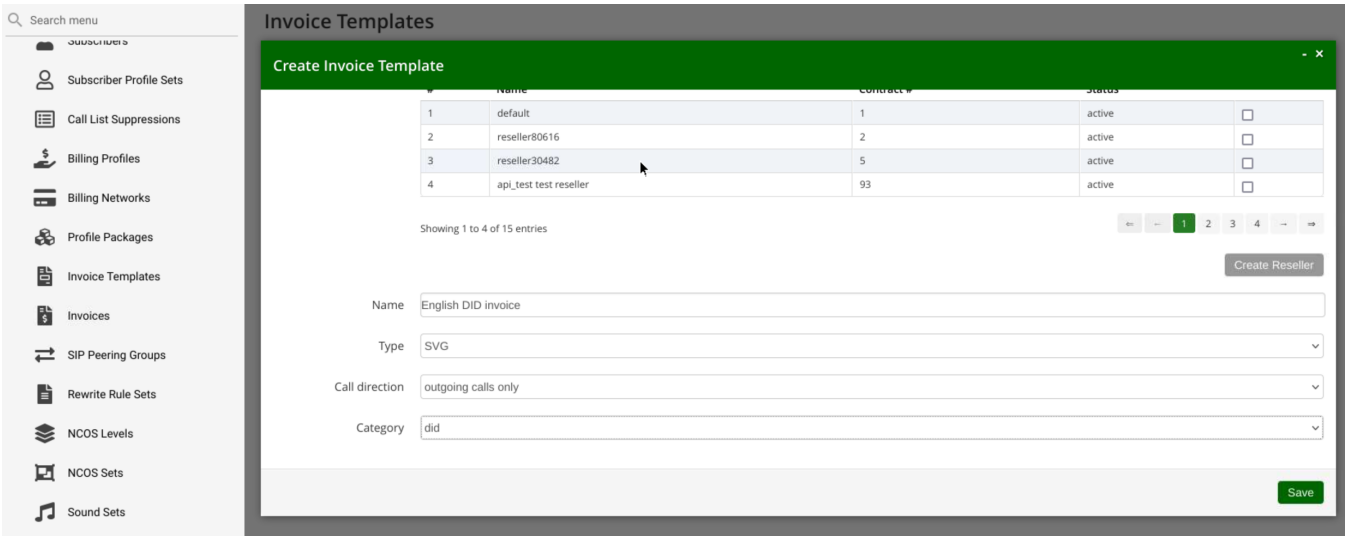

Figure 90. Create Invoice Templates

Invoice Template Content

After registering the new invoice template you can change the invoice template structure in WYSIWYG SVG editor and preview the result of the invoice generation based on the template.

The invoice template is a XML SVG source, which describes the content, look and position of the text lines, images or other invoice template elements. The Sipwise C5 provides embedded WYSIWYG SVG editor to customize the default template. The Sipwise C5 svg-edit has some changes in layers management, image edit, user interface, but this basic introduction may still be useful.

To enter in the edit mode, click the three dots beside the newly created template and click on the *Edit Template* button.

Template refers to the owner reseller contact ("rescontact"), contract ("customer" or "contract"), 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 the "Show variables" button on the top of the screen to see full list of variables with the fields:

Copyright © Sipwise GmbH - All rights reserved

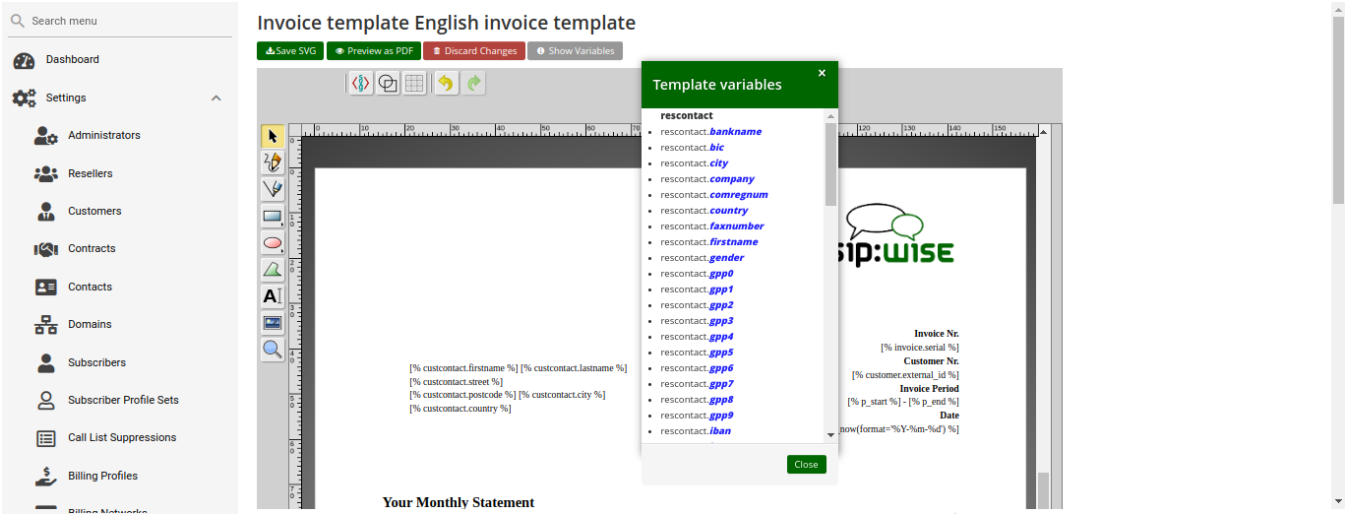

Figure 91. Invoice Template Variables

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 implementing the desired template changes, save the invoice template changes.

To return to the Sipwise C5 invoice template **default** content you can press on the "Discard changes" button.

IMPORTANT The "Discard changes" operation cannot 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 an invoice period. Available only to customer invoices with *outgoing* call direction and *incoming* call direction.
- **DID ZoneCalls**: page with zone costs like in the summary, but broken down for each of the customer contract's subscribers. Available to DID invoices only.

To see all invoice template layers, press on "Layers" vertical sign on right side of the svg-edit interface:

Copyright © Sipwise GmbH - All rights reserved

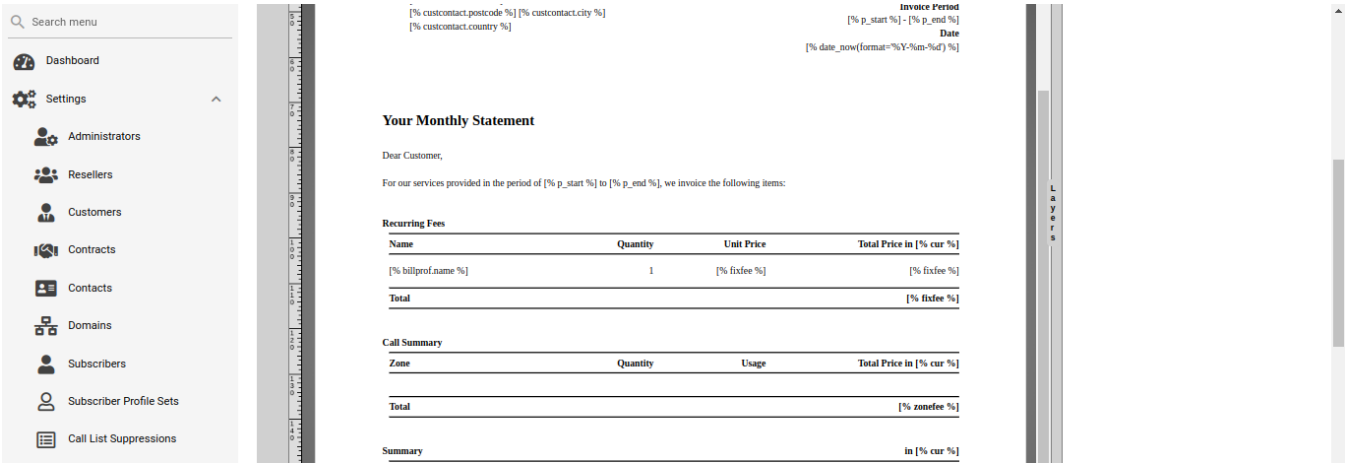

Figure 92. Open Invoice Templates Layers

Side panel with layers list will be shown.

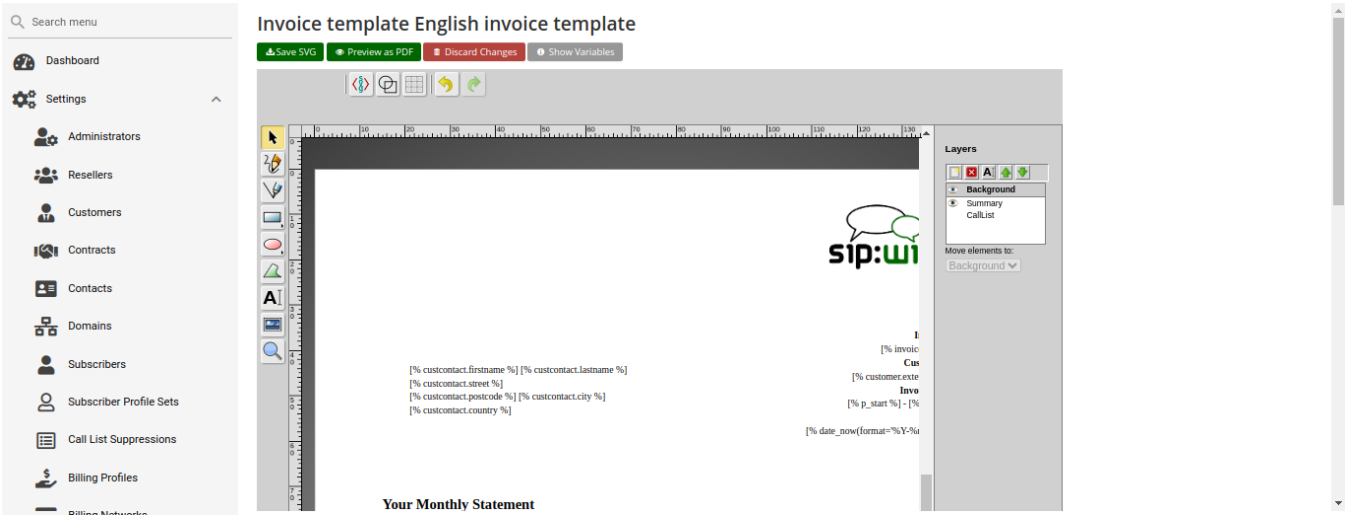

Figure 93. Invoice Templates Layers

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 a good idea to keep the visibility of the "Background" layer on, so look of the generated page will be seen.

Edit SVG XML Source

Sometimes it may be convenient to edit svg source directly and svg-edit makes it possible to do it. After press on the <svg> icon in the top left corner of the svg-edit interface:

Copyright © Sipwise GmbH - All rights reserved

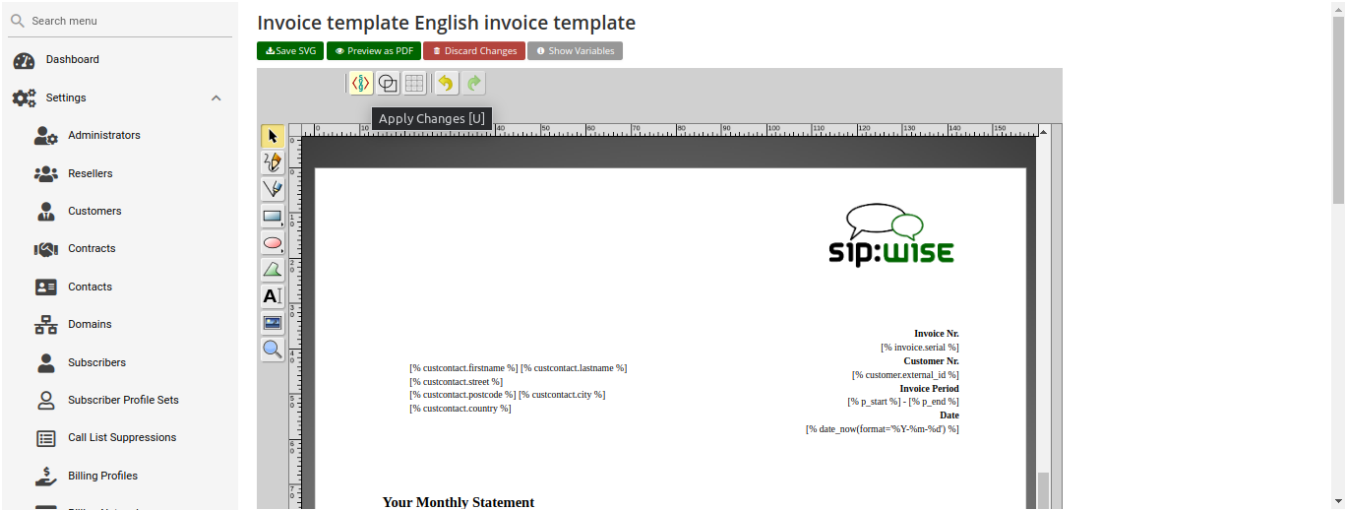

Figure 94. Edit Invoice Templates XML Source

SVG XML source of the invoice template will be shown.

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

NOTE Template keeps sizes and distances in pixels.

IMPORTANT When editing svg xml source, please change things inside special comment mark-up "<!--{ }-" very carefully. Failure to do so may result in potential disruptions to the invoice generation process. 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:

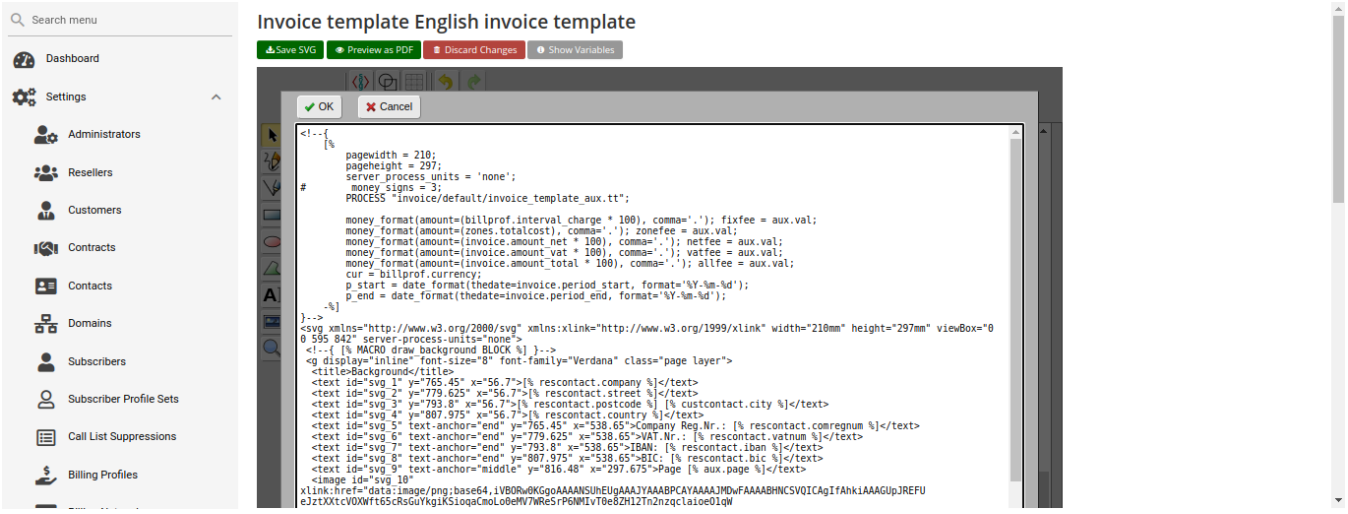

Figure 95. Invoice Templates XML Source

And then save invoice template changes.

NOTE You can copy and keep the SVG source of your template as a file on the disk before

 Copyright © Sipwise GmbH - All rights reserved

> 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.

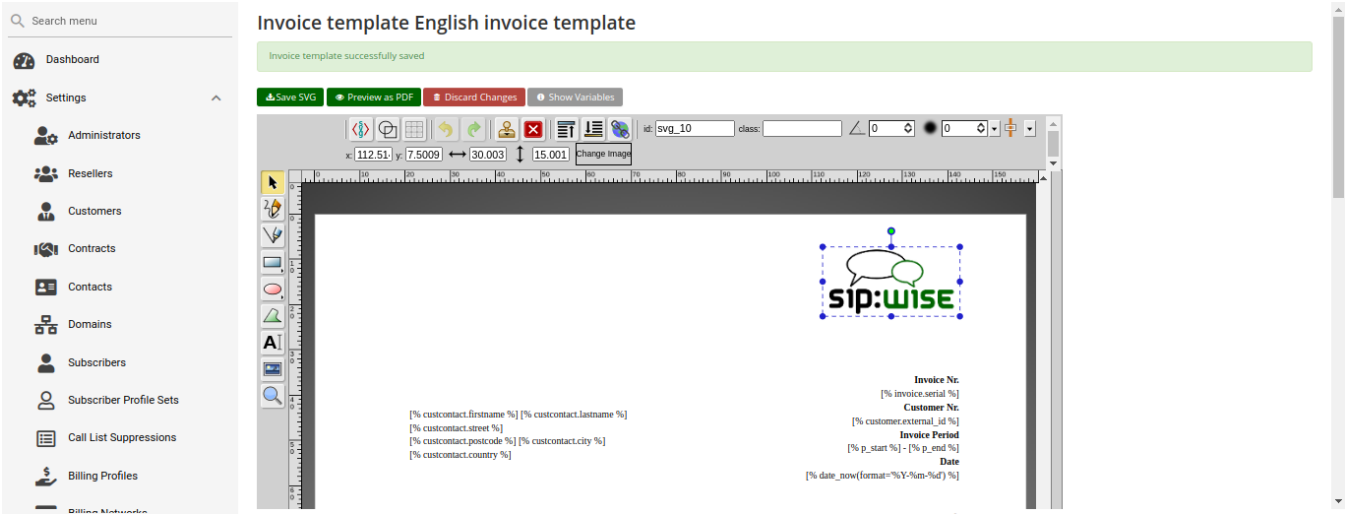

Figure 96. Change Invoice Templates Logo

After image uploaded save invoice template changes.

Save and Preview Invoice Template Content

To save invoice template content changes press button "Save SVG".

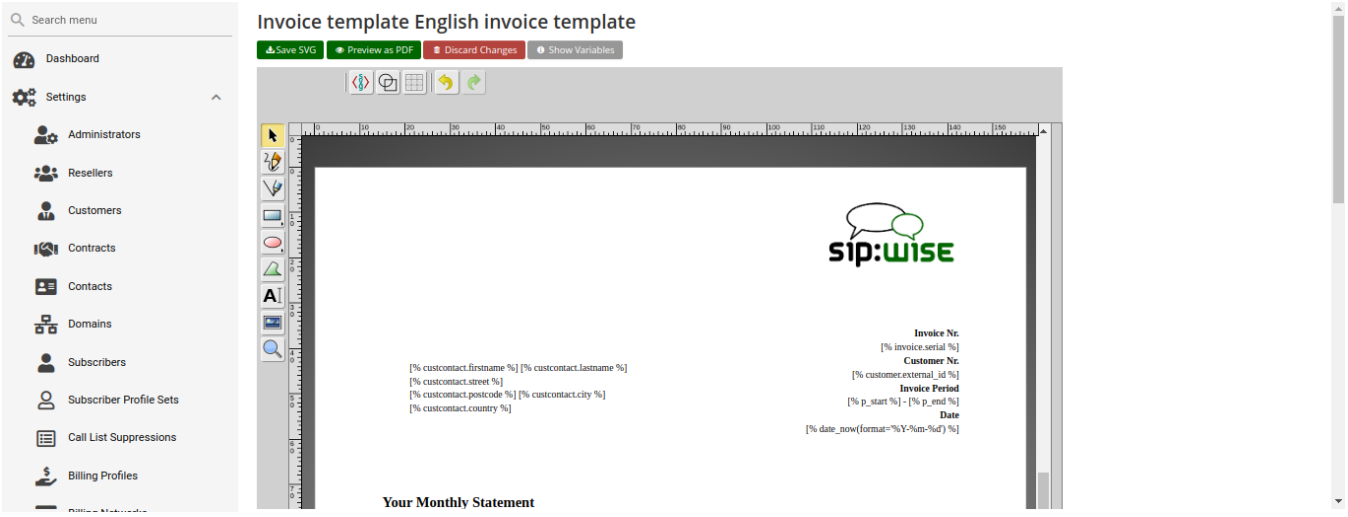

Figure 97. Invoice Templates New Logo

You will see message about successfully saved template. You can preview your invoice look in PDF format. Press on "Preview as PDF" button.

Copyright © Sipwise GmbH - All rights reserved

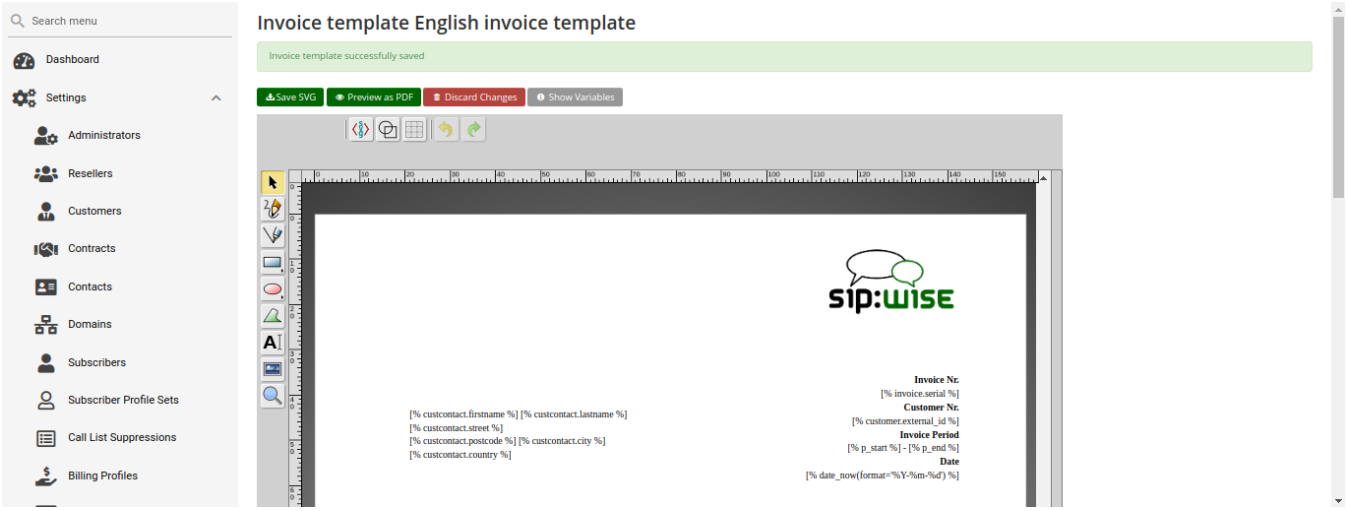

Figure 98. Preview Invoice Templates

Invoice preview will be opened in the new window.

NOTE Example fake data will be used for preview generation.

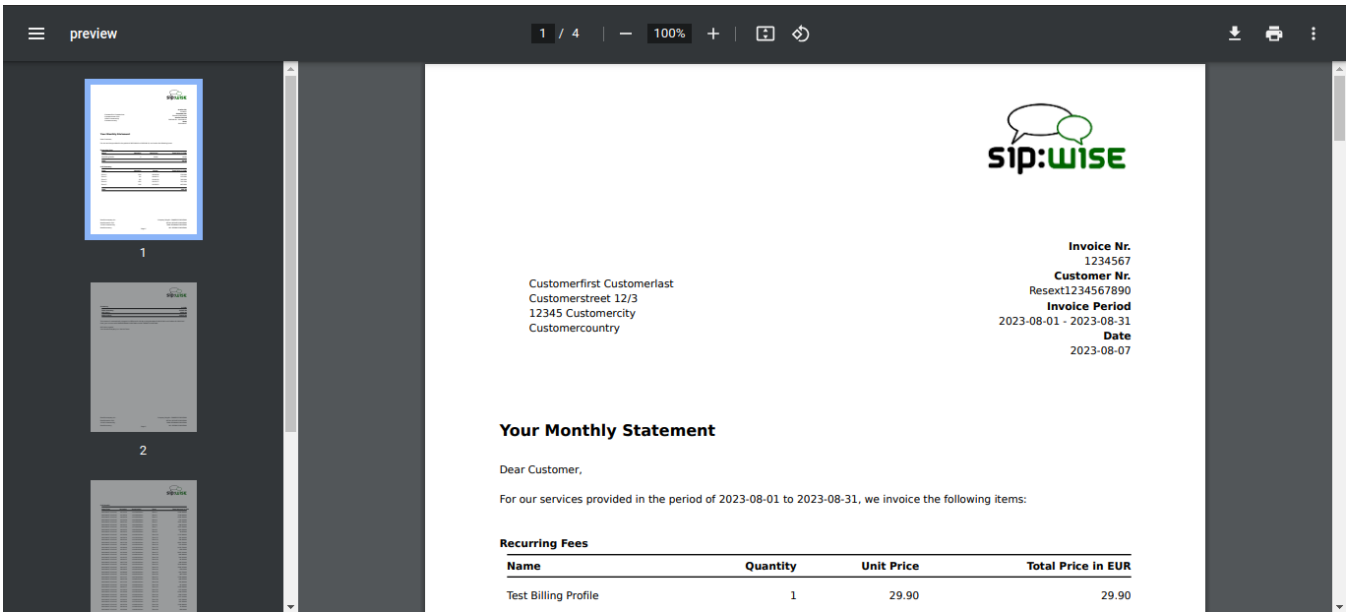

Figure 99. Invoice Templates PDF Preview

7.22. Email Reports and Notifications

7.22.1. Email Events

The Sipwise C5 makes it possible to customize the content of the emails sent on the following actions:

- Web password reset requested. An email will be sent to the subscriber, for whom the password reset was initiated. If the subscriber does not have an email, an email will be sent to the customer, who owns the subscriber.
- Administrator password reset requested. An email will be sent to the administrator, for whom the password reset was initiated. If the administrator does not have an email, an error message will

 Copyright © Sipwise GmbH - All rights reserved

appear when requesting the password reset.

- New subscriber created. Email will be sent to the newly created subscriber or to the customer, who owns the new subscriber.
- Email with the invoice. The email will be sent to the customer.

7.22.2. Initial Template Values and Template Variables

Default email templates for each of the email events are inserted on the initial Sipwise C5 database creation. Content of the default template is described in the corresponding sections. Default email templates are not linked to any reseller and cannot be changed through the Sipwise C5 Panel. They will be used to initialize default templates for the newly created reseller.

Each email template refers to the values from the database using special mark-ups "[%" and "%]". Each email template has a fixed set of the variables. Variables cannot be added or changed without changes in the Sipwise C5 Panel code.

7.22.3. Subscriber Password Reset Email Template

An email will be sent after the subscriber or the subscriber administrator requests a password reset for the subscriber account. If subscriber does not have an email, an email will be sent to the customer owning the subscriber.

The default content of the password reset email template is:

Template name	**passreset_default_email**
From	default@sipwise.com
Subject	Password reset email

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

The following variables will be provided to the email template:

- [%**url**%]: specially generated url where subscriber can define his new password.
- [%**subscriber**%]: username@domain of the subscriber, which password was requested for reset.

7.22.4. Administrator Password Reset Email Template

An email is sent when the administrator requests a password reset for their account. If the administrator does not have an email address, an error message will be displayed when requesting the password reset.

Copyright © Sipwise GmbH - All rights reserved

The default content of the password reset email template is:

Template name	**admin_passreset_default_email**
From	default@sipwise.com
Subject	Password reset email

Body	`Dear Customer,`
	
	`Please go to [%url%] to set your password and log into your admin interface.`
	
	`Your faithful Sipwise system`
	
	`--`
	`This is an automatically generated message. Do not reply.`

The following variables will be provided to the email template:

- [%**url**%]: specially generated url where administrator can define his new password.
- [%**admin**%]: username@domain of the administrator, which password was requested for reset.

7.22.5. New Subscriber Notification Email Template

An email is sent when a new subscriber is created. An email will be sent to the newly created subscriber if they have an email. Otherwise the email will be sent to the customer who owns the subscriber.

NOTE By default, the email content template is addressed to the customer. Please consider this when creating the subscriber with an email.

Template name	**subscriber_default_email**
From	default@sipwise.com
Subject	Subscriber created

Body	`Dear Customer,`
	
	`A new subscriber [%subscriber%] has been created for you.`
	
	`Your faithful Sipwise system`
	
	`--`
	`This is an automatically generated message. Do not reply.`

The 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.

Copyright © Sipwise GmbH - All rights reserved

7.22.6. Invoice Email Template

Template name	**invoice_default_email**
From	default@sipwise.com
Subject	Invoice #[%invoice.serial%] from [%invoice.period_start_obj.ymd%] to [%invoice.period_end_obj.ymd%]

Body	`Dear Customer,`
	
	`Please find your invoice #[%invoice.serial%] for [%invoice.period_start_obj.month_name%], [%invoice.period_start_obj.year%] in attachment of this letter.`
	
	`Your faithful Sipwise system`
	
	`--`
	`This is an automatically generated message. Do not reply.`

The variables passed to the email template:

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

NOTE

Invoice fields

- [%invoice.**serial**%]
- [%invoice.**amount_net**%]
- [%invoice.**amount_vat**%]
- [%invoice.**amount_total**%]
- [%invoice.**period_start_obj**%]
- [%invoice.**period_end_obj**%]

The fields [%invoice.period_start_obj%] and [%invoice.period_end_obj%] provide methods of the perl package DateTime for the invoice start date and end date. Further information about DateTime can be obtained from the package documentation:

man DateTime

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

NOTE

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

- [%provider.**gender**%]
- [%provider.**firstname**%]
- [%provider.**lastname**%]

Copyright © Sipwise GmbH - All rights reserved

- [%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**%]

7.22.7. Email Templates Management

Email templates linked to the resellers can be customized in the email templates management interface. For the administrative account, the email templates from all resellers will be displayed, while for the reseller account, only their own email templates will be visible.

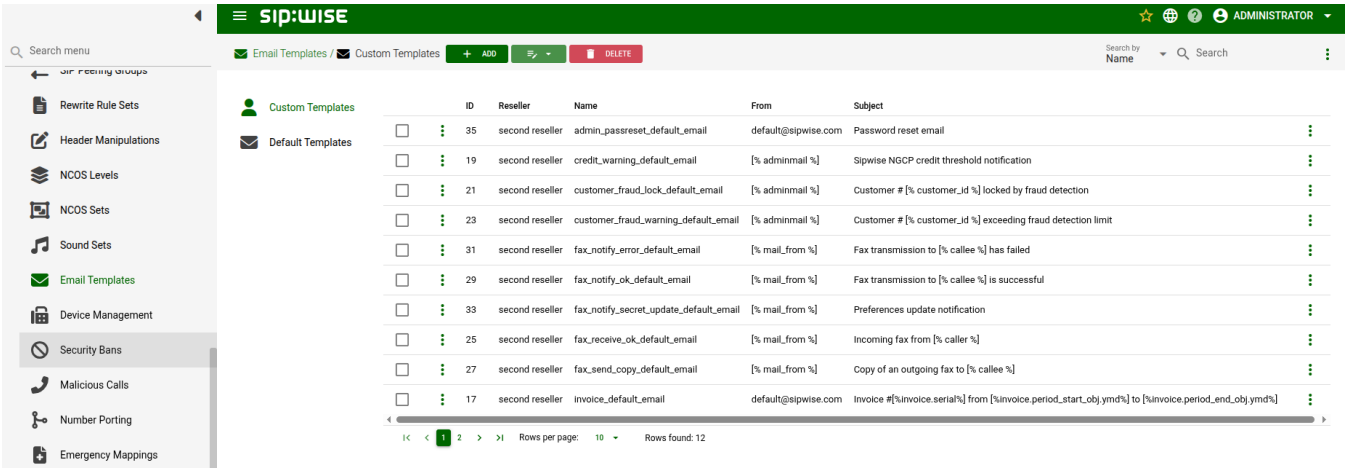

Figure 100. Email Templates

To create new email template go to *SettingsEmail Templates Custom Templates* and press the button "Add".

 Copyright © Sipwise GmbH - All rights reserved

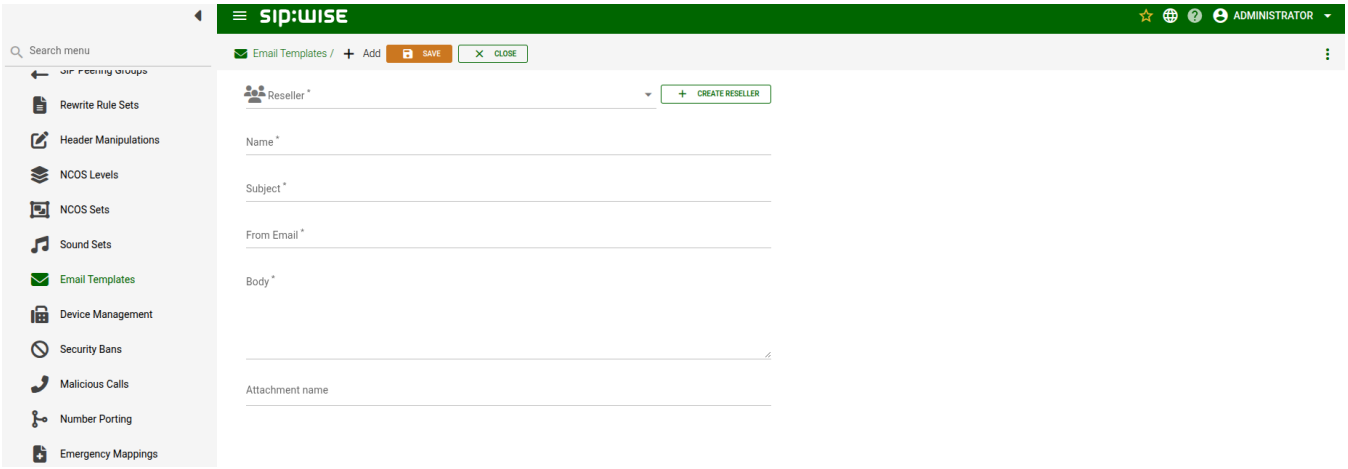

Figure 101. Create Email Templates

On the email template form all fields are mandatory:

- **Reseller**: The reseller who owns this email template.
- **Name**: Currently only an email template with the following names will be considered by the Sipwise C5 on the Email event :

  admin_passreset_default_email;

  passreset_default_email;

  subscriber_default_email;

  invoice_default_email;
- **From Email Address**: An email address which will be used in the From field in the email sent by the Sipwise C5 .
- **Subject**: Template of the email subject. Subject will be processed with the same template variables as the email body.
- **Body**: Email text template. Will be processed with appropriate template variables.

7.23. Calling Line Identification Restriction

The Calling Line Identification Restriction (CLIR) feature prevents the calling party's number from being displayed to the called party. It is the opposite of Calling Line Identification Presentation (CLIP).

7.23.1. How It Works

When CLIR is activated the Sipwise C5 platform does the following:

1. The From header is set to 'From: "Anonymous" <sip:anonymous@anonymous.invalid>"'.
2. The "Privacy: id" SIP header is added to tell the downstream SIP entities to hide the identity.
3. Optionally, only if the 'pai_clir' preference is set, the P-Asserted Identity PAI header is replaced with 'anonymous@anonymous.invalid'.

Copyright © Sipwise GmbH - All rights reserved

7.23.2. How to Activate It

It can be activated using the dedicated 'clir' caller's preference in the administrative interface or in the CSC panel. Alternatively it can be enabled permanently or per single call by the subscriber using the Vertical Service Code (VSC) Interface.

By default, the 'clir' feature is not activated for calls done between subscribers belonging to the same PBX customers. In case it is required to anonymize also internal PBX calls, then the 'clir_intrapbx' has to be enabled as well.

The 'clir_override' callee's preference can be used to force the display of the caller number even if the CLIR feature was originally activated.

7.24. Connected Line Identification Restriction

The Connected Line Identification Restriction (COLR) feature prevents the calling party's name and number from being displayed to the caller party. It is the opposite of Connected Line Identification Presentation (COLP).

7.24.1. How It Works

When COLR is activated the Sipwise C5 platform does the following:

1. The P-Asserted Identity PAI header in the upstream messages (e.g. the 180 Ringing and then 200 OK messages) is replaced with 'anonymous@anonymous.invalid'.

7.24.2. How to Activate It

It can be activated using the dedicated 'colr' callee's preference in the administrative interface or in the CSC panel. Alternatively it can be enabled permanently or per single call by the subscriber using the Vertical Service Code (VSC) Interface.

7.25. Do Not Disturb

The Do Not Disturb (DND) feature allows users to block incoming calls temporarily. When activated, calls are diverted to a predefined message informing the caller that the user is not accepting calls. The DND helps avoid interruptions during meetings, rest periods, or focused work.

The DND feature can be activated in 3 different ways:

- by the administrators from the administrative web interface.
- by the subscriber itself using the Vertical Service Code (VSC) Interface.
- by the subscriber itself using the CSC panel, if activated.

The default return code and reason in case of call rejected because of DND can be customized using the config.yml parameters: `kamailio.proxy.early_rejects.dnd.announce_code` and `kamailio.proxy.early_rejects.dnd.announce_reason`. The announcement can be customized as well using the sound_set handle 'callee_dnd' under the 'early_rejects' group.

 Copyright © Sipwise GmbH - All rights reserved

7.26. The Vertical Service Code Interface

Vertical Service Codes (VSC) are codes a user can dial on his phone to provision specific features for his subscriber account. The format is *<code>*<value> to activate a specific feature, and #<code> or #<code># to deactivate it. The *code* parameter is a two-digit code, e.g. 72. The *value* parameter is the value being set for the corresponding feature.

IMPORTANT The *value* user input is normalized using the Rewrite Rules Sets assigned to domain as described in Configuring Rewrite Rule Sets.

By default, the following codes are configured for setting features. The examples below assume that there is a domain rewrite rule normalizing the number format *0<ac><sn>* to *<cc><ac><sn>* using *43* as country code.

7.26.1. Call Forward Setup Using Vertical Service Codes

- **72** - Call Forward Unconditional

  `*72*<dest_number>` creates the Call Forward Unconditional, e.g. to 431000 by dialing `*72*01000`.

  `#72` deletes the Call Forward Unconditional.

- **90** - Call Forward on Busy

  `*90*<dest_number>` creates the Call Forward on Busy, e.g. to 431000 by dialing `*90*01000`.

  `#90` deletes the Call Forward on Busy.

- **92** - Call Forward on Timeout

  `*92*<timeout>*<dest_number>*` creates the Call Forward on Timeout, e.g. after 30 seconds of ringing to 431000 by dialing `*92*30*01000`.

  `#92` deletes the Call Forward on Timeout.

- **93** - Call Forward on Not Available

  `*93*<dest_number>` creates the Call Forward on Not Available, e.g. to 431000 by dialing `*93*01000`.

  `#93` deletes the Call Forward on Not Available.

- **96** - All Call Forwards

  `#96` deletes at once all the Call Forwards previously configured using VSC.

  `#96#` disable at once all the Call Forwards previously configured using VSC.

7.26.2. Manage Recent Calls Using Vertical Service Codes

- **95** - Redial the Last Dialed Number

  `*95*` allows to redial the last dialed number by the subscriber. Note: the feature has to be enabled for the subscriber/domain using preference *last_number_redial*.

- **71** - Play the Last Caller's Number

  `*71*` allows to hear a voice announcement of the last caller's number, after the announcement dial the key defined in *semsvsccallback_last_caller_confirmation_key* to return the call. Note: it is not possible to return a call if the caller's number is unavailable.

Copyright © Sipwise GmbH - All rights reserved

- **74** - Redial the Last Caller's Number

  `*74*` allows to return the call to the last caller's number who called you without hearing the last call ID announcement. Note: it is not possible to return a call if the caller's number is unavailable.

- **20** - Delete Recent Calls records

  `*20*` allows to remove the records of the recent calls to and from you.

IMPORTANT

In order to use the feature codes related to recent calls, the *kamailioproxy store_recentcalls* preference in */etc/ngcp-config/config.yml* has to be set to `yes`. Additionally, to handle caller numbers coming from Peers (e.g. PSTN), set the *kamailioproxyforeign_domain_via_peer* preference to `yes` in */etc/ngcp-config/config.yml* file.

7.26.3. General Vertical Service Codes

- **50** - Speed Dial Slot

  `*50*<slot_number><dest_number>` sets a new Speed Dial Slot, e.g. set slot 1 to 431000 by dialing `*50*101000`, which then can be used by dialing `*1`.

- **55** - One-Shot Reminder Call

  `*55*<time>` sets One-Shot Reminder Call, e.g. to 08:30 by dialing `*55*0830`.

- **31** - Calling Line Identification Restriction

  `*31*<num_to_call>` calls the provided number enabling/disabling Calling Line Identification Restriction for that single call, e.g. to call 431000 anonymously make sure to have subscriber's caller preferences 'clir' disabled and dial `*31*01000`. If 'clir' preference is enabled, this will disable Calling Line Identification Restriction for this particular call.

  `*31*` enables Calling Line Identification Restriction permanently.

  `#31#` disables Calling Line Identification Restriction permanently.

  `**31#` enables/disables Calling Line Identification Restriction permanently depending on the previous state.

- **32** - Block Incoming Anonymous Calls

  `*32*` enables Block Incoming Anonymous Calls.

  `#32#` disables Block Incoming Anonymous Calls.

- **77** - Connected Line Identification Restriction

  `*77*` enables Connected Line Identification Restriction permanently.

  `#77#` disables Connected Line Identification Restriction permanently.

  `**77#` enables/disables Connected Line Identification Restriction permanently depending on the previous state.

- **78** - Do not Disturb (DND)

  `*78*` enables Do not Disturb (DND).

  `#78#` disables Do not Disturb (DND).

  `**78#` enables/disables Do not Disturb (DND) depending on the previous state.

- **80** - Call Block Override PIN

Copyright © Sipwise GmbH - All rights reserved

`*80*<pin><num_to_call>` calls the provided number using Call Block Override PIN. The PIN has to be the one 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.

- **88** - Deactivate features at once

  `#88` deactivates preference and/or deletes all the Call Forwards at once. By default following features will be disabled: "cfu,cfb,cft,cfna,cfs,cfo,cfr,clir,colr,dnd". It can be customized in the sems configuration files with the parameter 'vsc_off_pref_list'. ATTENTION: at the moment the module supports only CFs and boolean preferences.

7.26.4. Configuration of Vertical Service Codes

You can change any of the codes (but not the format) in */etc/ngcp-config/config.yml* in the section *semsvsc*. After the changes, execute *ngcpcfg apply "changed VSC codes"*.

CAUTION If you have the EMTAs under your control, make sure that the specified VSCs do not overlap with EMTA-internal VSCs, because the VSC calls must be sent to the Sipwise C5 via SIP like normal telephone calls.

7.26.5. Voice Prompts for Vertical Service Code Configuration

Table 16. 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.

Copyright © Sipwise GmbH - All rights reserved

Prompt Handle	Related VSC	Message
vsc_cfna_off	93 (Call Forward on Not Available)	Your call forward while not reachable has successfully been deactivated.
vsc_speeddial	50 (Speed Dial Slot)	Your speed dial slot has successfully been stored.
vsc_reminder_on	55 (One-Shot Reminder Call)	Your reminder has successfully been activated.
vsc_reminder_off	55 (One-Shot Reminder Call)	Your reminder has successfully been deactivated.
vsc_blockinclir_on	32 (Block Incoming Anonymous Calls)	Your rejection of anonymous calls has successfully been activated.
vsc_blockinclir_off	32 (Block Incoming Anonymous Calls)	Your rejection of anonymous calls has successfully been deactivated.
vsc_clir_on	31 (Calling Line Identification Restriction)	Your calling line identification restriction service has successfully been activated.
vsc_clir_off	31 (Calling Line Identification Restriction)	Your calling line identification restriction service has successfully been deactivated.
vsc_colr_on	77 (Connected Line Identification Restriction)	Your connected line identification restriction service has successfully been activated.
vsc_colr_off	77 (Connected Line Identification Restriction)	Your connected line identification restriction service has successfully been deactivated.
vsc_dnd_on	78 (Do not Disturb)	Your do not disturb service has successfully been activated.
vsc_dnd_off	78 (Do not Disturb)	Your do not disturb service has successfully been deactivated.
vsc_service_off	88 (Deactivate features at once)	All your feature services have successfully been deactivated.

7.27. Call Recording

7.27.1. Introduction to Call Recording Function

The Sipwise C5 provides an opportunity to record call media content and store that in files.

Some characteristics of the Call Recording:

- Call Recording function can store both unidirectional (originating either from the caller, or from the

 Copyright © Sipwise GmbH - All rights reserved

callee) or bidirectional (combined) streams from calls, resulting in 1, 2 or 3 physical files as output respectively.

- The location and format of the files is configurable.
- File storage is planned to occur on an NFS shared folder.
- Media files can be stored on an NFS shared folder, in the central database, and/or in a cloud storage service. Storage on the local file system is also possible, but not recommended due to the increased I/O load that it can cause.
- Activation of call recording may happen generally for a *Domain / Peer / Subscriber* through the Sipwise C5 admin web interface.

IMPORTANT Sipwise C5's Call Recording function is not meant for individual call interception purpose! Sipwise provides its Lawful Interception solution for that use case.

- Querying or deletion of existing recordings may happen through the REST API.
- Listing recordings of a subscriber is possible on Sipwise C5's admin web interface.

The Call Recording function is implemented using Sipwise C5's ***rtpengine*** module.

NOTE There are 2 *rtpengine* daemons employed when call recording is enabled and active. The *main rtpengine* takes care of forwarding media packets between caller and callee, as usual, while the *secondary rtpengine* (recording) daemon is responsible for storing call data streams in the file system.

Call Recording is disabled by default. Enabling and configuration of Call Recording takes place in 2 steps:

1. Enabling the feature on the Sipwise C5 by setting configuration parameters in the main `config.yml` configuration file.
2. Activating the feature for a *Domain / Peer / Subscriber.*

7.27.2. Information on Files and Directories

Sipwise C5's Call Recording function can save recorded streams to an **NFS shared folder**, and/or into the **central database** as binary blobs, and/or upload them to a **cloud storage service** such as Amazon S3 or Google Cloud Storage. Multiple storage options can be enabled simultaneously.

IMPORTANT Since the call data amount may be significant (depending on the number of calls and the duration of calls), storing recorded streams on Sipwise C5's local disk is *highly discouraged*. However if you *have to* store recorded streams as files in the local file system, please contact the Sipwise Support team in order to get the appropriate configuration of the Call Recording function.

NFS Storage

The NFS share gets mounted during startup of the recording daemon. If the NFS share cannot be mounted for some reason, the recording daemon will not start.

File names have the format: `<call_ID>-<random>-<SSRC>.<extension>` by default, where:

Copyright © Sipwise GmbH - All rights reserved

- `call_ID:` SIP Call-ID of the call being recorded
- `random:` is a string of random characters, unique for each recorded call. Its purpose is to avoid possible filename collisions if a Call-ID ever gets reused.
- `SSRC:` is the RTP SSRC for unidirectional recordings, or "mix" for the bidirectional (combined) audio.
- `extension:` is either "mp3" or "wav", depending on the configuration (rtpengine.recording.output_format)

There might be 1, 2 or 3 files produced as recorded streams. The **number of files** depends on the configuration:

1. `rtpengine.recording.output_mixed = 'yes'` (combined stream required)

   `rtpengine.recording.output_single = 'no'` (unidirectional streams not required)

2. `rtpengine.recording.output_mixed = 'no'` (combined stream not required)

   `rtpengine.recording.output_single = 'yes'` (unidirectional streams required)

3. `rtpengine.recording.output_mixed = 'yes'` (combined stream required)

   `rtpengine.recording.output_single = 'yes'` (unidirectional streams required)

The format of the file names can be configured by setting the output pattern (see below) in the configuration file.

Database Storage

Each recorded call and each recorded stream will always produce a respective entry in the database, but normally only metadata is stored. When database storage is enabled, the actual content of the recorded stream will be stored in the database as a binary blob in addition to the metadata.

Cloud Storage

Recorded streams can be uploaded to a supported cloud storage service. Currently Amazon S3 (AWS) and Google Cloud Storage are supported, as well as services providing a compatible API (e.g. MinIO). The names of uploaded objects follow the same pattern as the file names described under **NFS Storage**.

Required configuration parameters are described below.

7.27.3. Configuration

The Call Recording function can be enabled and configured on the Sipwise C5 by changing the following configuration parameters in `config.yml` file:

```
rtpengine:
  ...
  recording:
    egress: no
    enable: no
    mix_method: direct
```


Copyright © Sipwise GmbH - All rights reserved

```
    mp3_bitrate: '48000'
    nfs_host: 192.168.1.1
    nfs_remote_path: /var/recordings
    output_chgrp: _ngcp-admin
    output_chmod: '0660'
    output_chmod_dir: '0770'
    output_chown: root
    output_dir: /var/lib/rtpengine-recording
    output_format: wav
    output_mixed: yes
    output_pattern: '%c-%r-%t'
    output_single: yes
    resample: no
    resample_to: '16000'
    spool_dir: /var/spool/rtpengine
    storage_db: no
    storage_file: yes
    storage_gcs:
      api_key: ''
      enable: no
      scope: ''
      service_account: ''
      tls_verify: yes
      uri: ''
    storage_local: no
    storage_memory: no
    storage_s3:
      access_key: ''
      enable: no
      host: ''
      path: ''
      port: 0
      region: ''
      secret_key: ''
      tls_verify: yes
    transcribe:
      enable: no
    vsc:
      pause_recording: ''
      pause_resume_recording: ''
      start_pause_resume_recording: ''
      start_recording: ''
      start_stop_recording: ''
      stop_recording: ''
```

Enabling Call Recording

Enabling the function requires changing the value of `rtpengine.recording.enable` parameter to "yes". In order to make the new configuration active, it is necessary to use the following command:

Copyright © Sipwise GmbH - All rights reserved

```
ngcpcfg apply 'Activated call recording'
```

Description of configuration parameters:

- `enable`: when set to "yes" Call Recording service is enabled. Default: `no`
- `egress`: when enabled, uses RTP streams leaving the Sipwise C5 platform for recording purposes, instead of received RTP
- `mix_method`: for mixed output, select between mixing all input audio streams into a single output audio stream (`direct`), or into a multi-channel output file with one channel per stream (`channels`)
- `mp3_bitrate`: the bit rate used when recording to MP3 format. Default: `48000` (48 kbps)
- `nfs_host`: IP address of the NFS host that provides storage space for recorded streams. Default: `192.168.1.1`. Set to an empty string if NFS storage is not desired
- `nfs_remote_path`: the remote path (folder) where files of recorded streams are stored on the NFS share. Default: `/var/recordings`
- `output_chgrp`: Unix group to change file ownership to after recording has finished. Default: `_ngcp-admin` to allow access to the files from the web panel
- `output_chmod`: Unix permissions to set for recorded files. Default: `0660` to allow read-only access from file owner and group members
- `output_chmod_dir`: Unix permissions to set for created directories. Default: `0770` to allow read-only access from file owner and group members
- `output_chown`: Unix user to set file ownership to. Default: `root` to disallow modifications from non-administrator users
- `output_dir`: is the local mount point for the NFS share, and thus where the final audio files will be written. Default: `/var/lib/rtpengine-recording`

  CAUTION Normally you do not need to change the default setting. If you do change the value, please be aware that the recorded files will be written by *root* user in that directory.

- `output_format`: possible values are `wav` (uncompressed PCM) or `mp3` (MPEG layer 3 compressed). Default: `wav`
- `output_mixed`: `yes` means that a file containing a mixed stream of caller and callee voice data is produced. Default: `yes`
- `output_pattern`: allows customising the format of output file names. Default: `%c-%r-%t` matching the format described above
- `output_single`: `yes` means that a separate file for each stream direction is produced, i.e. for each stream originating from caller and callee. Default: `yes`
- `resample`: when set to `yes` the call data stream will be resampled before storing it in the file. Default: `no` to record with the original sampling rate
- `resample_to`: the sample rate used for resampling output. Default: `16000` (16 kHz)
- `spool_dir`: is the place for temporary metadata files that are used by the recording daemon and the main rtpengine daemon for their communication. Default: `/var/spool/rtpengine`

  CAUTION It is advisable not to alter the default setting unless there is a valid reason to do

 Copyright © Sipwise GmbH - All rights reserved

so. Sipwise has tested the Call Recording function extensively with the default configuration.

- `storage_db`: enables or disables storage of recording content to the database. Default: `no`
- `storage_file`: enables or disables writing of recording files, either to NFS or to the local file system. Default: `yes`
- `storage_gcs`: contains parameters for uploads to Google Cloud Storage

  `api_key`: if authentication via service account (OAuth2/JWT) is not in use, a secret API key must be provided to access the storage service

  `enable`: to enable content uploads to GCS

  `scope`: authentication scope to request access for, when OAuth2/JWT authentication via service account is in use. If left blank, an automatic default value will be used, for example `https://www.googleapis.com/auth/cloud-platform`

  `service_account`: JSON service account file to use for OAuth2/JWT authentication. Must be readable by the daemon

  `tls_verify`: can be used to disable verification of TLS certificates if a local service is used. Default: `yes`

  `uri`: full storage URI to post uploads to. Normally contains the name of the storage bucket. Example: `https://storage.googleapis.com/upload/storage/v1/b/examplebucket/o`
- `storage_local`: when enabled, allows storage on the local file system instead of on an NFS shared folder. Default: `no`

CAUTION Do not enable local file storage unless the impact on the system load has been fully considered.

- `storage_memory`: relevant only if file storage (NFS or local) is disabled. For all non-file based remote storage options (database, cloud storage), normally recordings are first stored in a temporary file, which is then deleted after the content has been moved to its final storage place. With this option enabled, media content is temporarily stored in memory only. Default: `no`
- `storage_s3`: options to configure uploads to S3 (Amazon AWS, etc.) storage

  `access_key`: the "access key" required to authenticate to the service

  `enable`: to enable content uploads to S3

  `host`: host name (not URI) of the storage service. May or may not have to include the name of the storage bucket. Example: `examplebucket.s3.amazonaws.com` or `minio.example.com`

  `path`: path component of the object to upload. May be left blank, for example if the host name includes the name of the bucket, or may need to set to the name of the bucket. Default blank

  `port`: allows use of a storage service running on a non-standard port. Default zero (use default)

  `region`: region code to use for authentication and for uploads if required by the service. May be left blank. Example: `eu-west-2`

  `secret_key`: the secret authentication key (matching the access to) to use for authentication

  `tls_verify`: can be used to disable verification of TLS certificates if a local service is used. Default `yes`
- `transcribe`: to enable use of an external transcription service for recorded calls

Copyright © Sipwise GmbH - All rights reserved

- `vsc`: configures various vertical service codes to control recording from within a call (see below)

If Call Recording is enabled you can see 2 *rtpengine* processes running when checking the Sipwise C5 system state with the *ngcp-service* tool:

```
root@sp1:/etc/ngcp-config# ngcp-service summary
Ok Service                         Managed    Started   Status
-- ------------------------------- ---------- --------- ---------
…
   kamailio-lb                     managed    by-ha     active
   ngcp-voisniff                   managed    by-ha     active
   rtpengine                       managed    by-ha     active
   rtpengine-recording             managed    by-ha     active
…
```

Activating Call Recording

Activating Call Recording for e.g. a *Subscriber:* please use Sipwise C5's admin web interface or api for this purpose. On the web interface one has to navigate as follows: *SettingsSubscribersselect subscriberPreferencesNAT and Media Flow Controlrecord_call* and select one of the following options:

- Off (never): call recording disabled
- On (always): all the received or placed calls will be recorded
- On demand using VSC (initially disabled): call are not recorded by default but recording can be activated during the call using DTMF VSC codes
- On demand using VSC (initially enabled): all call are recorded by default but recording can be paused/stopped during the call using DTMF VSC codes

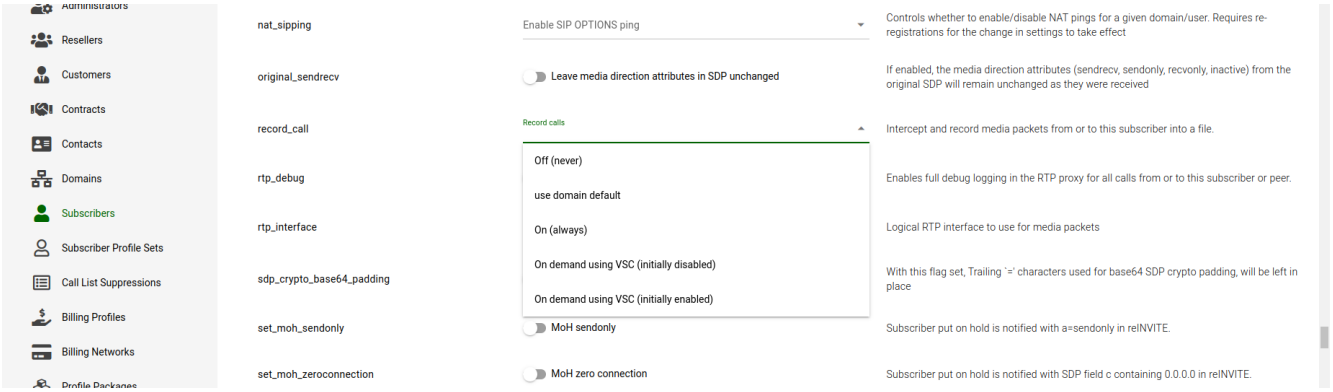

Figure 102. Activating Call Recording

NOTE The call recording function may be activated for a single *Subscriber, a Domain and a Peer server* in the same way: *Preferences NAT and Media Flow Control record_call*. When activating call recording for a *Domain or Peer* this effectively activates the function for all subscribers that belong to the selected domain, and for all calls with a local endpoint going through the selected peer server, respectively.

 Copyright © Sipwise GmbH - All rights reserved

7.27.4. Pre-Recording Announcement

Many country regulations require that an informative announcement is played to the caller before the call is actually recorded. The Sipwise C5 allows you to configure your own custom announcement with few simple steps.

First create a system sound set for the feature. In *SettingsSound Sets* either use your already existing *Sound Set* or create a new *Sound Set* and then assign it to your domain or subscribers. In the *Sound Set* there is an announcement *early_mediaannounce_before_recording* for that purpose.

Once the *Sound Set* is created the subscriber's preference *play_announce_before_recording* of the callee must be enabled under *SubscriberPreferencesApplications* menu. The same parameter can be set in the Domain's or Customer's preferences to enable this feature for all its subscribers. Set the preference to "Always" or "External/Internal calls only" depending if you want to play the message only for incoming calls from PSTN or only for incoming calls from local subscribers. There also exists the options "Never", and "default".

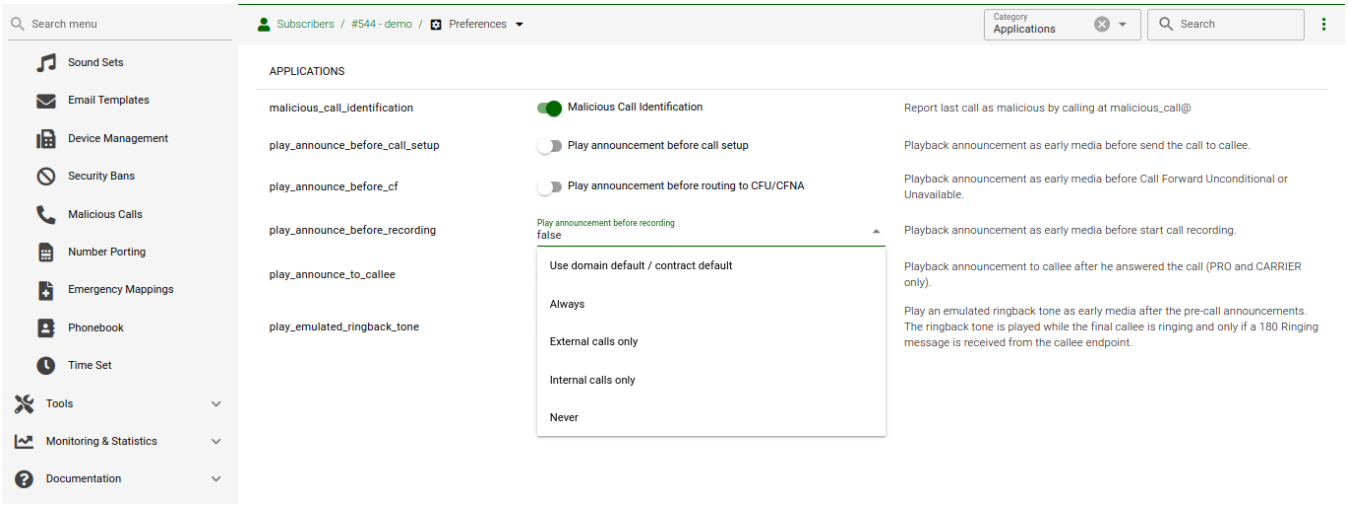

Figure 103. Setting Call Recording Announcement on Subscriber Preferences

NOTE The announcement will be played to the caller before the call is routed to the callee.

IMPORTANT In case of **CFU** or **CFNA** with Pre-Recording Announcement feature enabled on both the forwarder and the final callee, only the Announcement of the final callee will be played to the caller. In case of **CFB** and **CFT**, instead, the announcement of the forwarder will be played first, then the announcement of final callee will be played after the call forward is executed.

7.27.5. Call Recording VSC

The call recording can also be activated and terminated during the call using VCS codes. In particular, the possible actions are:

- pause_recording: the recording is put on pause, the recording continues but no audio is captured (default code not assigned)
- pause_resume_recording: the recording is put on pause or resumed depending on the previous state (default code not assigned)
- start_pause_resume_recording: the recording is put on pause, resumed or started depending on the

Copyright © Sipwise GmbH - All rights reserved

previous state (default code not assigned)

- start_recording: the recording is started (default code not assigned)
- start_stop_recording: the recording is started or stopped depending on the previous state (default code not assigned)
- stop_recording: the recording is stopped (default code not assigned)

To customize the default VSC codes, see section `rtpengine.recording.vsc` in `config.yml` file. An acoustic signal will inform the participants of the beginning/ending of the registration. The acoustic signal can be customized editing the `kamailio.proxy.vcs_recording_announce_file` config.yml option.

7.27.6. Call Recording List

It is possible to **list existing call recordings** of a *Subscriber* through the admin web interface of the Sipwise C5. In order to do so, please navigate to: *Settings* *Subscribers* *select subscriber* *Details* *Call Recordings*

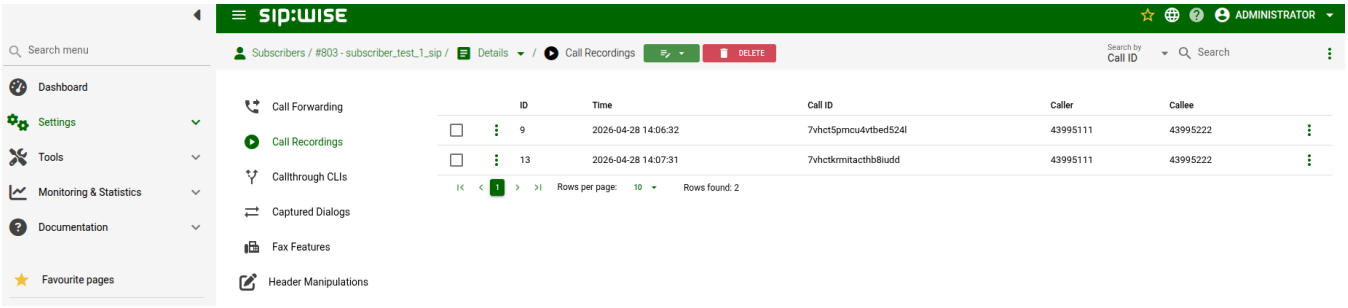

Figure 104. Listing Call Recordings

If you select an item in the list, besides the main properties such as the time of call and the SIP Call-ID, you can retrieve the details of the related call (press the *Call Details* button), get the list of recorded files (press the *Recorded Files* button) or *Delete* the recorded call.

When selecting *Call Details* you will see the most important accounting data of the call. Furthermore you can see the SIP *Call Flow* or the complete *Call Details* if you press the respective buttons.

 Copyright © Sipwise GmbH - All rights reserved

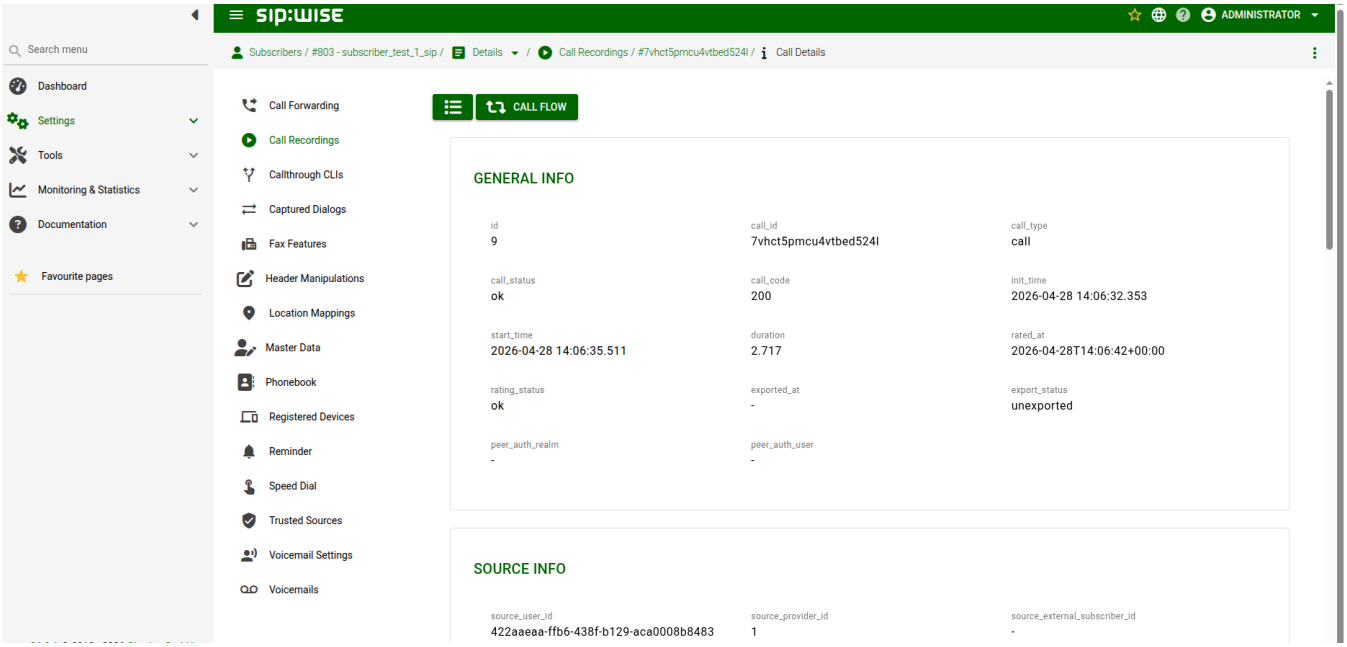

Figure 105. Listing Call Recordings Details

When navigating to *Recorded Files* of a call you will be presented with a list of files. For each file item:

- `type` of stream is shown, that can be either "mixed" (combined voice data), or "single" (voice data of caller or callee)
- file `format` is shown, that can be either "wav", or "mp3"
- you can download the file by pressing the *Play* button

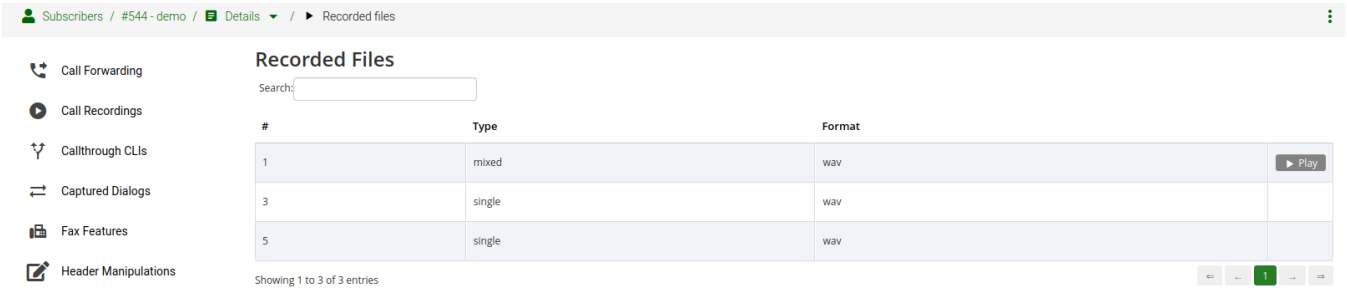

Figure 106. Listing Files for a Recording

7.27.7. REST API

The Sipwise C5 REST API provides methods for querying and deletion of existing recording data. The full documentation of the available API methods is available on the admin web interface of the Sipwise C5, as usual.

The following API methods are provided for managing Call Recordings:

- CallRecordings:

  Provides information about the calls recorded in the system; can also be used to delete a recording entry

  accessible by the path: `/api/callrecordings` (collection) or `/api/callrecordings/id`

Copyright © Sipwise GmbH - All rights reserved

(single item)

Supported HTTP methods: OPTIONS, GET, DELETE

- CallRecordingStreams:

  Provides information about recorded streams, such as start time, end time, format, mixed/single type, etc.; can also be used to delete a recorded stream

  accessible by the path: /api/callrecordingstreams (collection) or /api/callrecordingstreams/id (single item)

  Supported HTTP methods: OPTIONS, GET, DELETE

- CallRecordingFiles:

  Provides information about recorded streams, such as start time, end time, format, mixed/single type, etc.; additionally returns the file content too

  accessible by the path: `/api/callrecordingfiles` (collection) or `/api/callrecordingfiles/id` (single item)

  Supported HTTP methods: OPTIONS, GET

7.28. Media Transcoding and Transrating

7.28.1. Overview

Sipwise C5 offers the capability to convert RTP media between several supported codecs, a feature known as transcoding. While this feature is always available on Sipwise C5, it is engaged only when a subscriber, peer, or domain is explicitly configured for it. By default, Sipwise C5 lets RTP endpoints negotiate the codec to use among themselves without interfering.

IMPORTANT	Media transcoding is a relatively CPU-intensive feature. As such, each individual node of a Sipwise C5 performing media transcoding can only support a limited number of concurrent calls for which transcoding is active.

7.28.2. Supported Codecs

The following audio codecs, which are commonly found in use by SIP/RTP clients, are currently supported for transcoding.

- G.711 (μ-Law and a-Law)
- G.722
- G.723.1
- G.729
- GSM
- AMR (narrowband and wideband, the latter also known as AMR-WB)
- Opus
- Speex
- DTMF event packets (**telephone-event**)
- Comfort noise

 Copyright © Sipwise GmbH - All rights reserved

Some codecs operate at different sampling rates than other codecs. If transcoding happens between two such codecs, the audio will be resampled as necessary. Similarly, if transcoding happens between a mono (1-channel) and a stereo (2-channel) codec, the audio will be up-mixed and down-mixed as necessary.

7.28.3. Configuration

Transcoding can be engaged for individual subscribers, peers, or domains on their respective preferences page in the Sipwise C5 admin web interface.

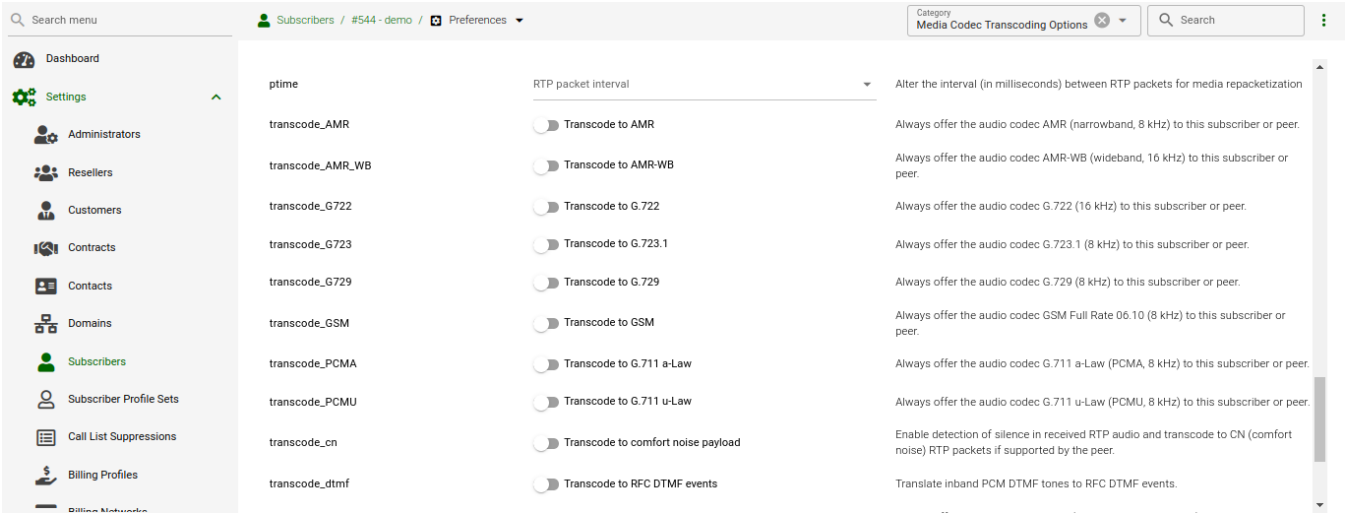

Figure 107. Transcoding Configuration

Setting any of the transcoding options for a domain makes it affect all the subscribers in this domain.

Individual options are described below.

ptime

Packetisation time in milliseconds. Normally the Sipwise C5 lets the RTP endpoints select and negotiate the packetisation time they want to use. Setting this option to anything other than `unchanged` will engage transrating via the transcoding engine towards this subscriber or peer, even if none of the other transcoding options are set, in which case the media will be transrated (repacketised).

For example, setting this to `40 ms` would mean that each RTP packet sent towards this subscriber or peer would contain 40 milliseconds worth of audio, even if the other side of the call sends media that is packetised differently. It would also make the Sipwise C5 indicate towards this subscriber or peer that it would prefer to receive audio in 40 millisecond packets (through the `a=ptime` SDP attribute).

transcode_...

Enabling one of these options adds the selected codecs to the list of codecs offered to this subscriber or peer, even if the original list of offered codecs did not include it. If this additional codec ends up being accepted by this subscriber or peer, then it will be transcoding to the first supported codec that was originally offered.

For example, if a calling RTP client A indicates support for PCMA (G.711 a-Law) as well as G.722, and calls a subscriber B that is configured for transcoding to G.729, then subscriber B would be offered

Copyright © Sipwise GmbH - All rights reserved

PCMA, G.722, and G.729 by the Sipwise C5. If subscriber B then accepts G.729 and starts sending G.729, the Sipwise C5 would engage its transcoding engine and transcode the audio to PCMA (because PCMA and not G.722 was the codec preferred by A) before forwarding it to A. Vice versa, PCMA arriving from A would be transcoded to G.729 before being sent to B. (If B were to reject G.729 and instead starts to send PCMA or G.722, no transcoding would happen.)

Notes on individual codecs:

- **AMR** is available in both narrowband (`AMR` operating at 8 kHz) and wideband (`AMR-WB` operating at 16 kHz) variants. These are distinct codecs and can be configured for transcoding separately or together.
- **Opus** always operates at 48 kHz, but is supported in both mono and stereo (1 and 2 audio channels respectively). Both can be offered at the same time if so desired.
- **Speex** is supported at sampling rates of 8, 16, and 32 kHz. These can be configured separately for transcoding, or together.
- **DTMF** is not an actual audio codec, but rather represents transcoding between DTMF event packets and in-band DTMF audio tones. This is described in more detail below.
- **CN** refers to comfort noise payloads. More information about this is below.

…_bitrate

Some codecs (Opus and G.723.1 in particular) can be configured for different bitrates, which would impact the amount of network bandwidth they use, as well as the audio quality produced. For Opus, different bitrates can be selected for their mono and stereo instances. Selecting a bitrate has no effect if transcoding to the respective codec is not engaged.

AMR-Specific Options

AMR and AMR-WB support dynamic mode switching (adapting the bitrate) during runtime. The configurable bitrate therefore is only the initial bitrate at which the encoder is started, and is further constrained by the **mode-set** option.

The **mode-set** is a list of comma-separated AMR modes, e.g. **0,1,2,3**. If a **mode-set** is received from a remote peer, then only the modes (bitrates) given in the list will be used for the encoder. If a specific bitrate is also configured, then this bitrate will be used as the highest possible starting bitrate. If this bitrate is not allowed by the **mode-set**, then the next lower bitrate will be used. If no lower bitrates are permitted, then the next higher bitrate will be used.

For outgoing AMR offers, a **mode-set** can be configured in the preferences page, and it will be honoured just like a **mode-set** that is received from a remote peer. A remote AMR peer may support only a specific subset of AMR modes, which would need to be configured in the **mode-set**. The Sipwise C5 supports encoding and decoding all possible AMR modes.

If no **mode-set** is received nor configured, then all bitrates are permitted.

AMR has two basic payload variants: **bandwidth-efficient** and **octet-aligned**. A remote AMR peer may support only one of them, in which case the correct one must be configured in the preferences. The default is **bandwidth-efficient**. The Sipwise C5 supports sending and receiving both and will honour the request made by a remote AMR peer for one or the other.

Further options that may be required by a remote AMR peer to restrict permissible mode changes are:

 Copyright © Sipwise GmbH - All rights reserved

mode-change-period restricts mode changes to be made only every other packet if set to **2** (the default is **1**); **mode-change-capability** advertises the capability to restrict the mode changes as such without actually restricting them; and **mode-change-neighbor** restricts mode changes to be made only to neighbouring modes if enabled (the default is disabled).

AMR supports requesting specific encoder modes from the remote peer via **Codec Mode Requests** (CMR). The Sipwise C5 will honour a CMR received from the remote peer and switch encoder mode accordingly. Then optionally, if no further CMRs are received from the peer, the Sipwise C5 can then try to increase encoder bitrate again to improve audio quality. This can be enabled by setting the **mode change interval** preference to non-zero. It specifies an interval in milliseconds at which to switch to a higher encoder bitrate if no CMR was received during that time, and if a higher bitrate is available and allowed by the **mode-set**.

In the opposite direction, the Sipwise C5 can optionally request higher bitrates from the remote peer by sending CMRs out. This can be enabled by setting the **CMR interval** preference to non-zero. It specifies an interval in milliseconds at which the Sipwise C5 will request a higher bitrate from the remote encoder if a higher bitrate is available and it was not being used during that time.

always_transcode

Setting this flag instructs the Sipwise C5 to always engage transcoding to the first (preferred) codec indicated by an RTP endpoint, even if another codec is available that is supported by both parties to a call. Enabling this flag can potentially engage the transcoding engine for a call even if none of the other transcoding options are set.

For example: Subscriber A is calling subscriber B. Subscriber A is indicating support for PCMA and G.722. Subscriber B answers the call, rejects PCMA but accepts G.722, and starts sending G.722 to A. Normally the Sipwise C5 would not get involved and would let G.722 pass between A and B. But if subscriber B has the always_transcode flag set, the Sipwise C5 would now start transcoding the G.722 sent by B into PCMA before forwarding it to A, because PCMA was indicated as the preferred codec by A. Vice versa, PCMA arriving from A would be transcoded into G.722 and then forwarded to B.

DTMF Transcoding

The Sipwise C5 supports transcoding between DTMF event packets (using the RTP telephone-event type payload) and DTMF tones carried in-band in the audio stream. DTMF transcoding is supported in both directions: transcoding DTMF event packets to DTMF tones, and DTMF tones in an audio stream and transcoding them to DTMF event packets.

Support for DTMF transcoding can be enabled in one of two ways:

- Enabling the setting transcode_dtmf for a subscriber, peer, or domain. This is useful if the subscriber, peer, or domain requires support for DTMF event packets, but the calling entity might only support DTMF tones carried in-band in the audio stream.
- Enabling the setting always_transcode for a subscriber, peer, or domain. This is useful for the reverse case: if the subscriber, peer, or domain might only support DTMF tones carried in-band in the audio stream, but the calling entity requires support for DTMF event packets.

Enabling DTMF transcoding for any call requires that all audio passes through the transcoding engine, as well as a DSP for detecting DTMF tones in one direction. This carries an additional performance impact with it, and so DTMF transcoding should only be enabled when absolutely necessary.

Copyright © Sipwise GmbH - All rights reserved

DTMF Conversion of INFO Messages

The Sipwise C5 supports the conversion of SIP INFO messages with application/dtmf-relay payload to DTMF event or PCM DTMF inband tone, depending on whether the destination participant supports the telephone-event RTP payload type or not. DTMF conversion of INFO messages works only in one way direction: DTMF events or PCM DTMF inband tones are not converted back to DTMF INFO messages.

The preference kamailio.proxy.allow_info_method has to be set to yes in config.yml in order to make the DTMF INFO message conversion working.

Enabling DTMF conversion of INFO messages for any call requires that all audio passes through the transcoding engine, as well as a DSP for detecting DTMF tones in one direction. This carries an additional performance impact with it, and so it should only be enabled when absolutely necessary.

NOTE At the moment this feature is for internal use only. External generated INFO messages will not be converted but passed through.

Comfort Noise

RTP supports a special payload format for carrying comfort noise (CN) audio without having to encode it as actual audio. Support for CN is optional and negotiation is normally left up to the endpoints. If CN transcoding is enabled, the Sipwise C5 can transcode CN payloads to audio noise and vice versa.

When active and when CN is supported by one endpoint but not the other, any received CN payloads are converted to audio noise and this noise is then inserted into the audio stream. In the reverse direction, the Sipwise C5 can optionally detect silence audio frames and then convert these to predefined CN payloads.

To enable silence detection, the setting rtpengine.silence_detect in config.yml must be set to a non-zero value. This value denotes the silence detection threshold in percent and therefore must be between 0.0 and 100.0. Any audio samples that are lower than the configured threshold will be treated as silence, and if all audio samples in an audio frame are detected as silence, then that frame will be replaced by a CN frame. To only replace audio frames that contain complete silence with CN frames, a very low but non-zero value can be used here, such as 0.1%.

NOTE Certain audio codecs always leave a residual DC offset in the audio samples, even when encoding complete silence. G.711 a-Law (PCMA) is one such example, for which the minimum silence detection threshold is 0.013%.

When a silence audio frame is replaced by a CN frame, the generated CN payload is not based on the audio that is being replaced, but rather taken from the static and predefined setting rtpengine.cn_payload in config.yml. This is a list of comfort noise parameters according to RFC 3389. The first value in the list is mandatory and denotes the volume of the noise payload in negative dBov (meaning larger values result in quieter audio, with zero being the loudest), while all subsequent values are optional and denote spectral information about the noise. The default is the single value 32, which describes noise with a volume of -32 dBov without any spectral information.

7.28.4. T.38 Transcoding

In addition to transcoding between audio codecs, the Sipwise C5 supports transcoding between T.38 fax transmissions (over UDPTL transport) and T.30 fax data over regular audio channels. The audio codec commonly used to carry T.30 data is G.711, but any other audio codec that is supported for transcoding can also be used, provided it offers a high enough bitrate and audio quality.

 Copyright © Sipwise GmbH - All rights reserved

Two settings to control T.38 transcoding are available for subscribers, peers, and domains:

- The setting t38_decode instructs the Sipwise C5 to accept an offered T.38 session towards the subscriber, peer, or domain, and translate it into a regular audio call carrying T.30 fax data. By default, G.711 (both µ-Law and a-Law) will be offered. If any other codecs are selected through the transcode_... options, then only those codecs will be offered and the G.711 default will be omitted. Non-T.38 offers are not affected by these settings.
- The setting `t38_force` forces any audio call towards this subscriber, peer, or domain to be translated to a T.38 session, regardless of whether the audio media actually carries T.30 fax data or not. This is useful if it is known that the remote destination is a fax endpoint supporting T.38.

7.28.5. DTX Jitter Buffer

DTX is short for **discontinuous transmission** and describes an event during which a remote RTP client intermittently stops sending RTP. Some codecs explicitly support this as a feature (e.g. AMR and AMR-WB) while in other instances a sending RTP client may erroneously produce gaps in the RTP stream. If a receiving RTP client is not expecting such a behaviour, the resulting audio output may have unpleasant characteristics, such as stuttering or dropping the audio to complete silence.

By default the Sipwise C5 will not attempt to correct such a DTX behaviour and will simply pass through any DTX gaps untouched. However, for any call with active audio transcoding, the Sipwise C5 can be configured to enable a DTX jitter buffer in order to fill in DTX gaps with comfort noise to produce a steady output RTP stream.

Configuration

The master switch to enable the DTX jitter buffer is found in `config.yml` under the setting `rtpengine.dtx_delay`. This value denotes the processing delay in milliseconds (i.e. the length of the jitter buffer) and should be set to just slightly higher than the highest expected ingress jitter. The default value of zero disables the DTX buffer.

When a DTX event is detected, the gap will be filled in using an RFC 3389 compliant comfort noise generator. The parameters for this generator are set under `rtpengine.dtx_cn_params`. The default is just a single value of 35, specifying white noise at -35 dBov. Higher numbers lead to quieter noise. Subsequent numbers (i.e. from the second number onward) are optional spectral parameters.

The maximum duration for which comfort noise should be generated can be set using `rtpengine.max_dtx` in seconds. The default is unlimited (zero).

If an incoming RTP stream experiences a timer drift (i.e. the RTP clock runs slower or faster than expected), then the timer handling the DTX buffer will be shifted forward or backward by `rtpengine.dtx_shift` milliseconds (default 5). Timer drifts can lead to DTX buffer overflows, which are detected if more than `rtpengine.dtx_buffer` packets with an RTP delay of more than `rtpengine.dtx_lag` milliseconds are held in the buffer (defaults 10 and 100, respectively).

AMR and AMR-WB explicitly support DTX and provide their own comfort noise format (called SID) to fill in DTX gaps. If `rtpengine.amr_dtx` is set to `native` (the default) then the native AMR SID/CN generator will be used to produce the noise to fill in DTX gaps. If this setting is set to `CN` then the generic CN generator (which is used for other codecs) is used instead.

Copyright © Sipwise GmbH - All rights reserved

7.29. Transcription Services

Voicemail messages as well as recordings from the call recording feature can optionally be sent to an external service to be transcribed into a textual form.

As these services are not provided directly by the Sipwise Sipwise C5, it is the responsibility of the operator of the platform to consider all possible implications (e.g. related to privacy laws) of sending voice data to an external service and have it transcribed.

7.29.1. Primary Configuration

The interaction with the external transcription service is facilitated by the platform's task-agent, and as such the task-agent service must be enabled in `config.yml`:

```
task_agent:
  enable: yes
```

Furthermore, the transcription component itself must be configured:

```
transcripts:
  service: whisper_local
  node: '*|state=active,role=web'
  auth: api_key
  api_key: ''
  oauth:
    algorithm: RS256
    scope: https://www.googleapis.com/auth/cloud-platform
    service_key_file: /etc/ngcp-config/shared-files/.....json
    uri: https://oauth2.googleapis.com/token
  google:
    ...
  lemonfox:
    ...
  openai:
    ...
  whisper_local:
    ...
```

The `service` parameter is used to select from one of the supported transcription services. Currently this is OpenAI, Google, or Lemonfox. The `whisper_local` service is available as an experimental proof-of-concept provider using a locally-run instance of Whisper and is not meant for production use.

The `node` parameter is in task-agent syntax and is relevant in multi-node platform setups (i.e. PRO and CARRIER). It describes one or several nodes of the platform to which to delegate the transcription task to. As this typically involves communicating with an external API, the node in question must be able to access that API. The default nominates the active "web" node to facilitate this.

NOTE Load sharing between different nodes of a CARRIER installation is not currently available, and so use of this feature on a CARRIER platform currently requires a custom

 Copyright © Sipwise GmbH - All rights reserved

setup.

Authentication to the external service happens either with a secret API key (set `auth` to `api_key` and set the key as `api_key`), or via OAuth2/JWT (set `auth` to `oauth`). Which method to choose is dictated by the transcription service being used.

OAuth2/JWT Configuration

Use of OAuth2 requires a service account key file in JSON format provided by the transcription service. This key file contains several pieces of information required by the OAuth2 mechanism, most importantly a client identifier and a cryptographic private key. The configuration parameter `service_key_file` must point to this JSON file.

The `algorithm` must be set to match the type of private key in the key file (default `RS256` for RSA+SHA-256). The `scope` is the authentication scope to request access for, and the `uri` is the endpoint to request JWT tokens from. The defaults are usable for Google Cloud Services.

7.29.2. Google Cloud Speech-to-Text

The Google transcription service supports the following options:

```
google:
  audio:
    channels: 1
    sample_rate: 16000
  bucket: ''
  config:
    alternativeLanguageCodes:
    - es
    - de
    - it
    enableAutomaticPunctuation: 'true'
    enableSeparateRecognitionPerChannel: 'false'
    languageCode: en
```

The `audio` parameters describe the audio format used towards the Google service. The default is single-channel 16 kHz PCM. Audio in other formats is converted to this format before being sent to Google.

The `bucket` setting is needed to support transcription of audio longer than 1 minute or 10 MB. If set, audio data will be uploaded to the given Google Cloud Storage bucket first, and transcription will then be requested of the uploaded object in the storage bucket. Afterwards, the object will be deleted from the bucket.

All keys under `config` are given verbatim to the service as part of the transcription request. At least one language code must be given, and up to 3 alternative languages can be set.

7.29.3. OpenAI Speech-to-Text

The OpenAI API supports these options:

Copyright © Sipwise GmbH - All rights reserved

```
  openai:
    language: ''
    model: gpt-4o-transcribe
    prompt: ''
    temperature: 0
```

These options are sent to OpenAI as part of the transcription request. At least the model must be given. The language is a hint and can improve detection speed and accuracy. The temperature is a number between 0.0 and 1.0 and describes the randomness of the output, with higher values meaning more random. A value of zero enables automatic detection. The prompt is optional and can be a clue to the AI about what is being transcribed.

7.29.4. Lemonfox Speech-to-Text

Lemonfox provides an OpenAI-compatible API with some extra features. The configuration options are:

```
lemonfox:
  language: ''
  prompt: ''
  translate: no
```

As with OpenAI, language and prompt are optional. If `translate` is set to `yes` then the result will be translated to English.

7.29.5. Voicemail Transcripts

To enable transcription of voicemail messages, set:

```
asterisk:
  voicemail:
    transcribe:
      enable: yes
    transcribed_mailbody: You have received a new message from
${VM_CALLERID} in voicebox
      ${VM_MAILBOX} on ${VM_DATE}.\nThe contents of the message are
"@@@TRANSCRIPT@@@"
```

The transcript of each voicemail message is stored in the central database, together with the entry for the voicemail message itself. If email notifications for new voicemail messages are configured, then the string from `transcribed_mailbody` will be used instead of the regular `mailbody` string for the body of outgoing emails. This allows inclusion of the special template string `@@@TRANSCRIPT@@@` which will be replaced with the contents of the transcript.

NOTE Inclusion of the transcript in a voicemail email message necessarily incurs a delay in delivery of the email, as the message has to wait until the transcription has been completed.

Copyright © Sipwise GmbH - All rights reserved

7.29.6. Transcripts for Call Recordings

To transcribe call recordings, set the following:

```
rtpengine:
  recording:
    transcribe:
      enable: yes
```

The transcripts will be stored in the central database, together with the entry for each recorded stream.

7.30. Store Recent Calls and Redial

The Sipwise C5 allows to store the number of the last incoming and outgoing calls of each subscriber. To enable the feature, edit `config.yml` file and enable there `kamailio→proxy→store_recentcalls: yes`. Only the last incoming and outgoing call is stored.

Each subscriber can interact with their own personal stored records using Vertical Service Codes (see VSC). In particular a subscriber can:

- redial the last dialed number (this feature has to be enabled for each subscriber/domain using preference *last_number_redial*);
- hear a voice announcement of the last caller's number, then press the key defined in `sems→vsc→callback_last_caller_confirmation_key` parameter of `/etc/ngcp-config/config.yml` to return the call;
- return the call to the last caller's number without hearing the number announcement;
- delete the personal stored records of any recent calls to and from him.

NOTE It is not possible to return a call if the caller's number is unavailable (e.g. anonymous calls).

7.30.1. Configuring Recent Calls Sound Sets

The first step to do is to create a new Sound Set or edit an existing one. In particular, the files to upload are the ones contained in the *recent_calls* group:

Table 17. Recent Calls Sound Sets

Handle	Message played
recent_call_play_number	The last call you received was from ....
recent_call_confirmation	To call back this number press key ....
recent_call_anonymous	The last caller did not share his number, you cannot return the call to this person.
recent_call_empty	Your recent call history is empty.
recent_call_deleted	Your recent call history has been successfully deleted.

Copyright © Sipwise GmbH - All rights reserved

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

Then, remember to set the preference *sound_set* on the Domain or Subscriber level in order to assign the Sound Set.

7.30.2. Advanced Configuration

By default the expiration times for the most recent incoming and outgoing call per subscriber are 3600 seconds (1 hour) and 86400 seconds (1 day) respectively. If you wish to prolong or shorten the expiration time open `constants.yml` and set there `recentcalls: expire: 3600` and `recentcalls: out_expire: 86400` to a new value.

7.31. Time Sets Management

7.31.1. Time Sets Specifications and Data Description

The Sipwise C5 provides an administrative WEB and API interface to manage time sets.

Supported fields, input and output format are based on iCalendar EVENT specification.

Not all iCalendar and EVENT properties are supported, but those that are used for time points and periods definition or stated mandatory by the specification:

- CALENDAR supported properties:

  NAME
- EVENT supported properties:

  SUMMARY

  DTSTART

  DTEND

  RRULE

Important to mention that current implementation does not support these EVENT properties:

- DTSTAMP (UID is used in generated calendar ics file, both UID and DTSTAMP are ignored during uploading calendar file);
- DURATION (DTEND is used);
- RDATE
- EXDATE
- PRIORITY

Main EVENT property, that is used for time points and periods definition is RRULE. Current time sets implementation supports all properties described in the RRULE specification except WKST.

Default value for week start is MO (Monday).

 Copyright © Sipwise GmbH - All rights reserved

7.31.2. Web Configuration of the Time Sets

The time sets management section is provided in two variants. One is the main time sets management section and other is a section on reseller details page. Variants have minor differences. Functionality will be explained using the time sets dedicated interface. Differences will be explained below.

To create a time set navigate to *SettingsTime Set* and press the "Add" button.

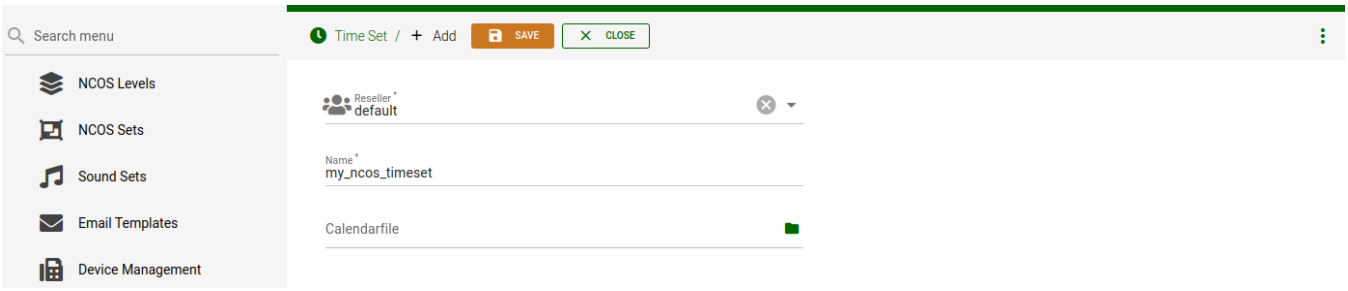

Figure 108. Time set creation

The required fields are:

- "Reseller" field, mandatory
- "Name" field, mandatory
- "Calendarfile", optional

NOTE The name value will be used if the iCalendar (ics) file is not going to be uploaded or the file does not have NAME property for the CALENDAR entry. If both NAME in the uploaded iCalendar (ics) file and form field "Name" are not empty then value from the form field will be taken.

WARNING While editing a timeset, if a calendar ics file will be uploaded to edit time set, all presented events will be deleted and events from the uploaded file will be added after it.

7.31.3. Web Configuration of the Time Set Events

A time set can contain set of events. Each time set event will be used to generate CALENDAR EVENT entry in the generated iCalendar file. All fields in the time set event forms represent properties of the iCalendar EVENT component.

To manage time set events click the three dots beside the desired time set and press the "Events" button. To create a new event press "Create Event" button. And the form to create event will be shown:

Copyright © Sipwise GmbH - All rights reserved

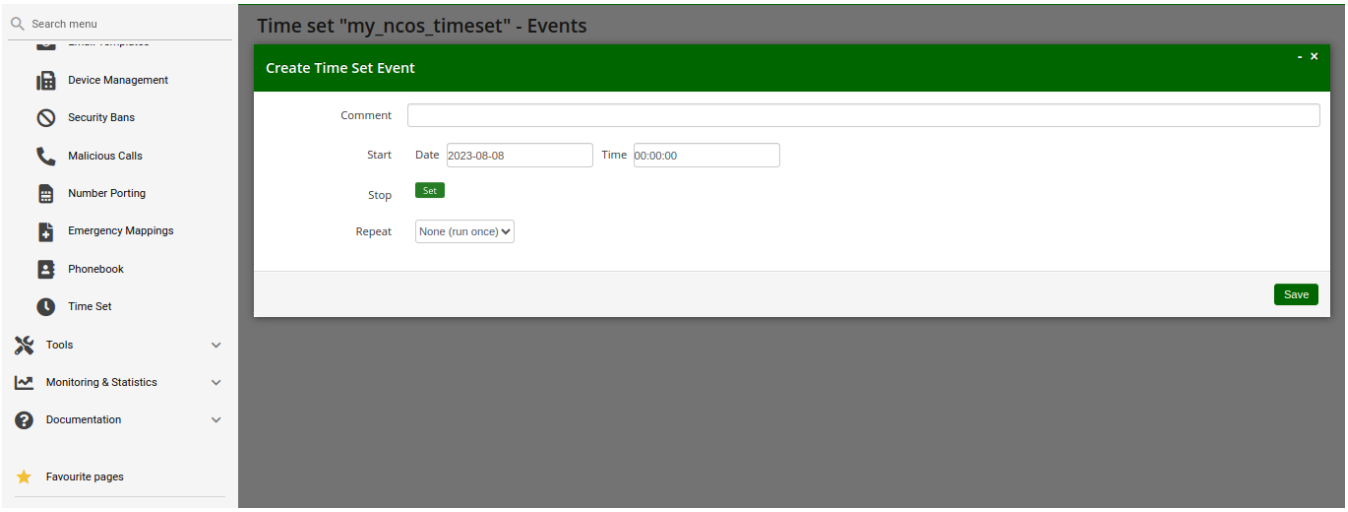

Figure 109. Time set event creation

Time Set Event Form Fields Explanation

"Start" field reflects DTSTART property of the EVENT. "Start" is mandatory and by default is set to the start of the current day. "Start" value format is datetime. "Stop" field reflects DTEND property of the EVENT. For the events within recurrence "Stop" will define duration of each iteration. To specify "Stop" datetime, press button "Set". Fields to enter DTEND date and time will appear:

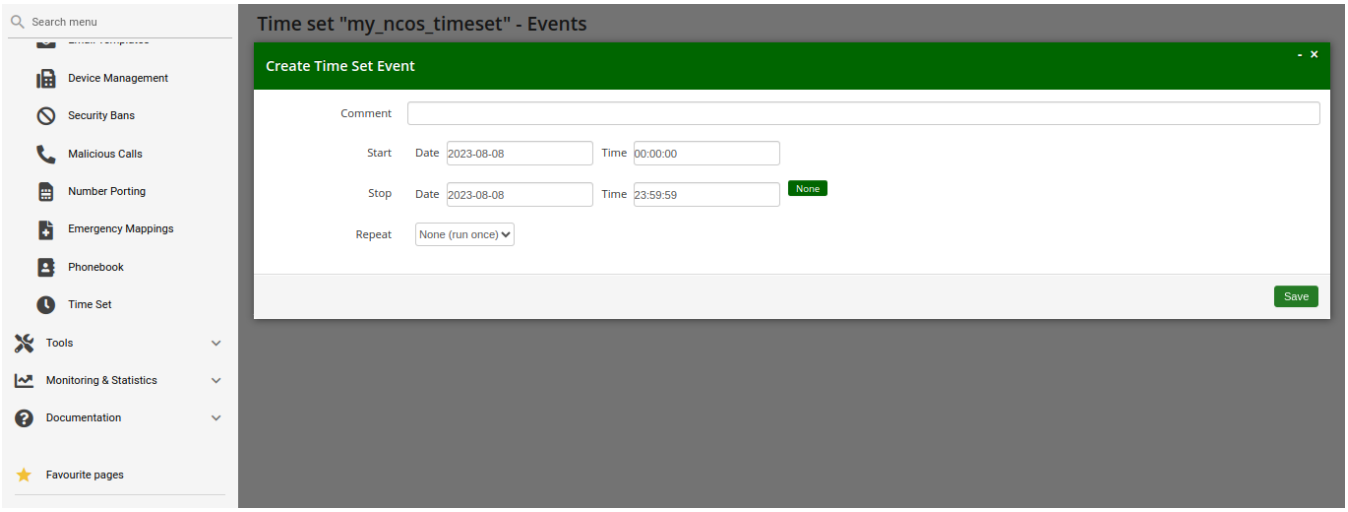

Figure 110. Time set event - Stop

To return "Stop" field to the empty value press button "None". Value in the form fields will be preserved, but newly created EVENT will have empty DTEND property. The rest of the fields in the form are optional. Most of them are not visible by default and will be shown if requested by user or required by data into other fields.

RRULE property of the EVENT is a recurrence rule and defines set of the iterations for the EVENT. To customize recurrence rule for the EVENT select proper repetition unit for the "Repeat" form field. Input field for the recurrence interval will appear left to the frequency select. According to the selected unit, FREQ property of the EVENT RRULE will be set to one of the: SECONDLY, MINUTELY, HOURLY, DAYLY, WEEKLY, MONTHLY, YEARLY.

 Copyright © Sipwise GmbH - All rights reserved

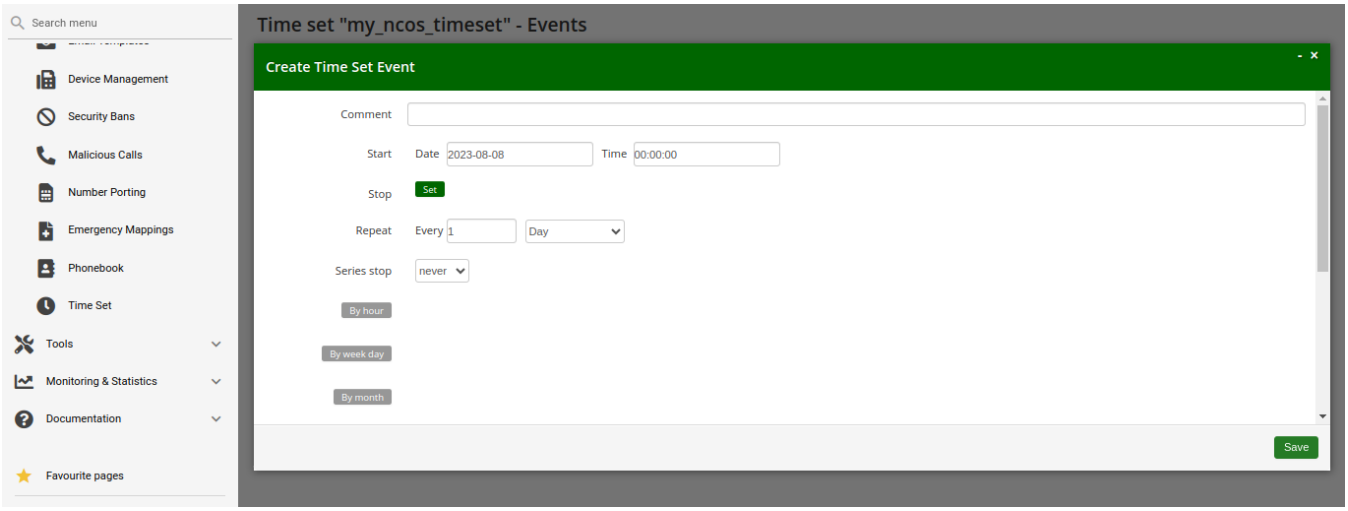

Figure 111. Time set event creation - Repeat

To specify end of the EVENT iterations, select "Series stop" value. For the "at date" option will be shown input for the date and time that will define UNTIL property of the EVENT RRULE.

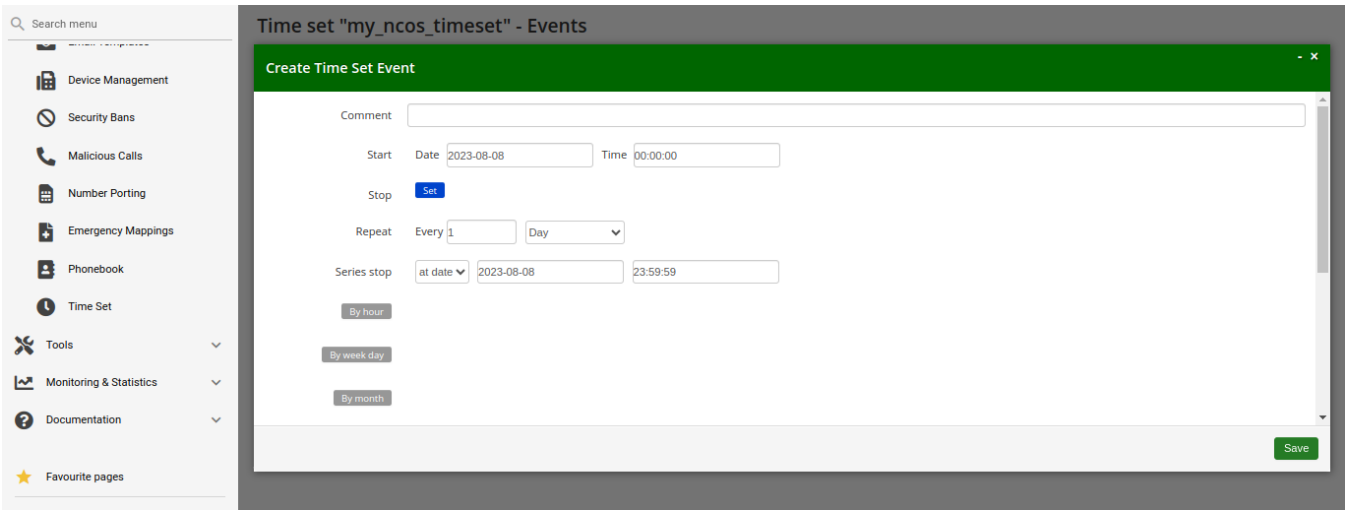

Figure 112. Time set event creation - Repeat End

"after" option, respectively, will put entered value to the COUNT property of the EVENT RRULE.

Form fields "By hour", "By week day", "By month", "By month day", "By set position", "By week number", "By year day", "By second" and "By minute" are not shown by default. To enter value to any of these fields press according button on the left. Button with field name is grayed off when corresponding EVENT property is empty.

When gray button with field name is pressed, field input control appears on the right. In the same time button with field name becomes orange, indicating that field value will be saved for the EVENT. Fields with checkbox controls have auxiliary button "Invert selection". When button "Invert selection" is pressed currently empty checkboxes become selected and currently selected checkboxes become empty.

When form data will be saved, checkboxes values will be saved as comma separated numbers. BYxxx RRULE properties expand or limit behavior of the FREQ according to the table in the RRULE specification. Field "By week day" has two variants of the input: checkbox for each week day and text

Copyright © Sipwise GmbH - All rights reserved

input.

Text input can be used, if "By week day" value is more complex than list of week days, separated by comma, for example for FREQ MONTHLY value "2TH,-3FR" in the "By week day" will mean second Thursday from the month start and third Friday from the month end in every month. Such value cannot be presented as checkboxes selection.

Fields "By set position" and "By year day" are text inputs. Value format for these fields is set of the [+/-]NUMBER values, separated by comma.

For the "By year day" minus sign in front of year day number means that this day should be taken by number from the end of the year.

For the "By set position" minus sign in front of the position of the iteration means that the iteration should be taken by number from the end of the generated iterations sequence.

After new event created, event will appear in time set event list. It will have column with rrule text description, buttons to request event edit form or event deletion.

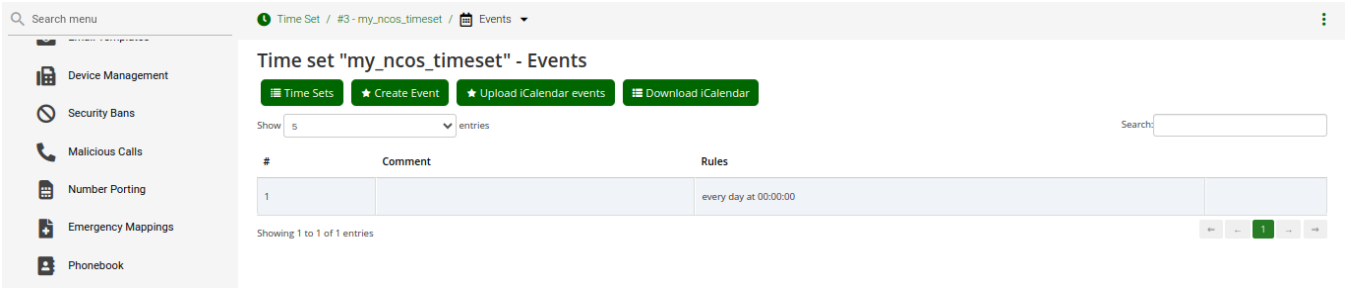

Figure 113. Time set events table

In events list section all events can be redefined uploading ics iCalendar file:

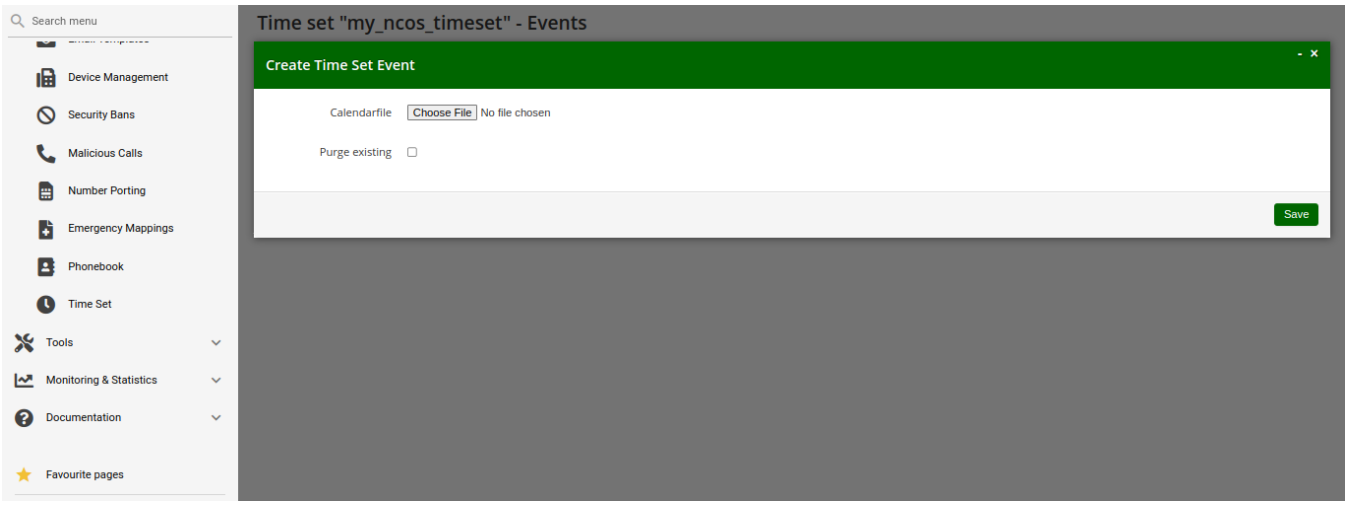

Figure 114. Time set events upload

If "Purge existing" option is selected, all existing time set events will be removed before creation of the events from the uploaded file.

To download the iCalendar ics file of the time set, press "Download iCalendar".

 Copyright © Sipwise GmbH - All rights reserved

7.31.4. Web Interface for Time Set Related to Reseller

Reseller details page provides list of the time sets connected to the reseller and allows create, edit, delete and download time set and has link to the time set events section.

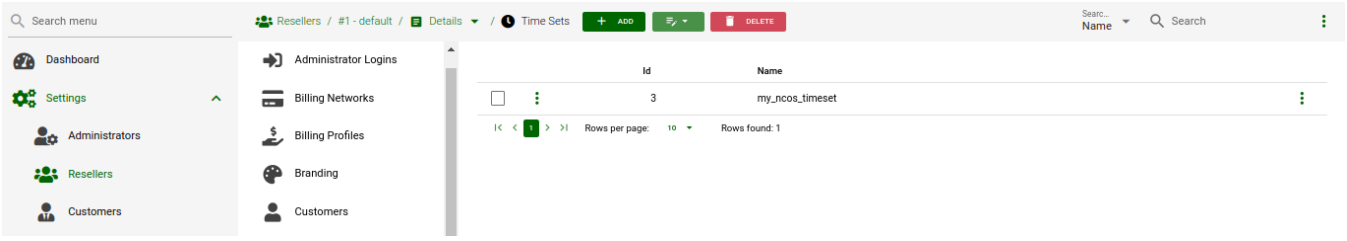

Figure 115. Time set interface for the reseller

In difference to the main time set interface, iCalendar ics file for the time set can be downloaded from the time set list pressing "Download" button.

The creation form does not have the "Reseller" field and is processed in the context of the current selected reseller.

7.31.5. REST API

Time sets management is possible using REST API entry point /api/timesets/.

Time sets API has possibility to get and return information both as "application/json" data and as "text/calendar" file.

To create time set with events the full specification of the all fields in json format can be used:

```
curl --request POST --user administrator:administrator --header 'Prefer:
return=representation' --header 'Content-Type: application/json'
'https://127.0.0.1:1443/api/timesets/' --data '{"reseller_id":"3",
"name":"api_test_timeset_name1","times":[{"start":"1971-01-01
00:00:01","until":"1997-01-01 23:59:59","end":"2020-12-31 23:59:59"}]}'
```

Also time set and events can be uploaded as ics iCalendar file:

```
curl --request POST --user administrator:administrator --header 'Prefer:
return=representation' --header 'Content-Type: multipart/form-data'
'https://127.0.0.1:1443/api/timesets/' --form
'json={"reseller_id":3,"name":"unique_name"}' --form
'calendarfile=@/path/to/calendar.ics'
```

Output of the GET request to the time set item can be text/calendar:

```
curl --request GET --user administrator:administrator --header 'Accept:
text/calendar' 'https://127.0.0.1:1443/api/timesets/12' \>
/path/to/download/calendar.ics
```

Copyright © Sipwise GmbH - All rights reserved

or application/json:

```
curl --request GET --user administrator:administrator --header 'Accept:
application/json' 'https://127.0.0.1:1443/api/timesets/12'
```

By default the API will send response in text/calendar format.

Output will be generated iCalendar including time set events:

```
curl --request GET --user administrator:administrator
'https://127.0.0.1:1443/api/timesets/12'
BEGIN:VCALENDAR
PRODID:-//Mozilla.org/NONSGML Mozilla Calendar V1.1//EN
NAME:api_test_timeset_name2
VERSION:2.0

BEGIN:VEVENT
UID:sipwise19@sipwise15
SUMMARY:unique_name event 19
DTSTART:19710101T000001
DTEND:20201231T235959
END:VEVENT
END:VCALENDAR+
```

7.32. Announcement Before Call Setup

This feature allows a callee to play a custom announcement to the caller every time they receive a call. The announcement is played in early media mode, therefore it can be used as a simple business welcome message or to inform the caller about a different cost of the call before it will be actually charged.

NOTE It can also be played in so-called established mode, controlled by the `conn_type` subscriber's preference. In this case the call will be automatically answered with 200OK and the announcement will be played in the established two-way media session. After that the system will continue to interconnect a caller with the final destination. And the ring-back tone in this case gets emulated.

The configuration of the announcement is similar to the activation of Pre-Recording Announcement and it requires few simple steps.

First create a system sound set for the feature. In *SettingsSound Sets* either use your already existing *Sound Set* or create a new *Sound Set* and then assign it to your domain or subscribers. In the *Sound Set* there is an announcement *early_mediaannounce_before_call_setup* for that purpose.

Once the *Sound Set* is created the subscriber's preference *play_announce_before_call_setup* must be enabled under *SubscriberPreferencesApplications* menu. The same parameter can be set in the Domain's or Customer's preferences to enable this feature for all its subscribers.

NOTE The announcement will be played to the caller before the call is routed to the callee.

Copyright © Sipwise GmbH - All rights reserved

IMPORTANT The announcement before call setup will always be played, even during the call forward or deflection (302 Moved) cases. That means, if the CF announcement is enabled on the call forward owner, then this will be played first, and if there is the announcement before call setup configured on the final destination, then this will be played afterwards. This behavior has been changed beginning from the mr13.4+ Sipwise C5 version.

NOTE This feature and *Pre-Recording Announcement* can be activated at the same time. In this case the *Announcement Before Call Setup* will be played first. If it is enabled in common with the CF announcement, then the order will be: 1. CF announcement is played; 2. Then the announcement before call setup is played; 3. Eventually the recording announcement is played.

7.33. Emulated Ringback Tone

A usual problem linked to the activation of pre-call announcements, like for example Announcement Before Call Setup, Pre-Recording Announcement, Announcement Before Call Forward, is the impossibility of the caller device or provider to generate again the ringback tones after the early media announcement is terminated. This usually happens even if a new '180 Ringing' message is sent back to the caller.

To overcome this problem the Sipwise C5 makes it possible to play a pre-recorded ringback tone to the caller party in early media mode. This feature can be activated on the caller Subscriber or Peer where the tone has to be played. The emulated ringback is played in a loop and it is stopped if one of the following conditions happens:

- a '183 Progress' message is sent back by the callee, this is because the callee most likely would like to play an announcement in early media mode
- a '200 OK' message is sent back by the callee
- the call fails

The configuration of the announcement is similar to the activation of Pre-Recording Announcement and it requires a few simple steps.

First create a system sound set for the feature. In *SettingsSound Sets* either use your already existing *Sound Set* or create a new *Sound Set* and then assign it to your domain or subscriber. In the *Sound Set* there is an announcement *early_mediaringback_tone* for that purpose.

Once the *Sound Set* is created the caller subscriber's preference *play_emulated_ringback_tone* must be enabled under *SubscriberPreferencesApplications* menu. The same parameter can be set in the Domain's or Customer's preferences to enable this feature for all its subscribers. It can also be enabled on Peer's preferences to activate the feature when the call is coming from the peer.

By default the callee's soundset is used to generate the emulated ringback tone. If you prefer to use the ringback tone uploaded in the caller soundset you have to set `config.yml` preference `kamailio.proxy.play_ringback_tone_of_caller` to 'yes'.

The same feature can be used to play back to the caller music on hold or other announcements while the callee endpoint is ringing. To do that just upload the media you prefer instead of a ringback tone.

IMPORTANT The callee has to send a '180 Ringing' message to trigger the emulated ringback tone to start.

Copyright © Sipwise GmbH - All rights reserved

NOTE The generation of the emulated ringback tone is currently limited to two specific scenarios: in the one specified above after a pre-call announcement and in a TPCC initiated call where the caller is already in conversation.

7.34. Subscriber Location Mappings

7.34.1. Overview

Subscriber Location Mappings enables incoming calls on a subscriber to also be routed and handled on additional "locations".

A location can represent:

- A SIP URI of an NGCP subscriber, which may have one or more endpoints registered to it.
- Remote peering destinations.

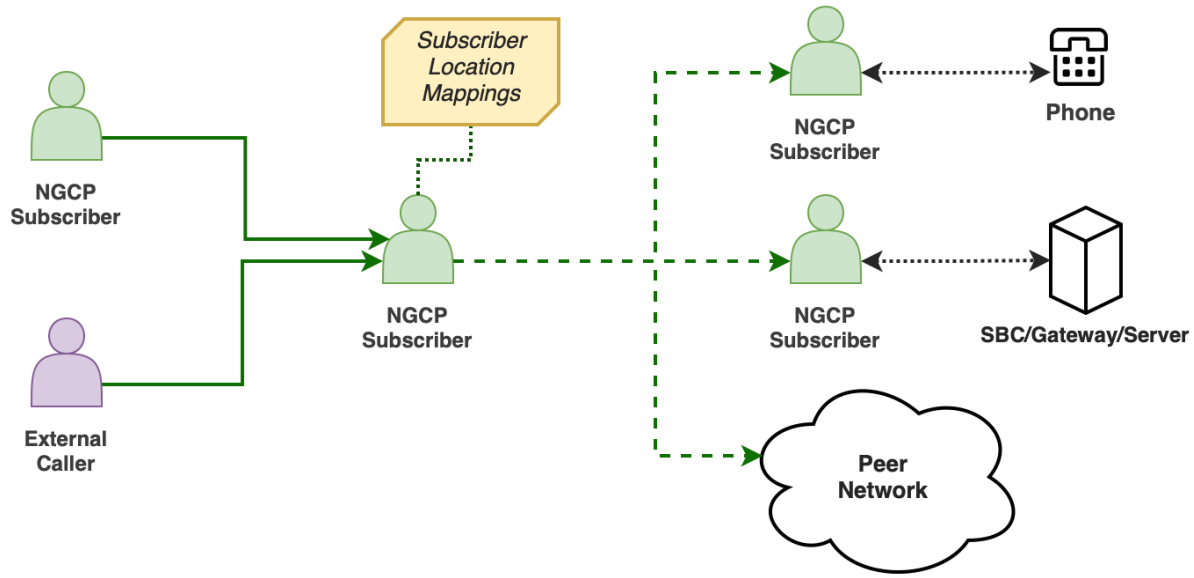

Figure 116. Subscriber Location Mappings - Overall View

The feature can be set within subscriber preferences, via the Admin Panel or REST API.

Its main advantages are as follows:

- **Definition of Several Call Branches** - By defining one or more location entries for a subscriber, it is possible to generate separate call branches for each destination, when an incoming call is received for that subscriber.
- **Set Triggering Conditions and Mode for Each Call Branch** - In order to trigger a call branch, conditions can be defined for both caller and callee numbers via regular expressions, as well as the definition of the call branch behavior to be applied.
- **Number Manipulations** - Number manipulations can be applied on the *username* part of the *Request-URI* (RURI) or the To field in a SIP INVITE message.

 Copyright © Sipwise GmbH - All rights reserved

Example Scenarios

There may be different scenarios in which the feature can be considered. Some examples, are:

Table 18. Scenario Examples.

Scenario	Description
NGCP Subscriber	• Incoming calls on an NGCP subscriber are (also) delivered to another NGCP subscriber. • Two cases are possible: re-routed or multiplied calls.
External Gateway	• There is a Session Border Controller (SBC) with a registration in NGCP that is intended to receive calls for NGCP subscribers. • Incoming calls on an NGCP subscriber are also routed to the SBC. • The SBC may route the calls to other PBX.
Remote Peer	• Incoming calls on an NGCP subscriber are (also) delivered to a remote peer according to the outbound peering logic in place. • The remote peer will receive the call with source equal to the original caller.

The following sections will provide more details about the feature.

7.34.2. Fields

Location mappings fields are summarized in the following table.

Table 19. Location Mapping Fields.

Field	Description
Location SIP URI	This field has two possible value options: • A SIP URI of an NGCP subscriber (i.e. *sip:user@domain*). Mandatory value for the `Add`, `Replace` and `Offline` modes. See "*Mode*" below. *NOTE: This entry does not refer to a SIP URI with location binding address.* • A fictitious or an empty name for the `Forward` mode (The value is ignored during the branch creation). See "*Mode*" below.
Caller Pattern	A regular expression pattern to match the `From` field of the INVITE (After Inbound Rewrite Rules). If left empty, it defaults to the `.` pattern.
Callee Pattern	A regular expression pattern to match the `R-URI`/`To` of the INVITE (After Inbound Rewrite Rules). If left empty, it defaults to the `.` pattern.

Copyright © Sipwise GmbH - All rights reserved

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Mode</td><td>Defines the behavior of the matched entry during the call termination:

<table>
<tr><td><code>Add</code></td><td>It appends the entry into the destinations list.
The defined Location SIP URI must have registered endpoints.
Multiple <code>Add</code> entries matched are added into the call branching process.</td></tr>
<tr><td><code>Replace</code></td><td>It replaces the subscriber's SIP URI as destination by this entry.
The defined Location SIP URI must have registered endpoints.</td></tr>
<tr><td><code>Offline</code></td><td>It appends the entry into the destinations list ONLY if the subscriber is offline (i.e. there are no registration records for the subscriber).</td></tr>
<tr><td><code>Forward</code></td><td>It appends the entry into the destinations list for a remote peer.
If this mode is set, it is not possible to call to another local subscriber.</td></tr>
</table>
"Mode" field is mandatory.</td></tr>
<tr><td>RURI/To Username</td><td>Replaces the username part of the <code>RURI/To</code> in a SIP INVITE with the value defined on this field. Useful for cases when the remote username is different than the one used for the location mapping.</td></tr>
<tr><td>External ID</td><td>An arbitrary string name (e.g. an identifier of a 3rd party provisioning/billing system). This is an optional attribute, which can be kept as empty.</td></tr>
<tr><td>Enabled</td><td>Enable/Disable the entry.</td></tr>
</table>

About the 'Forward' Mode

The location mappings "*Forward*" mode works differently when compared to the others:

- This mode does not require the *Location SIP URI* field, since it applies outbound peering logic to select the correct peer and automatically calculates the final destination.
- It creates a new parallel branch meant for peers only and it will behave like a Call Forward Unconditional (CFU). (NOTE: If you need to generate an internal call to local subscribers, use the `Add` mode instead.)
- The remote peer will receive the original caller as the source of the call (NOTE: User Provided Redirecting Number is intentionally disabled for this particular mode). Also, two different Call Detail Records (CDR) will be created: the first one from 'A' to 'B' (in which the CDR `call_type` field equals to '*call*'), and the second one from the original callee subscriber to the peer (with `call_type` = '*cfu*').
- To prevent the call to be looped back, the new parallel branch is terminated, if the inbound peer belongs to the same peer group of the outbound peer. However, it is possible to disable this check by activating the `allow_lm_forward_loop` peer preference ("*Internals*" section) of the outgoing

Copyright © Sipwise GmbH - All rights reserved

peer.

7.34.3. Provisioning via Admin Panel

Subscriber Location Mappings can be accessed at *Subscriber PreferencesLocation Mappings* section. To create a new entry, press the "add" button.

As shown in the example below, data can be entered and saved to create the new entry.

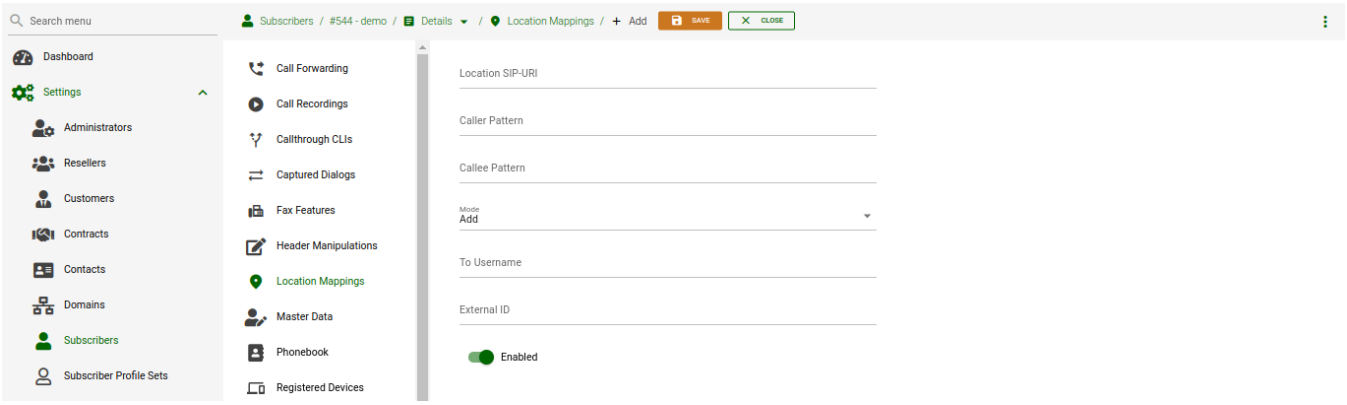

Figure 117. Subscriber Location Mappings: Create Entry

Parameters are the ones described on the "*Fields*" section.

Practical Example

Let us assume that an administrator will use the Admin Panel to provision some location mappings on a selected subscriber. The following *Subscriber Details* are considered:

Table 20. Practical Example - Subscriber Details.

Field	Value
SIP URI	sip:su1@siphomelab.com
Primary Number	4312521523
Alias Numbers	4312521523001 4312521523002

Then, on *Subscriber Preferences*, within the *Location Mappings* section, the following entries are defined:

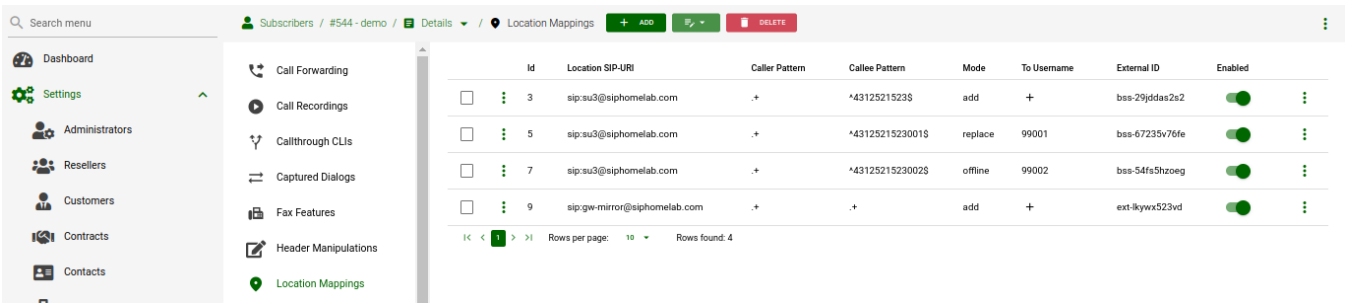

Figure 118. Practical Example - Subscriber Location Mappings

In the provided example, the logic is as follows:

Copyright © Sipwise GmbH - All rights reserved

Table 21. Practical Example - Logic & Result.

Logic	Result
• An incoming call to the subscriber number `4312521523` will be additionally sent to the devices registered with *sip:su3@siphomelab.com*.	• The call will ring on the registered devices of *sip:su1@siphomelab.com*. • The call will ring on the registered devices of *sip:su3@siphomelab.com*. • The call will ring on the registered devices of *sip:gw-mirror@siphomelab.com*. • When a device takes the call, the call will be cancelled on the rest of devices.
• An incoming call to the subscriber number `4312521523001` will replace the current subscriber SIP URI as destination (i.e. the call will not be terminated at the devices registered with subscriber *sip:su1@siphomelab.com*). • Instead, the call will be sent to the registered devices with location entry *sip:su3@siphomelab.com* with the "*username*" part of the INVITE replaced with number `990001`.	• The call will NOT ring on the registered devices of *sip:su1@siphomelab.com*. • The call will ring on the registered devices of *sip:su3@siphomelab.com* with the number `990001`. • The call will ring on the registered devices of *sip:gw-mirror@siphomelab.com*. • When a device takes the call, the call will be cancelled on the rest of devices.
• An incoming call to the subscriber number `4312521523002` will be additionally sent to the devices registered with *sip:su3@siphomelab.com* only if there are NO active registrations under this subscriber (i.e. *sip:su1@siphomelab.com* is offline). • Also, the "*username*" part of the INVITE will be replaced with number `990002`.	• If *sip:su1@siphomelab.com* is offline, then the call will ring on the registered devices of *sip:su3@siphomelab.com* with the number `990002`. • The call will ring on the registered devices of *sip:gw-mirror@siphomelab.com*. • When a device takes the call, the call will be cancelled on the rest of devices.
• Any incoming call to the subscriber *sip:su1@siphomelab.com* will be additionally sent to the devices registered with *sip:gw-mirror@siphomelab.com*.	• When a device takes the call, the call will be cancelled on the rest of devices.

7.35. STIR/SHAKEN

7.35.1. Overview

What Does STIR/SHAKEN Stand For?

STIR - Secure Telephony Identity Revisited (RFC8224, RFC8225 and RFC8226)

Copyright © Sipwise GmbH - All rights reserved

SHAKEN - Secure Handling of Asserted Information using tokens (RFC8588)

This technology gives an opportunity to make sure that the calling side (caller's number) is not spoofed, hence can help during the common work/inter-cooperation between ITSPs, for call spoofing detection. When it is used by ITSPs, it decreases the amount of robo-calls.

NOTE ITSP - Internet Telephony Service Provider. Is a general abbreviation for VoIP service providers, which can use IP-based technologies/protocols such as SIP protocol or H.323 stack of protocols to provide telephony services.

The Sipwise C5 is currently compliant with the list of general RFC standards:

- RFC8224 - Authenticated Identity Management in the Session Initiation Protocol (SIP) (this RFC obsoletes RFC 4474) ;
- RFC8588 - Personal Assertion Token (PaSSporT) Extension for signature-based handling of Asserted information using toKENs (SHAKEN) ;

Why Do You Need STIR/SHAKEN?

As the VoIP world grows new security mechanisms come into play, which can provide protection on different layers of VoIP. STIR/SHAKEN is one of these mechanisms and it works as a superstructure over the VoIP system. It can be considered as the Application layer of TCP/IP stack (in common with SIP).

This standard has been developed by IETF, to provide a new way to detect spoofed calls and to help ITSPs to protect their customers. This means a decrease in the amount of fraudulent events/bot calls. A notable strength of this technology is that it not only works within the scope of your VoIP system, but it is also dependent on other intermediary/termination ITSPs. Therefore it can bring a good quality improvement for a protection from undesired calls (bot-calling).

NOTE The SIP header declared for this purpose is called literally "Identity:".

IMPORTANT The "Identity-Info:" header is deprecated by RFC 8224 and now is stored as the ";info=" parameter within the "Identity:" header.

7.35.2. Operation of STIR/SHAKEN

All logical parts of STIR/SHAKEN fall into these sub-categories:

- Originating service provider - a system which performs an authentication (IP based, digest over MD5 etc.) of the calling party (SIP). It has to support all the needed RFC regulations to add/manipulate the Identity header. This entity does all the work to prepare the "Identity" header before sending a call out ;
- Terminating service provider - a remote VoIP system (call termination ITSP), which verifies if the call is compliant with the STIR/SHAKEN technology, then verifies it before accepting and sending to the edge subscriber. As well as with the Originating service provider, it has to support all the needed functionality to manipulate the Identity header. This entity does all the work on the Identity header of the received request, such as: decryption of the JWT token stored inside the "Identity" header and an attestation level verification ;
- Authentication service - is a superstructure/standalone authority used by the originating ITSP to obtain certificates/keys needed to encrypt the "Identity" header's value (JWT token). It is only used

Copyright © Sipwise GmbH - All rights reserved

for work with certificates and keys ;

- Verification service - is a superstructure/standalone authority used by a terminating ITSP to obtain certificates/keys needed to decrypt the "Identity" header's value (JWT token). The same as with the Authentication service, it is only used for work with certificates and keys ;

The Sipwise C5 system role will be:

- Originating service provider - when Sipwise C5 sends a call out to PSTN networks
- Terminating service provider - when Sipwise C5 is a recipient of the call coming from PSTN networks

NOTE This technology assumes that there is some centralized authority or a list of them, which will be used both by the originating and the terminating service providers.

How STIR/SHAKEN Works

- Firstly, a SIP call is originated and obtained by a specific ITSP (let us refer to it as ITSP A);
- ITSP A verifies the call source and its number, in order to define how to confirm validity (full, partial or gateway attestation);
- ITSP A creates a SIP Identity header that contains the information on the calling number, called number, attestation level and call origination, along with the certificate (important). All information in the Identity header gets encrypted into the JWT token, with the following list of parameters;
- The call (INVITE request) is sent out with the SIP Identity header (which has a reference to the certificate) to a certain destination ITSP (let us refer to it as ITSP B);
- ITSP B verifies the identity of the header and the certificate itself;
- ITSP B makes sure the attestation level is compliant with local security standards (A, B or C);
- ITSP B sends a call to the edge subscriber (with possibly added parameters to the PAI header - verstat);

Essentially this technology allows the destination user/service provider to verify that the original caller (calling side) is the one that it claims to be.

Possible Attestation Levels:

- Full attestation (A) - SP authenticates the calling party and confirms it is authorized to use this particular number. An example is SIP registration: we say that originating ITSP recognizes the entire phone number as being registered with the originating subscriber;
- Partial attestation (B) - SP verifies the call origin, but cannot fully confirm that the source of the call is authorized to use this number. An example is a calling party (number) from behind a remote PBX: you give a block of numbers to the customer and trust this customer entirely, but you cannot verify directly individuals that use those numbers;
- Gateway attestation (C) - SP authenticates the call's origin, but there is no way to verify the source. An example is a call received from an international gateway: you have a legal interconnection, but you do not know who is sitting behind and using source numbers, which you let pass through your network;

NOTE In most cases, the attestation level which will appear in calls involving Sipwise C5 - will be “A”. Unless you do not use a subscriber preference "trusted source" and/or a domain

 Copyright © Sipwise GmbH - All rights reserved

preference "unauth_inbound_calls".

7.35.3. Identity Header Constitution

- a JSON Web token - JWT, that in addition consists of the following parts (divided by the dot symbol '.'):

  header - stores: an encryption algorithm, used extension (usually 'shaken'), token type (usually 'passport'), location of the cert used to sign the token

  payload - stores: an attestation level, calling/called number, a timestamp when token was created (epoch), an origination identifier (most likely UUID)

  signature - some encoded string (binary data) in Base64 URL

- three parameters (currently):

  ';info=' - is a substitution of the deprecated Identity-Info header (will usually have a reference to a public key, which will be used to decrypt then a signature encrypted previously with a private key)

  ';alg=' - specifies the use of a cryptographic algorithm for the signature part of the JWS (for e.g. 'ES256')

  ';ppt=' - can extend the payload of PASSporT with additional JWT claims (requires that a relying party supports a particular extended claim, for e.g. 'shaken')

TIP Signature - is some encoded string (binary data) in Base64 URL, where its syntax is: *Base64URL(ES256 (Base64URL(JWT header).Base64URL(JWT payload)))*. Firstly the Base64 URL is built, then also the signature with the private key is built, and a URL to the public key certificate corresponding to that key is added (as a value in the ";info=" parameter).

NOTE The Receiving party will have to download the certificate and perform a signature verification. In case the check passes, it will trust the value stored in the attestation level parameter. And of course the value of caller/callee ("orig" / "dest") has to match the actual data (in SIP headers) to be sure it is not an attack and is not a call which reuses a value from another INVITE not related to the current one.

7.35.4. Implementation of STIR/SHAKEN in Sipwise C5

Currently the Sipwise C5 is using the Kamailio project's *Secsipid.so* module to provide all, or almost all required STIR/SHAKEN related features.

NOTE The secsipid module relies on libsecsipid from the Secsipidx project. Link to the SecSIPIDx project at Github: https://github.com/asipto/secsipidx

Secsipid - is a first implementation of STIR/SHAKEN IETF extensions (RFC8224, RFC8588) for asserting caller identity in Kamailio. The module is being developed further, and is reliable enough to be included in the Sipwise C5.

The configuration of the STIR/SHAKEN is done in `config.yml`, with the section: `kamailio.proxy.stir`. To see how it is being configured, please take a look at kamailio (Appendix "Configuration Overview" "config.yml Overview").

Domain preferences:

Copyright © Sipwise GmbH - All rights reserved

- *stir_pub_url* - Public key HTTP URL
- *stir_check* - Enable STIR Identity validation

A quick start with STIR/SHAKEN:

1. You need to identify the appropriate authorities with whom to establish contact in order to obtain the required certificates and keys.
2. Your certificates should be prepared for work with a needed domain (your SIP domain), then you upload the given private key(s) to your Sipwise C5 system ;
3. Basically a general parameter (in the `config.yml`) you need to take care of is 'domains', and of course the given keys to it. The rest can be left untouched, unless you have some preferences for your domain(s) ;
4. You need to define the related domain preferences (via the web panel) 'stir_pub_url' and 'stir_check' to enable SHAKEN for your domain(s);
5. After you are done with the configuration part, you need to apply changes with: `ngcpcfg apply "added stir-shaken support"` ;

IMPORTANT Applying the changes will imply a restart of the Proxy service, therefore it is strongly recommended to apply the changes outside of business hours.

IMPORTANT Remember that the kamailio user needs read permissions on the private-key files, otherwise it will not be able to consume them.

NOTE Keep in mind that you can have multiple domains and hence keys for them stored in the Sipwise C5. This depends on the amount of SIP domains you actually have, which use STIR/SHAKEN.

NOTE The fact that you enabled support for STIR/SHAKEN on your platform, does not mean all of the present domains will have to use it. Checks/manipulations will be only enabled for those domains that are defined in `kamailio.proxy.stir.domains` of `config.yml`.

Outbound Calls and STIR/SHAKEN

At the moment the Sipwise C5 supports 'A' and 'B' attestation levels when sending outbound calls:

- 'A' - corresponds to a normally registered subscriber at the Sipwise C5 platform ;
- 'B' - will be set in case the subscriber uses the preference "trusted source" and/or is affected by the domain preference "unauth_inbound_calls" ;

Inbound Calls and STIR/SHAKEN

As soon as you enable STIR/SHAKEN for a certain domain served by the Sipwise C5 system, calls coming to this domain will be challenged according to RFC standards dedicated to processing of the Identity header.

For now the Sipwise C5 supports the following list of 4XX response codes:

- 428 - indicates an absence of the Identity header or wrong extension added into the ';ppt=' parameter ;

 Copyright © Sipwise GmbH - All rights reserved

- 436 - indicates an inability to acquire the credentials needed by the verification service for validating the signature in an Identity header field ;
- 438 - indicates that no Identity header field with a valid and supported PASSporT object has been received ;

Normally, in case of an incoming call to a SIP domain with enabled STIR/SHAKEN, the "Identity" header's content will be decrypted, verified and the call will be sent further, according to the logic of the Sipwise C5 system setup.

7.36. Fileshare

7.36.1. Overview

Fileshare is a REST API endpoint, /api/fileshare, that provides NGCP users with a way to share data among other NGCP users or external users.

This feature can be provided to clients directly or integrated into own application services.

7.36.2. Example Scenarios



Figure 119. Fileshare File Upload

Copyright © Sipwise GmbH - All rights reserved



Figure 120. Fileshare File Download

 Copyright © Sipwise GmbH - All rights reserved



Figure 121. Fileshare Uploaded Files

7.36.3. Configuration

Fileshare is disabled by default and requires basic configuration in the `config.yml`

```
fileshare:
  enable: no
```

Copyright © Sipwise GmbH - All rights reserved

```
  limits:
    quota: 10737418240
    upload_size: 10485760
    user_files: 10
    user_quota: 20971520
  public_links: no
  ttl: 86400
```

- **fileshare.enable**: disabled by default, when enabled, **rest_api.enable** is required to be on as /api/fileshare uses API v2.
- **fileshare.limits.quota**: allowed database quota threshold in bytes, when the table size reaches the quota threshold all subsequent file uploads are denied.
- **fileshare.limits.upload_size**: single file upload size in bytes.
- **fileshare.limits.user_files**: amount of uploaded files per user (subscriber).
- **fileshare.limits.user_quota**: per subscriber uploaded files quota in bytes, calculated by the exact size of all uploaded files per user (subscriber)
- **fileshare.public_links**: defines whether uploaded files can be downloaded by anybody without authentication (the exact URL must be provided to the end user).
- **fileshare.ttl**: uploaded file expiration time in seconds, default is 86400 (1 day) client, e.g.: /api/fileshare/58f417bb-4d28-47cf-9673-0a9453b79cc5), access to the uploaded files collection via /api/fileshare always requires authentication.

7.36.4. Approach

File Upload

To upload a file the client must use POST and the file attached into the form named "file"

An example of file upload using 'curl':

```
curl -X POST -F 'file=@/tmp/sample_image.png' https://ngcp-
api/api/fileshare
```

NOTE One of the available authorization methods is required for the file upload (Basic Authentication or JWT)

There is an optional upload body parameter 'ttl', that can be provided to override the default ttl for the uploaded file.

An example of file upload with `ttl` using 'curl':

```
curl -X POST -F 'file=@/tmp/sample_image.png' -F ttl=3600 https://ngcp-
api/api/fileshare
```

Once a file is successfully uploaded a JSON response is returned with the following data:

Copyright © Sipwise GmbH - All rights reserved

```
{
  "id": "b994894d-882b-4672-a152-ff1e68ab7ee0",
  "name": "sample_image.png",
  "mime_type": "image/png",
  "ttl": "3600",
  "created_at": "2021-12-14T20:58:13.483Z",
  "expires_at": "2021-12-14T21:58:13.483Z",
  "size": 198791,
  "_links": {
    "self": {
      "href": "/api/fileshare/b994894d-882b-4672-a152-ff1e68ab7ee0"
    },
    "collection": {
      "href": "/api/fileshare"
    }
  }
}
```


- The upload name is taken from the provided filename.
- The MIME type is automatically determined by the server.

File Download

To download a file a full URL must be provided, e.g.:

```
curl -X GET https://ngcp-api/api/fileshare/b994894d-882b-4672-a152-
ff1e68ab7ee0
```

If `fileshare.public_links` is enabled in the `config.yml`, then there is no authorization required for the link to work, otherwise an authorization is required for the link to be used (any authenticated NGCP subscriber (or admin) user can use the link).

The response is a data stream.

Files Collection

As a subscriber (or as admin) you can retrieve all your uploaded files list as:

```
curl -X GET https://ngcp-api/api/fileshare/
```

The response is a JSON HAL collection.


```
{
  "_links": {
    "self": {
      "href": "/api/fileshare?page=1&rows=10"
    },
```


Copyright © Sipwise GmbH - All rights reserved

```
    "ngcp:fileshare": [
      {
        "href": "/api/fileshare/1ec51a35-92e7-46be-aa98-db69f11483a7"
      },
      {
        "href": "/api/fileshare/58f417bb-4d28-47cf-9673-0a9453b79cc5"
      }
    ],
    "collection": {
      "href": "/api/fileshare"
    }
  },
  "total_count": 2,
  "_embedded": {
    "ngcp:fileshare": [
      {
        "id": "1ec51a35-92e7-46be-aa98-db69f11483a7",
        "name": "doc.pdf",
        "mime_type": "application/pdf",
        "ttl": 86400,
        "created_at": "2021-12-11T11:46:39.000Z",
        "expires_at": "2021-12-12T11:46:39.000Z",
        "size": 10
      },
      {
        "id": "58f417bb-4d28-47cf-9673-0a9453b79cc5",
        "name": "sample_text.txt",
        "mime_type": "text/plain",
        "ttl": 3600,
        "created_at": "2021-12-14T18:17:15.000Z",
        "expires_at": "2021-12-14T19:17:15.000Z",
        "size": 10
      }
    ]
  }
```


NOTE It is also possible to receive the response in the OpenAPI format by providing an additional header, **Accept: application/json**


```
{
  "id": "1ec51a35-92e7-46be-aa98-db69f11483a7",
  "name": "doc.pdf",
  "mime_type": "application/pdf",
  "ttl": 86400,
  "created_at": "2021-12-11T11:46:39.000Z",
  "expires_at": "2021-12-12T11:46:39.000Z",
  "size": 10
},
{
  "id": "58f417bb-4d28-47cf-9673-0a9453b79cc5",
  "name": "sample_text.txt",
```


Copyright © Sipwise GmbH - All rights reserved

```
  "mime_type": "text/plain",
  "ttl": 3600,
  "created_at": "2021-12-14T18:17:15.000Z",
  "expires_at": "2021-12-14T19:17:15.000Z",
  "size": 10
}
```

File Removal

Files are removed automatically by the server based on their expiration time or manually per request.

```
curl -X DELETE https://ngcp-api/api/fileshare/b994894d-882b-4672-a152-
ff1e68ab7ee0
```

7.37. Batch Provisioning

7.37.1. Overview

Batch provisioning of subscribers can ensure that all details of the required subscriber settings are correctly stored in the database. For the easy and convenient operation, NGCP provides the possibility to enter or upload the necessary, variable subscriber data and execute the provisioning automatically. This can be achieved through the administrative web interface, command line interface, or also via REST API.

The advantages are as follows:

a. **Standardizing complex subscriber setup procedures** Batch provisioning allows to define provisioning templates, in which specific subscriber settings can be set. A template allows to define different types of input fields, parameters for internal calculations, and the setting of subscriber-related parameters.

b. **Scripting languages** Provisioning templates allows the definition of code snippets for parameter calculations. Code can be based on JavaScript or Perl programming languages.

c. **Transactional processing (Rollback behavior)** Batch provisioning is based on transactional processing. If provisioning fails at a specific point, it will rollback what was created up to that point, in order to avoid residual data in the database.

d. **Processing speed** Batch provisioning allows to process a large number of subscriber data with a considerable gain in processing speed.

The following sections will provide more details about these features.

7.37.2. Template Structure and Main Fields

Provisioning templates are commonly defined on the YAML language. The list of currently supported main keys are shown and described as follows.

Copyright © Sipwise GmbH - All rights reserved

Currently supported main template fields.

```
fields:
  #Structure that contains several input fields.

contract_contact:
  #Structure that contains several provisioning fields
  #for the contact of the customer.

contract:
  #Structure that contains several provisioning fields
  #for the customer.

contract_balance
  #Structure that contains several provisioning fields
  #for the contract cash and free time balance.

fraud_preferences
  #Structure that contains several provisioning fields
  #for the per-contract fraud limit overrides.

contract_preferences:
  #Structure that contains several provisioning fields
  #for customer preferences.

subscriber:
  #Structure that contains several provisioning fields
  #for subscriber details.

subscriber_preferences:
  #Structure that contains several provisioning fields
  #for subscriber preferences.

registrations:
  #Structure that contains several provisioning fields
  #for permanent registrations.

trusted_sources:
  #Structure that contains several provisioning fields
  #for trusted sources.

cf_mappings:
  #A structure that contains several provisioning fields
  #for call forward mappings.
```

The 'fields' Key

The fields key allows to define input hierarchical structures by providing additional key-value pair parameters. The supported attribute names are the ones defined in the Comprehensive Perl Archive Network (CPAN) `HTML::FormHandler::Field` Modules.

 Copyright © Sipwise GmbH - All rights reserved

Parameters must contain a **name** and data **type**. For the latter, some basic type values are: *Integer*, *Float*, *Text*, *Boolean*, *Checkbox* and *Select*. Additional attributes can be appended as well.

The following example aims to show how to declare different parameters within the `fields` key. Let us assume that an administrator wants to:

- Declare a parameter named "my_field1", which will be a mandatory input text labeled as "*This is field 1 (Text)*".
- Declare a parameter named "my_field2", which will be an optional integer input labeled as "*This is field 2 (Integer)*".
- Declare a parameter named "my_field3", which will be optional Boolean input labeled as "*This is field 3 (Boolean)*".
- Declare a parameter named "my_field4", which will be a mandatory drop-down list labeled as "*This is field 4 (Select)*" with two sub-options labeled "*optA-label*", "*optB-label*" with the static values "*optA-value*", "*optB-value*", respectively.

The resulting YAML declaration will be:

Example Form of Four Different Field Types.

```
fields:
  - name: my_field1
    label: "This is Field 1 (Text)"
    type: Text
    required: 1
  - name: my_field2
    label: "This is Field 2 (Integer)"
    type: Integer
  - name: my_field3
    label: "This is Field 3 (Boolean)"
    type: Boolean
  - name: my_field4
    label: "This is Field 4 (Select)"
    type: Select
    options:
      - label: optA-label
        value: optA-value
      - label: optB-label
        value: optB-value
    required: 1
```

The resulting web form will be shown as the picture below.

Copyright © Sipwise GmbH - All rights reserved

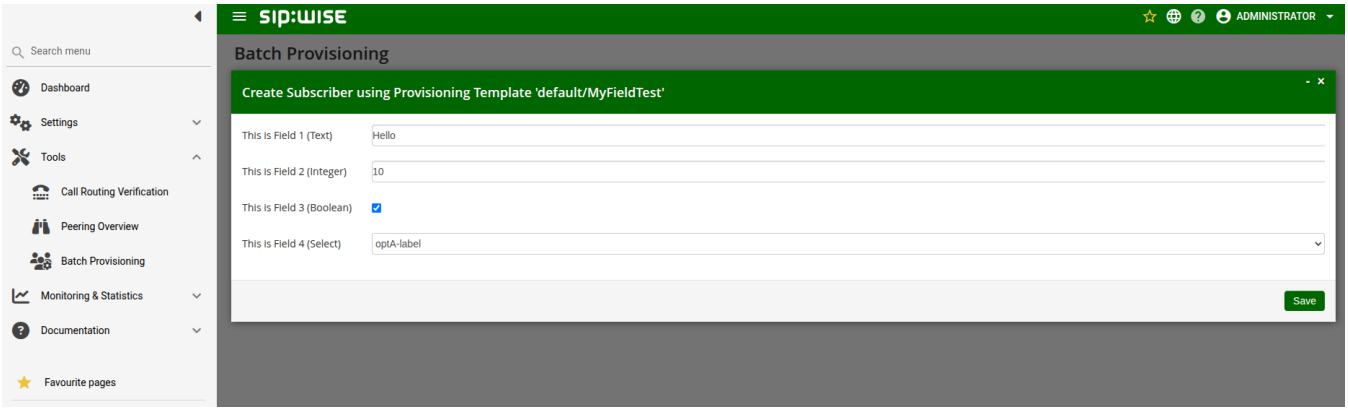

Figure 122. Resulting form of the example with four different field types

The 'purge' Attribute Name

The **purge** attribute name is a reserved word that can be declared within the **fields** tag, as a *Boolean* type. When set to **1**, and if there is an existing subscriber with the same data to be provisioned (e.g. same username, number or alias), then the existing subscriber will be terminated before the new one is created. On the opposite, if set to **0**, that row will be skipped.

NOTE The engine will not terminate anything other than a subscriber (Including its child items such as preferences, trusted sources, call forwards, etc.). Thus, the contract or contact is not terminated/deleted.

The 'calculated' Reserved Parameter Type

There is a special attribute type named calculated, which is not visible on form elements. It is declared for internal use to: (a) Store static values on attributes; (b) Store the output of computed operations coming from code snippets. As defined in `HTML::FormHandler`, the former requires declaring the **value** attribute name to store the data. The latter requires declaring the **value_code** attribute in order to store the data in the **value** attribute. Next sections will describe further details about this.

Rest of Template Keys

The rest of the template keys can contain any of the NGCP settings, as detailed in the table below.

Table 22. Rest of Template Keys.

Field Name	Sub-fields
`contract_contact`	Available field names can be taken from the `/api/customercontacts/` API properties (Please, check API documentation), or from the fields belonging to the `billing.contacts` SQL table.
`contract`	Available field names can be taken from the `/api/customers/` API properties (Please, check API documentation), or from the fields belonging to the `billing.contracts` SQL table.
`contract_balance`	Available field names can be taken from the `/api/customerbalances/` API properties (Please, check API documentation).

 Copyright © Sipwise GmbH - All rights reserved

`fraud_preferences`	Available field names can be taken from the `/api/customerfraudpreferences/` API properties (Please, check API documentation).
`contract_preferences`	Available field names can be taken from the `/api/customerpreferences/` API properties (Please, check API documentation).
`subscriber`	Available field names can be taken from the `/api/subscribers/` API properties (Please, check API documentation).
`subscriber_preferences`	Available field names can be taken from the `/api/subscriberpreferences/` API properties (Please, check API documentation).
`registrations`	Available field names can be taken from the `/api/subscriberregistrations/` API properties (Please, check API documentation).
`trusted_sources`	Available field names can be taken from the `/api/trustedsources/` API properties (Please, check API documentation).
`cf_mappings`	Available field names can be taken from the `/api/cfmappings/` API properties (Please, check API documentation).

The 'identifier' Sub-Field and Data Lookup

The **identifier** field is a comma-separated list composed by one or more JSON/SQL properties. It can be specified within any of the template keys of the table above. It is a string list that the batch processing engine uses for data lookup in the database.

For instance, let us assume that the following declaration has been defined:

Reference Example For The `identifier` Attribute.

```
contract_contact:
  identifier: "firstname, lastname, status"
  reseller: default
  firstname_code: "function() { return row.first_name; }"
  lastname_code: "function() { return row.last_name; }"
  status: "active"
contract:
  identifier: contact_id
  contact_id_code: "function() { return contract_contact.id; }"
  product: "Basic SIP Account"
  billing_profile: "Default Billing Profile"
```

Firstly, a *Contract* refers to a customer (i.e. a CloudPBX). When the engine processes a CSV file (or a Web form), it will search for a contract contact with the names specified in the **firstname** and **lastname** fields (e.g. "John, Doe"), and with the status field equals to "*active*".

If the contact entry does not yet exist, it will be created. At this stage, there are two situations:

Copyright © Sipwise GmbH - All rights reserved

a. Contact does not yet exist: A new one is created, with some new **contact_id** value.

b. Contact already exists: It is looked up and returns some existing **contact_id** value.

Regarding the **contract** tag, its identifier is the **contact_id** field. Again, the engine will try to look up for a contract with the **contact_id** previously obtained:

- When the contract contact was just created (Case (a) above), then there will neither be an existing contract. Hence, the contract is also created.
- In case (b), there is already a contact (e.g. "John Doe", which is active), then possibly also already a contract linked to it. Hence, the engine will not create another contract, but it looks up the existing one.
- Finally, the engine proceeds with creating subscribers and their corresponding settings.

Further sections will describe extra details on the template declaration.

7.37.3. Storing Data

When using internal attributes (i.e. defined by the calculated type), it is possible to store static data or code-calculated data (i.e. obtained from code snippets). Static data can be stored in any suitable attribute name according the template section in which will be declared. Notwithstanding, in order to store the output from a code snippet, that attribute name MUST be declared along with the `_code` suffix concatenated to it. Some examples:

- Within the **fields** key, the **value** attribute name needs to be used to store static data. Hence, the **value_code** attribute name must be declared in the template if a code snippet is put in place to calculate such data. When the data is computed, it will be stored in the **value** attribute.
- Within the **contract** key, **contact_id** is one of the several available attributes for a customer that can be declared in that section. If the value of this attribute needs to be computed, then the **contact_id_code** attribute name needs to be declared, along with the corresponding code snippet. Then, when the data is computed, it will be stored in the **contact_id** attribute.

7.37.4. Code Snippets

Code snippets allows to use programming logic to calculate attribute values. The supported programming languages are JavaScript and Perl.

The following directives must be declared to use code snippets in templates, for both JavaScript and Perl, respectively:

Code Snippet Declaration: JavaScript and Perl.

```
"function() {
  //JavaScript code logic
  ...
  return ... ; //Output to be stored in a *_code attribute name.
}"
```

```
"sub {
```

 Copyright © Sipwise GmbH - All rights reserved

```
  #Perl code logic
  ...
  return ... ; #Output to be stored in a *_code attribute name.
}”
```

Native JavaScript/Perl directives can be used as part of the code logic, such as string concatenations, number stripping, etc. Furthermore, there are reserved variable names and special functions that can be used in a code. These are detailed in the following sections.

Reserved Variable Names

Reserved variable names can be used in the definition of code snippets, summarized as follows.

Table 23. Reserved Variable Names.

Variable Name	Type	Description
`row`	*Object*	It represents the CSV file row (or the input form when filling out via Admin Panel) according the `fields` parameters defined in the template.
`contract_contact`	*Object*	Settings for the contact of the customer. It contains information such as the name, the postal and email addresses, among others.
`contract`	*Object*	Settings for customer. It refers to the `billing.contract` SQL table.
`contract_preferences`	*Object*	Settings for customer preferences.
`contract_balance`	*Object*	Customer cash balance and free time values.
`fraud_preferences`	*Object*	Customer fraud limit overrides.
`subscriber`	*Object*	Settings for subscriber details.
`subscriber_preferences`	*Object*	Settings for subscriber preferences.
`registrations`	*Object*	Settings for permanent registrations.
`trusted_sources`	*Object*	Settings for trusted sources.
`cf_mappings`	*Object*	Settings for call forward mappings.
`reseller`	*String*	Reseller name. The entity name that represents a collection of settings to provide telecommunication services to subscribers on the NGCP.
`billing_profile`	*String*	Billing profile name.
`profile_package`	*String*	Billing profile package name.
`domain`	*String*	SIP domain name. Particularly, it refers to the record for the `billing.domain` SQL table.
`provisioning_domain`	*String*	SIP domain name. Particularly, it refers to the `provisioning.voip_domain` SQL table.
`product`	*String*	The product type for the PBX customer. Either "*Basic SIP Account*" or "*Cloud PBX Account*".

Copyright © Sipwise GmbH - All rights reserved

now	*String*	It represents the transaction timestamp.

Special Functions

Apart from using native JavaScript/Perl functions, there are special utility functions available, summarized as follows.

Table 24. Special Functions.

Function Name	Arguments	Description

`split_number()`	*String*	It can split a number (entered as a text) into country code (`cc`), area code (`ac`) and subscriber number (`sn`). It MUST be used along with the definition of the `cc_ac_map` attribute name within the `fields` tag, as follows:
		
		`fields:`
		`- name: cc_ac_map`
		`  type: calculated`
		`  value:`
		`  - CC1:`
		`  - AC1:`
		`  - AC2:`
		`  ...`
		`  - CC2:`
		`  - AC1:`
		`  - AC2:`
		`  ...`
		
		Where:
		
		• `CC`: *Country Code number (e.g. 43, 56, etc.)*
		• `AC`: *Area Code number (e.g. 1, 2142, etc.)*

`debug()`	*String*	It prints 'debug' log information in the /var/log/ngcp/panel.log and /var/log/ngcp/panel-debug.log files when used via Admin Panel, or in the output of the `ngcp-provisioning-templates` script (depending on the `--log-level` option specified).
`info()`	*String*	It prints 'info' log information in the /var/log/ngcp/panel.log and /var/log/ngcp/panel-debug.log files when used via Admin Panel, or in the output of the `ngcp-provisioning-templates` script (depending on the `--log-level` option specified).

Copyright © Sipwise GmbH - All rights reserved

`warn()`	*String*	It prints 'warning' log information in the /var/log/ngcp/panel.log and /var/log/ngcp/panel-debug.log files when used via Admin Panel, or in the output of the `ngcp-provisioning-templates` script (depending on the `--log-level` option specified).
`error()`	*String*	It prints 'error' log information in the /var/log/ngcp/panel.log and /var/log/ngcp/panel-debug.log files when used via Admin Panel, or in the output of the `ngcp-provisioning-templates` script (depending on the `--log-level` option specified).

IMPORTANT

For Perl, the following core functions are disabled to prevent harmful code injection: *binmode(), close(), closedir(), dbmclose(), dbmopen(), eof(), fileno(), flock(), format(), getc(), read(), readdir(), rewinddir(), say(), seek(), seekdir(), select(), syscall(), sysread(), sysseek(), syswrite(), tell(), telldir(), truncate(), write(), print(), printf(), chdir(), chmod(), chown(), chroot(), fcntl(), glob(), ioctl(), link(), lstat(), mkdir(), open(), opendir(), readlink(), rename(), rmdir(), stat(), symlink(), sysopen(), umask(), unlink(), utime(), alarm(), exec(), fork(), getpgrp(), getppid(), getpriority(), kill(), pipe(), setpgrp(), setpriority(), sleep(), system(), times(), wait(), waitpid(), accept(), bind(), connect(), getpeername(), getsockname(), getsockopt(), listen(), recv(), send(), setsockopt(), shutdown(), socket(), socketpair(), msgctl(), msgget(), msgrcv(), msgsnd(), semctl(), semget(), semop(), shmctl(), shmget(), shmread(), shmwrite(), endgrent(), endhostent(), endnetent(), endpwent(), getgrent(), getgrgid(), getgrnam(), getlogin(), getpwent(), getpwnam(), getpwuid(), setgrent(), setpwent(), endprotoent(), endservent(), gethostbyaddr(), gethostbyname(), gethostent(), getnetbyaddr(), getnetbyname(), getnetent(), getprotobyname(), getprotobynumber(), getprotoent(), getservbyname(), getservbyport(), getservent(), sethostent(), setnetent(), setprotoent(), setservent(), exit(), goto().*

Finally, let us consider the following template section example:

Code-Snippet Example.

```
fields:
  - name: cc
    label: "Country Code:"
    type: Text
    required: 1
  - name: ac
    label: "Area Code:"
    type: Text
    required: 1
  - name: sn
    label: "Subscriber Number:"
    type: Text
    required: 1
  - name: sip_username
    type: calculated
```

Copyright © Sipwise GmbH - All rights reserved

```
    value_code: "function() {
                    return row.cc.concat(row.ac).concat(row.sn);
                }"
```

Here, the text input fields `cc`, `ac` and `sn` along with the internal field `sip_username` are declared. It is required for this case that `sip_username` must be a string concatenation of the `cc`, `ac` and `sn` fields, respectively. Therefore, `sip_username` type is declared as `calculated` and its value must be declared through the `value_code` attribute name. This attribute shows a code snippet which contains a JavaScript function that performs the string concatenation. The `row` variable refers to the CSV file row, or the form input when filling out the data through Admin Panel. Finally, when the data is computed, it will be stored in the `value` attribute.

7.37.5. Batch Provisioning via Admin Panel

Batch provisioning can be enabled as a global, system-wide configuration in the main configuration file `config.yml` through the `www_admin.batch_provisioning_features` property set to `1`. The feature is available for all administrative users of the platform, that covers users with generic administrator role and users with "customer care" role. These users will be referred as "admin users" later in this document.

When set, the Batch Provisioning page can be accessed through the *ToolsBatch Provisioning* menu entry. The interface will offer the option to create a new provisioning template, or edit an existing one.

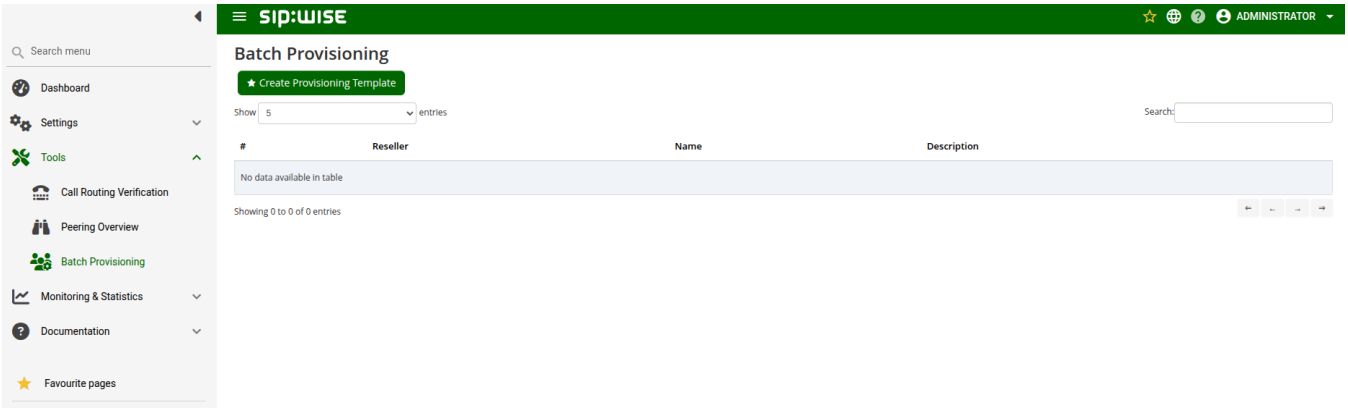

Figure 123. Main Batch Provisioning Page

For creation, press on the '*Create Provisioning Template*' button. This will pop-up a new window, as shown below.

Copyright © Sipwise GmbH - All rights reserved

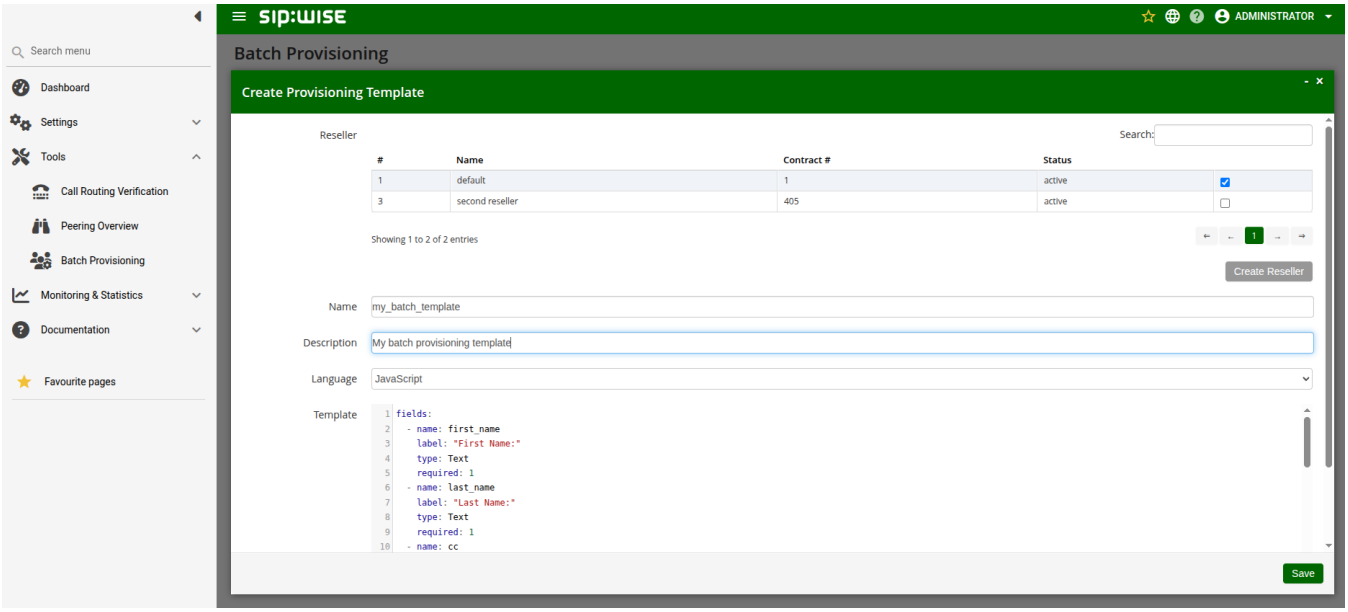

Figure 124. Create Provisioning Template

The parameters are as follows:

- **Reseller**: The reseller this template belongs to.
- **Name**: A free form string used to identify the provisioning template in the Admin Panel. This may be edited at any time.
- **Description**: Information about the provisioning template.
- **Language**: Scripting language used in the provisioning template for the 'calculated' fields. Either *JavaScript* (default) or *Perl*.
- **Template**: The provisioning template to be used. By default, there is a built-in template that has been created to set basic options. Please, check appendix for further details.

Once a template has been defined, there are two options available for data input:

- **Open Form**
- **Upload CSV**

Manual Entry for a Single Subscriber

This option is available when clicking on the '*Open Form*' button. A new window will be displayed to enter the variable data of the subscriber in several form fields:

Copyright © Sipwise GmbH - All rights reserved

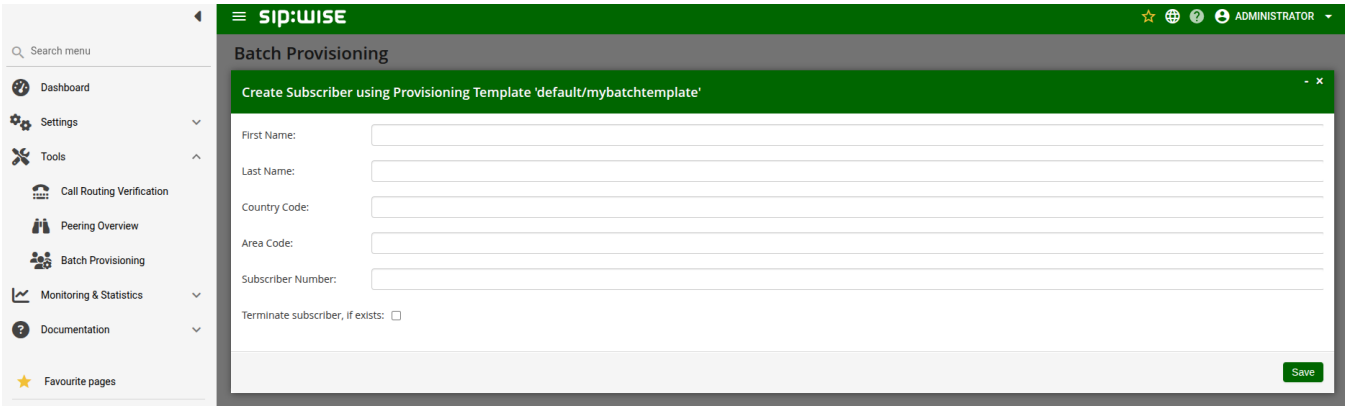

Figure 125. Single Subscriber Provisioning

The fields displayed will depend on what is defined on the provisioning template. If the default built-in template is in use, then parameters are as follows:

- **First Name**: The given name of the customer in which the subscriber will belong to.
- **Last Name**: The surname of the customer in which the subscriber will belong to.
- **Country Code (CC)**: Country code of the subscriber telephone number.
- **Area Code (AC)**: Area code of the subscriber telephone number.
- **Subscriber Number (SN)**: Telephone number of the subscriber.
- **Terminate subscriber, if exists**: If ticked, it will terminate any existing subscriber with that name.

Bulk Entry for More Subscribers

This option is available when clicking on the '*Upload CSV*' button, which allows to upload a CSV file that contains a line of data for each subscriber. A new window will be displayed to enter the data:

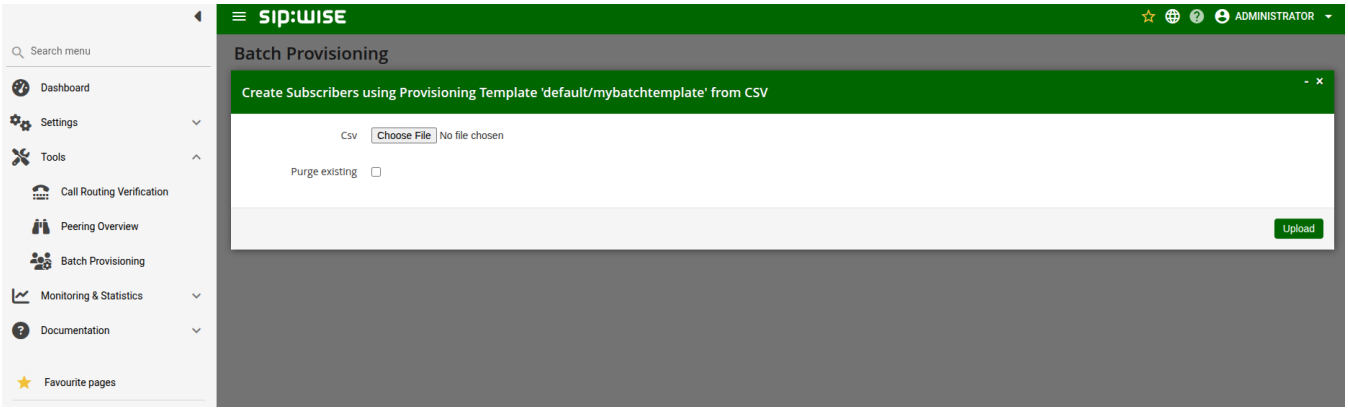

Figure 126. Bulk Subscriber Provisioning

The parameters are as follows:

- **CSV**: The CSV file to upload. The file has to follow the format defined in the template.
- **Purge Existing**: If ticked, it will terminate any existing subscribers with that name. This checkbox is equivalent to specify the value 1 in each row of the CSV file.

 Copyright © Sipwise GmbH - All rights reserved

7.37.6. Batch Provisioning via CLI

NGCP offers the `ngcp-provisioning-template` command line tool for batch provisioning, which allows to run a 'provisioning template' from database or from config.yml file. This will produce a subscriber setup including required billing contact, contract, preferences, etc. from an input form defined by that template.

Usage:

For templates defined in *config.yml* file:

```
ngcp-provisioning-template "provisioning-template-name" [options]
```

For templates defined in database:

```
ngcp-provisioning-template "reseller-name/provisioning-template-name"
[options]
```

NOTE For templates defined in database, different resellers could each have a provisioning template with the same name.

The form fields can be passed as command line options, as described below.

Table 25. Usage of The ngcp-provisioning-template Command.

Option	Description
`--help`	Prints a brief help message and exits.
`--db-host=db-host-IP-address`	The host of the ngcp database to connect to. If omitted, the database connection settings of ngcp-panel will be used.
`--db-port=db-host-port`	The port of the ngcp database to connect to. Only relevant if `--db-host` is specified.
`--db-user=db-username`	The database user for the ngcp database to connect to. Only relevant if `--db-host` is specified.
`--db-password=db-password`	The database user password (if any) for the ngcp database to connect to. Only relevant if `--db-host` is specified.
`--file=csv-filename`	Specify a CSV filename to process. Each row represents form values for one subscriber to create.
`--[input-attribute -name]=[attribute-value]`	Provide an input attribute name and its value, as defined within the `fields` tag in the template (Attribute must not be internal, but form visible). Only relevant if no `--file` is specified.
`--purge`	Terminate an existing subscriber with duplicate number/aliases first.
`--log-level`	Verbosity level of printed messages while processing (`error`, `warn`, `info`, `debug`).

Copyright © Sipwise GmbH - All rights reserved

By default, database connection parameters are read from the `/etc/ngcp-panel/provisioning.conf` configuration file. For developing and/or testing reasons, it is possible though to use different database connection parameters, which can be specified with the `--db-*` arguments.

If no CSV file is specified with the `--file` option, each of the non-internal input parameters defined in the template (within the fields tag) can be provided as part of the command line options. For instance, if `first_name` input variable is declared in the template, then such option can be entered as `--first_name` (e.g. `--first_name=John`).

Batch provisioning is based on transactional processing. If provisioning fails at a specific point, it will rollback what was created up to that point, in order to avoid residual data in the database.

TIP When developing a template, it is best to use the `ngcp-provisioning-template` script with debug enabled, since it is possible to check log messages directly on the command output.

Some examples for single subscriber use, and CSV-file use.

Single subscriber (Debug level on):

```
ngcp-provisioning-template "default/My Template" \
  --log-level=debug \
  --first_name=Peter \
  --last_name=White \
  --cc=43 \
  --ac=1 \
  --sn=2521523 \
  --purge
```

CSV File (Debug level on):

```
ngcp-provisioning-template "default/My Template" \
  --log-level=debug \
  --file=bulk.csv \
  --purge
```

7.37.7. Batch Provisioning via API

Batch provisioning is also available in REST API through the `/api/provisioningtemplates/` path. Please, check API documentation for details about available HTTP methods, properties and query parameters.

The following sub-sections aim to highlight common use cases. Let us assume the following template base data will be used.

Table 26. Example Of API Provisioning Data.

Field	Value

 Copyright © Sipwise GmbH - All rights reserved

Reseller	default
Reseller ID	1
SIP Domain	mydomain.com
Product	Basic SIP Account
Template Name	*My API Provisioning Template*
Template Language	js (JavaScript)
Template Content	Same as Built-in Template
API IP <Address:Port>	127.0.0.1:1443
API Key File	./apiclient.pem

Template Creation

For template creation, the HTTP `POST` method is available. Template metadata can be specified in JSON notation using the `name`, `description`, `reseller_id`, and `lang` attributes. Template definition can be provided in JSON notation within the `template` attribute, as shown below.

Example of Template Creation via API.

```
curl -ki --cert ./apiclient.pem -X POST \
  -H 'Connection: close' \
  -H 'Content-Type: application/json' \
  "https://127.0.0.1:1443/api/provisioningtemplates/" \
  --data-binary '{
    "description" : "My API-Created Template",
    "name" : "My API Provisioning Template",
    "reseller_id" : 1,
    "lang" : "js",
    "template" : {
      "fields" : [
        {
           "name" : "first_name",
           "label" : "First Name:",
           "type" : "Text",
           "required" : "1"
        },
        {
          "name" : "last_name",
          "label" : "Last Name:",
          "type" : "Text",
          "required" : "1"
        },
        {
          "label" : "Country Code:",
          "name" : "cc",
          "type" : "Text",
          "required" : "1"
        },
```

Copyright © Sipwise GmbH - All rights reserved

```
        {
          "name" : "ac",
          "label" : "Area Code:",
          "type" : "Text",
          "required" : "1"
        },
        {
          "name" : "sn",
          "label" : "Subscriber Number:",
          "type" : "Text",
          "required" : "1"
        },
        {
          "name" : "sip_username",
          "type" : "calculated",
          "value_code" : "function() { return
row.cc.concat(row.ac).concat(row.sn); }"
        },
        {
          "name" : "purge",
          "label" : "Terminate subscriber, if exists:",
          "type" : "Boolean"
        }
      ],
      "contract_contact" : {
        "identifier" : "firstname, lastname, status",
        "reseller" : "default",
        "firstname_code" : "function() { return row.first_name; }",
        "lastname_code" : "function() { return row.last_name; }",
        "status" : "active"
      },
      "contract" : {
        "product" : "Basic SIP Account",
        "billing_profile" : "Default Billing Profile",
        "identifier" : "contact_id",
        "contact_id_code" : "function() { return contract_contact.id; }"
      },
      "subscriber" : {
        "domain" : "mydomain.com",
        "primary_number" : {
          "cc_code" : "function() { return row.cc; }",
          "ac_code" : "function() { return row.ac; }",
          "sn_code" : "function() { return row.sn; }"
        },
        "username_code" : "function() { return row.sip_username; }",
        "password_code" : "function() { return row.sip_password; }"
      },
      "subscriber_preferences" : {
        "gpp0" : "test"
      }
    }
```

 Copyright © Sipwise GmbH - All rights reserved

```
  }'
```

Template Request

For template request, the HTTP `GET` method is available. A specific template can be fetched according the id value of the template, which corresponds to the concatenation of reseller name, the `/` symbol and the template name. Based on the previous example creation, the corresponding value is "*default/My API Provisioning Template*".

Example of Template Request via API.

```
curl -ki --cert ./apiclient.pem -X GET \
  -H 'Connection: close' \
  "https://127.0.0.1:1443/api/provisioningtemplates/default/My API
Provisioning Template/"
```

NOTE On batch provisioning templates, the `id` parameter is a *String* type (Rather than an *Integer* type, as commonly used on other API commands).

`GET` method will output the template definition within the template attribute, which output format can be controlled by using the `?format=[value]` directive. The available values are: `yml`, `yaml` or `json`. Example:

```
curl -ki --cert ./apiclient.pem -X GET \
  -H 'Connection: close' \
  "https://127.0.0.1:1443/api/provisioningtemplates/default/My API
Provisioning Template/?format=json"
```

Additionally, as templates can be stored on database or in config.yml file, it is possible to use the Boolean editable parameter to list: (a) Database-stored templates only (`true`); or (b) Templates defined on config.yml file only (`false`). Example:

```
curl -ki --cert ./apiclient.pem -X GET \
  -H 'Connection: close' \
  "https://127.0.0.1:1443/api/provisioningtemplates/?editable=false"
```

NOTE Templates defined on config.yml will be displayed on Admin Panel, but they cannot be edited.

Template Update

For template update, the HTTP `PUT` and `PATCH` methods are available.

For `PUT`, it is possible to set the whole template at once only. Let us assume that the `gpp0` attribute (located within the `subscriber_preferences` tag) will be modified to other value, meanwhile the rest of template properties will remain the same.

Copyright © Sipwise GmbH - All rights reserved

Example of Template Update Via API (PUT).

```
curl -ki --cert ./apiclient.pem -X PUT \
  -H 'Connection: close' \
  -H 'Content-Type: application/json' \
  "https://127.0.0.1:1443/api/provisioningtemplates/default/My API
Provisioning Template/" \
  --data-binary '{
    "description" : "My API-Created Template",
    "name" : "My API Provisioning Template",
    "reseller_id" : 1,
    "lang" : "js",
    "template" : {
      "fields" : [
        {
          "name" : "first_name",
          "label" : "First Name:",
          "type" : "Text",
          "required" : "1"
        },
        {
          "name" : "last_name",
          "label" : "Last Name:",
          "type" : "Text",
          "required" : "1"
        },
        {
          "label" : "Country Code:",
          "name" : "cc",
          "type" : "Text",
          "required" : "1"
        },
        {
          "name" : "ac",
          "label" : "Area Code:",
          "type" : "Text",
          "required" : "1"
        },
        {
          "name" : "sn",
          "label" : "Subscriber Number:",
          "type" : "Text",
          "required" : "1"
        },
        {
          "name" : "sip_username",
          "type" : "calculated",
          "value_code" : "function() { return
row.cc.concat(row.ac).concat(row.sn); }"
        },
        {
```


Copyright © Sipwise GmbH - All rights reserved

```
          "name" : "purge",
          "label" : "Terminate subscriber, if exists:",
          "type" : "Boolean"
        }
      ],
      "contract_contact" : {
        "identifier" : "firstname, lastname, active",
        "reseller" : "default",
        "firstname_code" : "function() { return row.first_name; }",
        "lastname_code" : "function() { return row.last_name; }",
        "status" : "active"
      },
      "contract" : {
        "product" : "Basic SIP Account",
        "billing_profile" : "Default Billing Profile",
        "identifier" : "contact_id",
        "contact_id_code" : "function() { return contract_contact.id; }"
      },
      "subscriber" : {
        "domain" : "mydomain.com",
        "primary_number" : {
          "cc_code" : "function() { return row.cc; }",
          "ac_code" : "function() { return row.ac; }",
          "sn_code" : "function() { return row.sn; }"
        },
        "username_code" : "function() { return row.sip_username; }",
        "password_code" : "function() { return row.sip_password; }"
      },
      "subscriber_preferences" : {
        "gpp0" : "My new gpp0 value"
      }
    }
  }'
```

For `PATCH`, it is possible to modify main template attributes at once only. Sub-attributes belonging to a section will need to be replaced as a whole. Let us assume that the `description` attribute and the `gpp0` attribute (located within the `template.subscriber_preferences` section) will be updated to different values, respectively.

Example of Template Update Via API (PATCH).

```
curl -ki --cert ./apiclient.pem -X PATCH \
  -H 'Connection: close' \
  -H 'Content-Type: application/json-patch+json' \
  "https://127.0.0.1:1443/api/provisioningtemplates/default/My API
Provisioning Template/" \
  --data-binary '[
    { "op" : "replace", "path" : "/description", "value" : "My new
description for API-Created Template" },
    { "op" : "replace", "path" : "/template", "value" : {
      "fields" : [
```

Copyright © Sipwise GmbH - All rights reserved

```
        {
          "name" : "first_name",
          "label" : "First Name:",
          "type" : "Text",
          "required" : "1"
        },
        {
          "name" : "last_name",
          "label" : "Last Name:",
          "type" : "Text",
          "required" : "1"
        },
        {
          "label" : "Country Code:",
          "name" : "cc",
          "type" : "Text",
          "required" : "1"
        },
        {
          "name" : "ac",
          "label" : "Area Code:",
          "type" : "Text",
          "required" : "1"
        },
        {
          "name" : "sn",
          "label" : "Subscriber Number:",
          "type" : "Text",
          "required" : "1"
        },
        {
          "name" : "sip_username",
          "type" : "calculated",
          "value_code" : "function() { return
row.cc.concat(row.ac).concat(row.sn); }"
        },
        {
          "name" : "purge",
          "label" : "Terminate subscriber, if exists:",
          "type" : "Boolean"
        }
      ],
      "contract_contact" : {
        "identifier" : "firstname, lastname, status",
        "reseller" : "default",
        "firstname_code" : "function() { return row.first_name; }",
        "lastname_code" : "function() { return row.last_name; }",
        "status" : "active"
      },
      "contract" : {
        "product" : "Basic SIP Account",
        "billing_profile" : "Default Billing Profile",
```


Copyright © Sipwise GmbH - All rights reserved

```
        "identifier" : "contact_id",
        "contact_id_code" : "function() { return contract_contact.id; }"
      },
      "subscriber" : {
        "domain" : "mydomain.com",
        "primary_number" : {
          "cc_code" : "function() { return row.cc; }",
          "ac_code" : "function() { return row.ac; }",
          "sn_code" : "function() { return row.sn; }"
        },
        "username_code" : "function() { return row.sip_username; }",
        "password_code" : "function() { return row.sip_password; }"
      },
      "subscriber_preferences" : {
        "gpp0" : "My new gpp0 value"
      }
     }
    }
  ]'
```

Template Deletion

For template deletion, the HTTP `DELETE` method is available. It is required to provide the `id` value (*String*) of the template. Based on the examples, the value is "*default/My API Provisioning Template*".

Example of Template Deletion Via API.

```
curl -ki --cert ./apiclient.pem -X DELETE \
  -H 'Connection: close' \
  "https://127.0.0.1:1443/api/provisioningtemplates/default/My API
Provisioning Template/"
```

API Data Provisioning: Single Subscriber

A `POST` can be requested to provision a single subscriber by using JSON. Let us assume that the following single subscriber data will be provisioned:

Table 27. Example Of Single Subscriber Provisioning Via API.

Customer Contact Info		Subscriber			Purge?
First Name	Surname	CC	AC	SN	
Mark	Brown	56	2	33445511	yes

Then, the corresponding API command will be:

Example of Single Subscriber Provisioning Via API.

```
curl -ki --cert ./apiclient.pem -X POST \
```

Copyright © Sipwise GmbH - All rights reserved

```
  -H 'Connection: close' \
  -H 'Content-Type: application/json' \
  "https://127.0.0.1:1443/api/provisioningtemplates/default/My API
Provisioning Template/" \
  --data-binary '{
    "first_name" : "Mark" ,
    "last_name" : "Brown" ,
    "cc" : "56" ,
    "ac" : "2",
    "sn" : "33445511" ,
    "purge" : true
  }'
```

API Data Provisioning: Batch of Subscribers

To bulk-upload subscribers, it is required to specify the Content-Type as `text/csv` and POST the CSV file in the request body. Let us assume that the following subscriber data will be provisioned in the bulk.csv file:

Table 28. Example Of Bulk Subscriber Provisioning Via API.

Customer Contact Info		Subscriber(s)			Purge?
First Name	Surname	CC	AC	SN	
James	Smith	56	2	33445511	yes
Richard	Stone	56	2	33445522	yes
John	Doe	43	123	1001 1002 1003 1004 1005	yes

The CSV file will contain:

```
James, Smith, 56, 2, 33445511, 1
Richard, Stone, 56, 2, 33445522, 1
John, Doe, 43, 123, 1001, 1
John, Doe, 43, 123, 1002, 1
John, Doe, 43, 123, 1003, 1
John, Doe, 43, 123, 1004, 1
John, Doe, 43, 123, 1005, 1
```

Then, the corresponding API command will be:

Example of Bulk Subscriber Provisioning Via API.

```
curl -ki --cert ./apiclient.pem -X POST \
  -H 'Connection: close' \
```

 Copyright © Sipwise GmbH - All rights reserved

```
  -H 'Content-Type: text/csv' \
  "https://127.0.0.1:1443/api/provisioningtemplates/default/My API
Provisioning Template/" \
  --data-binary '@./bulk.csv'
```

7.38. Call List Suppressions

7.38.1. Overview

Call List Suppressions allows to filter specific numbers from Call History and Invoices. The feature aims to support country regulations such as gender violation or abuse of women, in which calls made to specific service numbers should not be made visible in any way to the aggressor.

The feature can be set via Admin Panel.

Its main advantages are as follows:

a. **Filter Calls** Matched call numbers (against a regular expression) will not appear at all.

b. **Obfuscate Calls** Matched call numbers (against a regular expression) will be replaced by a given label.

The following sections will provide more details about the feature.

7.38.2. Provisioning via Admin Panel

Call List Suppressions can be accessed at *SettingsCall List Suppressions* section. The interface will offer the option to create a new entry, edit any of the existing ones, or upload/download via CSV.

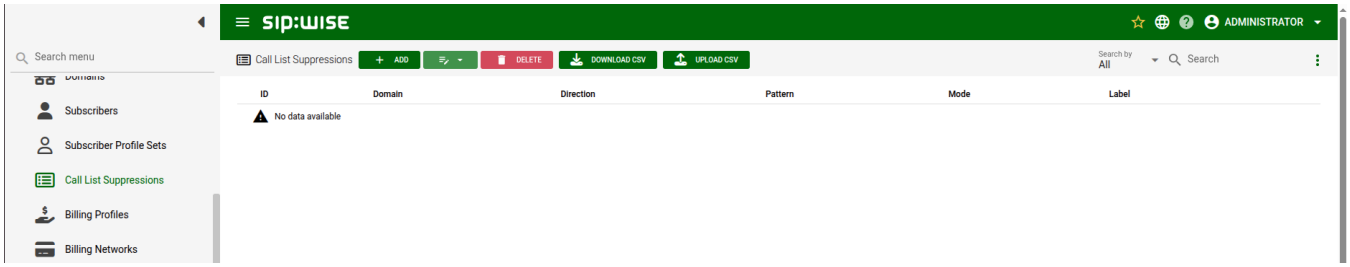

Figure 127. Call List Suppressions

7.38.3. Single Entry

For creation, press on the "*Add*" button.

Copyright © Sipwise GmbH - All rights reserved

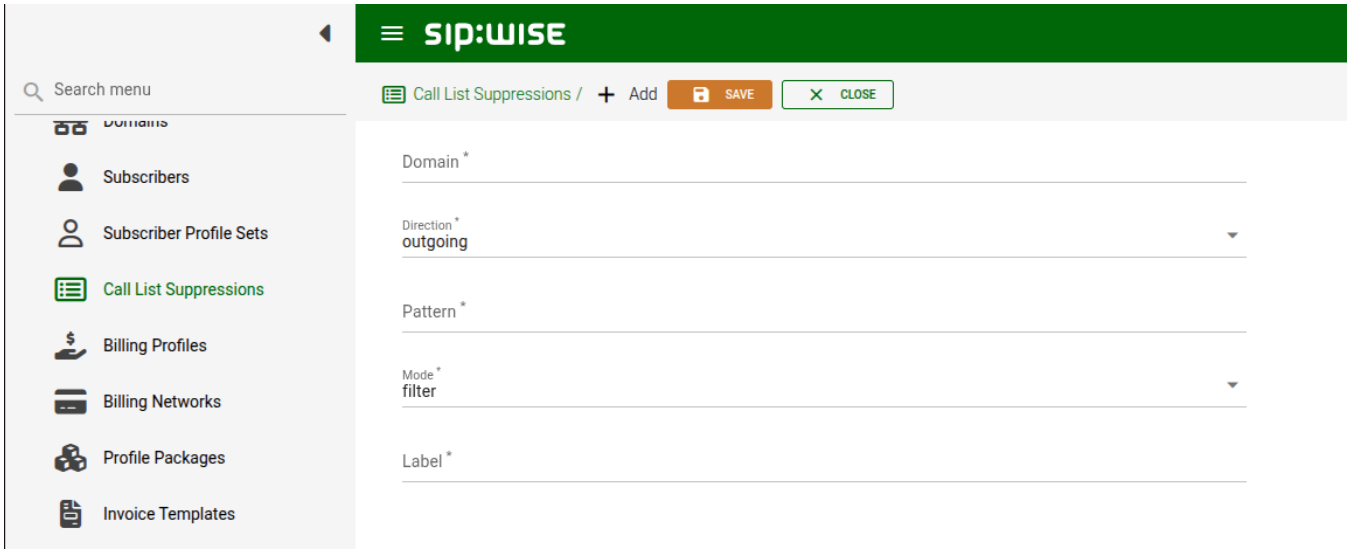

Figure 128. Subscriber Location Mappings: Create Entry

Parameters are as follows:

- **Domain**: The domain of subscribers in which this call list suppression applies to. An empty domain means to apply it to subscribers of any domain.
- **Direction**: The direction (*outgoing* or *incoming*) of calls this call list suppression applies to.
- **Pattern**: A regular expression the dialed number (CDR *destination_user_in* field) has to match in case of *outgoing* direction, or the inbound number (CDR *source_cli* field) in case of incoming direction.
- **Mode**: It defines the suppression mode. For subscriber and subscriber admins, the *filter* option means that matching calls will not appear at all, while the *obfuscate* option means that the number is replaced by the given label. Also, the mode can be disabled by selecting the *disabled* option.
- **Label**: A string name to be defined for this list. Admin and reseller users see it for *filter* mode suppressions. For the *obfuscate mode*, this label will be used as a replacement string.

7.38.4. Bulk Entry

This option is available when clicking on the "*Upload CSV*" button, which allows to upload a CSV file that contains a line of data for each list. A new window will be displayed to enter the data:

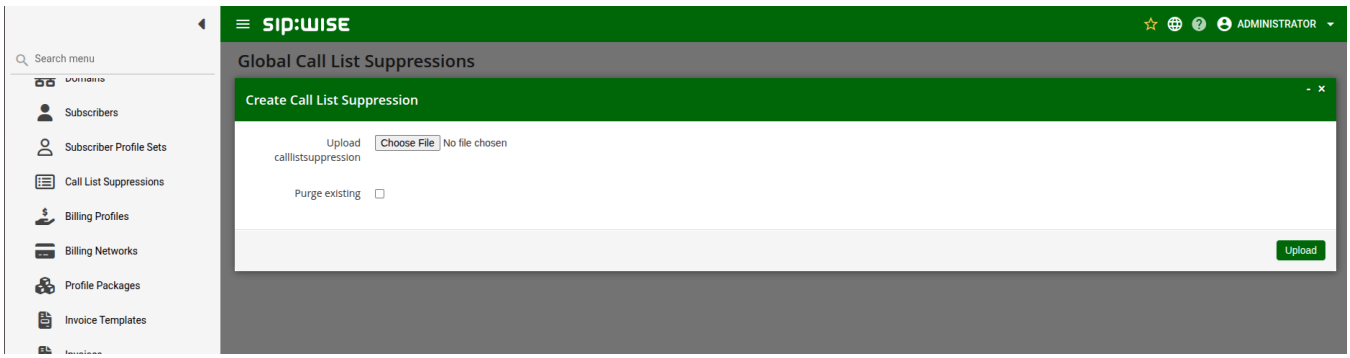

Figure 129. Bulk Provisioning

The parameters are as follows:

 Copyright © Sipwise GmbH - All rights reserved

- **CSV**: The CSV file to upload. The file has to follow the same order as defined for single entries.
- **Purge Existing**: If ticked, it will delete any existing list before provisioning the new one.

7.38.5. Practical Example

Let us assume that an administrator will use Admin Panel to provision two call list suppression on a selected number. The following details are considered:

Table 29. Practical Example - Input Data.

Domain	Direction	Pattern	Mode	Label
voipdomain.com	`outgoing`	^571016$	`filter`	*cls-a*
voipdomain.com	`outgoing`	^571017$	`obfuscate`	*cls-b*

In the provided example, the logic is as following:

Table 30. Practical Example - Logic & Result.

Logic	Result
An outgoing call to the destination number 571016 is made.	• The call will not be shown in the call history (*Conversations* section) of the Customer Self-Care interface. • An NGCP administrator will be able to see the call in the call history via Admin Panel, with the label "*filtered: cls-a*". • The call will not be shown on NGCP-generated invoices.
An outgoing call to the destination number 571017 is made.	• The call will be shown in the call history (*Conversations* section) of the Customer Self-Care interface as "*Call to cls-b*". • An NGCP administrator will be able to see the call in the call history, with the label "*obfuscated: cls-b*". • The call will be shown as "*cls-b*" in NGCP-generated invoices.

7.39. Message Body Filtering

In some cases it is required to filter out from message bodies some specific multipart/mixed body because not supported by the endpoints. This can be simply achieved using the `config.yml` option `kamailio.lb.filter_content_type`. The filter will be executed on:

- all the INVITE requests that contain body
- all the replies with return code 200 and 183 that contain body

To configure it, proceed following the next steps:

1. set the `kamailio.lb.filter_content_type.enable` option to "yes".
2. set the `kamailio.lb.filter_content_type.action` to "filter" or "drop". If one of the listed content-types is found, the first option will strip out all the bodies except for application/sdp from the message, the second one will simply drop the message.

Copyright © Sipwise GmbH - All rights reserved

3. list in `kamailio.lb.filter_content_type.content_type_list` the content_types and the direction (request, reply or all) where the action has to be applied.
4. apply the changes executing *ngcpcfg apply 'enabled filter_content_type'*.

NOTE The action 'drop' works only for 'requests' and not for 'replies'.

For example, to filter out from all the reply messages the 'application/isup' content-type and from all the messages the 'application/vnd.etsi.cug+xml', you have to configure the following:

```
kamailio:
  lb:
    filter_content_type:
      enable: yes
      action: filter (or drop)
      content_type_list:
      - content_type: application/isup
        direction: reply
      - content_type: application/vnd.etsi.cug+xml
        direction: all
```

NOTE `kamailio.lb.filter_content_type` definition substitutes the already existing `kamailio.lb.remove_isup_body_from_replies`, extending its functionality to other content types. The migration from the previous setting to the new one is automatically done during the upgrade.

7.40. Generation of 181 Call Is Being Forwarded

7.40.1. Overview

SIP message **181 - Call Is Being Forwarded** is an optional provisional response that can be sent towards the caller user to inform it of an ongoing call forward. The message can also contain History-Info headers necessary to the caller to identify which is the new destination of the call.

7.40.2. How to Enable It

By default Sipwise C5 does not generate any 181 message. To enable this feature two steps are required:

- set the `config.yml` option `kamailio.proxy.cf_send_181.enable` to "yes"
- set the `config.yml` option `b2b.sbc.reset_tag_on_fork` to "yes"
- set the preference *outbound_history_info* in subscriber/domain/peer preferences

The first preference will actually activate the 181 message generation. The second one, instead, allows SEMS to forward back to caller all the provisional messages even if they contain a different To-Tag. Without this second preference set, the 181 message is sent towards the caller but all the following 18x messages are blocked by SEMS. The third one adds the History-Info headers with the information of the forwarded and of the final destination to the 181 message.

Copyright © Sipwise GmbH - All rights reserved

7.40.3. How It Works

When a call forward is triggered in Sipwise C5, Kamailio Proxy stores in the Redis database the History-Info headers that are added to the outgoing INVITE message. Kamailio LB receives the outgoing INVITE and it generates a *182 - Connecting* message back towards the caller. The 182 message is received by Kamailio Proxy that converts it in a *181 - Call Is Being Forwarded* message and adds the History-Info headers previously stored in the Redis DB, if any. The message is then sent back to the caller.

Copyright © Sipwise GmbH - All rights reserved

Chapter 8. Configuration Framework

The Sipwise C5 provides a configuration framework for consistent and easy to use low level settings management. A basic usage of the configuration framework only needs two actions already used in previous chapters:

- Edit */etc/ngcp-config/config.yml* file.
- Execute *ngcpcfg apply 'my commit message'* command.

Low level management of the configuration framework might be required by advanced users though. This chapter explains the architecture and usage of Sipwise C5 configuration framework. If the basic usage explained above fits your needs, feel free to skip this chapter and return to it when your requirements change.

A more detailed workflow of the configuration framework for creating a configuration file consists of 8 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 post-build actions after the build process, such as synchronizing the database credentials, grants and timezones. This step is called *post-build* in the configuration framework.
- 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.

8.1. Configuration Templates

The Sipwise C5 provides configuration file templates for most of the services it runs. These templates are stored in the directory */etc/ngcp-config/templates*.

Example: Template files for */etc/kamailio/proxy/kamailio.cfg* are stored in */etc/ngcp-config/templates/etc/kamailio/proxy/*.

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

8.1.1. .tt2, .customtt.tt2 and .patchtt.tt2 Files

These files are the main template files that will be used to generate the final configuration file for the running service. They contain all the configuration options needed for a running Sipwise C5 system. The configuration framework will combine these files with the values provided by *config.yml*, *constants.yml* and *network.yml* to generate the appropriate configuration file.

 Copyright © Sipwise GmbH - All rights reserved

Example: Let us say we are changing the IP used by kamailio load balancer on interface *eth0* to IP 203.0.113.42. 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/lb/kamailio.cfg.tt2* will show a line like this:

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

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

```
listen=udp:203.0.113.42:5060
```

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

Instead of editing .tt2 files, the configuration framework recognises .customtt.tt2 files. These files are the same as .tt2, but they have higher priority when the configuration framework creates the final configuration files. If you need to introduce changes in a template, you must always copy the required .tt2 file to .customtt.tt2, make changes in the latter file only and leave the .tt2 file untouched. This way, the system will use the new custom configuration allowing you to switch back to the original one quickly.

Example: We will create */etc/ngcp-config/templates/etc/lb/kamailio.cfg.customtt.tt2* and use it for our customized configuration. In this example, we will append a comment at the end of the template.

```
cd /etc/ngcp-config/templates/etc/kamailio/lb
cp kamailio.cfg.tt2 kamailio.cfg.customtt.tt2
echo '# This is my last line comment' >> kamailio.cfg.customtt.tt2
ngcpcfg apply 'my commit message'
```

The ngcpcfg command will generate */etc/kamailio/lb/kamailio.cfg* from our custom template instead of the general one:

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

WARNING users have to upgrade all .customtt.tt2 manually every time .tt2 is upgraded, as ngcpcfg completely ignores new code in .tt2 received from new package version.

The huge drawback of .customtt.tt2 files are necessity to keep them up-to-date manually. Keeping them outdated will cause the system misbehaviour as different components will use different code version (as new .tt2 version will be overwritten by old .customtt.tt2).

The .patchtt.tt2 concept should help users here. It will minimise the manual efforts by using the "patch(1)" utility. The ngcpcfg tool is searching for .patchtt.tt2 files every time 'ngcpcfg build' has been called. If .patchtt.tt2 is detected, the ngcpcfg tool will try to apply .patchtt.tt2 on .tt2 and store result in

Copyright © Sipwise GmbH - All rights reserved

.customtt.tt2 if no conflicts noticed during patching. Further building process happens in a common way. Example:

```
root@spce:~# ngcpcfg build /etc/kamailio/lb/kamailio.cfg
spce: yml configs were validated successfully
spce: configs were checked successfully
spce: Validating patch '/etc/ngcp-
config/templates/etc/kamailio/lb/kamailio.cfg.patchtt.tt2'
spce: Applying patch '/etc/ngcp-
config/templates/etc/kamailio/lb/kamailio.cfg.patchtt.tt2'
spce: Successfully created '/etc/ngcp-
config/templates/etc/kamailio/lb/kamailio.cfg.customtt.tt2'
spce: Requested patchtt operation has finished successfully.
Loading /etc/ngcp-config/config.yml in memory: OK
Loading /etc/ngcp-config/network.yml in memory: OK
Loading /etc/ngcp-config/constants.yml in memory: OK
spce: Generating /etc/kamailio/lb/kamailio.cfg: OK
spce: Executing postbuild for /etc/kamailio/lb/kamailio.cfg
root@spce:~#
```

To convert some/all the current .customtt.tt2 users can use command *ngcpcfg patch --from-customtt [<customtt_file>]*:

```
root@spce:~# ngcpcfg patch --from-customtt /etc/ngcp-
config/templates/etc/kamailio/lb/kamailio.cfg.customtt.tt2
spce: Validating customtt '/etc/ngcp-
config/templates/etc/kamailio/lb/kamailio.cfg.customtt.tt2'
spce: Creating patchtt file '/etc/ngcp-
config/templates/etc/kamailio/lb/kamailio.cfg.patchtt.tt2'
spce: Requested customtt operation has finished successfully.
root@spce:~#
```

Here is the example of newly created .patchtt.tt2 file:

```
root@spce:~# cat /etc/ngcp-
config/templates/etc/kamailio/lb/kamailio.cfg.patchtt.tt2
@@ -1799,3 +1799,4 @@
 }

 # vim: ft=cfg
+# This is my last line comment
root@spce:~#
```

See more details about .patchtt.tt2 files below in patchtt section.

TIP The .tt2 files use the Template Toolkit language. Therefore you can use all the feature this excellent toolkit provides within ngcpcfg's template files (all the ones with the .tt2 suffix).

Copyright © Sipwise GmbH - All rights reserved

8.1.2. Using patchtt for Generation of a Relevant customtt File

Keeping custom modifications directly in the .customtt.tt2 templates is NOT recommended as templates become outdated with every software upgrade.

A better way is to handle custom modifications using .patchtt.tt2 files (e.g. /etc/ngcp-config/templates/etc/cron.d/cleanup-tools.patchtt.tt2). In this case, on every "ngcpcfg patch", a .patchtt.tt2 file will be applied on top of the .tt2 file and the result will be saved into the customtt file and used commonly as described in the previous section. "ngcpcfg patch" is the first step on "ngcpcfg build" that guarantees the latest upstream templates with the availability of the necessary local changes on every configuration apply.

TIP The patch to be applied to the corresponding .tt2 template file is selected in the following order (highest to lowest): *.patchtt.tt2.$HOSTNAME *.patchtt.tt2.$PAIRNAME *.patchtt.tt2.$HA_NODE *.patchtt.tt2

NOTE If a suitable patchtt file is found for a template, then the ngcpcfg patch command will overwrite the corresponding customtt file, if any.

Creating a patchtt File

Let us see how to introduce custom changes into a template through a patchtt file. For example, we need to change the accounting records cleanup time, which is defined in *cleanup-tools.tt2*. Here is how to do this:

- Go to the corresponding templates directory:

```
cd /etc/ngcp-config/templates/etc/cron.d/
```

- Duplicate the required .tt2 file to .customtt.tt2

```
cp ./cleanup-tools.tt2 ./cleanup-tools.customtt.tt2
```

- Introduce the necessary changes to the duplicated file:

```
vim ./cleanup-tools.customtt.tt2
```

- Create the patchtt file from your customtt file and recheck it:

```
ngcpcfg patch --from-customtt ./cleanup-tools.customtt.tt2
cat ./cleanup-tools.patchtt.tt2
```

- Apply and push the changes

```
ngcpcfg apply "Change acc-cleanup time from 00 to 02 hours"
ngcpcfg push all
```

Copyright © Sipwise GmbH - All rights reserved

You will notice that the "ngcpcfg apply" command has generated the customtt file for the corresponding template:

```
root@web01a:/etc/ngcp-config/templates/etc/cron.d# ls -l ./cleanup-
tools*
-rw------- 1 root root 932 Jan  4 11:11 ./cleanup-tools.customtt.tt2
-rw-r--r-- 1 root root 630 Jan  4 11:08 ./cleanup-tools.patchtt.tt2
-rw-r--r-- 1 root root 932 Dec 18 15:09 ./cleanup-tools.tt2
```

Now, even if cleanup-tools.tt2 slightly changes after a software upgrade, "ngcpcfg apply" will still preserve your custom changes.

NOTE If in a new release the .tt2 file gets changed in the same lines where you had introduced custom changes (e.g. your changes were temporary until a feature is implemented properly in a new software release), the apply process will fail and ask you to review the corresponding .patchtt.tt2 file. Then, check it and either correct if it is still required or remove it.

TIP To convert all existing customtt files to patchtt files use the command: **ngcpcfg patch --from-customtt**

8.1.3. .prebuild and .postbuild Files

After creating the configuration files, the configuration framework can execute some commands before and after placing that file in its target directory. These commands usually are used for changing the file's owner, groups, or any other attributes. There are some rules these commands need to match:

- They have to be placed in a *.prebuild* or *.postbuild* file in the same path as the original *.tt2* file.
- The file name must be the same as the configuration file, but having the mentioned suffixes.
- The commands must be *bash* compatible.
- The commands must return 0 if successful.
- The target configuration file is matched by the environment variable *output_file*.

Example: We need *www-data* as owner of the configuration file */etc/ngcp-ossbss/provisioning.conf*. The configuration framework will by default create the configuration files with root:root as owner:group and with the same permissions (rwx) as the original template. For this particular example, we will change the owner of the generated file using the *.postbuild* mechanism.

```
echo 'chgrp www-data ${output_file}' \
  > /etc/ngcp-config/templates/etc/ngcp-
ossbss/provisioning.conf.postbuild
```

8.1.4. .services Files

.services files are pretty similar and might contain commands that will be executed after the *build* process. There are two types of *.services* files:

- The particular one, with the same name as the configuration file it is associated to.

 Copyright © Sipwise GmbH - All rights reserved

Example: */etc/ngcp-config/templates/etc/asterisk/sip.conf.services* is associated to */etc/asterisk/sip.conf*

- The general one, named *ngcpcfg.services* that is associated to every file in its target directory. Example: */etc/ngcp-config/templates/etc/asterisk/ngcpcfg.services* is associated to every file under */etc/asterisk/*

When the *services* step is triggered all *.services* files associated to a changed configuration file will be executed. In case of the general file, any change to any of the configuration files in the directory will trigger the execution of the commands.

TIP If the service script has the execute flags set (*chmod +x $file*) it will be invoked directly. If it does not 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 do not overwrite any existing files if following this example.

Example:

```
echo 'ngcp-service restart mariadb' \
  > /etc/ngcpcfg-config/templates/etc/mysql/my.cnf.services
echo 'ngcp-service restart asterisk' \
  > /etc/ngcpcfg-config/templates/etc/asterisk/ngcpcfg.services
```

In this example we created two *.services* files. Now, each time we trigger a change to */etc/mysql.my.cnf* or to */etc/asterisk/** we will see that MySQL or Asterisk services will be restarted by the ngcpcfg system.

8.2. 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 are not supposed to be edited by the user. Do not manually edit this file unless you really know what you are 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 changing settings without having to manually edit this file.

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

Copyright © Sipwise GmbH - All rights reserved

8.3. ngcpcfg and Its Command Line Options

The ngcpcfg utility supports the following command line options:

8.3.1. apply

The *apply* option is a short-cut for the options "check && build && post-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.

8.3.2. 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/
```

8.3.3. cat

The *cat* option concatenates the YAML configuration files, including any local override files and prints them to standard output.

8.3.4. check

Check the syntax of the YAML files and validate them against schemas.

8.3.5. clean

Clean */etc/ngcp-config/* from uncommitted or unapplied changes, reset the git state and remove any possibly existing git stashed changes. It will also proposed to delete unnecessary git branches.

8.3.6. commit

The *commit* option records any changes done to the configuration tree inside */etc/ngcp-config*. The commit option should be executed when you have modified anything inside the configuration tree.

8.3.7. decrypt

Decrypt /etc/ngcp-config.tgz.pgp (or /etc/ngcp-config-crypted.tgz.gpg for backwards compatibility)

Copyright © Sipwise GmbH - All rights reserved

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.

8.3.8. del

Deletes a specified key from a specified YAML configuration file.

8.3.9. diff

Show uncommitted changes between ngcpcfg's Git repository and the working tree inside */etc/ngcp-config*. Iff the tool does not 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.

8.3.10. edit

Convenience option to edit the YAML files.

8.3.11. encrypt

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

8.3.12. get

Prints a value for a specified key from the YAML files.

8.3.13. help

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

8.3.14. initialise

The *initialise* option sets up the ngcpcfg framework. This option is automatically executed by the installer for you, so you should not have to use this option in normal operations mode.

8.3.15. log

Prints the log for local changes.

8.3.16. patch

Generate customtt files using default templates and patchtt files.

8.3.17. post-build

Performs needed actions after building the files, and before starting the services, such as synchronizing the database credentials, grants and timezones.

Copyright © Sipwise GmbH - All rights reserved

8.3.18. services

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

8.3.19. set

Set a specified key to a value in a specified YAML configuration file.

8.3.20. show

Display the details for a specified 'change-id', or display the latest changes if no 'change-id' is specified.

8.3.21. status

The *status* option provides a human readable interface to check the state of the configuration tree. If you are unsure what should be done as next step or if want to check the current state of the configuration tree, invoke *ngcpcfg status*.

If everything is OK and nothing needs to be done the output should look like:

```
# ngcpcfg status
Checking state of ngcpcfg:
OK:   has been initialised already (without shared storage)
Checking state of configuration files:
OK:   nothing to commit.
Checking state of /etc files
OK:   nothing to commit.
```

If the output does not say "OK", follow the instructions provided by the output of 'ngcpcfg status'.

NOTE Further details regarding the ngcpcfg tool are available through 'man ngcpcfg' on the Sipwise C5.

Copyright © Sipwise GmbH - All rights reserved

Chapter 9. Network Configuration

Starting with version 2.7, Sipwise C5 uses a dedicated *network.yml* file to configure the IP addresses of the system. The reason for this is to be able to access all IPs of all nodes for all services from any particular node in case of a distributed system on one hand, and in order to be able to generate */etc/network/interfaces* automatically for all nodes based on this central configuration file.

9.1. General Structure

The basic structure of the file looks like this:

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

Some more complete, sample configuration is shown in network.yml Overview section of the handbook.

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

In Sipwise C5 systems there is only one host entry in the file, and it is always named *self*.

9.1.1. 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:

Copyright © Sipwise GmbH - All rights reserved

mgmt: This entry means the host is acting as management node for the platform. In a Sipwise C5 system this option must always be set. The management node exposes the admin and CSC panels to the users and the APIs to external applications and is used to export CDRs.

lb: This entry means the host is acting as SIP load-balancer for the platform. In a Sipwise C5 system this option must always be set. The SIP load-balancer acts as an ingress and egress point for all SIP traffic to and from the platform.

proxy: This entry means the host is acting as SIP proxy for the platform. In a Sipwise C5 system this option must always be set. The SIP proxy acts as registrar, proxy and application server and media relay, and is responsible for providing the features for all subscribers provisioned on it.

db: This entry means the host is acting as the database node for the platform. In a Sipwise C5 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 Sipwise C5 system this option must always be set. The RTP relay node runs the *rtpengine* Sipwise C5 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 Sipwise C5 CE systems.
- *status*: one of 'online', 'offline', 'inactive'. 'inactive' means that the node is up but is not ready to work in the cluster (installing process). 'offline' means that the node is not reachable. 'online' is a normal working node.

9.1.2. Interface Parameters

- `hwaddr`: MAC address of the interface
- `ip`: IPv4 address of the node
- `v6ip`: IPv6 address of the node; optional
- `netmask`: IPv4 netmask
- `v6netmask`: IPv6 netmask
- `gateway`: IPv4 gateway address
- `v6gateway`: IPv6 gateway address
- `advertised_ip`: the IP address that is used in SIP messages when Sipwise C5 system is behind NAT/SBC. For example when running Sipwise C5 in a cloud environment, the server might not have a public IP. Then *load-balancer* component of Sipwise C5 needs to know what his public domain is (`advertised_ip`).
- `type`: type of services that the node provides; these are usually the VLANs defined for a particular Sipwise C5 system.

 Copyright © Sipwise GmbH - All rights reserved

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 fraud detection and others.

`aux_ext`: interface for potentially insecure external components like remote system log collection service.

`rtp_ext`: main (external) interface for media traffic

`sip_ext`: main (external) interface for SIP signalling traffic between NGCP and other SIP endpoints

`sip_ext_incoming`: additional, optional interface for incoming SIP signalling traffic

`sip_int`: internal SIP interface used by Sipwise C5 components (*lb, proxy, etc.*)

`ssh_ext`: command line (SSH) remote access interface

`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

`ha_int`: HA (High Availability) communication interface between the services

NOTE Please note that, apart from the standard ones described so far, there might be other *types* defined for a particular Sipwise C5 system.

- `vlan_raw_device`: tells which physical interface is used by the particular VLAN
- `post_up`: routes can be defined here (interface-based routing), for example:

```
post_up:
- route add -host 203.0.113.42 gw 192.168.1.1 dev vlan70
- route add -net 10.11.12.0/21 gw 192.168.1.2 dev vlan300
- route del -host 203.0.113.42 gw 192.168.1.1 dev vlan70
- route del -net 10.11.12.0/21 gw 192.168.1.2 dev vlan300
```

- `bond_XY`: specific to "bond0" interface only; these contain Ethernet bonding properties

9.2. 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.

9.2.1. Additional Entries in `/etc/hosts`

The file `/etc/hosts` is generated by a template, containing entries for basic host configuration (localhost and basic IPv4/IPv6), and the IPs of other nodes in PRO/CARRIER configurations.

To add extra entries in this file, it can be done in several ways:

- `etc_hosts_global_extra_entries` at the global level, added to all hosts

Copyright © Sipwise GmbH - All rights reserved

- `etc_hosts_global_extra_entries` at the host level, which overrides the global one if for some reason the whole content is undesired for a particular host (e.g. to have some but not all of the "default" global entries)
- `etc_hosts_local_extra_entries` at the host level, which are added only to the hosts where this entry is present, if for some reason it is desired to have extra entries only visible in some subset of the hosts

The behaviour is the same in all cases, to append the entries directly to `/etc/hosts`.

Example of both in a configuration file:

```
---
hosts_common:
  etc_hosts_global_extra_entries:
  - 10.100.1.1 server-1 server-1.internal.example.com
  - 10.100.1.2 server-2 server-2.internal.example.com
hosts:
  db01b:
    etc_hosts_local_extra_entries:
    - 127.0.1.1 local-alias-1.db01b
    - 127.0.2.1 local-alias-2.db01b
    - 172.30.52.180 db01b.example.com
    ...
  web01a:
    etc_hosts_local_extra_entries:
    - 127.0.1.1 local-alias-1.web01a
    - 127.0.2.1 local-alias-2.web01a
    - 172.30.52.168 web01a.example.com
    etc_hosts_global_extra_entries:
    - 10.100.1.1 server-1 server-1.internal.example.com
    ...
```

With this, the additional output in `/etc/hosts` for `db01b` will be:

```
# local extra entries for host 'db01b'
127.0.1.1 local-alias-1.db01b
127.0.2.1 local-alias-2.db01b
172.30.52.180 db01b.example.com

# global extra entries
10.100.1.1 server-1 server-1.internal.example.com
10.100.2.1 server-2 server-2.internal.example.com
```

and in `web01a`:

```
# local extra entries for host 'web01a'
127.0.1.1 local-alias-1.web01a
127.0.2.1 local-alias-2.web01a
```


Copyright © Sipwise GmbH - All rights reserved

```
172.30.52.168 web01a.example.com

# global extra entries overridden for host 'web01a'
10.100.1.1 server-1 server-1.internal.example.com
```

9.2.2. Extra SIP Sockets

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

The correct format consists of a label and value like this:

```
    extra_sockets:
      port_5064: udp:10.15.20.108:5064
      test: udp:10.15.20.108:6060
```

The label is shown in the outbound_socket peer preference (if you want to route calls to the specific peer out via specific socket); the value must contain a transport specification as in example above (udp, tcp or tls). After adding execute `ngcpcfg apply`:

```
ngcpcfg apply 'added extra socket'
```

The direction of communication through this SIP extra socket is incoming+outgoing. The Sipwise C5 will answer the incoming client registrations and other methods sent to the extra socket. For such incoming communication no configuration is needed. For the outgoing communication the new socket must be selected in the outbound_socket peer preference. For more details read the next section Extra SIP and RTP Sockets that covers peer configuration for SIP and RTP in greater detail.

IMPORTANT In this section you have just added an extra SIP socket. RTP traffic will still use your *rtp_ext* IP address.

9.2.3. Extra SIP and RTP Sockets

If you want to use an additional interface (with a different IP address) for SIP signalling and RTP traffic you need to add your new interface in the */etc/network/interfaces* file. Also the interface must be declared in */etc/ngcp-config/network.yml*.

Suppose we need to add a new SIP socket and a new RTP socket on VLAN 100. You can use the *ngcp-network* tool for adding interfaces without having to manually edit this file:

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

The generated file should look like the following:

Copyright © Sipwise GmbH - All rights reserved

```
..
..
    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 Sipwise C5 will answer the incoming client registrations and other methods sent to this address and remember the interfaces used for clients' registrations to be able to send incoming calls to him from the same interface.

In order to use the interface for the outbound SIP communication it is necessary to add it to extra_sockets section in */etc/ngcp-config/config.yml* and select in the outbound_socket peer preference. So if using the above example we want to use the vlan100 IP as source interface towards a peer, the corresponding section may look like the following:

```
    extra_sockets:
      port_5064: udp:10.15.20.108:5064
      test: udp:10.15.20.108:6060
      int_100: udp:192.168.1.3:5060
```

The changes have to be applied:

```
ngcpcfg apply 'added extra SIP and RTP socket'
```

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.

 Copyright © Sipwise GmbH - All rights reserved

9.2.4. Alternative RTP Interface Selection Using ICE

Normally, each interface that was configured with a type that starts with 'rtp_' can be selected individually as RTP interface in the Domain/Subscriber/Peer preferences. For example, if the interface types 'rtp_ext', 'rtp_int', and 'rtp_int_100' have been configured, the Domain/Subscriber/Peer preferences will allow the RTP interfaces to be selected as either 'ext', 'int', or 'int_100' in addition to "default".

The same 'rtp_' interface type can be configured on multiple interfaces. If this is the case, and if ICE ('Interactive Connectivity Establishment') is enabled for a Domain/Subscriber/Peer, it is possible to use ICE to automatically negotiate which interface should be used for RTP communications. ICE must be supported by the remote client for this to work.

For example, 'rtp_ext' can be configured on multiple interfaces like so (abbreviated):

```
..
..
    eth0.100:
      type:
        - rtp_ext
..
    eth0.150:
      type:
        - rtp_ext
..
    eth1:
      type:
        - rtp_ext
..
..
```

In this example, the RTP interface 'ext' will be available for selection in the Domain/Subscriber/Peer preferences. If selected and if ICE is enabled, the addresses of all three interfaces will be presented to the remote client, and ICE will be used to negotiate which one of them will be used for communications. This can be useful in multi-homed environments, or when remote clients are on private networks.

9.2.5. Extended RTP Port Range Using Multiple Interfaces

If the RTP port range configured via the config.yml keys rtpengine.minport and rtpengine.maxport is not sufficient to handle all concurrent calls, it is possible to load-balance the RTP ports across multiple interfaces. This is useful if the RTP proxy runs out of ports and if not enough additional ports are available.

To enable this, multiple interfaces with different addresses must be configured, and interface types of the format 'rtp_NAME:SUFFIX' must be assigned to them. For example, if the RTP interface named 'ext' should be load-balanced across three interfaces, they can be configured like so (abbreviated):

```
..
..
```

Copyright © Sipwise GmbH - All rights reserved

```
    eth0.100:
      type:
        - rtp_ext:1
..
    eth0.150:
      type:
        - rtp_ext:2
..
    eth1:
      type:
        - rtp_ext:3
..
..
```

In this example, all three given RTP interface types will be available for selection in the Domain/Subscriber/Peer preferences individually (as 'ext:1' and so on), but in addition to that, an interface named just 'ext' will also be available for selection. If 'ext' is selected, only one of the three RTP interfaces will be selected in a round-robin fashion, thus increasing the number of available RTP ports threefold. The round-robin algorithm only selects an interface if it actually has RTP ports available.

Copyright © Sipwise GmbH - All rights reserved

Chapter 10. Software Upgrade

Sipwise C5 can be upgraded to mr26.3.1 from previous LTS release, or any non-LTS release since the previous LTS-release.

The mr26.3.1 maintenance release uses the new upgrade approach commonly known as 'A/B Upgrade'

ngcp-upgrade requires a special partitioning schema of disk subsystem. The server has 3 separate partitions:

ngcp-data - it stores data files, like databases files, logs, etc.

ngcp-root and ngcp-fallback - they are equal size and contain the software, OS files and NGCP packages. One of them is the current root '/' partition, the other one is mounted as /ngcp-fallback directory.

See more details in The default disk partitions

During the upgrade the second partition is formatted and the target version is installed into it. After the reboot the system will be started from this partition and previous one becomes the /ngcp-fallback directory.

WARNING Please, pay particular attention that this partitioning schema is mandatory and if your system does not have it - create it beforehand.

WARNING This is the only supported upgrade schema. The old, in-place upgrade is not supported and technically not possible.

10.1. Release Notes

Please find the complete release notes and changelog of Sipwise C5 version mr26.3.1 on our WEB site.

10.2. Overview

The Sipwise C5 software upgrade procedure to mr26.3.1 will perform several fundamental tasks, and it is split into 2 stages:

First stage:

- do pre-upgrade checks
- format fallback partition
- install Debian to fallback partition
- install NGCP packages to fallback partition
- copy current configuration to fallback partition
- upgrade the NGCP configuration schema in fallback partition
- update grub configuration so after reboot the current fallback partition becomes the active one

Second stage:

Copyright © Sipwise GmbH - All rights reserved

- reboot to new partition. This step needs to be taken care manually during maintenance window
- upgrade the NGCP database schema

The 1st stage is safe to run outside the maintenance window - it changes nothing on currently running system, only prepares the fallback partition.

Also it can be done in parallel on all the nodes.

WARNING It still can affect the system in terms of CPU, network and disk load, to download, unpack and install the packages.

WARNING Grub configuration is changed so in case of any reboot (expected or not) system will be booted to mr26.3.1 partition. It will not affect the installation though as no NGCP services are run there.

The 2nd stage should be run during maintenance window with enabled maintenance mode in configuration.

WARNING Please make sure to have remote access to the system via out-of-band management (like IPMI, iLO, IMM, iDRAC, KVM, etc.)

10.3. Pre-Upgrade Checks

It is recommended to execute the preparatory steps in this chapter a few days before the actual software upgrade. They do not cause a service downtime, so it is safe to execute them during peak hours.

10.3.1. Log into the NGCP Server

Run the terminal multiplexer under the *sipwise* user (to reuse the Sipwise .screenrc settings that are convenient for working in multiple windows):

```
screen -S my_screen_name_for_ngcp_upgrade
```

Become root inside your screen session:

```
sudo -s
```

10.3.2. Check the Overall System Status

Check the overall system status:

```
ngcp-status
```

10.3.3. Evaluate and Update Custom Modifications

For the below steps, investigate and make sure you understand why the custom modifications were

Copyright © Sipwise GmbH - All rights reserved

introduced and if they are still required after the software upgrade. If the custom modifications are not required anymore, remove them (e.g. if a bug was fixed in the target release and the existing patch becomes irrelevant).

WARNING If you directly change the working configuration (e.g. add custom templates or change the existing ones) for some reason, then the system must be thoroughly tested after these changes have been applied. Continue with the software upgrade preparation only if the results of the tests are acceptable.

Find the local changes to the template files:

```
ngcp-customtt-diff-helper
```

The script will also ask you if you would like to download the templates for your target release. To download the new templates separately, execute:

```
ngcp-customtt-diff-helper -d
```

In the tmp folder provided by the script, you can review the patchtt files or merge the current customtt with the new tt2 templates, creating the new customtt.tt2 files. Once you do this, archive the new patchtt/customtt files to reapply your custom modifications after the software upgrade:

```
ngcp-customtt-diff-helper -t
```

Find all available script options with the "-h" parameter.

10.3.4. Check System Integrity

Changes made directly in tt2 templates will be lost after the software upgrade. Only custom changes made in customtt.tt2 or added by patchtt.tt2 files will be kept. Hence, check the system for locally modified tt2 files on **all** nodes:

```
ngcp-status --integrity
```

10.3.5. Check the Configuration Framework Status

Check the configuration framework status on **all** nodes. All checks must show the "OK" result and there must be no actions required:

```
ngcpcfg status
```

Run "apt-get update" and ensure that you do not have any warnings and errors in the output.

WARNING If the installation uses locally specified mirrors, then the mirrors must be switched to the Sipwise APT repositories (at least for the software upgrade). Otherwise, the

Copyright © Sipwise GmbH - All rights reserved

public Debian mirrors may not provide packages for old Releases anymore or at least provide outdated ones!

10.3.6. KeyDB to Valkey Migration

If not done already after the upgrade to mr12.5.x version, you must migrate the key-value database from KeyDB to Valkey before being able to upgrade to mr27.0.x or later. See the Appendix chapter for detailed information about the steps to execute.

10.4. Pre-Upgrade Steps

WARNING Ensure you are using the latest hot-fix of ngcp-upgrade-ce package from current repository before proceeding.

From node run:

```
apt update
apt install ngcp-upgrade-ce
```

10.4.1. Set the Proper Software Repositories

WARNING Ensure you are using the Sipwise APT repositories.

Public Debian mirrors may not provide packages for old Debian releases anymore. Also, they might be outdated. Consider using Sipwise repositories for the time of the upgrade.

10.4.2. Prepare the Upgrade

WARNING Starting from Sipwise C5 version mr13.2.1 the *man_allowed_ips* property will be depreciated. Any existing whitelisted IPs stored *man_allowed_ips* property can be migrated to the *allowed_ips* property with the help of the ngcp-migrate-allowed-ips script. This script transfers all the ips set in the *man_allowed_ips* preference onto *allowed_ips* for all subscribers, and domains. Additionally, the script resolves any associated conflicts should they arise.

Run the following command on all the nodes to install the package responsible for upgrading Sipwise C5 to a newer release:

```
ngcp-prepare-upgrade mr26.3.1
```

NOTE Do not worry, ngcp-upgrade-carrier does not exist, ngcp-upgrade-pro is used in Carrier too.

10.4.3. Run Upgrade Checks

There is a list of checks before the actual upgrade so it is wise to run them beforehand to detect and fix all the issues.

Copyright © Sipwise GmbH - All rights reserved

```
ngcp-upgrade-pre-checks mr26.3.1
```

10.4.4. `ngcp-upgrade` Options

The following options in `ngcp-upgrade` can be specially useful in some instances of upgrade:

- **`--step-by-step`**: confirm before proceeding to next step. With this option the upgrade operation is performed confirming every step before execution, with the possibility to instruct to continue without confirming further steps until the end (if confirmation is only needed for some steps at the beginning).
- **`--pause-before-step STEP_NAME`**: pause execution before step, given by the name of the script (e.g. "backup_mysql_db"). This option can be useful in several scenarios, for example:

  to help to debug problems or work around known problems during upgrades. In this case the operator can pause at a given step known to be problematic or right before a problematic set, perform some manual checks or changes, then continue the upgrade until another step (with confirmation like with the recent option `--step-by-step`), or continue without stop until the end

  another use might be to help to speed up upgrades when it involves several nodes: they can all proceed in parallel when it is known to be safe to do so; then perform some parts in lock-step (some nodes waiting until others finish with some stage); then continue in parallel until the end
- **`--skip-db-backup`**: This will speed-up the process in cases where it is deemed unnecessary, and this is very likely in the upgrade of nodes other than the first.

10.5. Upgrading Sipwise C5 CE

Make sure you are prepared to spend about one hour upgrading the system. Note that a short service downtime is possible during the system reboot to the upgraded partition.

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

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

10.5.1. The Custom Modification Handling (Optional)

During the execution of ngcp-upgrade in step **place_customtt_files**, you will be asked to place customtt/patchtt files for the new system to */ngcp-fallback/etc/ngcp-config*.

10.5.2. First Stage of Upgrade

You can run 1st stage outside the maintenance window.

To do this run the upgrade script as *root*:

```
ngcp-upgrade --target mr26.3.1
```

Copyright © Sipwise GmbH - All rights reserved

There is a "stop" step in the upgrade scenario, ngcp-upgrade stops there. At this time the new system in the fallback partition is ready.

10.5.3. Enable Maintenance Mode

The maintenance mode of the Sipwise C5 will disable some background services (for instance: *ngcp-mediator*) during the software upgrade. It thus prevents the system from getting into an inconsistent state while the upgrade is being performed. You can activate maintenance mode by applying a simple configuration change as described later.

- Enable maintenance mode:

```
ngcpcfg set /etc/ngcp-config/maintenance.yml "general.maintenance=yes"
```

- Apply configuration changes by executing:

```
ngcpcfg apply 'Enabling maintenance mode before the upgrade to mr26.3.1'
```

10.5.4. Second Stage of Upgrade

Before reboot switch boot record, run as *root*:

```
/usr/share/ngcp-upgrade/ngcp-switch-root-partition
```

Reboot the node and run as *root*:

```
ngcp-upgrade --target mr26.3.1
```

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.

10.6. Post-Upgrade Steps

10.6.1. Disabling Maintenance Mode

In order to disable the *maintenance mode*, do the following:

- Disable the maintenance mode:

```
ngcpcfg set /etc/ngcp-config/maintenance.yml "general.maintenance=no"
```

- Apply the changes to configuration templates:

Copyright © Sipwise GmbH - All rights reserved

```
ngcpcfg apply 'Disable the maintenance mode after the upgrade to
mr26.3.1'
```

10.6.2. Post-Upgrade Checks

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

NOTE You can find a backup of some important configuration files of your existing installation under */ngcp-data/backup/ngcp-mr26.3.1-** (where *** is a place holder for a timestamp) in case you need to roll back something at any time. A log file of the upgrade procedure is available at */ngcp-data/ngcp-upgrade/$FROM-mr26.3.1/logs/*.

10.7. Applying the Latest Hotfixes

If your current release is already the latest or you prefer to be on the LTS release, we still suggest applying the latest hotfixes and critical bug fixes.

Prior to preparing your patchtt, you can check if they need to be updated with this command.

```
ngcp-customtt-diff-helper -c
```

If your patchtt needs to be updated, execute all steps as described in Pre-upgrade checks. They include the system checks, customtt/patchtt preparation and others. It is important to execute all the steps from the above chapter.

10.7.1. Enable Maintenance Mode

The maintenance mode of the Sipwise C5 will disable some background services (for instance: *ngcp-mediator*) during the update. It thus prevents the system from getting into an inconsistent state while the update is being performed. You can activate maintenance mode by applying a simple configuration change on the **`standby management`** node as described later.

- Pull pending configuration (if any):

```
ngcpcfg pull
```

- Enable maintenance mode:

```
ngcpcfg set /etc/ngcp-config/maintenance.yml "general.maintenance=yes"
```

- Apply configuration changes by executing:

```
ngcpcfg apply 'Enabling maintenance mode before updating to the latest
```

Copyright © Sipwise GmbH - All rights reserved

```
mr26.3.1 packages'
```

10.7.2. Update ngcp-system-tools and ngcp-support Packages

Update the ngcp-system-tools and ngcp-support packages to the latest version to obtain the most recent ngcp-update script.

```
apt install -y ngcp-system-tools-ce ngcp-support-bin
```

10.7.3. Apply Hotfixes

```
ngcp-update
```

10.7.4. If There Are Custom Configuration Templates

When running ngcp-update, the script will check the current patchtt files. If your current patchtt can be applied correctly during the ngcp-update, the script will continue. Otherwise, the script will stop and will let you know which patchtt cannot be applied so that you can fix it.

After fixing the patchtt, continue the update by running:

```
ngcp-update -a
```

10.7.5. Disable Maintenance Mode

In order to disable the *maintenance mode*, do the following:

- Disable the maintenance mode:

```
ngcpcfg set /etc/ngcp-config/maintenance.yml "general.maintenance=no"
```

- Apply the changes to configuration templates:

```
ngcpcfg apply 'Disabling maintenance mode after updating to the latest
mr26.3.1 packages'
```

Execute the final checks as described in the **Post-upgrade checks** section.

Copyright © Sipwise GmbH - All rights reserved

Chapter 11. Backup, Recovery and Maintenance

11.1. Sipwise C5 Backup

For any service provider it is important to maintain a reliable backup policy as it enables prompt services restoration after any force majeure event. Hence, we strongly suggest you to configure a backup procedure. The Sipwise C5 can be integrated with any Debian compatible backup software.

11.1.1. 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 folder */etc/ngcp-config/* must be included in the backup as well. It contains the system specific configuration (like SSL keys). Also you might have some local modifications. We suggest backing up the whole */etc* folder to preserve the *etckeeper* history to be able to answer when and who changed particular configuration files in the past.

- Exported CDRs (optional)

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

11.2. Recovery

In the worst case scenario, when the system needs to be recovered from a total loss, you only need 4 steps to get the services back online:

- Install Sipwise C5 as explained in chapter 2.
- Restore the */etc/ngcp-config/* directory from the backup, overwriting your local files.
- Restore 'mysql.encryption.key' in constants.yml if new MariaDB instance/binlogs were encrypted (DB is encrypted by default). See the detailed information in MariaDB data restoration remarks.
- Restore the database from the latest MySQL dump.
- Apply the changes to bring the original configuration into effect:

```
ngcpcfg apply 'restored the system from the backup'
```

11.3. 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.

Copyright © Sipwise GmbH - All rights reserved

To reset the database to its original state you can use a script provided by CE: * Execute *ngcp-reset-db*. It will assign new unique passwords for Sipwise C5 services and reset all services. The script will also create dumps for all Sipwise C5 databases.

11.4. Accounting Data (CDR) Cleanup

Sipwise Sipwise C5 offers ways to cleanup, backup or archive old accounting data—i.e. CDRs—that is not necessary for further processing any more, or must be deleted according to the law. There are some Sipwise C5 components designed for this purpose and they are commonly called *cleanuptools*. These are configurable scripts that interact with NGCP's `accounting` and `kamailio` databases, or remove exported CDR files in order to clean or archive the unnecessary data.

11.4.1. Cleanuptools Configuration

The configuration parameters of *cleanuptools* are located in the main Sipwise C5 configuration file: `/etc/ngcp-config/config.yml`. Please refer to the `config.yml` file description: Cleanuptools Configuration Data for configuration parameter details.

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

```
> ngcpcfg apply 'Modified cleanuptools 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 Sipwise C5 system administrator can also select the time when cleanup scripts are run, by modifying the schedule here: `/etc/cron.d/cleanup-tools`

11.4.2. Accounting Database Cleanup

The script responsible for cleaning up the database is: `ngcp-cleanup-acc`

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

An extract from a sample configuration file is provided here:

```
############

batch = 10000
archive-target = /ngcp-data/backup/cdr
compress = gzip

username = dbcleaner
password = rcKamRdHhx7saYRbkJfP
host = localhost
port = 3306

redis-batch = 10000
redis-port = 6379
```


Copyright © Sipwise GmbH - All rights reserved

```
connect accounting
keep-months = 2
use-partitioning = yes
timestamp-column = cdr_start_time
backup cdr_cash_balance_data
backup cdr_time_balance_data
backup cdr_relation_data
backup cdr_tag_data
backup cdr_mos_data
backup cdr_export_status_data
backup cdr_group
timestamp-column = first_cdr_start_time
backup cdr_period_costs
timestamp-column = start_time
backup cdr

archive-months = 2
archive cdr_cash_balance_data
archive cdr_time_balance_data
archive cdr_relation_data
archive cdr_tag_data
archive cdr_mos_data
archive cdr_export_status
archive cdr_group
archive cdr_period_costs
archive cdr

cleanup-days = 1
use-partitioning = no
timestamp-column = cdr_start_time
cleanup int_cdr_cash_balance_data
cleanup int_cdr_time_balance_data
cleanup int_cdr_relation_data
cleanup int_cdr_tag_data
cleanup int_cdr_group
cleanup int_cdr_export_status
timestamp-column = start_time
cleanup int_cdr

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

connect-redis 21
connect kamailio
time-column = time_hires
cleanup-days = 3
cleanup-mode = mysql
cleanup-redis acc:entry::*
```

Copyright © Sipwise GmbH - All rights reserved

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

maintenance = no
```

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

There are 4 types of actions the database cleanup script can take:

- backup database tables
- archive database tables
- cleanup database tables
- cleanup redis databases

These actions are discussed in following sections.

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

Backup Database Tables

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

Configuration values that govern the backup procedure are:

- `time-column`: The name of the column in the table to use for determining which month a record belongs to. Must be a "datetime" column.
- `timestamp-column`: The name of the column in the table to use for determining which month a record belongs to. Must be a "decimal(13,3)" column.
- `use-partitioning`: If a table is partitioned using the time-column (timestamp-column) and the value is set to "yes", then moving/deleting records are instant operations, done by managing the partitions. Otherwise the usual method is used to delete/move records by chunks.
- `batch`: How many rows to include per transaction when processing in chunks. If unset or 0, it does them all at once.
- `keep-months`: How many months worth of records to keep in the table and not move into the monthly backup tables.

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

Copyright © Sipwise GmbH - All rights reserved

Archive Database Tables

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

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

- `archive-months`: Uses the same logic as the "keep-months" variable. If set to "12" and the script was started on December 15th, it will start archiving with the December table of the previous year. Archiving continues month by month, going backwards in time, until the script encounters a missing table.
- `archive-target`: Target directory for writing the SQL dump files. If explicitly specified as "/dev/null", then no actual archiving will be performed, but instead the tables will only be dropped.
- `compress`: If set to "gzip", then gzip the dump files after creation. If unset, do not compress.
- `host`, "username" and "password": As dumping is performed by an external command, those variables are reused from the "connect" statement.

Cleanup Database Tables

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

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

The `cleanup` statement works exactly like the `backup` statement, but does not actually backup anything, but rather only deletes old records. Additional configuration parameters required by the `cleanup` procedure:

- `cleanup-days`: Any record older than these many days will be deleted.

Cleanup Redis Databses

With the advent of persisting kamailio.acc record data in a redis keystore, a separate `cleanup-redis <key pattern>` statement was introduced. It will remove old redis entries, whose keys match the given redis SCAN pattern. Typically this has to be done for the redis 21 database (acc records), e.g.:

- `cleanup-redis acc:entry::*`

`connect-redis <redis database number>` has to be used instead of `connect +<mariadb database name>`, which is needed to initially connect before any `<backup>`, `<archive>` and `<cleanup>` operations.

The `cleanup-redis` statement works similar to the `cleanup` statement, with some additional options below:

- `time-column`: The name of the field in a redis entry denoting the record timestamp in epoch seconds.

Copyright © Sipwise GmbH - All rights reserved

- `cleanup-mode`: If set to "delete", aged entires will be simply removed. If set to "mysql", they will be inserted into a database table first. The latter requires `connect` to open the database additionally.
- `redis-batch`: Chunk size of redis entries with matching keys to look at. Note that redis processing always works in chunks, there is no partitioning.
- `cleanup-days`: Any entry older than these many days will be deleted.

11.4.3. Exported CDR Cleanup

The script responsible for cleaning up exported CDR files is: `ngcp-cleanup-cdr-files`

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

A sample configuration file is provided here:

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

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

Configuration values that define the files to be deleted:

- `enable`: Enable (`yes`) or disable (`no`) exported CDR cleanup.
- `max_age_days`: Gives the expiration time of the exported CDR files in days. There is a general value which may be overridden by a local value provided at a specific path. The local value is valid for the particular path only.
- `paths`: an array of path definitions

  path: a path where CDR files are to be found and deleted; this may contain wildcard characters

  `wildcard`: Enable (`yes`) or disable (`no`) using wildcards in the `path`

 Copyright © Sipwise GmbH - All rights reserved

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`

Copyright © Sipwise GmbH - All rights reserved

Chapter 12. Security, Performance and Troubleshooting

Once Sipwise C5 is in production, security and maintenance become really important. In this chapter, we will go through a set of best practices for any production system.

12.1. Sipwise SSH Access to Sipwise C5

The Sipwise C5 provides SSH access to the system for the Sipwise operational team for debugging and final tuning. Operational team uses user 'sipwise' which can be logged in through SSH key only (password access is disabled) from dedicated access server 'jump.sipwise.com' only.

To completely remove Sipwise access to your system, please execute as user root:

```
root@myserver:~# ngcp-support-access --disable && apt-get install ngcp-
support-noaccess
```

NOTE | you have to execute the command above on each node of your Sipwise C5 system!

WARNING | please ensure that the script completes 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 completes successfully:

```
  * Support access successfully enabled.
```

12.1.1. Secure System and User SSH Keys

The Sipwise C5 defaults to generating and using internally secure SSH system and user keys.

That does not cover user installed keys, or systems that have been upgraded from previous defaults.

Because system and user SSH keys cannot easily be deprecated without remote user involvement, this is a process that is not performed automatically during upgrade. Instead there is a 'ngcp-check-ssh-keys' command that will perform various checks on the system, and report either errors or warnings for the issues it detects. Its manual page documents these issues and the general procedures to follow to resolve them.

Copyright © Sipwise GmbH - All rights reserved

12.2. Firewalling

12.2.1. Firewall Framework

The Sipwise C5 runs a wide range of services. In order to secure the platform while allowing access to Sipwise C5 , Sipwise C5 configuration framework provides a set of predefined network zones. Services are aggregated into appropriate zones by default. Zones are assigned to network interfaces (and VLANs if applicable) in /etc/ngcp-config/network.yml.

CAUTION Though the default firewall setup provided by Sipwise C5 configuration framework provides a safe setup for Sipwise C5, security audits of the platform performed by qualified engineers before commissioning the platform into service are strongly recommended. Customization of the setup requires in-depth knowledge of firewalling principles in general and the 'netfilter' facility in particular.

Table 31. Sipwise C5 network zones

Zone name	Description
ha_int	Internal HA (High Availability) communication interface between the services
rtp_ext	Interface for external RTP media relay between Sipwise C5 and endpoints (e.g. user agents, peers)
sip_ext	Interface for external SIP signalling between Sipwise C5 and endpoints (e.g. user agents, peers)
sip_int	Interface for internal signalling, e.g. between load-balancers, proxies and application servers
ssh_ext	Interface providing external access to Sipwise C5 command line interface
web_ext	Interface providing access to the customers' self-care Web panel
web_int	Interface for access to the administrative Web panel, its REST APIs and internal API communications

NOTE Additional custom zones may be configured, but will not be automatically integrated into the firewall configuration.

To facilitate firewall functionality, Sipwise C5 uses the Kernel's 'netfilter' facility and 'iptables-persistent' as an interface to 'netfilter'. 'Netfilter' is using 'tables' and within that 'chains' to store rules in this hierarchy: 'table' 'chain' 'rule'. Default firewall setups of Sipwise C5 do not use netfilter tables 'nat' and 'raw', but only default table 'filter'.

NOTE Custom 'nat' rules for IPv4 and IPv6 may be added in file /etc/ngcp-config/config.yml in sections 'securityfirewallnat_rules4' and 'securityfirewallnat_rules6'.

Each 'chain' deploys a 'default policy' handling packets which did not trigger any rule in a particular 'chain'.

Table 32. Sipwise C5 'netfilter' default policies

Copyright © Sipwise GmbH - All rights reserved

Chain	Default policy	Description
INPUT	DROP	Handling all packets directly destined for a Sipwise C5 node (only packets matching a rule are allowed)
FORWARD	DROP	Handling all packets received by a Sipwise C5 node and destined for another, non-local IP destination (no default rules added)
OUTPUT	ACCEPT	Handling all packets originating on a Sipwise C5 node (no default rules added)
rtpengine	N/A	Container for rtpengine rule to allow the rule to persist even when the Kernel module is unloaded (e.g. during upgrades)

The default firewall setup provided by Sipwise C5:

- adds rules to INPUT to allow access to the platform only via specific services
- blocks all traffic from and to FORWARD
- allows all OUTPUT traffic
- allows adding additional rules to further restrict access

12.2.2. Sipwise C5 Firewall Configuration

The Sipwise C5 comes with a preconfigured set of firewall rules, which can be enabled and configured in `/etc/ngcp-config/config.yml` in section `security→firewall`. Refer to security for available configuration options.

Firewall configuration is applied by running `ngcpcfg apply`. However, this will not activate new rules automatically to avoid inadvertent self-lockout. To finally activate new firewall rules run `ngcp-iptables-apply`. This will prompt for another system logon to verify access remains available. If the prompt is not confirmed, firewall rules will automatically be reverted to the previous state re-enabling access to the command line.

IMPORTANT `ngcp-iptables-apply` needs to be enforced on each active and standby Sipwise C5 node providing the 'mgmt', 'lb', or 'rtp' roles (Please, refer to Available Host Options for further details). Additionally, any changes made into firewall rules will require executing this command again on the corresponding nodes.

CAUTION The Sipwise C5 firewall subsystem by default is disabled in '/etc/ngcp-config/config.yml' key `security.firewall.enable: no`. This is to avoid blocking any traffic inadvertently during installation. After the firewall subsystem has been configured appropriately, it needs to be enabled by setting `security.firewall.enable: yes` in '/etc/ngcp-config/config.yml'.

12.2.3. System Rules

The following set of rules is added by the system upon activation of the firewall subsystem. Individual system rules are configured in '/etc/ngcp-config/templates/etc/iptables/rules.v4.tt2' and '/etc/ngcp-config/templates/etc/iptables/rules.v6.tt2'

Table 33. Firewall system rules

Copyright © Sipwise GmbH - All rights reserved

Zone	Chain	Target	Rule	Description
n/a	INPUT	ACCEPT	`-i lo -j ACCEPT`	Accept all packets received by local loopback interface
n/a	OUTPUT	ACCEPT	`-o lo -j ACCEPT`	Accept all packets sent over local loopback interface
n/a	INPUT (IPv4)	DROP	`! -i lo -s 127.0.0.0/8 -j DROP`	Ignore all packets received on an external interface claiming to be from the loopback interface
n/a	INPUT (IPv6)	DROP	`! -i lo -s ::1/128 -j DROP`	Ignore all packets received on an external interface claiming to be from the loopback interface
all	INPUT	ACCEPT	`-m state --state RELATED,ESTABLISHED -j ACCEPT`	Accept all incoming packets tied to 'related' or 'established' connections
all	INPUT (IPv4)	ACCEPT	`-p icmp -m icmp --icmp -type 8 -j ACCEPT`	Accept all ICMP 'echo' messages
all	INPUT (IPv4)	ACCEPT	`-p icmp -m icmp --icmp -type 0 -j ACCEPT`	Accept all ICMP 'echo reply' messages
all	INPUT (IPv6)	ACCEPT	`-A INPUT -p ipv6-icmp -j ACCEPT`	Accept all ICMPv6 messages
all	ngcp_cluster	ACCEPT	`-s '<node_ip>' -j ACCEPT`	Set of rules white-listing all IP-addresses owned by Sipwise C5 platform for incoming traffic
ssh_ext	ngcp_ssh_ext	ACCEPT	`-A ngcp_ssh_ext -s '<sshd.permit_support_from>' -j ACCEPT`	List of rules to accept incoming packets for SSH on all 'ssh_ext' interfaces from hosts configured in '/etc/ngcp-config/config.yml' with key 'sshd.permit_support_from'
mon_ext	ngcp_mon_ext_snmp	ngcp_mon_ext	`-A ngcp_mon_ext -s '<snmpclient_ip>' -j ACCEPT`	Set of rules for all 'mon_ext' interfaces based on a list of IPs for all SNMP communities configured in 'snmpd.communities'
mon_ext	ngcp_mon_ext_prom	ngcp_mon_ext	`-A ngcp_mon_ext -s '<promclient_ip>' -j ACCEPT`	Set of rules for all 'mon_ext' interfaces based on a list of IPs for all Prometheus clients configured in 'prometheus.external.clients'
'ZONE'	ngcp_'ZONE'	ACCEPT	`-A ngcp_'ZONE' -j ACCEPT`	Default permissive allow rule for each remaining zone

Copyright © Sipwise GmbH - All rights reserved

Zone	Chain	Target	Rule	Description
all	INPUT	ngcp_cluster	`-j ngcp_cluster`	Divert all incoming packets to the 'cluster' chain
mon_ext	INPUT	ngcp_mon_ext_snmp	`-i 'mon_ext interface' -p udp --dport 161 -j ngcp_mon_ext_snmp`	Divert SNMP packets to the 'mon_ext_snmp' chain
mon_ext	INPUT	ngcp_mon_ext_prom	`-i 'mon_ext interface' -p tcp --dport 9090 -j ngcp_mon_ext_snmp`	Divert Prometheus packets to the 'mon_ext_prom' chain
ssh_ext	INPUT	ngcp_ssh_ext	`-i 'ssh_ext interface' -p tcp --dport 'sshd.port' -j ngcp_ssh_ext`	Divert connection requests to the SSH service to the 'ssh_ext' chain
web_ext	INPUT	ngcp_web_ext	`-i 'web_ext interface' -p tcp --dport '<www_admin.http_csc.port>' -j ngcp_web_ext`	Divert connection requests to the customer self-care HTTPS service to the 'web_ext' chain
web_ext	INPUT	ngcp_web_ext	`-i 'web_ext interface' -p tcp --dport '<autoprov.server.port>' -j ngcp_web_ext`	Divert connection requests to the device auto-provisioning service to the 'web_ext' chain
web_ext	INPUT	ngcp_web_ext	`-i 'web_ext interface' -p tcp --dport '<autoprov.server.bootstrap_port>' -j ngcp_web_ext`	Divert connection requests to the device bootstrapping service to the 'web_ext' chain
web_int	INPUT	ngcp_web_int	`-i 'web_int interface' -p tcp --dport '<www_admin.http_admin.port>' -j ngcp_web_int`	Divert connection requests to the platform administration HTTPS service to the 'web_int' chain
sip_ext	INPUT	ngcp_sip_ext	`-i 'sip_ext interface' -p udp --dport '<kamailio.lb.port>' -j ngcp_sip_ext`	Divert UDP packets sent to the SIP load balancer to the 'sip_ext' chain
sip_ext	INPUT	ngcp_sip_ext	`-i 'sip_ext interface' -p tcp --dport '<kamailio.lb.port>' -j ngcp_sip_ext`	Divert connection requests to the TCP SIP load balancer to the 'sip_ext' chain
sip_ext	INPUT	ngcp_sip_ext	`-i 'sip_ext interface' -p tcp --dport '<kamailio.lb.tls.port>' -j ngcp_sip_ext`	Divert connection requests to the TCP/TLS SIP load balancer to the 'sip_ext' chain
sip_ext	INPUT	ngcp_sip_ext	`-i 'sip_ext interface' -p tcp --dport '<pushd.port>' -j ngcp_sip_ext`	Divert connection requests to mobile push service to the 'sip_ext' chain
api_int	INPUT	ngcp_api_int	`-i 'api_int interface' -p tcp --dport '<ossbss.port>' -j ngcp_api_int`	Divert connection requests to internal API service to the 'api_int' chain

Copyright © Sipwise GmbH - All rights reserved

Zone	Chain	Target	Rule	Description
rtp_ext	INPUT	ngcp_rtp_ext	`-i 'rtp_ext interface' -p udp --dport '<rtpengine.minport>':'<rtpengine.maxport>' -j ngcp_rtp_ext`	Divert external RTP and RTCP to the 'rtp_ext' chain
rtp_int	INPUT	ngcp_rtp_int	`-i 'rtp_int interface' -p udp --dport '<rtpengine.minport>':'<rtpengine.maxport>' -j ngcp_rtp_int`	Divert internal RTP and RTCP to the 'rtp_int' chain

CAUTION

To function correctly, the *rtpengine* requires an additional *iptables* rule installed. This rule (with a target of `rtpengine` along with a chain called `rtpengine`) is automatically installed and removed when the rtpengine starts and stops, so normally you do not 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. As the required rule may not clearly be visible when listing the currently installed rules (either via *iptables* or via *nft*), the command (run as root or under *sudo*) *rtpengine --nftables-status* can be used to confirm that the required rules are correctly installed. These rules can be manipulated (installed or removed) as needed via the commands *rtpengine --nftables-start* and *rtpengine --nftables-stop* respectively. The `rtpengine` service itself does not need to be restarted in order to do this.

NOTE

Some of the parameters used to populate the firewall rules automatically may contain hostnames instead of IP addresses. Since firewall rules need to be configured based on IP addresses by design, Sipwise C5 configuration framework will lookup such hostnames during 'ngcpcfg apply' and expand them to the IP addresses as returned by 'gethostbyname'. If DNS resolving changes for such hostnames due to changes to DNS the rules will not update automatically. Another run of 'ngcpcfg apply' will be needed to re-perform the lookup and update the rules to reflect changes in DNS. If this step is omitted, clients may be locked out of the system.

NOTE

By default, the rules for the 'rtp_ext' zone are created with a target of `ACCEPT`. It is optionally possible to create these rules with another *iptables* chain as target, and instruct the RTP proxy to dynamically manage individual rules for each running call in this chain. If this is enabled, the chain with the name given in the `/etc/ngcp-config/config.yml` key `rtpengine→firewall_iptables_chain` will be created as empty, leaving the effective target for UDP packets within the RTP port range as the table's default policy (normally `DROP`). The RTP proxy will then dynamically create one `ACCEPT` rule for each open RTP media port in the given chain when a call starts, and delete it when the call is finished. It should be noted that dynamically creating and deleting iptables rules can incur a significant performance overhead, especially in scenarios with high call volumes, and it is therefore not recommended to enable this feature in such cases.

Copyright © Sipwise GmbH - All rights reserved

12.2.4. Custom Rules

The Sipwise C5 configuration framework makes it possible to add custom rules to the firewall setup in '/etc/ngcp-config/config.yml'. Custom rules can be added in a variety of ways, depending on the intended use and purpose.

Each zone described above corresponds to a set of sections in the YAML file (one for IPv4, one for IPv6), with each section containing a list of rules applicable to that zone. By default, each zone contains a single, permissive, catch-all `ACCEPT` rule to ensure that enabled services work out of the box, with the exception of the `ssh_ext` and `mon_ext` zones, which are restricted by default.

For example, the list of rules applicable to IPv4 RTP sent to any `rtp_ext` interface can be found as:

```
security:
  firewall:
    services:
      rtp_ext:
        rules4:
      - -j ACCEPT
```

This list can be extended to either disallow specific sets of traffic and allow everything else for that service (a list of specific `DROP` rules followed by a catch-all `ACCEPT`), or allow only a specific set of traffic and disallow everything else (a list of specific `ALLOW` rules followed by a catch-all `DROP`).

Rules are in free-form `iptables` syntax, but must not include the insert operation (*-A* or *-I*) nor the name of the respective chain.

For example, to allow external SIP only from a list of certain IP networks:

```
security:
  firewall:
    services:
      sip_ext:
        rules4:
      - -s 198.51.100.0/24 -j ACCEPT
      - -s 233.252.0.0/24 -j ACCEPT
      - -j DROP
        rules6:
      - -s 2001:db8:123:abc::/64 -j ACCEPT
      - -j DROP
```

NOTE Depending on the "default policy" set for the `INPUT` chain (see above), the final catch-all `ALLOW` or `DROP` rule may be omitted.

Input filter rules that do not correspond to one of the pre-defined zones can be added in one of the `security.firewall.input_rules…` lists. `input_rules4…` applies to IPv4 and `input_rules6…` applies to IPv6. These lists come in two varieties: The rules from the _pre list are inserted **before** the system-defined rules corresponding to network zones, while the rules from the _post list are inserted **after** the system-defined rules. As such, in a normal firewall setup the _pre list is mostly useful to restrict things that otherwise would be allowed, while the _post list is mostly useful to allow things that

 Copyright © Sipwise GmbH - All rights reserved

otherwise would be restricted.

Additional rules that apply to one of the other iptables chains or tables (the nat table, or a filter chain that is not INPUT) can be added in the nat_rules and filter_rules lists respectively. Unlike the other lists or rules, these lists use the full iptables syntax, i.e. they must include the insert operation (normally -A for "append") and the chain name (such as FORWARD or the name of a custom chain).

Custom chains can be created as needed by adding them to the custom_chains list. These chains can be populated through rules in filter_rules or nat_rules, and can be referred to as a jump target from any other rule (*-j* + the name of the custom chain). This can be useful to create sets of rules that can be applied to multiple zones (e.g. a shared set of allowed addresses).

Example custom rule to allow-list all IPv4 traffic from network interface eth1.301, effectively making VLAN 301 a trusted network:

```
security:
  firewall:
    input_rules4_post:
      - -i eth1.301 -j ACCEPT
```

Example custom rule to accept incoming traffic from monitoring station 203.0.113.93 for an optionally installed check_mk agent:

```
security:
  firewall:
    input_rules4_post:
      - -p tcp -s 203.0.113.93 --dport 6556 -j ACCEPT
```

To add hosts or networks to the SSH allow-list, they can be either added to key 'sshd.permit_support_from' in '/etc/ngcp-config/config.yml' or they can be added to list for the appropriate zone:

```
security:
  firewall:
    services:
      ssh_ext:
        rules4:
          - -s 198.51.100.0/24 -j ACCEPT
          - -s 203.0.113.93 -j ACCEPT
```

NOTE In custom rules keys from '/etc/ngcp-config/config.yml' cannot be referenced. Thus, the values need to be manually looked up, hard coded, and kept in sync manually. This is by design of YAML.

12.2.5. Example Firewall Configuration Section

An example for Sipwise C5 firewall configuration in '/etc/ngcp-config/config.yml', enabling both the firewall subsystem and the logging facility, and adding a custom chain to allow administration (SSH and

Copyright © Sipwise GmbH - All rights reserved

HTTPS admin panel) only from a single address, may look like:

```
security:
  firewall:
    custom_chains:
      - admins
    enable: yes
    filter_rules4:
      - -A admins -s 198.51.100.42 -j ACCEPT
      - -A admins -j DROP
    filter_rules6:
      - -A admins -j DROP
    input_rules4_post: []
    input_rules4_pre: []
    input_rules6_post: []
    input_rules6_pre: []
    logging:
      days_kept: '7'
      enable: yes
      file: /var/log/firewall.log
      tag: NGCPFW
    nat_rules4: []
    nat_rules6: []
    policies:
      forward: DROP
      input: DROP
      output: ACCEPT
    services:
      api_int:
        rules4:
        - -j ACCEPT
        rules6:
        - -j ACCEPT
      mon_ext:
        rules4: []
        rules6: []
      rtp_ext:
        rules4:
        - -j ACCEPT
        rules6:
        - -j ACCEPT
      rtp_int:
        rules4:
        - -j ACCEPT
        rules6:
        - -j ACCEPT
      sip_ext:
        rules4:
        - -j ACCEPT
        rules6:
        - -j ACCEPT
```

 Copyright © Sipwise GmbH - All rights reserved

```
      sip_int:
        rules4:
        - -j ACCEPT
        rules6:
        - -j ACCEPT
      ssh_ext:
        rules4:
        - -j admins
        rules6:
        - -j admins
      web_ext:
        rules4:
        - -j ACCEPT
        rules6:
        - -j ACCEPT
      web_int:
        rules4:
        - -j admins
        rules6:
        - -j admins
```

12.3. Password Management

The Sipwise C5 comes with some default passwords the user should change during the deployment of the system. They have been explained in the previous chapters of this handbook.

IMPORTANT Many Sipwise C5 services use MySQL backend. Users and passwords for these services are created during the installation. These passwords are unique for each installation, and the connections are restricted to localhost. You should not change these users and passwords.

12.3.1. The "root" Account

The Sipwise C5's super-user account comes with a preconfigured password. It is imperative that this password is changed by the operator immediately after Sipwise C5 is shipped and before it is connected to any potentially unsecure public or private network using a secure password in compliance with existing password policies of the operator. The "root" password must not be shared outside of the operator's organization including Sipwise engineers. The "root" password must not be shared in any publicly accessible communications including email or ticketing systems.

To change the root password log into the freshly deployed system as "root" using the preconfigured password and execute:

```
root@myserver:~# passwd
```

Then follow the prompts to change the password.

The Vagrant/VirtualBox Sipwise C5 images come with more default credentials which should be changed immediately:

Copyright © Sipwise GmbH - All rights reserved

- 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 images (skip the step if you use Vagrant).

12.3.2. The "administrator" Account

The Sipwise C5 Web-interface comes with a preconfigured "administrator" account deployed with a default password. This account can be considered Sipwise C5 application super-user and has far-reaching access to application specific settings via the Web-interface. It is imperative that the password for this account is changed by the operator immediately after Sipwise C5 is shipped and before it is connected to any potentially unsecure public or private network using a secure password in compliance with existing password policies of the operator. The "administrator" password must not be shared outside of the operator's organization including Sipwise engineers. The "administrator" password must not be shared in any publicly accessible communications including email or ticketing systems.

The password for the "administrator" account can be changed via the Web-interface.

12.3.3. The "cdrexport" Account

The login for the system account *cdrexport* is disabled by default. Although this is a jailed account, it has access to sensitive information, namely the Call Detail Records of all calls. SSH keys should be used to log in this user, or alternatively a really strong password should be used when setting the password via *passwd cdrexport*.

12.3.4. The MySQL "root" User

The *root* user in MySQL has no default password. A password should be set using the *mysqladmin password* command.

12.3.5. The "ngcpsoap" Account

Generate new password for user *ngcpsoap* to access the provisioning interfaces, see the details in REST API.

12.4. UI/API Security

The UI/API security configuration is located in config.yml as:

```
www_admin:
  security:
    jwt:
      expires: 36000
    login:
      ban_enable: yes
      ban_increment: 300
      ban_max_time: 3600
      ban_min_time: 300
      max_attempts: 5
```

 Copyright © Sipwise GmbH - All rights reserved

```
    password:
      allow_recovery: '0'
      max_length: 40
      min_length: 12
      musthave_digit: 3
      musthave_lowercase: 3
      musthave_specialchar: 3
      musthave_uppercase: 3
      sip_autogenerate: '0'
      sip_expose_subadmin: '1'
      sip_keep_last_used: 12
      sip_validate: no
      web_autogenerate: '0'
      web_expose_subadmin: '1'
      web_keep_last_used: 12
      web_max_age_days: 0
      web_validate: no
```

12.4.1. Authentication Security

There is a progressive ban system in place (enabled by default) that upon unsuccessful 'max_attempts' logins bans the user on UI/API (that includes JWT) for 'ban_min_time + ban_progression_stage * ban_increment'. * 'ban_progression_state' is 0 by default and increments by 1 every time the user is banned, and resets back to 0 upon successful login. 'ban_max_time' is the absolute max ban time that will not be exceeded by the progression. * when a user is banned the UI does not return any specific indication that the user is banned and API returns 401 Unauthorized as per invalid credentials. * bans are per User+Domain+IP address.

12.4.2. Password Validation

With password validation enabled via 'sip_validate = yes' (SIP passwords) or 'web_validate = yes' (UI/API passwords for admin and subscriber users), every time a password insert/update attempt is validated for the following states:

- password strength, the score is calculated based on multiple factors and passwords with low score (aka: weak passwords) are not allowed.
- by default the password must be 12-40 length and contain at least 3 of lower-case characters, upper-case characters, digits and special chars combined (corresponding to the config options: 'min_length', 'max_length', 'musthave_lowercase', 'musthave_uppercase', 'musthave_digit', musthave_specialchar').
- password cannot match the username.
- it cannot be one of last used passwords, corresponding to 'sip_keep_last_used' (for SIP passwords) and 'web_keep_last_used' (for admin and subscriber passwords).

When one of the password validation options (e.g. 'musthave_digit') is set to 0 - it is considered as disabled.

Copyright © Sipwise GmbH - All rights reserved

12.4.3. Max Password Age (Password Expiration)

'web_max_age_days' (default: 0) can be used to enable password expiration for admin and subscriber users. When set to anything greater than 0 and the password expires - the following behaviour happens for the user:

- logged in UI users are automatically redirected to the "Password Change" page, where after successful password change they are redirected back to their previous page.
- UI users who log in with the expired password are redirected to the "Password Change" page and afterwards to the default page (dashboard).
- all API v1 requests return 403 Forbidden "Password expired" except PUT/PATCH operations towards /api/admins or /api/subscribers for the authenticated user that contain a new password.

12.5. Managing POSIX Users

The interface to create and delete system or non-system users from the command-line is provided by the 'adduser' package. This package contains the 'adduser', 'deluser', 'addgroup' and 'delgroup' programs that will apply the system policy on ID allocation ranges, home permissions, and any other security parameter.

NOTE Directly using the 'useradd', 'userdel', 'groupadd' and 'groupdel' programs can cause unexpected behavior, such as too permissive home directories. This does not apply to the other programs from the same 'passwd' package to manage POSIX users, such as 'usermod', 'groupmod', 'chfn' and other similar programs.

12.6. LDAP Authentication for POSIX Users

The Sipwise C5 can be configured to accept LDAP authentication for POSIX users. This applies to both local console and remote SSH access.

The LDAP authentication is done through SSSD, a component that abstracts remote account management and handles caching to cope with remote system connectivity issues.

To enable this authentication method, the 'sssd.enable' option needs to be set to 'yes', and the LDAP host needs to be specified in 'sssd.ldap.host'. And finally, the LDAP base DN to use for searches needs to be specified in 'sssd.ldap.base_dn'.

NOTE The LDAP host must be accessed via LDAPS (LDAP over TLS), as this is required for authentication requests. Using unencrypted LDAP will not work. The TLS connection is assumed to involve a trusted certificate, otherwise self-signed certificates would need to be placed on the system CA certificates store.

Depending on the layout of the LDAP accounts the 'sssd.ldap.access_filter' might need to be changed from the default, for example to allow for group based access controls.

NOTE By default on the Sipwise C5, 'su' requires users to be on the 'wheel' group to be able to use it at all. If the account on LDAP is not part of such group, then these users will be unable to use 'su'.

 Copyright © Sipwise GmbH - All rights reserved

12.7. Remote 'root' Logins via SSH

To mitigate possible system intrusions from the outside, it is commonly held best practice to disable remote 'root' (administrator) logins. With remote root logins disabled, it is still possible to log into the system via SSH to a regular, non-privileged user account, and then use 'sudo' or 'su' to elevate account privileges to 'root' administrator level.

WARNING For system setup purposes, the default setting on the Sipwise C5 is to permit remote 'root' logins via SSH. It is strongly recommended to change this default setting as soon as possible.

Once you have created a user account for yourself that can be used to gain 'root' privileges, remote 'root' logins can be disabled via the config switch *sshdpermit_root_login* in */etc/ngcp-config/config.yml*.

```
sshd:
  permit_root_login: yes
```

Valid options are:

- 'yes' - permit remote root logins via SSH. Not recommended.
- 'no' - prohibit all remote root logins via SSH.
- 'prohibit-password' - permit remote root logins, except when password authentication is used. This is a good setting if you still wish to access the 'root' account directly using public-key authentication, but also want to prohibit remote brute-force attacks against the 'root' account password.
- 'forced-commands-only' - identical to 'prohibit-password' but with the additional restriction that only SSH config sections that have a forced command (via *command=* or *ForceCommand*) are permitted to log in. For advanced users.

12.8. 'sudo-io': Logging Input/Output of Commands Run Through 'sudo'

It is possible to enable logging for the input and output of commands run through 'sudo', along other meta-information like timing, to be able to reproduce sessions.

The logs are saved in directories with a special structure, the numbers are for a so-called "session" or sequence:

```
$BASE_DIR/$USERNAME/00/00/00
```

For example:

```
/var/log/sudo-io/root/00/00/00
```

and within them, several files for tty input and output, stdin/stdout/stderr, and other ancillary files.

Copyright © Sipwise GmbH - All rights reserved

New session use *00/00/01*, *00/00/02* and so on.

This is disabled by default, but can be configured in */etc/ngcp-config/config.yml* with the following options:

```
sudo:
  logging:
    enable: no
    exclude_users: []
    max_sessions: 0
```

Valid options are:

- 'enable': 'no' (default) or 'yes'; to use the feature or not
- 'exclude_users': list of users whose commands are excluded from logging
- 'max_sessions': 0 by default, which means unlimited. If set to a positive integer, the sequence returns to 0 after reaching it, and starts to overwrite the files in the subdir '00/00/00' instead of increasing indefinitely.

WARNING This sudo plugin saves all input and output including passwords typed in shell prompts (even when they are not echoed to the screen), and this is unencrypted, so please consider the security implications.

WARNING Even if in some versions the compression of these files is enabled, some commands can use huge amounts of data and fill-up the available disk space very quickly, and full disk will cause serious problems (failures in services, or even preventing external access to the system to fix the problems). Please consider setting up monitoring of free space, or mitigating measures like setting a small amount of sessions saved, taking data out the system often for preservation and deleting sessions from previous days, etc.

WARNING Additionally, this has a performance impact (both CPU and I/O) on every command run with sudo that triggers logging, negligible for most commands but significant when the input or output is large.

12.9. SSL Certificates

The Sipwise C5 provides default, self-signed SSL certificates for SSL connections. These certificates are common for every installation. Before going to production state, the system administrator should provide SSL certificates for the web services. These certificates can either be shared by all web interfaces (*provisioning*, *administrator interface* and *customer self care interface*), or separate ones for each them can be used.

- Generate the certificates. The *customer self care interface* certificate should be signed by a certification authority to avoid browser warnings.
- Upload the certificates to the system
- Set the path to the new certificates in */etc/ngcp-config/config.yml*:

  ossbssapacheautoprovsslcertfile and *ossbssapacheautoprovsslcertkeyfile* for the *provisioning interface*.

 Copyright © Sipwise GmbH - All rights reserved

ossbssapacherestapisslcertfile and *ossbssapacherestapisslcertkeyfile* for the *REST interface*.

www_adminhttp_adminsslcertfile and *www_adminhttp_adminsslcertkeyfile* for the *admin interface*.

www_adminhttp_cscsslcertfile and *www_adminhttp_cscsslcertkeyfile* for the *customer self care interface*.

- Apply the configuration changes with *ngcpcfg apply 'added web ssl certs'*.

The Sipwise C5 also provides the self-signed SSL certificates for SIP over TLS services. The system administrator should replace them with certificates signed by a trusted certificate authority if he is going to enable it for the production usage (config.yml section: *kamailiolbtls* (TLS is enabled by default and uses self-signed SSL certificates)).

- Generate the certificates.
- Upload the certificates to the system
- Set the path to the new certificates in */etc/ngcp-config/config.yml*:

  kamailiolbtlsdomains<domain>sslcertfile and *kamailiolbtlsdomains<domain>sslcertkeyfile* .
- Apply the configuration changes with *ngcpcfg apply 'added kamailio certs'*.

12.10. Securing Your Sipwise C5 Against SIP Attacks

The Sipwise C5 allows you to protect your VoIP system against SIP attacks, in particular **Denial of Service** and **brute-force attacks**. Let us go through each of those attacks and let us see how to configure your system in order to face such situations and react against them.

12.10.1. Denial of Service

As soon as you have packets arriving on your Sipwise C5 server, it will require a bit of time of your CPU. Denial of Service attacks are aimed to break down your system by sending floods of SIP messages in a very short period of time and keep your system busy to handle such huge amount of requests. Sipwise C5 allows you to block such kind of attacks quite easily, by configuring the following section in your /etc/ngcp-config/config.yml :

```
kamailio:
  lb:
    security:
      dos_ban_enable: yes
      dos_ban_time: 3600
      dos_reqs_density_per_unit: 50
      dos_sampling_time_unit: 2
      dos_whitelisted_ips: []
      dos_whitelisted_subnets: []
```

As soon as Sipwise C5 receives more than 50 messages from the same IP in a time window of 2 seconds, that IP will be blocked for 3600 sec, and you will see in the kamailio-lb.log a line saying:

```
Nov 9 00:11:53 sp1 lb[41958]: WARNING: <script>: IP '203.0.113.42' is
```

Copyright © Sipwise GmbH - All rights reserved

```
blocked and banned - R=<null> ID=304153-3624477113-
19168@tedadg.testlab.local
```

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

```
# ngcp-kamctl lb fifo htable.dump ipban
```

Excluding SIP endpoints from banning

There may be some SIP endpoints that send a huge traffic towards Sipwise C5 from a specific IP address. A typical example is a *SIP Peering Server*.

CAUTION Sipwise C5 supports handling such situations by excluding all defined *SIP Peering Servers* from DoS protection mechanism.

The Sipwise C5 platform administrator may also add whitelisted IP addresses manually in `/etc/ngcp-config/config.yml` at `kamailio.lb.security.dos_whitelisted_ips` and `kamailio.lb.security.dos_whitelisted_subnets` parameters.

12.10.2. Bruteforcing SIP Credentials

This is a very common attack that can be checked via the */var/log/ngcp/kamailio-proxy.log* file. There will be INVITE/REGISTER messages coming in with strange usernames. Attackers are trying to spoof/guess subscriber's credentials, which allow them to call out. The very first protection against these attacks is: **ALWAYS USE STRONG PASSWORD**. Nevertheless, Sipwise C5 allows you to detect and block such attacks quite easily by configuring one or both the following methods in `/etc/ngcp-config/config.yml` file

Subscriber Ban

This is the most used method and allows to block a subscriber after a predefined number of failed authentication attempts. The disadvantage of this type of ban is that an attacker can easily block any legitimate subscriber just knowing its username.

```
kamailio:
  lb:
    security:
      failed_auth_attempts: 3
      failed_auth_ban_enable: yes
      failed_auth_ban_time: 3600
```

It may be required to increase the number of failed attempts by adjusting the ban time (e.g. some users can be banned accidentally because they are not writing the right password). If a user tries to authenticate an INVITE/REGISTER (or more in general, any request containing an *Authorization* or *Proxy-Authorization* SIP header) and if it fails more than 3 times, the *user@domain* will be blocked for 3600 seconds (see *failed_auth_ban_time* on */etc/ngcp-config/config.yml*). In this case, you will see the following lines in */var/log/ngcp/kamailio-lb.log* file:

 Copyright © Sipwise GmbH - All rights reserved

```
Nov 9 13:31:56 sp1 lb[41952]: WARNING: <script>: Consecutive
Authentication Failure for 'sipvicous@mydomain.com' UA='sipvicous-
client' IP='203.0.113.42' - R=<null> ID=313793-3624525116-
589163@testlab.local
```

12.10.3. IP Ban

This type of ban is similar to the previous one but the IP address of the source user is blocked in case of excessive failed authentication attempts. This method blocks the IP even if the failed attempts are done using different subscribers usernames.

```
kamailio:
  lb:
    security:
      failed_ip_auth_attempts: 3
      failed_ip_auth_ban_enable: yes
      failed_ip_auth_ban_time: 3600
```

It may be required to increase the number of failed attempts by adjusting the ban time (e.g. some users can be banned accidentally because they are not writing the right password). If a user tries to authenticate an INVITE/REGISTER (or more in general, any request containing an *Authorization* or *Proxy-Authorization* SIP header) and if it fails more than 3 times, the *IP* will be blocked for 3600 seconds (see *failed_ip_auth_ban_time* on */etc/ngcp-config/config.yml*). In this case, you will see the following lines in */var/log/ngcp/kamailio-lb.log* file:

```
Nov 9 13:31:56 sp1 lb[41952]: WARNING: <script>: Consecutive IP
Authentication Failure for 'sipvicous@mydomain.com' UA='sipvicous-
client' IP='203.0.113.42' - R=<null> ID=313793-3624525116-
589163@testlab.local
```

Both the banned IPs and banned users will be shown in the Admin web interface by accessing the *SettingsSecurity Bans* menu. To retrieve the same information from *Kamailio* memory, use the following command:

```
# ngcp-kamctl lb fifo htable.dump auth
```

```
# ngcp-kamctl lb fifo htable.dump ip_auth
```

12.11. Topology Hiding

12.11.1. 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

Copyright © Sipwise GmbH - All rights reserved

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.

12.11.2. Topology Masking Mechanism

Concealment of sensitive information using this mechanism is achieved through encoding the original content of selected SIP headers. Then Sipwise C5 will create a new SIP URI using a preselected IP address and the encoded content as URI parameter, finally re-assembling the SIP header.

Examples for encoded SIP headers:

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

The *load-balancer* element of Sipwise C5 has an SBC role, from the SIP peers point of view. The *LB* offers topology masking function that can be activated through a configuration change. By default the function is disabled.

IMPORTANT Topology masking is not applied to outbound REGISTER message. It also excludes OPTIONS message to avoid overloading in case of natping high traffic. You can exclude additional SIP messages using the config.yml section `kamailio.lb.security.topos.exclude`.

Configuration of Topology Masking

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

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

Meaning of the configuration parameters:

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

TIP Any valid, preferably private network address can be used. The suggestion is however

 Copyright © Sipwise GmbH - All rights reserved

> to use an address that is not used by any other SIP endpoint or intermediary element in the network.

Considerations for Topology Masking

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

The most common example is the consequence that **SIP message size may grow** when applying topology masking. The fact that SIP messages become larger may even prevent Sipwise C5 from communicating successfully with another SIP entity (a peer SBC, for example). This can be expected under the following circumstances:

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

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

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

12.11.3. Topology Hiding Mechanism

This mechanism achieves topology hiding by stripping the SIP routing headers that show topology details and storing those data in the associative data structure (hash) in the Redis DB so that it can look it up when a reply or in-dialog SIP message comes in. From the signaling perspective it simulates a SBC (Session Border Controller) on the LB.

Considerations for Topology Hiding

This mechanism offers some benefits over the older topology masking approach:

- It enables the Sipwise C5 to interconnect with SIP endpoints that are not capable of operating through a SIP proxy.
- The message size is decreased because of stripping the SIP Record-Route, Route and Via header fields.
- It solves the interoperability issues with SIP ALG in some cases.
- It retains also the lightweight nature and the efficient operation.

The module uses the auto-expiration of the Redis keys so it can cause temporary spikes in the memory usage and redis keys count until produced data is cleaned up by redis.

Configuration of Topology Hiding

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

Copyright © Sipwise GmbH - All rights reserved

```
topos:
  enable: no
  redis_db: 24
```

In order to activate the function, you should set *enable: 'yes'* in */etc/ngcp-config/config.yml* and leave the Redis DB number unchanged, then execute *ngcpcfg apply "activated topos"*.

12.12. System Requirements and Performance

The Sipwise C5 is a very flexible system, capable of serving from hundreds to several tens of thousands of subscribers in a single node. The system comes with a default configuration, capable of serving up to 50.000 subscribers in a *normal* environment. But there is no such thing as a *normal* environment. And Sipwise C5 has sometimes to be tuned for special environments, special hardware requirements or growing traffic.

NOTE If you have performance issues with regards to disk I/O please consider enabling the 'noatime' mount option for the root filesystem. Sipwise recommends the usage of 'noatime', though remove it if you use software which conflicts with its presence.

In this section some parameters will be explained to allow Sipwise C5 administrator tune the system requirements for optimum performance.

Table 34. Requirement_options

Option	Default value	Requirement impact
cleanuptoolsbinlog_days	15	Heavy impact on the harddisk storage needed for mysql logs. It can help to restore the database from backups or restore broken replication.
databasebufferpoolsize	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.
kamailiolbpkg_mem	16	This setting affects the amount of RAM the system will use. Each kamailio-lb worker will have this amount of RAM reserved. Lowering this setting up to 8 will help to release some memory depending on the number of kamailio-lb workers running. This can be a dangerous setting as the lb process could run out of memory. Use with caution.
kamailiolbshm_mem	1/16 * Total System RAM	The installer will set this value to 1/16 of the total system RAM. This setting does not change even if the system RAM does so it is 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.

Copyright © Sipwise GmbH - All rights reserved

Option	Default value	Requirement impact
kamailiolbtcp_children	8	Number of TCP workers kamailio-lb will spawn per listening socket. The value should be fine for a mixed UDP-TCP 50K subscriber system. Lowering this setting can free some RAM as the number of kamailio processes would decrease. For a test system or a pure UDP subscriber system 2 is a good value. 1 or 2 TCP workers are always needed.
kamailiolbtlsenable	yes	Enable or not TLS signaling on the system. Setting this value to "no" will prevent kamailio to spawn TLS listening workers and free some RAM.
kamailiolbudp_children	8	See *kamailiolbtcp_children* explanation
kamailioproxychildren	8	See *kamailiolbtcp_children* explanation. In this case the proxy only listens udp so these children should be enough to handle all the traffic. It could be set to 2 for test systems to lower the requirements.
kamailioproxy*_expires		Set the default and the max and min registration interval. The lower it is more REGISTER requests will be handled by the lb and the proxy. It can impact in the network traffic, RAM and CPU usage.
kamailioproxynatping_interval	30	Interval for the proxy to send a NAT keepalive OPTIONS message to the nated subscriber. If decreased, this setting will increase the number of OPTIONS requests the proxy needs to send and can impact in the network traffic and the number of natping processes the system needs to run. See *kamailioproxynatping_processes* explanation.
kamailioproxynatping_processes	7	Kamailio-proxy will spawn this number of processes to send keepalive OPTIONS to the nated subscribers. Each worker can handle about 250 messages/second (depends on the hardware). Depending on the number of nated subscribers and the *kamailioproxynatping_interval* parameter the number of workers may need to be adjusted. The number can be calculated like nated_subscribers/natping_interval/pings_per_second_per_process. For the default options, assuming 50K nated subscribers in the system the parameter value would be 50.000/30/250 = (6,66) 7 workers. 7 is the maximum number of processes kamailio will accept. Raising this value will cause kamailio not to start.
kamailioproxyshm_mem	1/16 * Total System RAM	See *kamailiolbshm_mem* explanation.
rateomatenable	yes	Set this to no if the system should not perform rating on the CDRs. This will save CPU usage.
rsyslogexternal_log	0	If enabled, the system will send the log messages to an external server. Depending on the *rsyslogexternal_loglevel* parameter this can increase dramatically the network traffic.

Copyright © Sipwise GmbH - All rights reserved

Option	Default value	Requirement impact
rsyslogngcp_logs_preserve_days	93	This setting will set the number of days ngcp logs under */var/log/ngcp* will be kept in disk. Lowering this setting will free a high amount of disk space.

TIP In case of using virtualized environment with limited amount of hardware resources, you can use the script *ngcp-toggle-performance-config* to adjust Sipwise C5 configuration for high/low performance:

```
root@spce:~# /usr/sbin/ngcp-toggle-performance-config
/usr/sbin/ngcp-toggle-performance-config - tool to adjust Sipwise C5
configuration for low/high performance

  --help                Display this usage information
  --high-performance    Adjust configuration for system with normal/high
performance
  --low-performance     Adjust configuration for system with low
performance (e.g. VMs)

root@spce:~#
```

12.13. Troubleshooting

The Sipwise C5 platform provides detailed logging and log files for each component included in the system via rsyslog. The main folder for log files is */var/log/ngcp/*, it contains a list of self explanatory log files named by component name.

The Sipwise C5 is a high performance system which requires compromise between traceability (maximum amount of debug information being written to hard drive) and productivity (minimum load on IO subsystem). This is the reason why different log levels are configured for the provided components by default.

Most log files are designed for debugging Sipwise C5 by Sipwise operational team while main log files for daily routine usage are:

Log file	Content	Estimated size
/var/log/ngcp/api.log	API logs providing type and content of API requests and responses as well as potential errors	medium
/var/log/ngcp/panel.log /var/log/ngcp/panel-debug.log	Admin Web UI logs when performing operational tasks on the ngcp-panel	medium
/var/log/ngcp/cdr.log	mediation and rating logs, e.g. how many CDRs have been generated and potential errors in case of CDR generation or rating fails for particular accounting data	medium

Copyright © Sipwise GmbH - All rights reserved

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

12.13.1. Collecting Call Information from Logs

The easiest way to fetch information about a single call among the log files is the search for the SIP CallID (a unique identifier for a SIP dialog). The call ID is used as call marker in almost all the VoIP related log file, such as */var/log/ngcp/kamailio-lb.log*, */var/log/ngcp/kamailio-proxy.log*, */var/log/ngcp/sems.log*, */var/log/ngcp/sems-b2b.log* or */var/log/ngcp/rtp.log*. Example of kamailio-proxy.log line:

```
Nov 19 00:35:56 sp1 proxy[7475]: NOTICE: <script>: New request on proxy
- M=REGISTER R=sip:sipwise.local
F=sip:jdoe@sipwise.local T=sip:jdoe@sipwise.local IP=10.10.1.10:5060
(127.0.0.1:5060) ID=364e467677662103497793 4e055d19ea@127.0.0.1 UA='SIP-
UA 203.0.113.42'
```

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 comes from (in this case Load Balancer)
- ID=364e467677662103497793 4e055d19ea@127.0.0.1 : The SIP CallID.
- UAIP=10.10.1.10 : The User Agent source IP

Copyright © Sipwise GmbH - All rights reserved

- UA='SIP-UA 203.0.113.42' : The SIP User Agent header

In order to collect the full log related to a single call, it is 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
```

12.13.2. Collecting SIP Traces

The Sipwise C5 platform provides several tools to collect SIP traces. It can be used Sipwise C5 *ngrep-sip* tool to collect SIP traces, for example to fetch traffic in text format from outbound and among load balancer, proxy and sems :

```
# ngrep-sip b
```

see the manual to know all the options:

```
# man ngrep-sip
```

The *ngrep* debian tool can be used in order to make a SIP trace and save it into a *.pcap* file :

```
# ngrep -s0 -Wbyline -d any -O /tmp/SIP_trace_file_name.pcap port 5062
or port 5060
```

The *sngrep* debian graphic tool as well can be used to visualize SIP trace and save them in a *.pcap* file :

```
# sngrep
```

12.14. Secure remote 'syslog' logging

When enabling remote 'syslog' logging, it is strongly remmended to enable the use of a secure transport, as the log entries contain sensitive and private data, which should not be exposed in plain text through the network.

To enable secure remote 'syslog' logging, you need to configure the 'rsyslog.external_logging.N.tls.enable' where N is the entry in the list of remote servers. The secure transport works with both 'udp' and 'tcp', via 'DTLS' and 'TLS' respectively. In addition the certificate authority certificate and the server certificate and key files need to be specified, so that the connection can be established securely.

 Copyright © Sipwise GmbH - All rights reserved

Chapter 13. Monitoring and Alerting

13.1. Internal Monitoring

13.1.1. System Monitoring Backend

The platform uses the *Prometheus* monitoring backend on new installations and on upgraded systems that have been migrated.

The platform uses various monitoring backend services to monitor many aspects of the system, including CPU, memory, swap, disk, filesystem, network, processes, NTP, Nginx, Redis and MySQL.

The gathered information is stored in *VictoriaMetrics* which is a long-term storage backend for *Prometheus*. NOTE: Both *VictoriaMetrics* and *Prometheus* can act as the prometheus server implementation, and are mutually exclusive in their execution.

13.1.2. Sipwise C5 Specific Monitoring via ngcp-witnessd

The platform uses the internal *ngcp-witnessd* service to monitor Sipwise C5 specific metrics or system metrics currently not tracked by the monitoring backend (via *Prometheus* exporters), including HA status, MTA, Kamailio, SIP and MySQL.

The gathered information is stored in *VictoriaMetrics* in the *ngcp* namespace on its time-series database.

TIP Some of the data gathering can be disabled (most are enabled by default) through the `config.yml` file, and those data points will then either be missing from the database or be initialized with a stub value. This will then cascade into other subsystems using this monitoring information, such as *Grafana* dashboards. The enable/disable flags can be found in the `witnessd.gather` section.

13.1.3. Monitoring Data in the Monitoring Backend

The platform uses *VictoriaMetrics* as a long-term *Prometheus* time series database to store most of the metrics collected in the system.

The monitoring data is used by the statistics dashboard powered by *Grafana*.

The monitoring data can also be accessed directly by various means. On new installations by using the *promtool* command-line tool; or by using the HTTP API with *curl* (or other HTTP fetchers), or with the *NGCP::Prometheus::HTTP* perl module.

Monitoring Metrics

See Prometheus Monitoring Metrics for detailed information about the list of *ngcp* namespaced metrics stored in the *Prometheus* monitoring database.

PromQL

See https://prometheus.io/docs/prometheus/latest/querying/basics/ for information about PromQL, the query language used by *Prometheus*.

Copyright © Sipwise GmbH - All rights reserved

TIP To get the list of all metrics for a specific namespace the following query can be used {__name__=~"^namespace_.+"}.

13.2. Statistics Dashboard

The platform's administration interface (described in Kick-Off) provides a graphical overview based on *Grafana* of the most important system health indicators, such as memory usage, load averages and disk usage. VoIP statistics, such as the number of concurrent active calls, the number of provisioned and registered subscribers, etc. is also present.

Copyright © Sipwise GmbH - All rights reserved

Chapter 14. Licenses

The Sipwise C5 implements *software licensing* primarily for the commercial products PRO and CARRIER. However as a CE platform operator you may also see a process running on the system: `"licensed"`. The only purpose of this software module is to collect anonymous statistics about the system usage, namely the following performance indicators are recorded:

- number of provisioned subscribers
- number of registered subscribers
- number of concurrent calls

The **anonymous usage statistics is enabled by default but you can disable it**. In order to do that you have to edit the main configuration file '/etc/ngcp-config/config.yml' and set `general.anonymous_usage_statistics` parameter to `no`. Then apply the new configuration with the usual command: `ngcpcfg apply "Disabled anon. usage stat"`

TIP If Sipwise C5 operator does not want to have the license client package (`ngcp-license-client` and `ngcp-license-module`) on his system at all, it is possible to replace it with a dummy package: `ngcp-license-client-dummy`. This dummy package does not contain any licensing software.

Copyright © Sipwise GmbH - All rights reserved

Chapter 15. Customer Self-Care

15.1. Customer Self-Care User Interface (CSC UI)

There are two ways for end users to maintain their subscriber settings:

- via the *Customer Self-Care Web Interface* and
- via *Vertical Service Codes* using their SIP phones.

15.2. The Customer Self-Care Web Interface

The Sipwise C5 provides a web panel for end users (CSC panel) to maintain their subscriber accounts, which is running on https://*ngcp-ip*. Every subscriber can log in there, change subscriber feature settings, view their call lists, retrieve voicemail/recording and trigger calls using the integrated webrtc-over-sip client.

The CSC UI is technically a Single Page Application, that is fully client side rendered and builds on top of the NCGP REST API as a part of the Service Oriented Architecture. The CSC UI source code is published under a GPL license on https://github.com/sipwise/ngcp-csc-ui and can be also used as a guideline towards own CSC UI implementation based on the same REST API methods.

15.2.1. Login Procedure

To log into the CSC panel, the end user has to provide his full **web username** (e.g. *user1@203.0.113.42*) and the **web password** defined in Creating a Subscriber. Once logged in, he can change his web password in the *Account* section. This will NOT change his SIP password, so if you control the end user devices, you can auto-provision the SIP password into the device and keep it secret, and hand over the web password to the customer. This way, the end user will only be able to place calls with this auto-provisioned device and not with an arbitrary soft-phone, but can nonetheless manage his account via the CSC panel.

15.2.2. 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.

Changing the Logo and Colors

For changing the branding logo on a reseller's admin web page and on the CSC panel you need to access the web interface **as Administrator** and navigate to *ResellerDetailsBranding* menu.

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

Copyright © Sipwise GmbH - All rights reserved

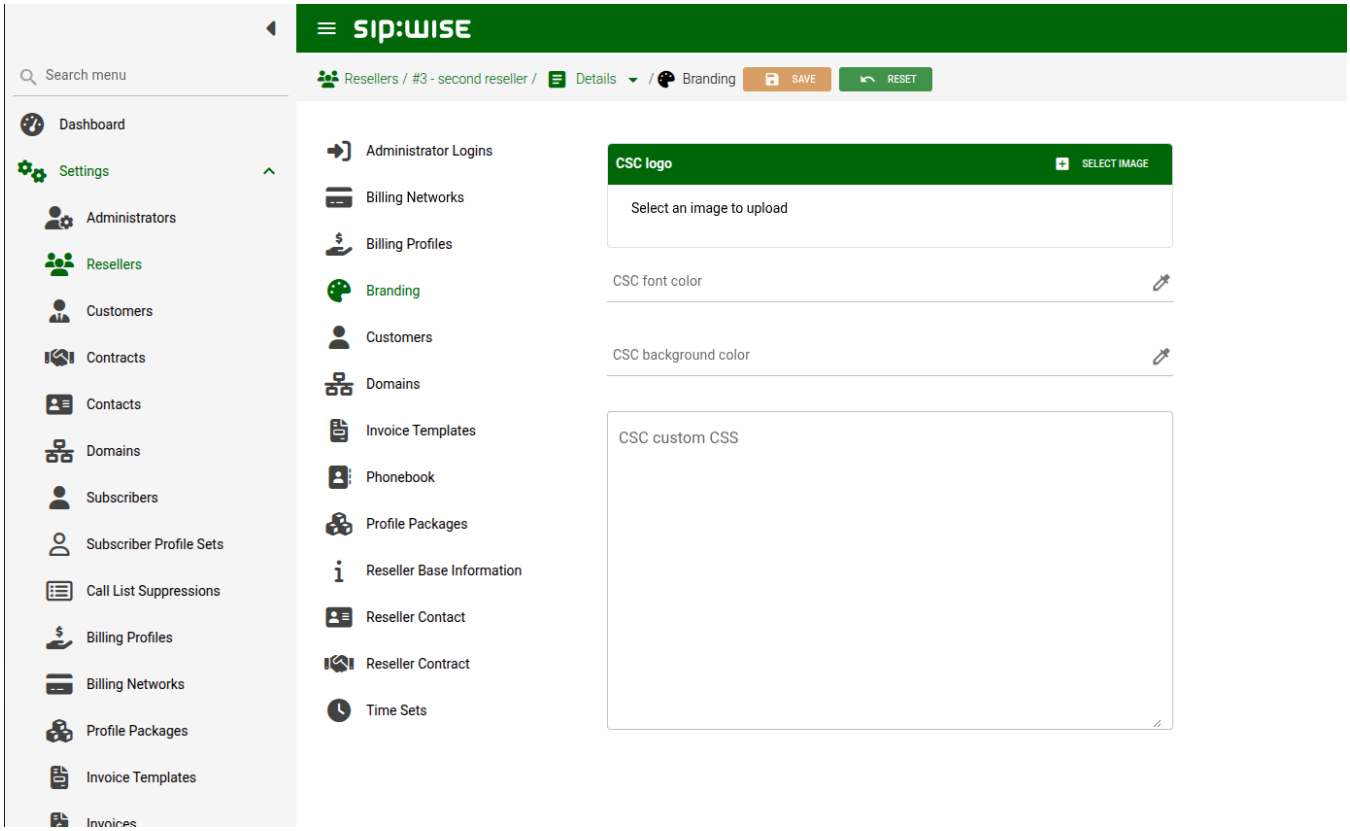

Figure 130. CSC Customisation

The web panel customisation happens as follows:

- **CSC Logo**: allows to upload an image for the new logo
  1. **CSC font color**: allows to select a color to use for the fonts
  2. **CSC background color**: allows to select a color to use as background
  3. **CSC custom color**: allows to customize the CSS style

Press the *Save* button to save changes.

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

Selecting Available Languages

You can also enable/disable specific languages a user can choose from in the CSC panel. Currently, English (*en*), German (*de*), Italian (*it*), Spanish (*es*) and French (*fr*) 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 the `config.yml` file. An example to set the English language as default:

```
ngcpcfg set /etc/ngcp-config/config.yml www_admin.force_language=en
```

Copyright © Sipwise GmbH - All rights reserved

```
ngcpcfg apply 'Set English as default on CSC'
```

15.3. The Voicemail Menu

Sipwise C5 offers several ways to access the Voicemail box.

The CSC panel allows your users to listen to voicemail messages from the web browser, delete them and call back the user who left the voice message. The user can set up voicemail forwarding to the external email and the PIN code needed to access the voicebox from any telephone also from the CSC panel.

To manage the voice messages from SIP phone: dial internal voicemail access number `2000`.

To change the access number: look for the parameter `sems.vsc.voicemail_number` in `config.yml`. 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 set up some DID number as external voicemail access number: for that, you should add a special rewrite rule (Inbound Rewrite Rule for Callee, see Configuring Rewrite Rule Sets.) on the incoming peer, to rewrite that DID to "voiceboxpass". Now when user calls this number the call will be forwarded to the voicemail server and he will be prompted for mailbox and password. The mailbox is the full E.164 number of the subscriber account and the password is the PIN set in the CSC panel.
- The user can also dial his own number from PSTN, if he set up 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.

Copyright © Sipwise GmbH - All rights reserved

Chapter 16. REST API

The Sipwise C5 provides the REST API interface for interconnection with 3rd party tools.

The Sipwise C5 provides a REST API to provision various functionality of the platform. The entry point - and at the same time the official documentation - is at *https://<your-ip>:1443/api*. It allows both administrators and resellers (in a limited scope) to manage the system.

You can either authenticate via username and password of your administrative account you are using to access the admin panel, or via SSL client certificates. With username and password it is recommended to stick with the *local* authentication mode rather than remote authentication (e.g. *LDAP*), in order to prevent overloading the remote server. For client certificate authentication, there are details provided in the online API documentation.

16.1. 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);
```

Copyright © Sipwise GmbH - All rights reserved

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/");
```


Copyright © Sipwise GmbH - All rights reserved

```
$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";
```

Copyright © Sipwise GmbH - All rights reserved

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

At that stage, the subscriber can connect via SIP, 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, cfb, cft, cfs, cfr and cfo via this API call, also all at once. Destinations can be hunting lists as described above or just a single number. Also, a time set can be provided to trigger call forwards only during specific time periods.

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

```
$req = HTTP::Request->new('GET',
"$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} = 'libswrate';
$prefs->{block_in_clir} = JSON::true; # reject incoming anonymous calls
$prefs->{block_in_list} = [ # reject calls from the following numbers:
```

 Copyright © Sipwise GmbH - All rights reserved

```
    '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 cannot recover the subscriber anymore.

```
$req = HTTP::Request->new('DELETE',
"$uri/api/subscribers/$subscriber_id");
$res = $ua->request($req);
if($res->code != 204) {
```

Copyright © Sipwise GmbH - All rights reserved

```
    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, log in or make calls/chats.

Resource 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";
```

16.2. API Performance Considerations

The REST API is designed with pagination support built-in. It is mandatory, to implement pagination in your API clients. If you circumvent pagination by setting the number of rows requested in one API call to a very high number the following side effects may appear:

1. An HTTP timeout at the gateway may occur. The default timeout limit is set to 60s. You can change it by creating a patchtt file for the following template: */etc/ngcp-config/templates/etc/nginx/sites-available/ngcp-panel_admin_api.tt2*.
2. Other parts of the system might become unresponsive due to mysql table locks. This especially applies to endpoints related to the Customers entity.

Copyright © Sipwise GmbH - All rights reserved

Appendix

Appendix A: Configuration Overview

config.yml Overview

`/etc/ngcp-config/config.yml` is the main configuration YAML file used by Sipwise C5. After every change, you need to run the command `ngcpcfg apply "my commit message"` to apply changes (followed by `ngcpcfg push` in the PRO version to apply changes to sp2). The following is a brief description of the main variables contained into `/etc/ngcp-config/config.yml` file.

apps

This section contains parameters for the additional applications that may be activated on Sipwise C5.

```
apps:
  malicious_call: no
```

- malicious_call: If set to 'yes', the Malicious Call Identification (MCID) application will be enabled.

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
```

Copyright © Sipwise GmbH - All rights reserved

```
    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.

autoprov

The following is the autoprovisioning section:

```
autoprov:
  hardphone:
    skip_vendor_redirect: no
  server:
    bootstrap_port: 1445
    ca_certfile: '/etc/ngcp-config/shared-files/ssl/client-auth-ca.crt'
    host: localhost
    port: 1444
    server_certfile: '/etc/ngcp-config/shared-files/ssl/myserver.crt'
    server_keyfile: '/etc/ngcp-config/shared-files/ssl/myserver.key'
    ssl_enabled: yes
  softphone:
    config_lockdown: 0
```


Copyright © Sipwise GmbH - All rights reserved

```
    webauth: 0
```

- autoprov.skip_vendor_redirect: Skip phone vendor redirection to the vendor provisioning web site.

sems-b2b (some parameters are only used with additional Cloud PBX module activated)

The following is the B2B section:

```
b2b:
  bindport: 5080
  dialog_publish_expires: 3600
  enable: yes
  highport: 19999
  lowport: 15000
  media_processor_threads: 10
  moh_codecs:
    codecs_list: PCMA,PCMU,telephone-event
    enable: no
    mode: whitelist
  permit_ext_dial_when_no_prompt_exists: no
  session_processor_threads: 10
  xmlrpcport: 8090
```

- b2b.enable: Enable sems-b2b service.

backuptools

The following is the backup tools section:

```
backuptools:
  cdrexport_backup:
    enable: no
  etc_backup:
    enable: no
  mail:
    address: noc@company.org
    error_subject: '[ngcp-backup] Problems detected during daily backup'
    log_subject: '[ngcp-backup] Daily backup report'
    send_errors: no
    send_log: no
  mysql_backup:
    enable: no
    exclude_dbs: 'syslog sipstats information_schema'
  replicas:
    peer: yes
    mgmt: no
  rotate_days: 7
  storage_dir: '/ngcp-data/backup/ngcp_backup'
  temp_backup_dir: '/ngcp-data/backup/ngcp_backup/tmp'
```

Copyright © Sipwise GmbH - All rights reserved

- backuptools.cdrexport_backup.enable: Enable backup of cdrexport (.csv) directory.
- backuptools.etc_backup.enable: Enable backup of /etc/* directory.
- backuptools.mail.address: Destination email address for backup emails.
- backuptools.mail.error_subject: Subject for error emails.
- backuptools.mail.log_subjetc: Subject for daily backup report.
- backuptools.mail.send_error: Send daily backup error report.
- backuptools.mail.send_log: Send daily backup log report.
- backuptools.mysql_backup.enable: Enable daily mysql backup.
- backuptools.mysql_backup.exclude_dbs: exclude mysql databases from backup.
- backuptools.replicas.peer: Enable or disable copying the backups to the peer node, for additional safety.
- backuptools.replicas.mgmt: Enable or disable copying the backups to the mgmt nodes, for additional safety, and so that the management nodes have consolidated backups for the entire cluster.
- backuptools.rotate_days: Number of days backup files should be kept. All files older than specified number of days are deleted from the storage directory.
- backuptools.storage_dir: Storage directory of backups.
- backuptools.storage_group: Name of the group that backup files should be owned by.
- backuptools.storage_user: Name of the user that backup files should be owned by.
- backuptools.temp_backup_dir: Temporary storage directory of backups.

cdrexport

The following is the cdrexport section:

```
cdrexport:
  default:
    admin_export_fields:
    - ...
    csv_esc: \
    csv_footer: ${checksum}
    csv_header: ${version},${rows,%04i}
    csv_sep: ','
    daily_folder: yes
    dir_reseller_time: no
    enable: yes
    export_conditions:
    - ...
    export_failed: no
    export_incoming: no
    export_joins:
    - ...
    export_nodes:
      hosts:
      roles:
```


Copyright © Sipwise GmbH - All rights reserved

```
      - mgmt
    exportpath: /home/jail/home/cdrexport
    full_names: yes
    intermediate: no
    max_rows_per_file: '5000'
    monthly_folder: yes
    reseller_export_fields:
    - ...
    write_empty: yes
  my_cdr_stream:
    admin_export_fields:
    - ...
```

- cdrexport.<stream name>: Each CDR export configuration is entitled as "stream" (default, my_cdr_stream, ...).
- cdrexport.<stream name>.enable: Enable or disable this cdr export stream.
- cdrexport.<stream name>.export_nodes.roles: Specify the node roles where to export CDRs.
- cdrexport.<stream name>.export_nodes.hosts: Specify the node hosts where to export CDRs.
- cdrexport.<stream name>.csv_sep: Specify the separator for exported CDRs in CSV format.
- cdrexport.<stream name>.csv_esc: Specify escape symbol to use for the separator in field values
- cdrexport.<stream name>.csv_header": header line in files
- cdrexport.<stream name>.csv_footer": footer line in files
- cdrexport.<stream name>.daily_folder: Specify whether or not a daily folder should be created.
- cdrexport.<stream name>.export_failed: Specify whether or not failed calls should be exported.
- cdrexport.<stream name>.export_incoming: Specify whether or not incoming calls should be exported.
- cdrexport.<stream name>.intermediate: Specify whether or not intermediate CDRs should be exported.
- cdrexport.<stream name>.exportpath: Specify the export path for CDR files.
- cdrexport.<stream name>.full_names: Specify whether or not to use the full name for CDR files instead of short ones.
- cdrexport.<stream name>.monthly_folder: Specify whether or not the system should create a monthly folder (e.g. folder '201801' for January 2018) for CDRs under the configured path.
- cdrexport.<stream name>.dir_reseller_time: Apply the reseller timezone for dates in daily/monthly folder names
- cdrexport.<stream name>.files_owner: Specify user name for created files (default: cdrexport).
- cdrexport.<stream name>.files_group: Specify group name for created files (default: cdrexport).
- cdrexport.<stream name>.files_mask: Specify the file permission mask for created files (default: 022).
- cdrexport.<stream name>.resellers: List of reseller names (including 'system' reseller) to write files for.
- cdrexport.<stream name>.write_empty: Specify whether or not to write files with 0 records for reseller/peers w/o activity.

Copyright © Sipwise GmbH - All rights reserved

- cdrexport.<stream name>.max_rows_per_file: Specify how many rows you can store in one single CSV file.
- cdrexport.<stream name>.admin_export_fields: Exported field list for the system administrator user.
- cdrexport.<stream name>.reseller_export_fields: Exported field list for the system resellers.
- cdrexport.<stream name>.export_joins: SQL joined tables
- cdrexport.<stream name>.export_conditions: SQL filter conditions

cdr-exporterd

The following is the cdr-exporterd section:

```
bulk_processor:
  cdr_exporterd:
    enable: yes
    streams:
      postgresql_teststream:
        type: postgresql
        blocksize: 1000
        limit: 5000
        multithreading: yes
        numofthreads: 2
        table_fields:
          - path: rating_status
            colname: column1
          - path: update_time
            colname: column2
          - path: id
            colname: column3
        tablename: table_to_sync
        schemaname: test
        username: postgres
        password: postgres
        host: 192.168.0.85
        port:
        txn_per_cdr: no
        ignore_unique: no
        multi_row_insert: yes
      oracle_teststream:
        type: oracle
        blocksize: 1000
        limit: 5000
        multithreading: yes
        numofthreads: 2
        table_fields:
          - path: rating_status
            colname: column1
          - path: update_time
            colname: column2
          - path: id
            colname: column3
```

 Copyright © Sipwise GmbH - All rights reserved

```
    tablename: TABLE_TO_SYNC
    servicename: XE
    host: 192.168.0.85
    schemaname: TEST
    username: SYSTEM
    password: oracle
    port:
    txn_per_cdr: no
    ignore_unique: no
    multi_row_insert: no # not supported for oracle
  mssql_teststream:
    type: mssqlserver
    blocksize: 100
    limit: 1000
    multithreading: yes
    numofthreads: 2
    table_fields:
      - path: rating_status
        colname: column1
      - path: update_time
        colname: column2
      - path: id
        colname: column3
    tablename: table_to_sync
    databasename: test
    host: 192.168.0.85
    username: sa
    password: 123
    port:
    txn_per_cdr: no
    ignore_unique: no
    multi_row_insert: yes
  mysql_teststream:
    type: mysql
    blocksize: 1000
    limit: 5000
    multithreading: yes
    numofthreads: 2
    table_fields:
      - path: rating_status
        colname: column1
      - path: update_time
        colname: column2
      - path: id
        colname: column3
    cdrs_export_status: |
      function(record) {
        return [[ record.id, "ok" ], [ (record.id * 1) + 2, "ok" ]];
      }
    tablename: table_to_sync
    databasename: test
    username: root
```

Copyright © Sipwise GmbH - All rights reserved

```
        password:
        host: 192.168.5.133
        port:
        txn_per_cdr: no
        ignore_unique: no
        multi_row_insert: yes
      sqlite_teststream:
        type: sqlite
        blocksize: 1000
        limit: 5000
        multithreading: yes
        numofthreads: 2
        table_fields:
          - path: rating_status
            colname: column1
          - path: update_time
            colname: column2
          - path: id
            colname: column3
        tablename: table_to_sync
        txn_per_cdr: no
        ignore_unique: no
        multi_row_insert: yes
        file_path: /home/jail/home/cdrexport/cdrexport.db
      ...
```

- bulk_processor.cdr_exporterd.enable: Enable or disable the ngcp-cdr-exporterd CDR replication daemon.
- bulk_processor.cdr_exporterd.streams.<stream name>: Each CDR replication configuration is entitled as "stream" (postgresql_teststream, oracle_teststream, …).
- bulk_processor.cdr_exporterd.streams.<stream name>.enable: Enable or disable this cdr replication stream.
- bulk_processor.cdr_exporterd.streams.<stream name>.type: Specify the target file or database type (mysql, mariadb, postgres, postgresql, oracle, sqlite, sqlserver, mssql, mssqlserver).
- bulk_processor.cdr_exporterd.streams.<stream name>.blocksize: Specify the page size when fetching (processing) records.
- bulk_processor.cdr_exporterd.streams.<stream name>.multithreading: no: read from NGCP CDR source table and write to target table/file in a single thread; yes: one thread to read from source and one or more threads to write to target.
- bulk_processor.cdr_exporterd.streams.<stream name>.numofthreads: Number of threads to write to target in case of multithreading: yes.
- bulk_processor.cdr_exporterd.streams.<stream name>.limit: The exporter process restarts (with clean memory, or new target file etc.) once the given limit is exceeded (or no more unexported CDRs left).
- bulk_processor.cdr_exporterd.streams.<stream name>.paging_txn: yes: fetch all unexported CDRs at once (make sure to set 'limit'), and load relations in same transaction; no: fetch record pages by individual SQL queries, using OFFSET and LIMIT.

Copyright © Sipwise GmbH - All rights reserved

- bulk_processor.cdr_exporterd.streams.<stream name>.export_joins: Definition of JOINs for the SQL page queries to fetch unexported CDR records. Same as cdrexport.<stream name>.export_joins.
- bulk_processor.cdr_exporterd.streams.<stream name>.export_conditions: Definition of additional WHERE criteria for the SQL page queries to fetch unexported CDR records. Same as cdrexport.<stream name>.export_conditions.
- bulk_processor.cdr_exporterd.streams.<stream name>.relations: Definition of related data of CDR records to gather for export. Each relation is identified by a dot path notation (similar to SQL), beginning with 'cdr' as the root element. The element value can be a true value to include (load) the relation records (and its sub elements), or a specify detailed options how to construct the CDR graph to process.
- bulk_processor.cdr_exporterd.streams.<stream name>.expand_cdrs: Specify a perl sub or javascript function that takes the CDR graph hash obj as input argument and is expected to return an array of CDR graphs. This can be used to expand/split a CDR into more CDRs.
- bulk_processor.cdr_exporterd.streams.<stream name>.cdrs_export_status: Specify a static export status ('unexported','ok','failed', 'skipped') value to set for processed CDRs, OR a perl sub or javascript function that takes a hash obj with (cdr_id, call_id, start_time) key fields of the processed CDR as input argument and is expected to return an array of (cdr_id, export_status) tuples to commit. This can be used to skip selected CDRs, effectively collapsing (related) CDRs into one.
- bulk_processor.cdr_exporterd.streams.<stream name>.table_fields: List of exported fields for tabular targets (database table, .csv, .xls, ...).
- bulk_processor.cdr_exporterd.streams.<stream name>.graph_fields: List of field glob patterns to filter the CDR graph object when exporting file formats with graph structure (.json, .xml, ...).
- bulk_processor.cdr_exporterd.streams.<stream name>.graph_fields_mode: Consider 'graph_fields' glob patterns as whitelist (include matching paths) or blacklist (exclude matching paths).
- bulk_processor.cdr_exporterd.streams.<stream name>.tablename: Target table name. It will be created using columns of TEXT data type supported by the remote RDBMS, if it does not exist yet.
- bulk_processor.cdr_exporterd.streams.<stream name>.databasename: Target database name (oracle, mssqlserver, mysql, mariadb).
- bulk_processor.cdr_exporterd.streams.<stream name>.servicename: Target database servicename (oracle).
- bulk_processor.cdr_exporterd.streams.<stream name>.sid: Target database SID (oracle).
- bulk_processor.cdr_exporterd.streams.<stream name>.schemaname: Target database schema name (oracle, postgres).
- bulk_processor.cdr_exporterd.streams.<stream name>.username Target database credentials.
- bulk_processor.cdr_exporterd.streams.<stream name>.password: Target database credentials.
- bulk_processor.cdr_exporterd.streams.<stream name>.host: Target database IP address.
- bulk_processor.cdr_exporterd.streams.<stream name>.port: Target database TCP port.
- bulk_processor.cdr_exporterd.streams.<stream name>.txn_per_cdr: Commit exported CDRs on by one or blockwise (see 'blocksize'). Enable this if there is a frequent chance of failures (i.e. from constraints or rowlocks) when committing to the target database, and avoid rolling back large blocks.
- bulk_processor.cdr_exporterd.streams.<stream name>.ignore_unique: Ignore (skip) unique constraint violations when committing to the target db. Note this is not yet supported for all target database types.

Copyright © Sipwise GmbH - All rights reserved

- bulk_processor.cdr_exporterd.streams.<stream name>.file_path Output file path pattern, e.g. for SQLite target database.
- bulk_processor.cdr_exporterd.streams.<stream name>.multi_row_insert: Use SQL INSERT statements with multiple rows to speed up committing to target db. Will be in effect only with txn_per_cdr: no.

cleanuptools

The following is the cleanup tools section:

```
cleanuptools:
  acc_cleanup_days: 90
  archive_targetdir: '/ngcp-data/backups/cdr'
  binlog_days: 15
  cdr_archive_months: 2
  cdr_keep_months: 2
  compress: gzip
  delete_old_cdr_files:
    enable: no
    max_age_days: 30
    paths:
      -
        max_age_days: ~
        path: '/home/jail/home/*/20[0-9][0-9][0-9][0-9]/[0-9][0-9]'
        remove_empty_directories: yes
        wildcard: yes
      -
        max_age_days: ~
        path: '/home/jail/home/cdrexport/resellers/*/20[0-9][0-9][0-
9][0-9]/[0-9][0-9]'
        remove_empty_directories: yes
        wildcard: yes
      -
        max_age_days: ~
        path: '/home/jail/home/cdrexport/system/20[0-9][0-9][0-9][0-
9]/[0-9][0-9]'
        remove_empty_directories: yes
        wildcard: yes
  sql_batch: 10000
  trash_cleanup_days: 30
```

- cleanuptools.acc_cleanup_days: CDR records in `acc` table in `kamailio` database will be deleted after this time
- cleanuptools.binlog_days: Time after MySQL binlogs will be deleted.
- cleanuptools.cdr_keep_months: Specify how many months of CDRs to keep in the main table (visible to call lists). Only relevant with cdr_cleanup_mode: 'backup'.
- cleanuptools.cdr_archive_months: Specify how many months worth of records to keep in monthly CDR backup tables instead of dumping them into archive files (under 'cleanuptools.archive_targetdir' folder) and dropping them from database. Only relevant with

Copyright © Sipwise GmbH - All rights reserved

cdr_cleanup_mode: 'backup'.

- cleanuptools.cdr_keep_days: Specify how many days of CDRs to keep in the main table (visible to call lists). Only relevant with cdr_cleanup_mode: 'cleanup'.
- cleanuptools.cdr_cleanup_mode: 'cleanup' will **delete** CDRs older than cdr_keep_days. 'backup' will move and archive CDRs older than cdr_keep_months.
- cleanuptools.fax_keep_months: Specify how many months of faxes to keep in the fax journal. Only relevant with fax_cleanup_mode: 'backup'.
- cleanuptools.fax_archive_months: Specify how many months worth of records to keep in monthly fax backup tables instead of dumping them into archive files (under 'cleanuptools.archive_targetdir' folder) and dropping them from database. Only relevant with fax_cleanup_mode: 'backup'.
- cleanuptools.fax_keep_days: Specify how many days of faxes to keep in the fax journal. Only relevant with fax_cleanup_mode: 'cleanup'.
- cleanuptools.fax_cleanup_mode: 'cleanup' will **delete** faxes older than fax_keep_days. 'backup' will move and archive faxes older than fax_keep_months.
- cleanuptools.delete_old_cdr_files:

  enable: Enable (`yes`) or disable (`no`) exported CDR cleanup.

  max_age_days: Gives the expiration time of the exported CDR files in days. There is a general value which may be overridden by a local value provided at a specific path. The local value is valid for the particular path only.

  paths: an array of path definitions

    path: a path where CDR files are to be found and deleted; this may contain wildcard characters

    wildcard: Enable (`yes`) or disable (`no`) using wildcards in the path

    remove_empty_directories: Enable (`yes`) or disable (`no`) removing empty directories if those are found in the given `path`

    max_age_days: the local expiration time value for files in the particular path
- cleanuptools.sql_batch: How many records to process within a single SQL statement.
- cleanuptools.trash_cleanup_days: Time after CDRs from acc_trash and acc_backup tables in kamailio database will be deleted.

For the description of *cleanuptools* please visit Cleanuptools Description section of the handbook.

cluster_sets

The following is the cluster sets section:

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

- cluster_sets.<label>: an arbitrary label of the cluster set; in the above example we have default

Copyright © Sipwise GmbH - All rights reserved

- 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

cron

The following is the cron section:

```
cron:
  crontab:
    users:
      allow: []
```

- cron.crontab.users.allow: Specify a list of usernames that are allowed to schedule cronjobs with crontab(1). By default no user besides root can use crontab(1).

database

The following is the database section:

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

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

faxserver

The following is the fax server section:

```
faxserver:
  enable: yes
  fail_attempts: '3'
  fail_retry_secs: '60'
  mail_from: 'Sipwise C5 FaxServer <voipfax@ngcp.sipwise.local>'
```

- faxserver.enable: 'yes'/'no' to enable or disable ngcp-faxserver on the platform respectively.
- faxserver.fail_attempts: Amount of attempts to send a fax after which it is marked as 'failed'.
- faxserver.fail_retry_secs: Amount of seconds to wait between "fail_attempts".
- faxserver.mail_from: Sets the email From Header for incoming fax.

general

The following is the general section:

```
general:
```


Copyright © Sipwise GmbH - All rights reserved

```
  adminmail: adjust@example.org
  companyname: sipwise
  lang: en
  production: yes
  timezone: localtime
```

- general.adminmail: Email address used by monit to send notifications to.
- general.companyname: Label used in SNMPd configuration.
- general.lang: Sets sounds language (e.g.: 'de' for German)
- general.production: Label to hint self-check scripts about installation mode.
- general.timezone: Sipwise C5 Timezone

intercept

The following is the legal intercept section:

```
intercept:
  enable: no
```

- intercept.enable: Enable ngcp-voisniff for Lawful Interception (additional Sipwise C5 module).

kamailio

The following is the kamailio section:

```
kamailio:
  lb:
    block_useragents:
      action: reject
      block_empty: no
      block_absent: no
      enable: no
      mode: blacklist
      ua_patterns: []
    cfgt: no
    debug:
      enable: no
      modules:
      - level: '1'
        name: core
      - level: '3'
        name: xlog
    debug_level: '1'
    debug_uri:
      enable: no
      redis_db: 27
      htable_idx_size: 4
    dns:
```

Copyright © Sipwise GmbH - All rights reserved

```
      dns_sctp_pref: 1
      dns_tcp_pref: 1
      dns_tls_pref: 1
      dns_try_naptr: no
      dns_udp_pref: 1
      use_dns_cache: on
    enable: yes
    external_sbc: []
    extra_sockets: ~
    filter_content_type:
      enable: yes
      action: filter
      content_type_list:
      - content_type: application/vnd.etsi.cug+xml
        direction: all
      - content_type: application/isup
        direction: reply
      - content_type: application/xml
        direction: request
    max_forwards: '70'
    mem_log: '1'
    mem_summary: '12'
    max_inv_lifetime: '180000'
    nattest_exception_ips:
    - 203.0.113.42
    - 203.0.113.43
    nattest_exception_nets:
    - 192.168.10.0/24
    - 192.168.11.0/24
    pkg_mem: '16'
    port: '5060'
    sdp_line_filter:
      enable: no
      remove_line_startswith: []
    security:
      dos_ban_enable: yes
      dos_ban_time: '300'
      dos_reqs_density_per_unit: '50'
      dos_sampling_time_unit: '5'
      dos_whitelisted_ips: []
      dos_whitelisted_subnets: []
      failed_auth_attempts: '3'
      failed_auth_ban_enable: yes
      failed_auth_ban_time: '3600'
      failed_ip_auth_attempts: '3'
      failed_ip_auth_ban_enable: no
      failed_ip_auth_ban_time: '3600'
      topoh:
        enable: no
        mask_callid: no
        mask_ip: 127.0.0.8
      topos:
```


Copyright © Sipwise GmbH - All rights reserved

```
        enable: no
        redis_db: 24
    shm_mem: '64'
    skip_contact_alias_for_ua_when_tcp:
      enable: no
      user_agent_patterns: []
    strict_routing_safe: no
    syslog_options: yes
    tcp_children: 1
    tcp_max_connections: '2048'
    tls:
      connection_timeout: 600
      domains:
      - ca_list: ''
        cipher_list:
MEDIUM:HIGH:!SSLv2:!SSLv3:!aNULL:!eNULL:!LOW:!3DES:!MD5:!EXP:!PSK:!DSS:!
RC4:!SEED:!ECDSA
        description: Default server TLS configuration for inbound
connections
        domain: default
        enable: yes
        require_certificate: no
        server_id: ''
        server_name: ''
        server_name_mode: ''
        sslcertfile: /etc/ngcp-config/shared-files/ssl/myserver.crt
        sslcertkeyfile: /etc/ngcp-config/shared-files/ssl/myserver.key
        tls_method: TLSv1.2+
        type: server
        verify_certificate: no
      - ca_list: ''
        cipher_list:
MEDIUM:HIGH:!SSLv2:!SSLv3:!aNULL:!eNULL:!LOW:!3DES:!MD5:!EXP:!PSK:!DSS:!
RC4:!SEED:!ECDSA
        description: Defaul client TLS configuration for outbound
connections
        domain: default
        enable: yes
        require_certificate: no
        server_id: ''
        server_name: ''
        server_name_mode: ''
        sslcertfile: /etc/ngcp-config/shared-files/ssl/myserver.crt
        sslcertkeyfile: /etc/ngcp-config/shared-files/ssl/myserver.key
        tls_method: TLSv1.2+
        type: client
        verify_certificate: no
      enable: yes
      init_mode: 0
      port: '5061'
      rand_engine: ''
      static: no
```

Copyright © Sipwise GmbH - All rights reserved

```
    udp_children: 1
  proxy:
    allow_cf_to_itself: no
    allow_info_method: no
    allow_msg_method: no
    allow_peer_relay: no
    allow_refer_method: no
    always_anonymize_from_user: no
    authenticate_bye: no
    cf_depth_limit: '10'
    cfgt: no
    check_prev_forwarder_as_upn: no
    children: 1
    decode_utu_header: no
    debug:
      enable: no
      modules:
      - level: '1'
        name: core
      - level: '3'
        name: xlog
    debug_level: '1'
    default_expires: '3600'
    default_expires_range: '30'
    dlg_timeout: '43200'
    early_rejects:
      block_admin:
        announce_code: '403'
        announce_reason: Blocked by Admin
      block_callee:
        announce_code: '403'
        announce_reason: Blocked by Callee
      block_caller:
        announce_code: '403'
        announce_reason: Blocked by Caller
      block_contract:
        announce_code: '403'
        announce_reason: Blocked by Contract
      block_in:
        announce_code: '403'
        announce_reason: Block in
      block_out:
        announce_code: '403'
        announce_reason: Blocked out
      block_override_pin_wrong:
        announce_code: '403'
        announce_reason: Incorrect Override PIN
      callee_busy:
        announce_code: '486'
        announce_reason: Busy Here
      callee_offline:
        announce_code: '480'
```


Copyright © Sipwise GmbH - All rights reserved

```
      announce_reason: Offline
    callee_tmp_unavailable:
      announce_code: '480'
      announce_reason: Temporarily Unavailable
    callee_tmp_unavailable_gp:
      announce_code: '480'
      announce_reason: Unavailable
    callee_tmp_unavailable_tm:
      announce_code: '408'
      announce_reason: Request Timeout
    callee_unknown:
      announce_code: '404'
      announce_reason: Not Found
    cf_loop:
      announce_code: '480'
      announce_reason: Unavailable
    emergency_invalid:
      announce_code: '404'
      announce_reason: Emergency code not available in this region
    emergency_unsupported:
      announce_code: '403'
      announce_reason: Emergency Calls Not Supported
    invalid_speeddial:
      announce_code: '484'
      announce_reason: Speed-Dial slot empty
    locked_in:
      announce_code: '403'
      announce_reason: Callee locked
    locked_out:
      announce_code: '403'
      announce_reason: Caller locked
    max_calls_in:
      announce_code: '486'
      announce_reason: Busy
    max_calls_out:
      announce_code: '403'
      announce_reason: Maximum parallel calls exceeded
    no_credit:
      announce_code: '402'
      announce_reason: Insufficient Credit
    peering_unavailable:
      announce_code: '503'
      announce_reason: PSTN Termination Currently Unavailable
    reject_vsc:
      announce_code: '403'
      announce_reason: VSC Forbidden
    relaying_denied:
      announce_code: '403'
      announce_reason: Relaying Denied
    unauth_caller_ip:
      announce_code: '403'
      announce_reason: Unauthorized IP detected
```

Copyright © Sipwise GmbH - All rights reserved

```
    emergency_priorization:
      enable: no
      register_fake_200: yes
      register_fake_expires: '3600'
      reject_code: '503'
      reject_reason: Temporary Unavailable
      retry_after: '3600'
    enable: yes
    enum_suffix: e164.arpa.
    expires_range: '30'
    filter_100rel_from_supported: no
    filter_failover_response: 408|500|503
    foreign_domain_via_peer: no
    fritzbox:
      enable: no
      prefixes:
      - 0$avp(caller_ac)
      - $avp(caller_cc)$avp(caller_ac)
      - \+$avp(caller_cc)$avp(caller_ac)
      - 00$avp(caller_cc)$avp(caller_ac)
      special_numbers:
      - '112'
      - '110'
      - 118[0-9]{2}
    ignore_auth_realm: no
    ignore_subscriber_allowed_clis: no
    keep_original_to: no
    lcr_stopper_mode: 0
    latency_limit_action: '100'
    latency_limit_db: '500'
    latency_log_level: '1'
    latency_runtime_action: 1000
    lnp:
      add_reply_headers:
        enable: no
        number: P-NGCP-LNP-Number
        status: P-NGCP-LNP-Status
      api:
        add_caller_cc_to_lnp_dst: no
        invalid_lnp_routing_codes:
        - ^EE00
        - ^DD00
        keepalive_interval: '3'
        lnp_request_blacklist: []
        lnp_request_whitelist: []
        port: '8991'
        reply_error_on_lnp_failure: no
        request_timeout: '1000'
        server: localhost
        tcap_field_fci: end.components.0.invoke.parameter
        tcap_field_lnp: ConnectArg.destinationRoutingAddress.0
        tcap_field_opcode: end.components.0.invoke.opCode
```


Copyright © Sipwise GmbH - All rights reserved

```
      enable: no
      execute_ncos_block_out_before_lnp: no
      skip_callee_lnp_lookup_from_any_peer: no
      strictly_check_ncos: no
      type: api
    lookup_peer_destination_domain_for_pbx: no
    loop_detection:
      enable: no
      expire: '1'
      max: '5'
    max_expires: '43200'
    max_gw_lcr: '128'
    max_registrations_per_subscriber: '5'
    mem_log: '1'
    mem_summary: '12'
    min_expires: '60'
    nathelper:
      sipping_from: sip:pinger@sipwise.local
    nathelper_dbro: no
    natping_interval: '30'
    natping_processes: 1
    nonce_expire: '300'
    pbx:
      hunt_display_fallback_format: '[H %s]'
      hunt_display_fallback_indicator: $var(cloud_pbx_hg_ext)
      hunt_display_format: '[H %s]'
      hunt_display_indicator: $var(cloud_pbx_hg_displayname)
      hunt_display_maxlength: 8
      ignore_cf_when_hunting: no
    peer_probe:
      available_treshold: '1'
      enable: yes
      from_uri_domain: probe.ngcp.local
      from_uri_user: ping
      interval: '10'
      method: OPTIONS
      reply_codes: class=2;class=3;code=403;code=404;code=405
      timeout: '5'
      unavailable_treshold: '1'
    perform_peer_failover_on_tm_timeout: yes
    perform_peer_lcr: no
    pkg_mem: '32'
    port: '5062'
    presence:
      enable: yes
      max_expires: '3600'
      reginfo_domain: example.org
    proxy_lookup: no
    push:
      apns_alert: New call
      apns_sound: incoming_call.xaf
      code_18x: 180
```

Copyright © Sipwise GmbH - All rights reserved

```
      reason_18x: 'Ringing'
      reply_18x: 'no'
    report_mos: yes
    set_ruri_to_peer_auth_realm: no
    shm_mem: '125'
    skip_pbx_loop: no
    stir:
      cache_dir: /var/cache/kamailio/stir/
      cache_expire: 3600
      domains:
      - name: <domain_name>
        private_key: <path_to_a_private_key_related_to_domain>
      enable: yes
      expire: 300
      libopt: []
      shaken:
        attestation_name: verstat
        attestation_values:
          failed: TN-Validation-Failed
          no_validation: No-TN-Validation
          not_present: TN-Validation-Not-Present
          passed: TN-Validation-Passed
          passed_A: TN-Validation-Passed-A
          passed_B: TN-Validation-Passed-B
          passed_C: TN-Validation-Passed-C
      timeout: 5
    store_recentcalls: no
    syslog_options: yes
    tcp_children: 1
    tm:
      fr_inv_timer: '180000'
      fr_timer: '9000'
      max_inv_lifetime: '180000'
    treat_600_as_busy: yes
    use_enum: no
    usrloc_dbmode: '1'
    voicebox_first_caller_cli: yes
    xfer_other_party_from: no
```

- kamailio.lb.block_useragents.action: one of [`drop`, `reject`] - Whether to silently drop the request from matching User-Agent or reject with a 403 message.
- kamailio.lb.block_useragents.block_empty: Enable/disable a rejection of messages with an empty User-Agent header (header present, but no value given).
- kamailio.lb.block_useragents.block_absent: Enable/disable a rejection of messages with an absent User-Agent header.
- kamailio.lb.block_useragents.enable: Enable/disable the User-Agent blocking.
- kamailio.lb.block_useragents.mode: one of [`whitelist`, `blacklist`] - Sets the mode of ua_patterns list evaluation (whitelist: block requests coming from all but listed User-Agents, blacklist: block requests from all listed User-Agents).

Copyright © Sipwise GmbH - All rights reserved

- kamailio.lb.block_useragents.ua_patterns: List of User-Agent string patterns that trigger the block action.
- kamailio.lb.cfgt: Enable/disable unit test config file execution tracing.
- kamailio.lb.debug.enable: Enable per-module debug options.
- kamailio.lb.debug.modules: List of modules to be traced with respective debug level.
- kamailio.lb.debug_uri.enable: Enable/disable sending SIP messages From/To specific subscriber to an inactive proxy node in order to debug/trace calls. Only makes sense on Sipwise C5 CARRIER appliance environment.
- kamailio.lb.debug_uri.redis_db: A number of internal Redis DB used by htable module to keep the subscribers values
- kamailio.lb.debug_uri.htable_idx_size: number to control how many slots (buckets) to create for the hash table (2^size). See kamailio htable docs for details.
- kamailio.lb.debug_level: Default debug level for `kamailio-lb`.
- kamailio.lb.dns.use_dns_cache: Enable/disable use of internal DNS cache.
- kamailio.lb.dns.dns_udp_pref: Set preference for each protocol when doing NAPTR lookups.In order to use remote site preferences set all dns_*_pref to the same positive value (e.g. dns_udp_pref=1, dns_tcp_pref=1, dns_tls_pref=1, dns_sctp_pref=1). To completely ignore NAPTR records for a specific protocol, set the corresponding protocol preference to -1.
- kamailio.lb.dns.dns_tcp_pref: See above.
- kamailio.lb.dns.dns_tls_pref: See above.
- kamailio.lb.dns.dns_sctp_pref: See above.
- kamailio.lb.dns.dns_try_naptr: Enable NAPTR support according to RFC 3263.
- kamailio.lb.enable: Enable/disable kamailio-lb service.
- kamailio.lb.external_sbc: SIP URI of external SBC used in the Via Route option of peering server.
- kamailio.lb.extra_sockets: Add here extra sockets for Load Balancer.
- kamailio.lb.max_forwards: Set the value for the Max Forwards SIP header for outgoing messages.
- kamailio.lb.mem_log: Specifies on which log level the memory statistics will be logged.
- kamailio.lb.mem_summary: Parameter to control printing of memory debugging information on exit or SIGUSR1 to log.
- kamailio.lb.max_inv_lifetime: Set INVITE transaction timeout per the whole transaction if no final reply for an INVITE arrives after a provisional message was received (whole transaction ringing timeout). It has to be equals or greater than kamailio.proxy.tm.fr_inv_timer.
- kamailio.lb.nattest_exception_ips: List of IPs that do not need the NAT test.
- kamailio.lb.nattest_exception_nets: List of IP networks (sub-nets) that do not need the NAT test. The format is network/mask, see an example of the 'kamailio' section.
- kamailio.lb.shm_mem: Shared memory used by Kamailio Load Balancer.
- kamailio.lb.pkg_mem: PKG memory used by Kamailio Load Balancer.
- kamailio.lb.port: Default listen port.
- kamailio.lb.remove_isup_body_from_replies: Enable/disable stripping of ISUP part from the message body.

Copyright © Sipwise GmbH - All rights reserved

- kamailio.lb.sdp_line_filter.enable: Enable/Disable filter of SDP lines in all the SIP messages.
- kamailio.lb.sdp_line_filter.remove_line_startswith: List of the SDP lines that should be removed. Attention: it removes all SDP attribute lines beginning with the listed strings in all media streams.
- kamailio.lb.security.dos_ban_enable: Enable/Disable DoS Ban.
- kamailio.lb.security.dos_ban_time: Sets the ban time.
- kamailio.lb.security.dos_reqs_density_per_unit: Sets the requests density per unit (if we receive more then * lb.dos_reqs_density_per_unit within dos_sampling_time_unit the user will be banned).
- kamailio.lb.security.dos_sampling_time_unit: Sets the DoS unit time.
- kamailio.lb.security.dos_whitelisted_ips: Write here the whitelisted IPs.
- kamailio.lb.security.dos_whitelisted_subnets: Write here the whitelisted IP subnets.
- kamailio.lb.security.failed_auth_attempts: Sets how many authentication attempts allowed before ban.
- kamailio.lb.security.failed_auth_ban_enable: Enable/Disable authentication ban.
- kamailio.lb.security.failed_auth_ban_time: Sets how long a user has to be banned.
- kamailio.lb.security.failed_ip_auth_attempts: Sets how many authentication attempts allowed before ban the IP.
- kamailio.lb.security.failed_ip_auth_ban_enable: Enable/Disable authentication IP ban.
- kamailio.lb.security.failed_ip_auth_ban_time: Sets how long a IP has to be banned.
- kamailio.lb.topoh.enable: Enable topology masking module (see the Topology Masking Mechanism subchapter for a detailed description).
- kamailio.lb.topoh.mask_callid: if set to `yes`, the SIP Call-ID header will also be encoded.
- kamailio.lb.topoh.mask_ip: an IP address that will be used to create valid SIP URIs, after encoding the real/original header content.
- kamailio.lb.topos.enable: Enable topology hiding module (see the Topology Hiding Mechanism subchapter for a detailed description).
- kamailio.lb.topos.redis_db: A number of internal Redis DB used by the topology hiding module.
- kamailio.lb.strict_routing_safe: Enable strict routing handle feature.
- kamailio.lb.syslog_options: Enable/disable logging of SIP OPTIONS messages to `kamailio-options-lb.log`.
- kamailio.lb.tcp_children: Number of TCP worker processes.
- kamailio.lb.tcp_max_connections: Maximum number of open TCP connections.
- kamailio.lb.tls.enable: Enable TLS socket.
- kamailio.lb.tls.port: Set TLS listening port.
- kamailio.lb.tls.rand_engine: Specify the random number generator engine for libssl. We suggest to not touch this option unless you know what you are doing.
- kamailio.lb.tls.static: If set to yes, the kamailio tlsa module will be used in substitution of the default tls one. It could help in case of crashes related to openssl.
- kamailio.lb.tls.init_mode: It controls how kamailio tls module is initialized and works at runtime. We suggest to not touch this option unless you know what you are doing.

Copyright © Sipwise GmbH - All rights reserved

- kamailio.lb.tls.connection_timeout: Sets the amount of time after which an idle TLS connection will be closed, if no I/O ever occurred after the initial open.
- kamailio.lb.tls.domains: A list of domains for which the TLS is activated tls parameters can be specified.
- kamailio.lb.tls.domains.<domain>.enable: Specify if the defined Server or Client TLS domain profiles have to be enabled or not.
- kamailio.lb.tls.domains.<domain>.type: Specify of the connaction type: server or client. A server TLS domain profile starts with [server:domain]. A client TLS domain profile starts with [client:domain].
- kamailio.lb.tls.domains.<domain>.domain: Server or Client TLS domain profile name. The 'domain' part can be: 'ip:port' - the IP address and port to match with the TLS connection; 'default' - (static string) for client and server profiles to be used when no other profile is matched; 'any' - (static string) for client and server profiles to be matched based on 'server_name', regardless TLS connection IP and port. There can be only one of each [server:default] and [client:default] profile definitions.
- kamailio.lb.tls.domains.<domain>.description: Description name of the TLS connection to simplify future debugging.
- kamailio.lb.tls.domains.<domain>.verify_certificate: Specify if Kamailio will force certificate verification when connecting to other SIP servers.
- kamailio.lb.tls.domains.<domain>.require_certificate: Specify if Kamailio will require a certificate from a client connecting to the TLS port.
- kamailio.lb.tls.domains.<domain>.tls_method: Specify the TLS protocol method.
- kamailio.lb.tls.domains.<domain>.ca_list: Specify the path of file containing a list of all trusted CAs certs.
- kamailio.lb.tls.domains.<domain>.cipher_list: Specify the list of supported SSL ciphers.
- kamailio.lb.tls.domains.<domain>.sslcertfile: Specify the SSL certificate to be used for the TLS handshake.
- kamailio.lb.tls.domains.<domain>.sslcertkeyfile: Specify the SSL certificate key to be used for the TLS handshake.
- kamailio.lb.tls.domains.<domain>.server_name: Sets the SNI (Server Name Indication) used for outbound connections. This is a TLS extension enabling one TLS server to serve multiple host names with unique certificates.
- kamailio.lb.tls.domains.<domain>.server_name_mode: Specify how to match the 'server_name' (SNI). If set to 1, then match the domain and all its subdomains. If set to 2, then match only the subdomains. If set to 0 (or anything else), then match only the domain given in 'server_name'.
- kamailio.lb.tls.domains.<domain>.server_id: The value for 'server_id' being used to match the TLS client config profile overriding the match on 'ip:port' and 'server_name'. It is currently not used in the kamailio configuration.
- kamailio.lb.udp_children: Number of UDP worker processes.
- kamailio.proxy.allow_cf_to_itself: Specify whether or not a Call Forward to the same subscriber (main number to an alias or viceversa) is allowed. To stop the CF loop a source number or a b-number have to be defined in the CF configuration.
- kamailio.proxy.allow_info_method: Allow INFO method.
- kamailio.proxy.allow_msg_method: Allow MESSAGE method.
- kamailio.proxy.allow_peer_relay: Allow peer relay. Call coming from a peer that does not match a

Copyright © Sipwise GmbH - All rights reserved

local subscriber will try to go out again, matching the peering rules.

- kamailio.proxy.allow_refer_method: Allow REFER method. Enable it with caution.
- kamailio.proxy.always_anonymize_from_user: Enable anonymization of full From URI (as opposed to only From Display-name part by default), has same effect as enabling the preference `anonymize_from_user` for all peers.
- kamailio.proxy.authenticate_bye: Enable BYE authentication.
- kamailio.proxy.cf_depth_limit: CF loop detector. How many CF loops are allowed before drop the call.
- kamailio.proxy.cfgt: Enable/disable unit test config file execution tracing.
- kamailio.proxy.check_prev_forwarder_as_upn: Enable/disable validation of the forwarder's number taken from the `Diversion` or `History-Info` header.
- kamailio.proxy.children: Number of UDP worker processes.
- kamailio.proxy.decode_utu_header: Default 'no'. If set to 'yes', the content of the User-to-User field received in 200Ok is decoded and saved in a dedicated field of the ACC records. The decoding consists in few steps: discard everything after the first occurrence of ';', remove the initial '04', hex decode the remaining part.
- kamailio.proxy.debug.enable: Enable per-module debug options.
- kamailio.proxy.debug.modules: List of modules to be traced with respective debug level.
- kamailio.proxy.debug_level: Default debug level for `kamailio-proxy`.
- kamailio.proxy.default_expires: Default expires value in seconds for a new registration (for REGISTER messages that contains neither Expires HFs nor expires contact parameters).
- kamailio.proxy.default_expires_range: This parameter specifies that the expiry used for the registration should be randomly chosen in a range given by `default_expires` +/- `default_expires_range` percent. For instance, if default_expires is 1200 seconds and default_expires_range is 50, the expiry is randomly chosen between [600,1800] seconds. If set to '0', default_expires is left unmodified.
- kamailio.proxy.dlg_timeout: Dialog timeout in seconds (by default 43200 sec - 12 hours).
- kamailio.proxy.early_rejects: Customize here the response codes and sound prompts for various reject scenarios. See the subchapter Configuring Early Reject Sound Sets for a detailed description.
- kamailio.proxy.emergency_prioritization.enable: Enable an emergency mode support.
- kamailio.proxy.emergency_prioritization.register_fake_200: When enabled, generates a fake 200 response to REGISTER from non-prioritized subscriber in emergency mode.
- kamailio.proxy.emergency_prioritization.register_fake_expires: Expires value for the fake 200 response to REGISTER.
- kamailio.proxy.emergency_prioritization.reject_code: Reject code for the non-emergency request.
- kamailio.proxy.emergency_prioritization.reject_reason: Reject reason for the non-emergency request.
- kamailio.proxy.emergency_prioritization.retry_after: Retry-After value when rejecting the non-emergency request.

TIP In order to learn about details of *emergency priorization* function of NGCP please refer to Emergency Priorization part of the handbook.

Copyright © Sipwise GmbH - All rights reserved

- kamailio.proxy.enable: Enable/disable kamailio-proxy service.
- kamailio.proxy.enum_suffix: Sets ENUM suffix - do not forget '.' (dot).
- kamailio.proxy.expires_range: Set randomization of expires for REGISTER messages (similar to default_expires_range but applies to received expires value).
- kamailio.proxy.filter_100rel_from_supported: Enable filtering of '100rel' from Supported header, to disable PRACK.
- kamailio.proxy.filter_failover_response: Specify the list of SIP responses that trigger a failover on the next available peering server.
- kamailio.proxy.foreign_domain_via_peer: Enable/disable of routing of calls to foreign SIP URI via peering servers.
- kamailio.proxy.fritzbox.enable: Enable detection for Fritzbox special numbers. Ex. Fritzbox add some prefix to emergency numbers.
- kamailio.proxy.fritzbox.prefixes: Fritzbox prefixes pattern to check. Ex. '0$avp(caller_ac)'.
- kamailio.proxy.fritzbox.special_numbers: Specifies Fritzbox special number patterns. They will be checked with the prefixes defined. Ex. '112', so the performed check will be 'sip:0$avp(caller_ac)112@' if prefix is '0$avp(caller_ac)'.
- kamailio.proxy.ignore_auth_realm: Ignore SIP authentication realm.
- kamailio.proxy.ignore_subscriber_allowed_clis: Set to 'yes' to ignore the subscriber's `allowed_clis` preference so that the User-Provided CLI is only checked against customer's `allowed_clis` preference.
- kamailio.proxy.lcr_stopper_mode: 0, default mode first rule to match will stop the gw matching process. 1, lcr will keep matching gws even if the rule has stopper value and after ordering gws by priority it will obey the first stopper value and discard the rest.
- kamailio.proxy.latency_limit_action: Limit of runtime in ms for config actions. If a config action executed by cfg interpreter takes longer than this value, a message is printed in the logs.
- kamailio.proxy.latency_limit_db: Limit of runtime in ms for DB queries. If a DB operation takes longer than this value, a warning is printed in the logs.
- kamailio.proxy.latency_log_level: Log level to print the messages related to latency. Default is 1 (INFO).
- kamailio.proxy.latency_runtime_action: Limit of runtime in ms for SIP message processing cycle. If the SIP message processing takes longer than this value, a warning is printed in the logs.
- kamailio.proxy.keep_original_to: Not used now.
- kamailio.proxy.lnp.add_reply_headers.enable: Enable/disable dedicated headers to be added after LNP lookup.
- kamailio.proxy.lnp.add_reply_headers.number: Name of the header that will contain the LNP number.
- kamailio.proxy.lnp.add_reply_headers.status: Name of the header that will contain the LNP return code (200 if OK, 500/480/... if an error/timeout is occurred).
- kamailio.proxy.lnp.api.add_caller_cc_to_lnp_dst: Enable/disable adding of caller country code to LNP routing number of the result ('no' by default, LNP result in E.164 format is assumed).
- kamailio.proxy.lnp.api.invalid_lnp_routing_codes [only for `api` type]: number matching pattern for routing numbers that represent invalid call destinations; an announcement is played in that case and the call is dropped.

Copyright © Sipwise GmbH - All rights reserved

- kamailio.proxy.lnp.api.keepalive_interval: Not used now.
- kamailio.proxy.lnp.api.lnp_request_whitelist [only for `api` type]: list of matching patterns of called numbers for which LNP lookup must be done.
- kamailio.proxy.lnp.api.lnp_request_blacklist [only for `api` type]: list of matching patterns of called numbers for which LNP lookup must not be done.
- kamailio.proxy.lnp.api.port: Not used now.
- kamailio.proxy.lnp.api.reply_error_on_lnp_failure: Specifies whether platform should drop the call in case of LNP API server failure or continue routing the call to the original callee without LNP.
- kamailio.proxy.lnp.api.request_timeout [only for `api` type]: timeout in milliseconds while Proxy waits for the response of an LNP query from *Sipwise LNP daemon*.
- kamailio.proxy.lnp.api.server: Not used now.
- kamailio.proxy.lnp.api.tcap_field_fci: path of the FCI INFO in the received tcap message
- kamailio.proxy.lnp.api.tcap_field_lnp: path of the LNP NUMBER in the received tcap/inap message
- kamailio.proxy.lnp.api.tcap_field_opcode: path of the FCI OPCODE in the received tcap message
- kamailio.proxy.lnp.enable: Enable/disable LNP (local number portability) lookup during call setup.
- kamailio.proxy.lnp.execute_ncos_block_out_before_lnp: if set to 'yes', the NCOS and BLOCK_OUT checks will be executed before the LNP lookup. Default is 'no', therefore the check are done after the LNP evaluation and rewriting.
- kamailio.proxy.lnp.skip_callee_lnp_lookup_from_any_peer: if set to 'yes', the destination LNP lookup is skipped (has same effect as enabling preference skip_callee_lnp_lookup_from_any_peer for all peers).
- kamailio.proxy.lnp.strictly_check_ncos: specify whether the NCOS LNP should be evaluated even if the LNP lookup was not previously executed or if it did not return any occurrence. If set to *yes*, a whitelist NCOS will fail if the LNP lookup does not return any match. The parameter has no impact on blacklist NCOS.
- kamailio.proxy.lnp.type: method of LNP lookup; valid values are: `local` (local LNP database) and `api` (LNP lookup through external gateways). *PLEASE NOTE:* the `api` type of LNP lookup is only available for Sipwise C5 PRO / CARRIER installations.
- kamailio.proxy.lookup_peer_destination_domain_for_pbx: one of [yes, no, peer_host_name] - Sets the content of destination_domain CDR field for calls between CloudPBX subscribers. In case of 'no' this field contains name of CloudPBX domain; 'yes': peer destination domain; 'peer_host_name': human-readable name of the peering server.
- kamailio.proxy.loop_detection.enable: Enable the SIP loop detection based on the combination of SIP URI, To and From header URIs.
- kamailio.proxy.loop_detection.expire: Sampling interval in seconds for the incoming INVITE requests (by default 1 sec).
- kamailio.proxy.loop_detection.max: Maximum allowed number of SIP requests with the same SIP URI, To and From header URIs within sampling interval. Requests in excess of this limit will be rejected with 482 Loop Detected response.
- kamailio.proxy.max_expires: Sets the maximum expires in seconds for registration. If set to '0', the check is disabled.
- kamailio.proxy.max_gw_lcr: Defines the maximum number of gateways in lcr_gw table
- kamailio.proxy.max_registrations_per_subscriber: Sets the maximum registration per subscribers.

Copyright © Sipwise GmbH - All rights reserved

- kamailio.proxy.mem_log: Specifies on which log level the memory statistics will be logged.
- kamailio.proxy.mem_summary: Parameter to control printing of memory debugging information on exit or SIGUSR1 to log.
- kamailio.proxy.min_expires: Sets the minimum expires in seconds for registration. If set to '0', the check is disabled.
- kamailio.proxy.nathelper.sipping_from: Set the From header in OPTIONS NAT ping.
- kamailio.proxy.nathelper_dbro: Default is "no". This will be "yes" on CARRIER in order to activate the use of a read-only connection using LOCAL_URL
- kamailio.proxy.natping_interval: Sets the NAT ping interval in seconds.
- kamailio.proxy.natping_processes: Set the number of NAT ping worker processes.
- kamailio.proxy.nonce_expire: Nonce expire time in seconds.
- kamailio.proxy.pbx.hunt_display_fallback_format: Default is '[H %s]'. Sets the format of the hunt group indicator that is sent as initial part of the From Display Name when subscriber is called as a member of PBX hunt group if the preferred format defined by the `hunt_display_format` and `hunt_display_indicator` cannot be used (as in the case of not provisioned subscriber settings). The '%s' part is replaced with the value of the `hunt_display_fallback_indicator` variable.
- kamailio.proxy.pbx.hunt_display_fallback_indicator: The internal kamailio variable that sets the number or extension of the hunt group. Default is `$var(cloud_pbx_hg_ext)` which is populated during call routing with the extension of the hunt group.
- kamailio.proxy.pbx.hunt_display_format: Default is '[H %s]'. Sets the format of hunt group indicator that is sent as initial part of the From Display Name when subscriber is called as a member of PBX hunt group. This is the preferred (default) indicator format with Display Name, where the '%s' part is replaced with the value of the `hunt_display_indicator` variable.
- kamailio.proxy.pbx.hunt_display_indicator: The internal kamailio variable that contains the preferred identifier of the hunt group. Default is `$var(cloud_pbx_hg_displayname)` which is populated during call routing with the provisioned Display Name of the hunt group.
- kamailio.proxy.pbx.hunt_display_maxlength: Default is '8'. Sets the maximum length of the variable used as the part of hunt group indicator in Display Name. The characters beyond this limit are truncated in order for hunt group indicator and calling party information to fit on display of most phones.
- kamailio.proxy.pbx.ignore_cf_when_hunting: Default is 'no'. Whether to disregard all individual call forwards (CFU, CFB, CFT and CFNA) of PBX extensions when they are called via hunt groups. Note that call forwards configured to local services such as Voicebox or Conference are always skipped from group hunting.
- kamailio.proxy.peer_probe.enable: Enable the peer probing, must be also checked per individual peer in the panel/API.
- kamailio.proxy.peer_probe.interval: Peer probe interval in seconds.
- kamailio.proxy.peer_probe.timeout: Peer probe response wait timeout in seconds.
- kamailio.proxy.peer_probe.reply_codes: Defines the response codes that are considered successful response to the configured probe request, e.g. `class=2;class=3;code=403;code=404;code=405`, with class defining a code range.
- kamailio.proxy.peer_probe.unavailable_treshold: Defines after how many failed probes a peer is considered unavailable.

Copyright © Sipwise GmbH - All rights reserved

- kamailio.proxy.peer_probe.available_treshold: Defines after how many successful probes a peer is considered available.
- kamailio.proxy.peer_probe.from_uri_user: From-userpart for the probe requests.
- kamailio.proxy.peer_probe.from_uri_domain From-hostpart for the probe requests.
- kamailio.proxy.peer_probe.method: [OPTIONS|INFO] - Request method for probe request.

TIP You can find more information about peer probing configuration in Configuration of Peer Probing of the handbook.

- kamailio.proxy.perform_peer_failover_on_tm_timeout: Specifies the failover behavior when maximum ring timeout (fr_inv_timer) has been reached. In case it is set to 'yes': failover to the next peer if any; in case of 'no' stop trying other peers.
- kamailio.proxy.perform_peer_lcr: Enable/Disable Least Cost Routing based on peering fees.
- kamailio.proxy.pkg_mem: PKG memory used by Kamailio Proxy.
- kamailio.proxy.shm_mem: Shared memory used by Kamailio Proxy.
- kamailio.proxy.port: SIP listening port.
- kamailio.proxy.presence.enable: Enable/disable presence feature
- kamailio.proxy.presence.max_expires: Sets the maximum expires value for PUBLISH/SUBSCRIBE message. Defines expiration of the presentity record.
- kamailio.proxy.presence.reginfo_domain: Set FQDN of Sipwise C5 domain used in callback for mobile push.
- kamailio.proxy.push.apns_alert: Set the content of 'alert' field towards APNS.
- kamailio.proxy.push.apns_sound: Set the content of 'sound' field towards APNS.
- kamailio.proxy.push.code_18x: code to be sent if reply_18x is 'yes'. Default: 180.
- kamailio.proxy.push.reason_18x: reason phrase to be sent if reply_18x is 'yes'. Default: 'Ringing'.
- kamailio.proxy.push.reply_18x: If set to 'yes' proxy will send a 18x message using code_18x and reason_18x config values after sending the PUSH notification. So caller will hear a fake ringing while the app is wakening.
- kamailio.proxy.report_mos: Enable MOS reporting in the log file.
- kamailio.proxy.set_ruri_to_peer_auth_realm: Set R-URI using peer auth realm.
- kamailio.proxy.stir.cache_dir: A path to the directory where to store cached public keys. This directory must be r/w for kamailio user.
- kamailio.proxy.stir.cache_expire: An interval in seconds after which a cached public key is considered expired.
- kamailio.proxy.stir.domains: A list of domains for which STIR is enabled, includes the 'name' - domain name (FQDN), the 'private_key' - a path to the private key.
- kamailio.proxy.stir.enable: Enable or disable STIR/SHAKEN support in Sipwise C5.
- kamailio.proxy.stir.libopt: Optional, set a libsecsipid option. The value has to be a list of options: *name=value*.
- kamailio.proxy.stir.shaken: A sub-block of options related to a treatment of incoming calls (mostly is used to define what are the values to be used in PAI header). For now Sipwise C5 only controls with it an optional parameter 'verstat' for the PAI header.

 Copyright © Sipwise GmbH - All rights reserved

- kamailio.proxy.stir.timeout: An interval in seconds after which the HTTP GET operation to download the public key times out.
- kamailio.proxy.skip_pbx_loop: Enable or disable the sems pbx loop for pbx subscribers
- kamailio.proxy.store_recentcalls: Store recent calls to redis (used by Malicious Call Identification application and VSCs related to recent calls redial).
- kamailio.proxy.syslog_options: Enable/disable logging of SIP OPTIONS messages to `kamailio-options-proxy.log`.
- kamailio.proxy.tcp_children: Number of TCP worker processes.
- kamailio.proxy.tm.fr_inv_timer: Set INVITE transaction timeout per branch if no final reply for an INVITE arrives after a provisional message was received (branch ringing timeout).
- kamailio.proxy.tm.fr_timer: Set INVITE transaction timeout if the destination is not responding with provisional response message.
- kamailio.proxy.tm.max_inv_lifetime: Set INVITE transaction timeout per the whole transaction if no final reply for an INVITE arrives after a provisional message was received (whole transaction ringing timeout). It has to be equals or greater than kamailio.proxy.tm.fr_inv_timer.
- kamailio.proxy.treat_600_as_busy: Enable the 6xx response handling according to RFC3261. When enabled, the 6xx response should stop the serial forking. Also, CFB will be triggered or busy prompt played as in case of 486 Busy response.
- kamailio.proxy.use_enum: Enable/Disable ENUM feature.
- kamailio.proxy.usrloc_dbmode: Set the mode of database usage for persistent contact storage.
- kamailio.proxy.voicebox_first_caller_cli: When enabled the previous forwarder's CLI will be used as caller CLI in case of chained Call Forwards.
- kamailio.proxy.xfer_other_party_from: If set to 'yes' transferred calls will have the number of the transferred party in the From header. Default is 'no', thus transferred calls have the number of the transferrer party in the From header.

ngcp-mediator

The following is the ngcp-mediator section:

```
mediator:
  interval: 10
```

- mediator.interval: Running interval of *ngcp-mediator*.

modules

The following is the modules section:

```
modules:
  - enable: no
    name: dummy
    options: numdummies=2
```

Copyright © Sipwise GmbH - All rights reserved

- modules: list of configs needed for load kernel modules on boot.
- enable: Enable/disable loading of the specific module (yes/no)
- name: kernel module name
- options: kernel module options if needed

monitoring

The following is the monitoring section:

```
monitoring:
  backend: prometheus
  filesystems:
  - /
  - /ngcp-data
  - /ngcp-fallback
  - /mnt/glusterfs
  interval: 10
  retention_policy_long_duration: 12
  retention_policy_short_duration: 15
  retrospect_interval: 30
  threshold:
    cpu_idle_min: '0.1'
    disk_used_max: '0.9'
    kamailio_lb_pkgmem_min: 1048576
    kamailio_lb_shmem_min: '1048576'
    kamailio_proxy_pkgmem_min: 1048576
    kamailio_proxy_shmem_min: '1048576'
    load_long_max: 0.75
    load_medium_max: 0.75
    load_short_max: 1.0
    mem_used_max: 0.98
    mta_queue_len_max: '15'
    mysql_replication_delay_max: 60
    sip_responsiveness_max: '15'
    sslcert_timetoexpiry: '30'
    sslcert_whitelist: []
    swap_free_min: 0.02
  timeout: 10
```

- monitoring.backend: The monitoring implementation backend to use. Valid value is: 'prometheus' (default).
- monitoring.filesystems: The filesystem mount points to monitor.
- monitoring.interval: The number of seconds between each data gathering iteration.
- monitoring.retention_policy_long_duration: The long term retention policy for metrics in the monitoring database in months.
- monitoring.retention_policy_short_duration: The short term retention policy for metrics in the monitoring database in days.

Copyright © Sipwise GmbH - All rights reserved

- monitoring.retrospect_interval: The number of seconds to look into the past, when checking for the last value for a data point.
- monitoring.threshold.*: These settings specify the thresholds that once crossed will make various components on the system (that is *ngcp-status*, *ngcp-collective-check*, *snmpd* or *monit*) emit alarms or warnings.
- monitoring.threshold.cpu_idle_min: Sets the minimum value for CPU usage (0.1 means 10%).
- monitoring.threshold.disk_used_max: Sets the maximum value for DISK usage (0.9 means 90%).
- monitoring.threshold.kamailio_lb_pkgmem_min: Sets the minimum value for Kamailio lb package memory usage per process.
- monitoring.threshold.kamailio_lb_shmem_min: Sets the minimum value for Kamailio lb shared memory usage.
- monitoring.threshold.kamailio_proxy_pkgmem_min: Sets the minimum value for Kamailio proxy package memory usage per process.
- monitoring.threshold.kamailio_proxy_shmem_min: Sets the minimum value for Kamailio proxy shared memory usage.
- monitoring.threshold.load_long_max/load_long_max/load_short_max: Base values for load (long, short, medium term) that will get scaled over the node CPU count to obtain the max values (base * cpu).
- monitoring.threshold.mem_used_max: Sets the maximum value for memory usage (0.7 means 70%).
- monitoring.threshold.mta_queue_len_max: Sets the maximum value for the MTA queue length.
- monitoring.threshold.mysql_replication_delay_max: Sets the maximum MySQL replication delay in seconds.
- monitoring.threshold.sip_responsiveness_max: Sets the maximum SIP responsiveness time timeout for the SIP options.
- monitoring.threshold.sslcert_timetoexpiry: Sets the number of days before a SSL certificate expiry starts to warn.
- monitoring.threshold.sslcert_whitelist: Sets a list of SSL certificate fingerprints to whitelist from the expiry check.
- monitoring.threshold.swap_free_min: Sets the minimum value for free swap (0.5 means 50%).
- monitoring.timeout: The timeout for backend queries in seconds, after which the query is considered to have failed due to lack of responsiveness.

nginx

The following is the nginx section:

```
nginx:
  status_port: 8081
  xcap_port: 1080
```

- nginx.status_port: Status port used by nginx server
- nginx.xcap_port: XCAP port used by nginx server

Copyright © Sipwise GmbH - All rights reserved

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.

ossbss

The following is the ossbss section:

```
ossbss:
  apache:
    port: 2443
    proxyluport: 1080
    restapi:
      sslcertfile: '/etc/ngcp-panel/api_ssl/api_ca.crt'
      sslcertkeyfile: '/etc/ngcp-panel/api_ssl/api_ca.key'
    serveradmin: support@sipwise.com
    servername: "\"myserver\""
    ssl_enable: yes
    sslcertfile: '/etc/ngcp-config/shared-files/ssl/myserver.crt'
    sslcertkeyfile: '/etc/ngcp-config/shared-files/ssl/myserver.key'
  frontend: no
  htpasswd:
    -
      pass: '{SHA}w4zj3mxbmynIQ1jsUEjSkN2z2pk='
      user: ngcpsoap
  logging:
    apache:
      acc:
        facility: daemon
        identity: oss
        level: info
      err:
        facility: local7
        level: info
    ossbss:
      facility: local0
      identity: provisioning
      level: DEBUG
    web:
      facility: local0
      level: DEBUG
```


Copyright © Sipwise GmbH - All rights reserved

```
  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).

pbx (only with additional Cloud PBX module activated)

The following is the PBX section:

```
pbx:
  enable: no
```

- pbx.enable: Enable Cloud PBX module.

Copyright © Sipwise GmbH - All rights reserved

prometheus

The following is the prometheus section:

```
prometheus:
  exporter_port:
    haproxy: 9101
    mysqld: 9104
    nginx: 9113
    node: 9100
    process: 9256
    redis: 9121
  external:
    clients: []
    enable: no
  flavor: victoria-metrics
  port: 9090
```

- prometheus.exporter_port.haproxy: Prometheus haproxy exporter listening port.
- prometheus.exporter_port.mysqld: Prometheus mysqld exporter listening port.
- prometheus.exporter_port.nginx: Prometheus nginx exporter listening port.
- prometheus.exporter_port.node: Prometheus node exporter listening port.
- prometheus.exporter_port.process: Prometheus process exporter listening port.
- prometheus.exporter_port.redis: Prometheus Redis exporter listening port.
- prometheus.external.clients: List of clients in CIDR notation allowed to connect to the prometheus service on the 'mon_ext' interface when the firewall is enabled. Default is no clients, and the firewall defaults to dropping any connection.
- prometheus.external.enable: Enable/disable exposing a prometheus server implementation on the 'mon_ext' interface, for external monitoring. Defaults to 'no'.
- prometheus.flavor: The prometheus server implementation to use. Either 'victoria-metrics' (default) or 'prometheus'.
- prometheus.port: Prometheus listening port.

pushd

The following is the pushd section:

```
pushd:
  apns:
    enable: yes
    endpoint: api.push.apple.com
    endpoint_port: 0
    extra_instances:
    - certificate: '/etc/ngcp-config/shared-
files/ssl/PushCallkitCert.pem'
      enable: yes
```


Copyright © Sipwise GmbH - All rights reserved

```
      key: '/etc/ngcp-config/shared-files/ssl/PushCallkitKey.pem'
      type: callkit
    http2_jwt:
      ec_key: '/etc/ngcp-config/shared-files/ssl/AuthKey_ABCDE12345.pem'
      ec_key_id: 'ABCDE12345'
      enable: yes
      issuer: 'VWXYZ67890'
      tls_certificate: ''
      tls_key: ''
      topic: 'com.example.appID'
    legacy:
      certificate: '/etc/ngcp-config/shared-files/ssl/PushChatCert.pem'
      feedback_endpoint: feedback.push.apple.com
      feedback_interval: '3600'
      key: '/etc/ngcp-config/shared-files/ssl/PushChatKey.pem'
    socket_timeout: 0
  domains:
  - apns:
      endpoint: api.push.apple.com
      extra_instances:
      - certificate: '/etc/ngcp-config/shared-files/ssl/PushCallkitCert-
example.com.pem'
        enable: no
        key: '/etc/ngcp-config/shared-files/ssl/PushCallkitKey-
example.com.pem''
        type: callkit
      http2_jwt:
        ec_key: '/etc/ngcp-config/shared-
files/ssl/AuthKey_54321EDCBA.pem'
        ec_key_id: '54321EDCBA'
        issuer: '09876ZYXWV'
        tls_certificate: ''
        tls_key: ''
        topic: 'com.example.otherAppID'
      legacy:
        certificate: '/etc/ngcp-config/shared-files/ssl/PushChatCert-
example.com.pem'
        feedback_endpoint: feedback.push.apple.com
        key: '/etc/ngcp-config/shared-files/ssl/PushChatKey-
example.com.pem'
    domain: example.com
    enable: yes
    android:
      key: 'google_api_key_for_example.com_here'
  enable: yes
  android:
    enable: yes
    key: 'google_api_key_here'
    priority:
      call: high
      groupchat: normal
      invite: normal
```

Copyright © Sipwise GmbH - All rights reserved

```
      message: normal
  muc:
    exclude: []
    force_persistent: 'true'
    owner_on_join: 'true'
  one_device_per_subscriber: no
  port: 45060
  processes: 4
  ssl: yes
  sslcertfile: /etc/ngcp-config/shared-files/ssl/CAsigned.crt
  sslcertkeyfile: /etc/ngcp-config/shared-files/ssl/CAsigned.key
  unique_device_ids: no
```

- pushd.enable: Enable/Disable the Push Notification feature.
- pushd.apns.enable: Enable/Disable Apple push notification.
- pushd.apns.endpoint: API endpoint hostname or address. Should be one of 'api.push.apple.com' or 'api.development.push.apple.com' for the newer HTTP2/JWT based protocol, or one of 'gateway.push.apple.com' or 'gateway.sandbox.push.apple.com' for the legacy protocol.
- pushd.apns.endpoint_port: API endpoint port. Normally 443 or alternatively 2197 for the newer HTTP2/JWT based protocol, or 2195 for the legacy protocol.
- pushd.apns.legacy: Contains all options specific to the legacy APNS protocol. Ignored when HTTP2/JWT is in use.
- pushd.apns.legacy.certificate: Specify the Apple certificate for push notification https requests from Sipwise C5 to an endpoint.
- pushd.apns.legacy.key: Specify the Apple key for push notification https requests from Sipwise C5 to an endpoint.
- pushd.apns.legacy.feedback_endpoint: Hostname or address of the APNS feedback service. Normally one of 'feedback.push.apple.com' or 'feedback.sandbox.push.apple.com'.
- pushd.apns.legacy.feedback_interval: How often to poll the feedback service, in seconds.
- pushd.apns.extra_instances: If the iOS app supports Callkit push notifications, they can be enabled here and the required separate certificate and key can be specified. Ignored if HTTP2/JWT is enabled.
- pushd.http2_jwt: Contains all options specific to the newer HTTP2/JWT based APNS API protocol.
- pushd.http2_jwt.ec_key: Name of file that contains the elliptic-curve (EC) cryptographic key provided by Apple, in PEM format.
- pushd.http2_jwt.ec_key_id: 10-digit identification string of the EC key in use.
- pushd.http2_jwt.enable: Master switch for the HTTP2/JWT based protocol. Disables the legacy protocol when enabled.
- pushd.http2_jwt.issuer: Issuer string for the JWT token. Normally the 10-digit team ID string for which the EC key was issued.
- pushd.http2_jwt.tls_certificate: Optional client certificate to use for the TLS connection.
- pushd.http2_jwt.tls_key: Optional private key for the client certificate to use for the TLS connection.
- pushd.http2_jwt.topic: Topic string for the JWT token. Normally the bundle ID for the iOS app.

Copyright © Sipwise GmbH - All rights reserved

- pushd.android.enable: Enable/Disable Google push notification.
- pushd.android.key: Specify the Google key for push notification https requests from Sipwise C5 to an endpoint.
- pushd.domains: Supports a separate set of push configurations (API keys, certificates, etc.) for all subscribers of the given domain.
- pushd.muc.exclude: list of MUC room jids excluded from sending push notifications.
- pushd.muc.force_persistent: Enable/Disable MUC rooms to be persistent. Needed for Sipwise C5 app to work with other clients.
- pushd.muc.owner_on_join: Enable/Disable all MUC participants to be owners of the MUC room. Needed for Sipwise C5 app to work with other clients.
- pushd.ssl: The security protocol Sipwise C5 uses for https requests from the app in the push notification process.
- pushd.sslcertfile: The trusted certificate file purchased from a CA
- pushd.sslcertkeyfile: The key file that purchased from a CA
- pushd.unique_device_ids: Allows a subscriber to register the app and have the push notification enabled on more than one mobile device.

qos

The QoS section allows configuring the ToS (Type of Service) feature:

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

- qos.tos_rtp: a ToS value for RTP traffic.
- qos.tos_sip: a ToS value for SIP traffic.

TIP The ToS byte includes both DSCP and ECN bits. So, specify the DSCP value multiplied by four (46x4=184) and, optionally, add the required ECN value to it (1, 2 or 3).

Set the rtpengine.control_tos parameter higher than zero to enable ToS.

ngcp-rate-o-mat

The following is the *ngcp-rate-o-mat* section:

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

- rateomat.enable: Enable/Disable *ngcp-rate-o-mat*
- rateomat.loopinterval: How long we shall sleep before looking for unrated CDRs again.

Copyright © Sipwise GmbH - All rights reserved

- rateomat.splitpeakparts: Whether we should split CDRs on peaktime borders.

ngcp-task-agent

The following is the ngcp-task-agent section:

```
ngcp-task-agent:
  enable: yes
```

- ngcp-task-agent.enable: Enable/Disable *ngcp-task-agent*.

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 has to try to call you.
- reminder.retry_time: Seconds between retries.
- reminder.wait_time: Seconds to wait for an answer.

rsyslog

The following is the rsyslog section:

```
rsyslog:
  external_logging:
  - address: 192.168.32.1
    enable: no
    loglevel: warning
    port: '514'
    proto: udp
    tls:
      enable: no
  ngcp_logs_max_size: 2G
  ngcp_logs_preserve_days: '93'
```

- rsyslog.external_logging: List of remote syslog servers.
- rsyslog.external_logging.address: Address of this remote syslog server.
- rsyslog.external_logging.enable: Enable or disable this remote syslog destination.

Copyright © Sipwise GmbH - All rights reserved

- rsyslog.external_logging.loglevel: Minimum log level to send to this syslog destination.
- rsyslog.external_logging.port: Port of this remote syslog server.
- rsyslog.external_logging.proto: Protocol (udp or tcp) for this remote syslog server.
- rsyslog.external_logging.tls.enable: Enable or disable TLS for remote syslog connection.
- rsyslog.external_logging.tls.auth_mode:Specify the authentication mode, as described in IETF's draft-ietf-syslog-transport-tls-12 Internet draft. Default is 'anon'.
- rsyslog.external_logging.tls.auth_peers: Specify the fingerprints for the permitted peers for the x509/fingerprint auth_mode. The fingerprint must be provided as the SHA1 or the SHA256 hex string of the certificate. Multiple values must be separated by comma (,).
- rsyslog.external_logging.tls.ca_certfile: Specify the CA certificate file for the remote syslog server.
- rsyslog.external_logging.tls.keyfile: Specify the client key file for the remote syslog server.
- rsyslog.external_logging.tls.certfile: Specify the client certificate file for the remote syslog server.
- rsyslog.ngcp_logs_max_size: Specify a maximum size for log files before they are rotated away.
- rsyslog.ngcp_logs_preserve_days: Specify how many days to preserve old rotated log files in /var/log/ngcp/old path.

rtpengine

The following is the rtp proxy section:

```
rtpengine:
  allow_userspace_only: yes
  cdr_logging_facility: ''
  control_tos: 0
  delete_delay: 30
  dtls_passive: no
  enable: yes
  final_timeout: 0
  firewall_iptables_chain: ''
  graphite:
    interval: 600
    prefix: rtpengine.
    server: ''
  log_level: '6'
  maxport: '40000'
  minport: '30000'
  num_threads: 0
  prefer_bind_on_internal: no
  recording:
    add_header_for_external_callrecording: ''
    egress: no
    enable: no
    lightweight_lawful_intercept:
      enable: no
      recording_pattern: lwli-%{uuid}-%{callee}-%Y%m%d%H%M%S
    log_level: 6
    mix_method: direct
```

Copyright © Sipwise GmbH - All rights reserved

```
    mp3_bitrate: '48000'
    nfs_host: 192.168.1.1
    nfs_remote_path: /var/recordings
    output_chgrp: _ngcp-admin
    output_chmod: '0660'
    output_chmod_dir: '0770'
    output_chown: root
    output_dir: /var/lib/rtpengine-recording
    output_format: wav
    output_mixed: yes
    output_pattern: '%c-%r-%t'
    output_single: yes
    resample: no
    resample_to: '16000'
    spool_dir: /var/spool/rtpengine
    storage_db: no
    storage_file: yes
    storage_local: no
    vsc:
      pause_recording: '555'
      pause_resume_recording: '666'
      start_pause_resume_recording: '777'
      start_recording: '888'
      start_stop_recording: '999'
      stop_recording: '000'
  rtcp_logging_facility: ''
  rtp_timeout: '60'
  rtp_timeout_onhold: '3600'
```

- rtpengine.allow_userspace_only: Enable/Disable the user space failover for rtpengine ('yes' means enable). By default rtpengine works in kernel space.
- rtpengine.cdr_logging_facility: If set, rtpengine will produce a CDR-like syslog line after each call finishes. Must be set to a valid syslog facility string (such as 'daemon' or 'local0').
- rtpengine.control_tos: If higher than 0, the control messages port uses the configured ToS (Type of Service) bits. See the QoS section below for details.
- rtpengine.delete_delay: After a call finishes, rtpengine will wait this many seconds before cleaning up resources. Useful for possible late branched calls.
- rtpengine.dtls_passive: If enabled, rtpengine will always advertise itself as a passive role in DTLS setup. Useful in WebRTC scenarios if used behind NAT.
- rtpengine.final_timeout: If set, any calls lasting longer than this many seconds will be terminated, no matter the circumstances.
- rtpengine.firewall_iptables_chain: If set, rtpengine will create an iptables rule for each individual media port opened in this chain.
- rtpengine.graphite.interval: Interval in seconds between sending updates to the Graphite server.
- rtpengine.graphite.prefix: Graphite keys will be prefixed with this string. Must include a separator character (such as a trailing dot) if one should be used.
- rtpengine.graphite.server: Graphite server to send periodic statistics updates to. Disabled if set to an

Copyright © Sipwise GmbH - All rights reserved

empty string. Must be in format 'IP:port' or 'hostname:port'.

- rtpengine.log_level: Verbosity of log messages. The default '6' logs everything except debug messages. Increase to 7 to log everything, or decrease to make logging more quiet.
- rtpengine.maxport: Maximum port used by rtpengine for RTP traffic.
- rtpengine.minport: Minimum port used by rtpengine for RTP traffic.
- rtpengine.num_threads: Number of worker threads to use. If set to 0, the number of CPU cores will be used.
- rtpengine.recording.add_header_for_external_callrecording: Specify the header to add for external Call Recording.
- rtpengine.recording.egress: Record egress (transmitted) audio streams instead of ingress (received).
- rtpengine.recording.enable: Enable support for call recording.
- rtpengine.recording.mp3_bitrate: If saving audio as MP3, bitrate of the output file.
- rtpengine.recording.log_level: Same as log_level above, but for the recording daemon.
- rtpengine.recording.nfs_host: Mount an NFS share from this host for storage.
- rtpengine.recording.nfs_remote_path: Remote path of the NFS share to mount.
- rtpengine.recording.output_chgrp: Change the file ownership to the given group name or group ID.
- rtpengine.recording.output_chmod: Change the mode of created files (in octal). Default is octal 0660. Can be set to 0 to disable and leave the mode to the umask.
- rtpengine.recording.output_chmod_dir: Change the mode of created directories (in octal). Default is octal 0770. Can be set to 0 to disable and leave the mode to the umask.
- rtpengine.recording.output_chown: Change the file ownership to the given user name or user ID.
- rtpengine.recording.output_dir: Local mount point for the NFS share.
- rtpengine.recording.output_format: Either 'wav' for PCM output or 'mp3'.
- rtpengine.recording.output_mixed: Create output audio files with all contributing audio streams mixed together.
- rtpengine.recording.mix_method: Mix all audio inputs together directly, or output them as individual channels.
- rtpengine.recording.output_single: Create separate audio files for each contributing audio stream.
- rtpengine.recording.output_pattern: File name pattern for created recording files. Defaults to '%c-%t' to include call ID and recording type.
- rtpengine.recording.resample: Resample all audio to a fixed bitrate ('yes' or 'no').
- rtpengine.recording.resample_to: If resampling is enabled, resample to this sample rate.
- rtpengine.recording.spool_dir: Local directory for temporary metadata file storage.
- rtpengine.recording.storage_file: Enable or disable storing of recordings as files.
- rtpengine.recording.storage_db: Enable or disable storing of recordings in database storage. EXPERIMENTAL MODE OF OPERATION.
- rtpengine.recording.storage_local: Enable or disable storing of recordings as local (non-NFS) files. UNSUPPORTED MODE OF OPERATION.
- rtpengine.recording.lightweight_lawful_intercept.enable: Enable or disable the Lightweight Lawful Interception. This will disable the standard call recording feature for end users.

Copyright © Sipwise GmbH - All rights reserved

- rtpengine.recording.lightweight_lawful_intercept.recording_pattern: File name pattern for created Lightweight Lawful Interception recording files. Defaults to 'lwli-%{uuid}-%{callee}-%Y%m%d%H%M%S' to include caller, callee, date and time.
- rtpengine.recording.vsc.start_recording: DTMF VSC to start recording if not already started or resume if paused.
- rtpengine.recording.vsc.stop_recording: DTMF VSC to stop recording if currently active or paused.
- rtpengine.recording.vsc.pause_recording: DTMF VSC to pause recording if currently active.
- rtpengine.recording.vsc.start_stop_recording: DTMF VSC to start recording if currently active or stop if running or paused.
- rtpengine.recording.vsc.pause_resume_recording: DTMF VSC to pause recording if currently active or resume if paused.
- rtpengine.recording.vsc.start_pause_resume_recording: DTMF VSC to start recording if not already started, pause if running, or resume if paused.
- rtpengine.rtcp_logging_facility: If set, rtpengine will write the contents of all received RTCP packets to syslog. Must be set to a valid syslog facility string (such as 'daemon' or 'local0').
- rtpengine.rtp_timeout: Consider a call dead if no RTP is received for this long (60 seconds).
- rtpengine.rtp_timeout_onhold: Maximum limit in seconds for an onhold (1h).

security

The following is the security section. Usage of the firewall subsection is described in Firewalling:

```
security:
  firewall:
    enable: no
    logging:
      days_kept: '7'
      enable: yes
      file: /var/log/firewall.log
      tag: NGCPFW
    nat_rules4: ~
    nat_rules6: ~
    policies:
      forward: DROP
      input: DROP
      output: ACCEPT
    rules4: ~
    rules6: ~
```

- security.firewall.enable: Enable/disable iptables configuration and rule generation for IPv4 and IPv6 (default: no)
- security.firewall.logging.days_kept: Number of days logfiles are kept on the system before being deleted (log files are rotated daily, default: 7)
- security.firewall.logging.enable: Enables/disables logging of all packets dropped by Sipwise C5 firewall (default: yes)

Copyright © Sipwise GmbH - All rights reserved

- security.firewall.logging.file: File firewall log messages go to (default: /var/log/firewall.log)
- security.firewall.logging.tag: String prepended to all log messages (internally DROP is added to any tag indicating the action triggering the message, default: NGCPFW)
- security.firewall.nat_rules4: Optional list of IPv4 firewall rules added to table nat using iptables-persistent syntax (default: undef)
- security.firewall.nat_rules6: Optional list of IPv6 firewall rules added to table nat using iptables-persistent syntax (default: undef)
- security.firewall.policies.forward: Default policy for iptables FORWARD chain (default: DROP)
- security.firewall.policies.input: Default policy for iptables INPUT chain (default: DROP)
- security.firewall.policies.output: Default policy for iptables OUTPUT chain (default: ACCEPT)
- security.firewall.rules4: Optional list of IPv4 firewall rules added to table filter using iptables-persistent syntax (default: undef)
- security.firewall.rules6: Optional list of IPv6 firewall rules added to table filter using iptables-persistent syntax (default: undef)

sems

The following is the SEMS section:

```
sems:
  bindport: 5080
  conference:
    enable: yes
    max_participants: 10
  debug: no
  highport: 50000
  lowport: 40001
  media_processor_threads: 10
  prepaid:
    enable: yes
  sbc:
    calltimer_enable: yes
    calltimer_max: 3600
    outbound_timeout: 6000
    profile:
    - custom_header: []
      name: ngcp
    - custom_header: []
      name: ngcp_cf
    sdp_filter:
      codecs: PCMA,PCMU,telephone-event
      enable: yes
      mode: whitelist
    session_timer:
      enable: yes
      max_timer: 7200
      min_timer: 90
      session_expires: 300
```

Copyright © Sipwise GmbH - All rights reserved

```
  session_processor_threads: 10
  vsc:
    auto_answer_code: '59'
    block_override_code: '80'
    blockinclir_code: '32'
    callback_last_caller_code: '74'
    callback_last_caller_confirmation_key: '3'
    callback_last_caller_interactive_code: '71'
    cf_disable_all_code: '96'
    cfb_code: '90'
    cfna_code: '93'
    cft_code: '92'
    cfu_code: '72'
    clir_code: '31'
    directed_pickup_code: '99'
    dnd_code: '78'
    enable: yes
    force_sendrecv_state: no
    last_number_redial_code: '95'
    park_code: '97'
    recent_calls_deletion_code: '20'
    reminder_code: '55'
    speedial_code: '50'
    unpark_code: '98'
    voicemail_deposit_code: '56'
    voicemail_number: '2000'
  xmlrpcport: 8090
```

- b2b.conference.enable: Enable/Disable conference feature.
- b2b.conference.max_participants: Sets the number of concurrent participant.
- b2b.highport: Maximum ports used by sems for RTP traffic.
- b2b.debug: Enable/Disable debug mode.
- b2b.lowport: Minimum ports used by sems for RTP traffic.
- b2b.prepaid.enable: Enable/Disable prepaid feature.
- b2b.sbc.calltimer_max: Set the default maximum call duration. Note that this value can be overwritten in subscriber/customer/domain preferences setting max_call_duration parameter. Attention: in case of call transfer done by the callee, with max_call_duration set, the timer will be restarted from 0 for the new transferred call.
- b2b.sbc.outbound_timeout: Set INVITE transaction timeout if the destination is not responding with provisional response message.
- b2b.sbc.profile.name: Profile's name where to add the custom headers in 'header_list' config parameter. Supported values: ngcp and ngcp_cf.
- b2b.sbc.profile.custom_header: List of the custom headers that has to be whitelisted (default) by sems sbc in the corresponding profile.
- b2b.sbc.session_timer.enable: If set to "no" all session timer headers are stripped off without considering the session timer related configuration done via the web interface. If set to "yes" the system uses the subscriber/peer configurations values set on the web interface. If set to

Copyright © Sipwise GmbH - All rights reserved

"transparent" no validation is performed on Session Timer headers, they are ignored by SEMS and therefore negotiated end-to-end.

- b2b.vsc.*: Define here the VSC codes.

sms

This section provides configuration of **S**hort **M**essage **S**ervice on the NGCP. Description of the SMS module is provided earlier in this handbook here.

In the below example you can see the default values of the configuration parameters.

```
sms:
  core:
    admin_port: '13000'
    smsbox_port: '13001'
  enable: no
  loglevel: '0'
  sendsms:
    max_parts_per_message: '5'
    port: '13002'
  smsc:
    dest_addr_npi: '1'
    dest_addr_ton: '1'
    enquire_link_interval: '58'
    host: 203.0.113.42
    id: default_smsc
    max_pending_submits: '10'
    no_dlr: yes
    password: password
    port: '2775'
    source_addr_npi: '1'
    source_addr_ton: '1'
    system_type: ''
    throughput: '5'
    transceiver_mode: '1'
    username: username
```

- sms.core.admin_port: Port number of admin interface of SMS core module (running on LB nodes).
- sms.core.smsbox_port: Port number used for internal communication between *bearerbox* module on LB nodes and *smsbox* module on PRX nodes. This is a listening port of the *bearerbox* module (running on LB nodes).
- sms.enable: Set to `yes` if you want to enable SMS module.
- sms.loglevel: Log level of SMS module; the default '0' will result in writing only the most important information into the log file.
- sms.sendsms.max_parts_per_message: If the SM needs to be sent as concatenated SM, this parameter sets the max. number of parts for a single (logical) message.
- sms.sendsms.port: Port number of *smsbox* module (running on PRX nodes).
- sms.smsc. : Parameters of the connection to an SMSC

Copyright © Sipwise GmbH - All rights reserved

dest_addr_npi: Telephony numbering plan indicator for the SM destination, as defined by standards (e.g. '1' stands for E.164)

dest_addr_ton: Type of number for the SM destination, as defined by standards (e.g. '1' stands for "international" format)

enquire_link_interval: Interval of SMSC link status check in seconds

host: IP address of the SMSC

id: An arbitrary string for identification of the SMSC; may be used in log files and for routing SMs.

max_pending_submits: The maximum number of outstanding (i.e. not acknowledged) SMPP operations between Sipwise C5 and SMSC. As a guideline it is recommended that no more than 10 (default) SMPP messages are outstanding at any time.

no_dlr: Do not request delivery report; when sending an SM and this parameter is set to `yes`, Sipwise C5 will not request DR for the message(s). May be required for some particular SMSCs, in order to avoid "Incorrect status report request parameter usage" error messages from the SMSC.

password: This is the password used for authentication on the SMSC.

port: Port number of the SMSC where Sipwise C5 will connect to.

source_addr_npi: Telephony numbering plan indicator for the SM source, as defined by standards (e.g. '1' stands for E.164)

source_addr_ton: Type of number for the SM source, as defined by standards (e.g. '1' stands for "international" format)

system_type: Defines the SMSC client category in which Sipwise C5 belongs to; defaults to "VMA" (Voice Mail Alert) when no value is given. (No need to set any value)

throughput: The max. number of messages per second that Sipwise C5 will send towards the SMSC. (Value type: float)

transceiver_mode: If set to `1` (yes / true), Sipwise C5 will attempt to use a TRANSCEIVER mode connection to the SMSC. It uses the standard transmit port of the SMSC for receiving SMs too.

username: This is the username used for authentication on the SMSC.

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).

sssd

The following is the sssd section:

Copyright © Sipwise GmbH - All rights reserved

```
sssd:
  enable: no
  ldap:
    host: localhost
    port: 636
    base_dn: ''
    access_filter: '(objectClass=posixAccount)'
```

- sssd.enable: Enable/disable LDAP authentication via SSSD.
- sssd.ldap.host: The LDAP hostname to use. The 'localhost' default is there for lack of a better hostname, but it is not currently set up with any account information. This would usually be set to the organization LDAP.
- sssd.ldap.port: The LDAP port to use for the ldaps protocol, that is LDAP over TLS.
- sssd.ldap.base_dn: The LDAP base DN to use for searches.
- sssd.ldap.access_filter: The access filter to apply on searches. The default setting is intended to work for POSIX accounts, but depending on how the LDAP accounts are set up and on the site policies, another filter might be needed.

sudo

The following is in the sudo section:

```
sudo:
  logging: no
  max_log_sessions: 0
```

- logging: enable/disable the I/O logging feature of sudo. See man page of 'sudoreplay(8)'.
- max_log_sessions: when I/O logging is enabled, specifies how many log sessions per individual user sudo should keep before it starts overwriting old ones. The default '0' means no limit.

ngcp-witnessd

The following is the ngcp-witnessd tool section:

```
witnessd:
  debug: no
  interval: ~
  gather:
    asr_ner_statistics: yes
    kamailio_concurrent_calls: yes
    kamailio_dialog_active: yes
    kamailio_dialog_early: yes
    kamailio_dialog_incoming: yes
    kamailio_dialog_local: yes
    kamailio_dialog_outgoing: yes
    kamailio_dialog_relay: yes
    kamailio_shmem: yes
```

Copyright © Sipwise GmbH - All rights reserved

```
    kamailio_usrloc_regdevices: yes
    kamailio_usrloc_regusers: yes
    peering_groups: yes
    mta_queue_len: yes
    mysql_global_status: yes
    mysql_slave_status: yes
    oss_provisioned_subscribers: yes
    sip_responsiveness: yes
    sip_stats_num_packets: yes
    sip_stats_num_packets_perday: yes
    sip_stats_partition_size: yes
```

- witnessd.interval: The number of seconds between each data gathering iteration, when the value is undefined, the code will fallback to use `monitoring.interval`.
- witnessd.gather.asr_ner_statistics: Enable ASR/NER statistics data.
- witnessd.gather.kamailio_*: Enable Kamailio statistics data.
- witnessd.gather.mta_queue_len: Enable MTA (exim4) queue length data.
- witnessd.gather.mysql_global_status: Enable global MySQL data.
- witnessd.gather.mysql_slave_status: Enable slave (replication) MySQL data.
- witnessd.gather.oss_provisioned_subscribers: Enable OSS provisioned subscribers count data.
- witnessd.gather.sip_*: Enable SIP statistics data.

www_admin

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

```
www_admin:
  apache:
    autoprov_port: 1444
  callingcard_features: 0
  callthru_features: 0
  conference_features: 1
  contactmail: adjust@example.org
  fastcgi_workers: 2
  fax_features: 1
  fees_csv:
    element_order:
      - source
      - destination
      - direction
      - zone
      - zone_detail
      - onpeak_init_rate
      - onpeak_init_interval
      - onpeak_follow_rate
      - onpeak_follow_interval
      - offpeak_init_rate
      - offpeak_init_interval
```


Copyright © Sipwise GmbH - All rights reserved

```
      - offpeak_follow_rate
      - offpeak_follow_interval
      - use_free_time
  http_admin:
    autoprov_port: 1444
    port: 1443
    serveradmin: support@sipwise.com
    servername: "\"myserver\""
    ssl_enable: yes
    sslcertfile: '/etc/ngcp-config/shared-files/ssl/myserver.crt'
    sslcertkeyfile: '/etc/ngcp-config/shared-files/ssl/myserver.key'
  http_csc:
    autoprov_bootstrap_port: 1445
    autoprov_port: 1444
    port: 443
    serveradmin: support@sipwise.com
    servername: "\"myserver\""
    ssl_enable: yes
    sslcertfile: '/etc/ngcp-config/shared-files/ssl/myserver.crt'
    sslcertkeyfile: '/etc/ngcp-config/shared-files/ssl/myserver.key'
  logging:
    apache:
      acc:
        facility: daemon
        identity: oss
        level: info
      err:
        facility: local7
        level: info
  security:
    admin_ldap:
      dn_format: ~
      manager_dn: ~
      manager_password: ~
      uri: localhost
    jwt:
      expires: 36000
    login:
      ban_enable: yes
      ban_increment: 300
      ban_max_time: 3600
      ban_min_time: 300
      max_attempts: 5
    password:
      allow_recovery: 0
      max_length: 40
      min_length: 12
      musthave_digit: 3
      musthave_lowercase: 3
      musthave_specialchar: 3
      musthave_uppercase: 3
      sip_autogenerate: 0
```

Copyright © Sipwise GmbH - All rights reserved

```
      sip_expose_subadmin: 1
      sip_keep_last_used: 12
      sip_validate: no
      web_autogenerate: 0
      web_expose_subadmin: 1
      web_keep_last_used: 12
      web_max_age_days: 0
      web_validate: no
    rsa_private_key_file: /etc/ngcp-config/shared-files/ngcp-
panel/rsa_private_key.pem
    rsa_public_key_file: /etc/ngcp-config/shared-files/ngcp-
panel/rsa_public_key.pem
    voucher_iv: ...
    voucher_key: ...
  speed_dial_vsc_presets:
    vsc:
      - '*0'
      - '*1'
      - '*2'
      - '*3'
      - '*4'
      - '*5'
      - '*6'
      - '*7'
      - '*8'
      - '*9'
```

- www_admin.http_admin.*: Define the Administration interface and certificates.
- www_admin.http_csc.*: Define the Customers interface and certificates.
- www_admin.contactmail: Email to show in the GUI's Error page.

constants.yml Overview

`/etc/ngcp-config/constants.yml` is one of the main configuration files that contains important (static) configuration parameters, like Sipwise C5 system-user data.

CAUTION Sipwise C5 platform administrator should not change content of `constants.yml` file unless absolutely necessary. Please contact Sipwise Support before changing any of the parameters within the `constants.yml` file!

maintenance.yml Overview

`/etc/ngcp-config/maintenance.yml` is one of the main configuration files that contains the maintenance configuration option.

general

The following is the general section:

```
general:
```


Copyright © Sipwise GmbH - All rights reserved

```
  maintenance: no
```

- general.maintenance: maintenance mode necessary for safe upgrades.

network.yml Overview

`/etc/ngcp-config/network.yml` is one of the main configuration files that contains network-related configuration parameters, like IP addresses and roles of the node(s) in Sipwise C5 system.

The next example shows a part of the `network.yml` configuration file. Explanation of all the configuration parameters is provided in Network Configuration section of the handbook.

Sample host configuration for Sipwise C5

A CE would look like:

```
  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
    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
```

Copyright © Sipwise GmbH - All rights reserved

```
      v6ip: '::1'
    role:
      - proxy
      - lb
      - mgmt
      - rtp
      - db
    status: 'online'
hosts_common:
  etc_hosts_global_extra_entries:
  - 10.100.1.1 server-1 server-1.internal.example.com
  - 10.100.1.2 server-2 server-2.internal.example.com
```

NOTE The option 'hosts_common' is optional and it allows administrator to provide extra entries in /etc/hosts.

The administrator can create new entries in network.yml to specify extra entries for the file /etc/hosts. One entry is global and two per-host, one of which is local only for the host, and the other overrides global for this host. These entries will be appended without further processing.

The example of adding new entries using 'ngcpcfg set':

```
ngcpcfg set --diff /etc/ngcp-config/network.yml \
  hosts_common.etc_hosts_global_extra_entries='["10.100.1.1 server-1
server-1.internal.example.com","10.100.1.2 server-2 server-
2.internal.example.com"]'
```

Global is useful if the entries are to be added to all hosts. These probably make more sense in most set-ups.

Per-host local is useful if the entries are only to be added to some node, but not needed or convenient to add to all of them.

Per-host override of the global config is useful if the global entries are to be added to a potentially large number of nodes and to be excluded only in a few of them, to not have to do the contrary (duplicate entries in many hosts except a couple).

Example of modifications to network.yml:

```
hosts_common:
  etc_hosts_global_extra_entries:
  - 10.100.1.1 server-1 server-1.internal.example.com
  - 10.100.1.2 server-2 server-2.internal.example.com
hosts:
  db01b:
    etc_hosts_local_extra_entries:
    - 127.0.1.1 local-alias-1.db01b
    - 127.0.2.1 local-alias-2.db01b
    - 172.30.52.180 db01b.example.com
  ...
```


Copyright © Sipwise GmbH - All rights reserved

```
web01a:
  etc_hosts_local_extra_entries:
  - 127.0.1.1 local-alias-1.web01a
  - 127.0.2.1 local-alias-2.web01a
  - 172.30.52.168 web01a.example.com
  etc_hosts_global_extra_entries:
  - 10.100.1.1 server-1 server-1.internal.example.com
...
```

With this, the output in /etc/hosts for *db01b* will be:

```
# local extra entries for host 'db01b'
127.0.1.1 local-alias-1.db01b
127.0.2.1 local-alias-2.db01b
172.30.52.180 db01b.example.com

# global extra entries
10.100.1.1 server-1 server-1.internal.example.com
10.100.2.1 server-2 server-2.internal.example.com
```

the content in /etc/hosts on *web01a* will be:

```
# local extra entries for host 'web01a'
127.0.1.1 local-alias-1.web01a
127.0.2.1 local-alias-2.web01a
172.30.52.168 web01a.example.com

# global extra entries overridden for host 'web01a'
10.100.1.1 server-1 server-1.internal.example.com
```

Copyright © Sipwise GmbH - All rights reserved

Appendix B: Basic Call Flows

General Call Setup

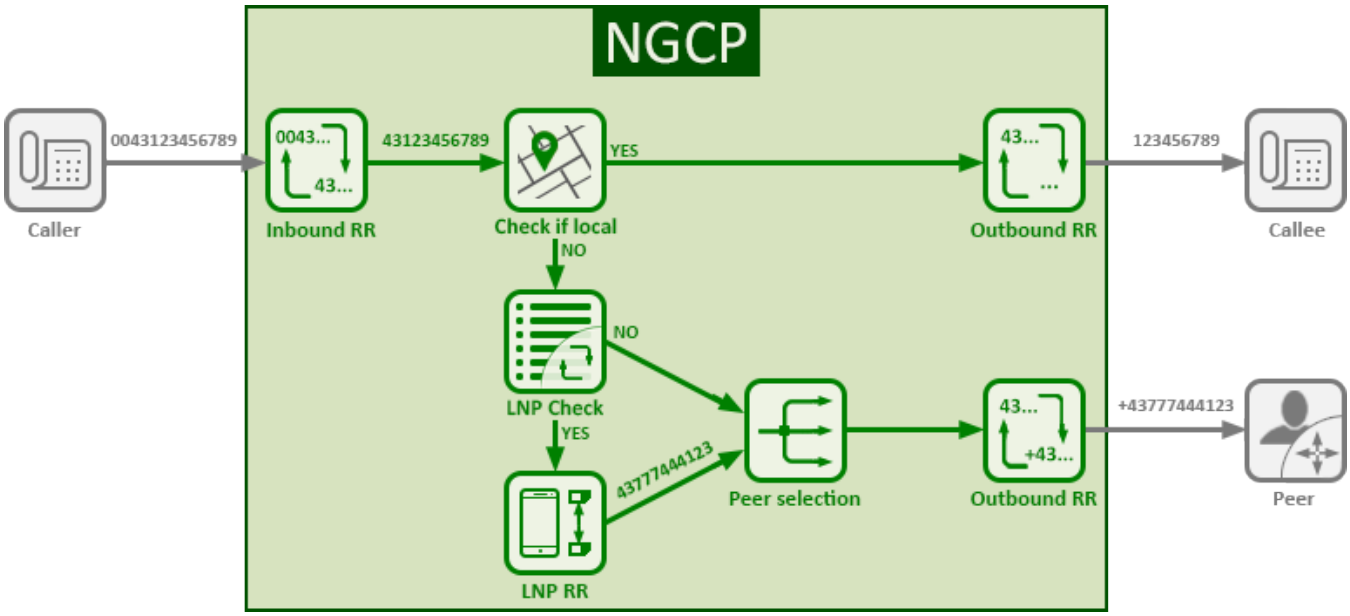

Figure 131. General Call Setup

Sipwise C5 performs the following checks when processing a call coming from a subscriber and terminated at a peer:

- Checks if the IP address where the request came from is in the list of trusted IP addresses. If yes, this IP address is taken as the identity for authentication. Otherwise, Sipwise C5 performs the digest authentication.
- When the subscriber is authorized to make the call, Sipwise C5 applies the Inbound Rewrite Rules for the caller and the callee assigned to the subscriber (if any). If there are no Rewrite Rules assigned to the subscriber, the ones assigned to the subscriber's domain are applied. At this stage the platform normalises the numbers from the subscriber's format to E.164.
- Matches the callee (called number) with local subscribers.

  If it finds a matching subscriber, the call is routed internally. In this case, Sipwise C5 applies the Outbound Rewrite Rules associated with the callee (if any). If there are no Rewrite Rules assigned to the callee, the ones assigned to the callee's domain are applied.

  If it does not find a matching subscriber, the call goes to a peer as described below.
- Queries the LNP database to find out if the number was ported or not. For details of LNP queries refer to the Local Number Porting chapter.

  If it was ported, Sipwise C5 applies the LNP Rewrite Rules to the called number.
- Based on the priorities of peering groups and peering rules (see Routing Order Selection for details), Sipwise C5 selects peering groups for call termination and defines their precedence.
- Within every peering group the weight of a peering server defines its probability to receive the call for termination. Thus, the bigger the weight of a server, the higher the probability that Sipwise C5 will send the call to it.
- Applies the Outbound Rewrite Rules for the caller and the callee assigned to a peering server when

Copyright © Sipwise GmbH - All rights reserved

sending the call to it.

Endpoint Registration

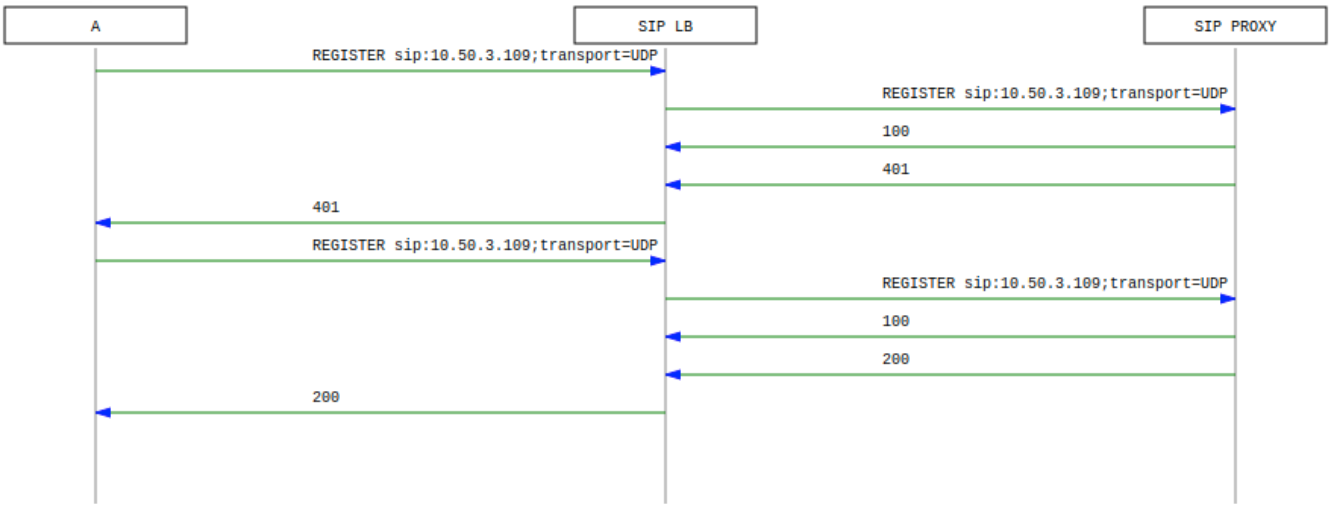

Figure 132. Registration Call-Flow

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@203.0.113.42:1234;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@203.0.113.42:1234;transport=UDP` and sent out to this address).

If NAT is detected, the SIP proxy sends an OPTIONS message to the registered contact every 30 seconds, in order to keep the NAT binding on the NAT device open. Otherwise, for subsequent calls to this contact, Sipwise C5 would not be able to reach the endpoint behind NAT (NAT devices usually drop a UDP binding after not receiving any traffic for ~30-60 seconds).

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

Basic Call

Copyright © Sipwise GmbH - All rights reserved

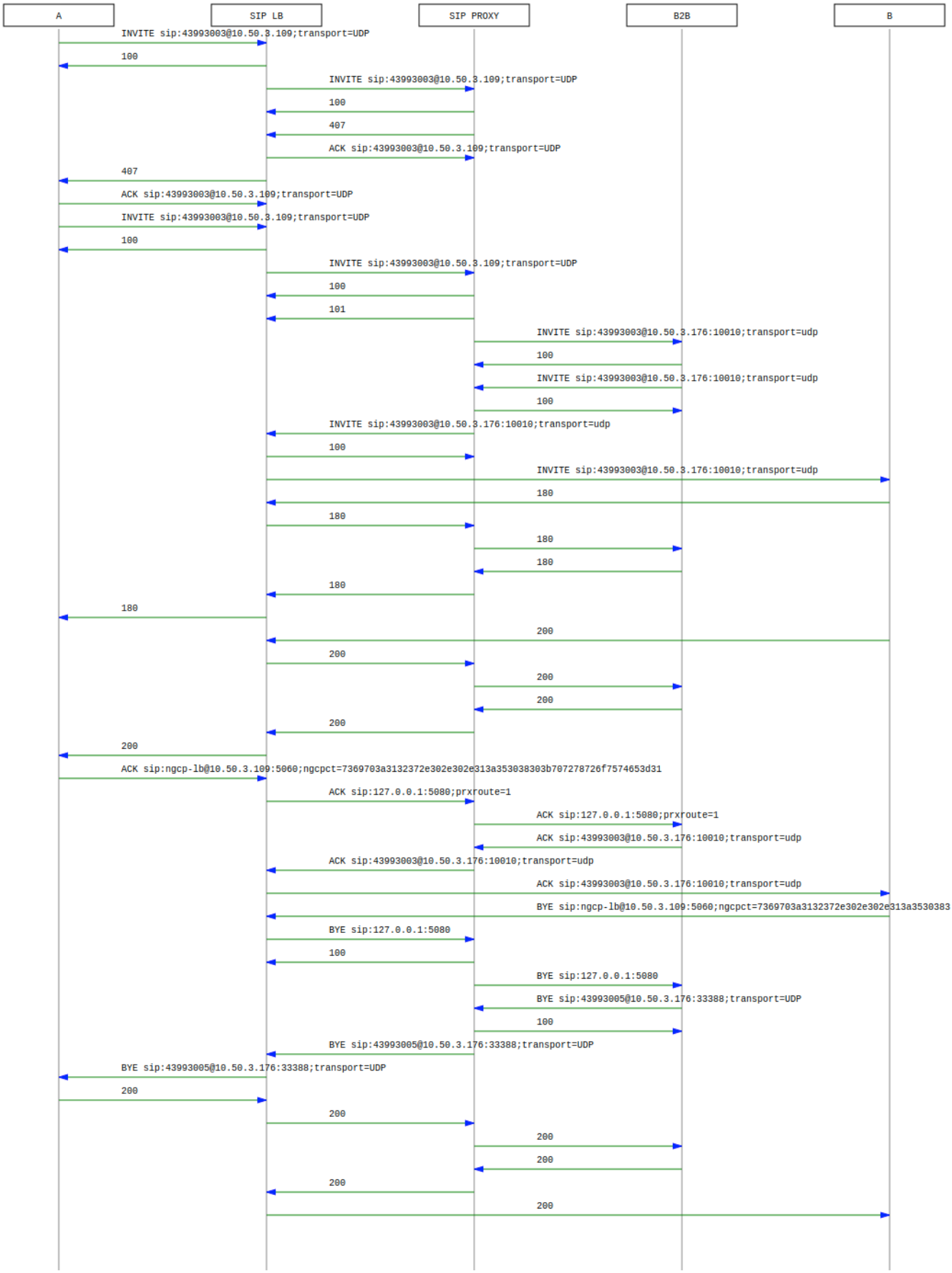

Figure 133. Basic Call Call-Flow

The calling party sends an INVITE (e.g. `sip:someuser@example.org`) via the SIP load-balancer to the

Copyright © Sipwise GmbH - All rights reserved

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@203.0.113.42:1234;transport=UDP`). If it is 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@203.0.113.42:5060`).

Once the proxy has finished working through the call features of both parties involved and has selected the final destination for the call, and - optionally - has invoked the Media Relay for this call, the INVITE is sent to the SIP B2BUA. The B2BUA creates a new INVITE message from scratch (using a new Call-ID and a new From-Tag), copies only various and explicitly allowed SIP headers from the old message to the new one, filters out unwanted media capabilities from the SDP body (e.g. to force audio calls to use G.711 as a codec) and then sends the new message back to the SIP proxy that forwards it to the SIP load-balancer to reach to the called party.

SIP replies from the called party are passed through the elements back to the calling party (replacing various fields on the B2BUA to match the first call leg again). If a reply with an SDP body is received by the SIP proxy (e.g. a 183 or a 200), the Media Relay is invoked again to prepare the ports for the media stream.

Once the 200OK is routed from the called party to the calling party, the media stream is fully negotiated, and the endpoints can start sending traffic to each other (either end-to-end or via the Media Relay). Upon reception of the 200OK, the SIP proxy writes a start record for the accounting process. The 200OK is also acknowledged with an ACK message from the calling party to the called party, according to the SIP 3-way handshake.

Either of the parties can tear down the media session at any time by sending a BYE, which is passed through to the other party. Once the BYE reaches the SIP proxy, it instructs the Media Relay to close the media ports, and it writes a stop record for accounting purposes. Both the start- and the stop-records are picked up by the *ngcp-mediator* service in a regular interval and are converted into a Call Detail Record (CDR), which will be rated by the *ngcp-rate-o-mat* process and can be billed to the calling party.

Session Keep-Alive

The SIP B2BUA acts as refresher for the Session-Timer mechanism as defined in RFC 4028. If the endpoints indicate support for session timers during call-setup, then the SIP B2BUA will use an UPDATE or re-INVITE message if enabled per peer, domain or subscriber via Provisioning to check if the endpoints are still alive and responsive. Both endpoints can renegotiate the timer within a configurable range. All values can be tuned using the Admin Panel or the APIs using Peer-, Domain- and Subscriber-Preferences.

TIP Keep in mind that the values being used in the signaling are always half the value being configured. So if you want to send a keep-alive every 300 seconds, you need to provision *sst_expires* to 600.

If one of the endpoints does not 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.

Besides the Session-Timer mechanism to prevent calls from being lost or kept open, there is a

Copyright © Sipwise GmbH - All rights reserved

maximum call length of 21600 seconds by default defined in the B2BUA. This is a security/anti-fraud mechanism to prevent overly long calls causing excessive costs.

Voicebox Calls



Figure 134. 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 network and 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 an 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 Sipwise C5 does not require your device to subscribe to the MWI service by sending a SUBSCRIBE (it would rather reject it). On the other hand, the endpoints need to accept unsolicited NOTIFY messages (that is, a NOTIFY without a valid subscription), otherwise the MWI service will not work with these endpoints.

Copyright © Sipwise GmbH - All rights reserved

Appendix C: Sipwise C5 Authentication Tokens

Sipwise C5 defines many system users and their authentication credentials in order to provide the necessary security level for its components / services.

Authentication with Password

Credentials of System Users

The list of system users that use password for their authentication is stored in `/etc/ngcp-config/constants.yml` configuration file.

CAUTION The table below shows the default usernames. Unless you have a good reason, please do not change the credentials in the file because this may render the related service unavailable!

Table 35. List of system users with password authentication

Purpose / Use case	Username	Configuration parameters: username / password
Generic system users		
		Config parameter root: credentials.
Autoprovisioning of Yealink devices	N/A	n/a / autoprov.yealink.p
Faxserver manager	faxserver	faxserver.manager.u / faxserver.manager.p
Fileshare service generic authentication	comx	fileshare.authentication.u / fileshare.authentication.p
Fileshare service app authentication	app	fileshare.simple_upload.u / fileshare.simple_upload.p
HA proxy service	N/A	n/a / haproxy.p
Key for Kamailio load balancer topology hiding	N/A	n/a / kamailio.lb.topoh_key.p
Password for Kamailio proxy UAC encryption	N/A	n/a / kamailio.proxy.uaccryptpw.p
LDAP server access	N/A	n/a / ldap.root.p
REST API internal access	system	rest_api.auth_system.u / rest_api.auth_system.p
Users in MySQL database		
		Config parameter root: credentials.mysql.
Asterisk voicemail application	asterisk	asterisk.u / asterisk.p
CDR exporter	exporter	cdrexport.u / cdrexport.p
Monitoring subsystem	nagios	monitoring.u / monitoring.p
Cleanuptools (DB purging) utility	dbcleaner	cleanuptools.u / cleanuptools.p

Copyright © Sipwise GmbH - All rights reserved

Purpose / Use case	Username	Configuration parameters: username / password
Debian system maintenance	debian-sys-maint	debian.u / debian.p
Fileshare service	fileshare	fileshare.u / fileshare.p
Kamailio read-write DB access	kamailio	kamailio.u / kamailio.p
Kamailio read-only DB access	kamailioro	kamailioro.u / kamailioro.p
LDAP access	ldap	ldap.u / ldap.p
Mediator (CDR pre-processing) service	mediator	mediator.u / mediator.p
Provisioning SOAP interface	soap	provisioning.u / provisioning.p
Rate-o-mat (CDR rating) service	rateomat	rateomat.u / rateomat.p
DB replicator	replicator	replicator.u / replicator.p
Rsyslog service	rsyslog	rsyslog.u / rsyslog.p
RTPEngine service	rtpengine	rtpengine.u / rtpengine.p
B2BUA prepaid module	sems_prepaid	sems.u / sems.p
Voisniff call capturing service	voisniff	voisniff.u / voisniff.p
SMS service credentials		
		Config parameter root: sms.
SMS core module admin password	N/A	core.admin_password
SMS API access token	N/A	panel_api_token
SMS sending	sipwise	sendsms.username / sendsms.password

Credentials for External Sipwise C5 Interfaces

Some Sipwise C5 services used by external entities and some external services that NGCP needs to connect to make use of password based authentication. The list of credentials is stored in `/etc/ngcp-config/config.yml` configuration file.

CAUTION The table below shows the default usernames. Changing some of them may render the related service unavailable!

Table 36. List of credentials used for external Sipwise C5 interfaces

Purpose / Use case	Username	Configuration parameters: username / password
Access to exported events (EDR files)	youruser	eventexport.transfer.username / eventexport.transfer.password
RTPEngine Homer RTCP monitoring service	2001	rtpengine.homer.capture_id
Voisniff Homer LI (Lawful Interception) service	2002	voisniff.daemon.homer.capt_id

Copyright © Sipwise GmbH - All rights reserved

Purpose / Use case	Username	Configuration parameters: username / password
SOAP API	ngcpsoap	ossbss.htpasswd[n].user / ossbss.htpasswd[n].pass
Sipwise C5 access to SMSC interface	username	sms.smsc.username / sms.smsc.password
TURN access for Sipwise C5 standard RTP media	turnuser	turnserver.user / turnserver.pass

Authentication with PKI (Keys and Certificates)

Keys and Certificates for Using Sipwise C5 Services

Some system users of Sipwise C5 authenticate themselves using PKI (*Public Key Infrastructure*), i.e. keys and certificates. The location of those keys and certificates may be selected in /etc/ngcp-config/config.yml file, using the configuration parameters as listed in the following table.

Table 37. List of keys and certificates used in authentication

Purpose / Use case	Configuration parameters: certificate / key
Device autoprovisioning server's CA certificate	autoprov.server.ca_certfile
Device autoprovisioning server's own credentials	autoprov.server.server_certfile / autoprov.server.server_keyfile
Load-balancer SIP service TLS connection server side credentials	kamailio.lb.tls.domains.<domain>.sslcertfile / kamailio.lb.tls.domains.<domain>.sslcertkeyfile
Generic HTTPS access server side credentials	ossbss.apache.sslcertfile / ossbss.apache.sslcertkeyfile
REST API access server side credentials	ossbss.apache.restapi.sslcertfile / ossbss.apache.restapi.sslcertkeyfile
Mobile Push notification client side credentials for iOS apps	pushd.apns.certificate
Mobile Push notification (local) server's server side credentials	pushd.sslcertfile / pushd.sslcertkeyfile
RTPEngine service HTTPS credentials	rtpengine.https.crt / rtpengine.https.key
TURN server's server side credentials	turnserver.sslcertfile / turnserver.sslcertkeyfile
LI (Lawful Interception) X2 / X3 interface client credentials	voisniff.daemon.li_x1x2x3.client_certificate
HTTPS-based admin interface ("Sipwise C5 web panel") server side credentials	www_admin.http_admin.sslcertfile / www_admin.http_admin.sslcertkeyfile
HTTPS-based CSC (Customer Self-Care) interface server side credentials	www_admin.http_csc.sslcertfile / www_admin.http_csc.sslcertkeyfile
HTTPS-based internal API server side credentials	www_admin.http_system.sslcertfile / www_admin.http_system.sslcertkeyfile

Copyright © Sipwise GmbH - All rights reserved

Location of Certificates

The following locations of keys and certificates are in use:

- /etc/ngcp-config/shared-files/ssl
- /etc/ssl/certs
- /etc/ssl/private
- /etc/kamailio

Copyright © Sipwise GmbH - All rights reserved

Appendix D: MariaDB Encryption

Overview

MariaDB encryption support (officially called "Data-at-Rest") enables innodb files, tables and binlogs data encryption so that if copied over the data is not usable without the master key. All the data accessed or modified by clients is encrypted/decrypted on the fly and transparent for the users. The feature comes with a price of 3% to 5% MariaDB performance loss (depending on the hardware, and CPU in particular).

Configuration

There are new options in `constants.yml`

```
mysql:
  encryption:
    enable: yes
    encrypt_binlog: yes
    key:
1;a356c82422a9031f2e472047ad8220eeea257d611849fbdc9f75b49933f75241
    threads: 1
```

NOTE: all changes in the configuration section will cause the MariaDB server to restart when ngcpcfg templates are applied.

- `mysql.encryption.enable`: Switch encryption on/off. Values: 'yes','no', Default: 'yes'. When enabled, all tables are being encrypted, it takes from a few seconds to several minutes for MariaDB to encrypt all the data (depending on the overall size) and the encryption procedure is performed in the background, while all the data continues to be fully accessible. Also all new tables are created encrypted by default and it is not possible to disable encryption for specific tables as the encryption is 'forced'.
- `mysql.encryption.encrypt_binlog`: Encrypt binlogs. Values: 'yes','no', Default: 'yes'. While it is preferred to have this option enabled by default, for scenarios where binlog files need to be parsed, this option can be turned off. It is also possible to use `mysqlbinlog` with `--read-from-remote-server` option to read encrypted binlogs.
- `mysql.encryption.key`: Encryption key. The value is randomly generated during the cfg-schema upgrade when the option is added into `constants.yml`. The key is located in /etc/mysql/keyfile and normally MUST NOT be changed. Changing or losing the key permanently will render all the MariaDB tablespaces data (databases/tables) unusable.
- `mysql.encryption.threads`: Amount of encryption threads. Default: 1 How many MariaDB encryption threads should be running, this value depends on how many tables are created/removed or the encryption keys are rotated.

What Is Not Encrypted

- `slow-queries log`
- `mysqld.err log`
- `general queries log, if enabled`

Copyright © Sipwise GmbH - All rights reserved

Data Restoration Remarks

- When restoring data from an sql backup from another platform it is safe to do that as the currently used 'encryption_key' (inside my.cnf) is not affected this way.
- When copying constants.yml file from another platform and the encryption is enabled, the current 'mysql.encryption.key' (inside constants.yml) must be restored in `constants.yml` to the same one the MariaDB server is originally started with or it will fail to start otherwise after `ngcpcfg apply`.

Copyright © Sipwise GmbH - All rights reserved

Appendix E: Disk Partitioning

This chapter documents possible disk partitioning on Sipwise C5 available after installation of the Sipwise C5. It should be helpful to understand the overall disk partitioning schema.

Supported IO Drives

At the moment the following drives are supported: HDD, SSD, and NVMe. We recommend installing NVMe type SSD storage for the best performance. Otherwise, install SATA SSDs for an average performance as SATA hard disks are a good option only for test/development purpose.

The exact model and size depend on the type of the system and the load. We recommend running the initial performance test on the selected hardware before going into production.

Hardware vs. Software RAID

The Sipwise C5 can be installed on the pre-configured hardware RAID. In this case, the actual Debian release must support the RAID adapter. Otherwise, the Install CD can configure software RAID 1 if two identical HDD/SSD/NVMe drives are installed (follow the on-screen wizard suggestions). The installation on a single/plain drive is also possible although not recommended for production platforms.

The Default Disk Partitions

The Sipwise C5 supports the modern concept of installing several releases side by side. The ability to switch between the releases simplifies software upgrades and enables rollbacks. You can find all the benefits here here.

The new partitioning logic is simple. The 'code' of services (e.g., kamailio, MariaDB) is separated from the 'data' (e.g., databases, CDR files) generated and processed by the 'code', and is located in a different partition of the disk. Additionally, there are two partitions for 'code' with different services versions. This way, the version of the code can be switched very quickly, by rebooting the system. The 'data' partition will be the same for both versions of the 'code', and it will always be mounted and ready to be used before the services start.

New partition layout:

```
  NAME                MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
  sda                   8:0    0    xG  0 disk              # Your disk
with size X Gb
  |-sda1                8:1    0    1M  0 part              # BIOS legacy
boot
  |-sda2                8:2    0  486M  0 part  /boot/efi   # UEFI boot
  `-sda3                8:3    0    yG  0 part              # LVM
partition
    `-md0               9:0    0    yG  0 raid1             # SW RAID (if
requested)
      |-ngcp-root     253:0    0   10G  0 lvm   /           # 'code'
partition
      |-ngcp-fallback 253:1    0   10G  0 lvm               # 'fallback'
partition
```

Copyright © Sipwise GmbH - All rights reserved

```
      `-ngcp-data     253:3    0    zG  0 lvm   /ngcp-data # 'data'
partition
                                                          # unassigned
space
```

- 1st partition: 1M BIOS boot, for BIOS/GPT (legacy) boot

  this allows fallback to grub-pc package (This partition must have its GUID set to 21686148-6449-6E6F-744E-656564454649 To switch to grub-pc, boot from a rescue/live CD, set to bios_grub with parted, then install grub to disk, so it properly embeds core.img)

- 2nd partition: ~500MB EFI System, for UEFI/GPT boot

  used as /boot/efi, if EFI support is available

- 3rd partition: LVM that is divided into:

  /dev/mapper/ngcp-root with 10GB (rootfs target)
  /dev/mapper/ngcp-fallback with 10GB (for rollback/install/upgrade)

  10% or >=500MB (whichever is bigger) of the remaining space is unassigned to allow LVM snapshots during maintenance

  /dev/mapper/ngcp-data is the **/ngcp-data** partition with the rest of the disk space for the whole platform 'data' (e.g., databases, CDR files, logs, etc.)

NOTE The installer can only boot from GPT and does not support ms-dos partitions anymore. The legacy 'BIOS' systems can also boot from GPT, while (U)EFI systems can only boot from GPT (and not from BIOS/legacy boot).

UEFI

UEFI installation is supported. The dedicated UEFI partition has been created on the disk during the installation (being the second partition in the list).

Swap Partition vs. File

IMPORTANT The Sipwise C5 performance heavily depends on the IO operations, hence if Swap is used (either the Swap file and/or the Swap partition), the performance might deteriorate. We highly recommend increasing RAM if the platform uses Swap during normal operation.

The Swap partition is no longer in use. The Sipwise C5 has been migrated to the Swap file on the 'data' partition. It gives the following benefits:

- more space is now available for the 'root', 'rollback' and 'data' partitions.
- the Swap file size can be easily changed on the fly (if necessary).
- the Swap file can be migrated to a new location easily: create a new Swap file with the necessary size and location using the 'mkswap' command and activate the new Swap file with 'swapon'. Add the new location to /etc/fstab. Now, you can deactivate the old swapfile with 'swapoff' and remove it to release the disk space.
- The main reason for the Swap partition usage, used to be 'data fragmentation' on hard disk drives (HDDs) and old types of filesystems. For modern SSD drives, the fragmentation issue is irrelevant and the 'ext4' filesystem does not require manual defragmentation either. The free space on fast

Copyright © Sipwise GmbH - All rights reserved

SSDs is more important nowadays, as it allows storing more 'data'.

Copyright © Sipwise GmbH - All rights reserved

Appendix F: NGCP Internals

This chapter documents internals of Sipwise C5 that should not be usually needed, but might be helpful to understand the overall system.

Pending Reboot Marker

The Sipwise C5 has the ability to mark a pending reboot for any server, using the file */run/reboot-required*. As soon as the file exists, several components will report about a pending reboot to the end-user. The following components report about a pending reboot right now: *ngcp-status*, *ngcpcfg status*, *motd*, *ngcp-upgrade*. Also, ngcp-upgrade will NOT allow proceeding with an upgrade if it notices a pending reboot. It might affect *rtpengine* dkms module building if there is a pending reboot requested by a newly installed kernel, etc.

Redis ID Constants

The list of current Sipwise C5 Redis DB IDs:

Service	central (role 'db')	pair	Release	Ticket	Description
sems	-	0	mr3.7.1+	-	HA switchover
rtpengine	-	1	mr3.7.1+	-	HA switchover
proxy	2	-	mr3.7.1+	-	Counter of hunting groups
proxy	3	-	mr3.7.1+	-	Concurrent dialog counters
proxy	-	4	mr3.7.1+	-	List of keys of the central counters
prosody	5	-	mr3.7.1 - mr14.1	-	XMPP cluster (obsolete)
sems-pbx	-	6	mr3.7 - mr9.5	-	Reserved (obsolete)
sems	7	-	mr4.1.1+	MT#12707	Sems malicious_call app
captagent	-	8	mr4.1 - mr7.1	-	Reserved (obsolete)
monitoring	9	-	mr4.3 - mr5.5	-	Reserved (obsolete)
proxy	10	-	mr4.3+	MT#16079	SIP Loop detection
ngcp-panel sessions	-	19	mr6.3+	TT#35523	Panel login sessions
proxy usrloc	20	-	mr6.2+	TT#32971	SIP registrations

Copyright © Sipwise GmbH - All rights reserved

Service	central (role 'db')	pair	Release	Ticket	Description
proxy acc	-	21	mr6.2+	TT#32971	Accounting records
proxy auth	-	22	mr6.2+	TT#32971	Subscriber data
proxy dialog	-	23	mr6.2+	TT#34100	Dialog data
lb topos	-	24	mr6.5+	TT#40617	Topos data
proxy b2b	25	-	mr8.0+	TT#64404	B2B in use by each subscriber
proxy 181 HIH	26	-	mr8.5+	TT#89301	HIH stored for 181 messages
lb debug_uri	-	27	mr9.1+	TT#93950	Send SIP messages to inactive node
websocket	-	30	mr7.1+	TT#49703	Reserved (future use)
websocket monitors	-	31	mr7.1+	TT#49703	Monitors
websocket subscriptions	-	32	mr7.1+	TT#49703	Subscriptions
proxy evapi	33	-	mr7.1+	TT#49703	Call event data
proxy push	34	-	mr10.0+	TT#131255	Suspended transactions
ngcp-task-agent	40	-	mr11.5+	MT#57970	ngcp-task-agent connection

Default Ports

The list of current Sipwise C5 default ports for NGCP services, in addition to the standard protocol ports. Most of these can be changed in 'config.yml' or 'constants.yml', and as such the canonical port registry for a system is the */etc/services* file on that system.

Service	Port	Protocol
approx-ro	9998	tcp
approx-rw	9999	tcp
asterisk-ctrl	5038	tcp
asterisk-sip	5070	udp
ffproxy	8082	tcp
freeswitch	5085	tcp
grafana	4000	tcp
haproxy-admin	8080	tcp

Copyright © Sipwise GmbH - All rights reserved

Service	Port	Protocol
kamailio-ctrl	5012	tcp
kamailio-sip	5062	tcp udp
kannel-admin	13000	tcp
kannel-smsbox	13001	tcp
kannel-smssend	13002	tcp
ngcp-ipxe	3000	tcp
ngcp-lnpd-rpc	8096	tcp
ngcp-lnpd-sip	5095	udp
ngcp-mariadb-central	3306	tcp
ngcp-mariadb-local	3306	tcp
ngcp-mariadb-pair	3306	tcp
ngcp-panel-admin	1443	tcp
ngcp-panel-api	1442	tcp
ngcp-redis-central	6379	tcp
ngcp-rest-api	2443	tcp
ngcp-websocket	3443	tcp
ngcp-witnessd	9800	tcp
nginx-status	8081	tcp
pbx-autoprov-cisco	1447	tcp
pbx-autoprov-https	1444	tcp
pbx-autoprov-http	1445	tcp
prometheus-mysqld-exporter	9104	tcp
prometheus-nginx-exporter	9113	tcp
prometheus-node-exporter	9100	tcp
prometheus-process-exporter	9256	tcp
prometheus-redis-exporter	9121	tcp
prometheus	9090	tcp
rtpengine-cli	2224	tcp
rtpengine-ctrl	2223	udp
rtpengine-https	2226	tcp
rtpengine-http	2225	tcp
sems-b2b	5080	tcp udp
sems-monit	5040	udp
sems-xmlrpc	8090	tcp
turn	443	tcp udp

Copyright © Sipwise GmbH - All rights reserved

Service	Port	Protocol
turn-tls	443	tcp udp

Monitoring Database Metrics

Prometheus Monitoring Metrics

The *Prometheus* monitoring database contains time series metrics of several monitoring sources.

The 'ngcp_' namespace is provided by *ngcp-witnessd*, and here are some of its current metrics subnamespaces:

ngcp_node_	Cluster node information.
ngcp_tls_certs_	TLS certificate information.
ngcp_monit_	Monit supervised processes information.
ngcp_mail_	MTA information.
ngcp_mysql_	MySQL database information.
ngcp_kamailio_	Kamailio statistics information.
ngcp_peer_	Peering information.
ngcp_sip_	SIP statistics information.
ngcp_cdr_	CDR statistics information.

The 'ngcp_node_' namespace consists of the following metrics:

ngcp_node_active	Cluster node HA state (boolean: 1/0).
ngcp_node_ha_proc_state	Cluster node GCS/CRM process state (boolean: stopped/running).
ngcp_node_ha_host_state	Cluster node host state (boolean: up/down).
ngcp_node_ha_node_state	Cluster node HA state (ngcp-check-active -q).

The 'ngcp_tls_certs_' namespace consists of the following metrics:

ngcp_tls_certs_expires_on	TLS certificate expiration information; the value is the expiration date as seconds since the epoch, with labels: **filename** of the certificate or bundle; **fingerprint** of the certificate, relevant on bundles).

The 'ngcp_monit_' namespace consists of the following metrics:

ngcp_monit_proc_info	Information about the process, the value is always 1, with labels: **name**, **pid**, **ppid**, **proc_status**, **monit_status**, **service_cur_status** is the current systemd service status, **service_exp_status** is the expected systemd service status.
ngcp_monit_proc_children	The number of children.

Copyright © Sipwise GmbH - All rights reserved

ngcp_monit_proc_uptime	The process uptime in seconds.
ngcp_monit_proc_cpu_ratio	The CPU usage in for this process.
ngcp_monit_proc_cpu_total_ratio	The CPU usage in for the process group.
ngcp_monit_proc_memory_bytes	The memory in bytes for this process.
ngcp_monit_proc_memory_total_bytes	The memory in bytes for the process group.
ngcp_monit_proc_port_response_seconds	The monitored port response in seconds.
ngcp_monit_proc_sock_response_seconds	The monitored socket response in seconds.
ngcp_monit_proc_data_collected	The timestamp when the data was collected.

The 'ngcp_mysql_' namespace consists of the following metrics:

ngcp_mysql_queries_per_second_average	Average of queries per second.
ngcp_mysql_global_status_innodb_buffer_pool_pages_total	Total number of InnoDB buffer pool pages.
ngcp_mysql_global_status_innodb_buffer_pool_pages_free	Number of free InnoDB buffer pool pages.
ngcp_mysql_global_status_innodb_buffer_pool_pages_dirty	Number of dirty InnoDB buffer pool pages.
ngcp_mysql_global_status_commands_total	Total number of commands used.
ngcp_mysql_slave_status_slave_io_running	Whether the IO slave is running.
ngcp_mysql_slave_status_slave_sql_running	Whether the SQL slave is running.
ngcp_mysql_slave_status_seconds_behind_master	Seconds behind master replication.
ngcp_mysql_slave_status_last_io_errno	Last IO error description.
ngcp_mysql_slave_status_last_sql_errno	Last SQL error description.

The 'ngcp_peer_' namespace consists of the following metrics:

ngcp_peer_group_info	Peer group information; the value is the group ID, and contains group **name**, **priority** and **description** labels.
ngcp_peer_host_info	Peer host information; the value is the host ID, and contains host **name**, **ip**, and **group_id** labels.
ngcp_peer_host_status	Peer host status, where the value is -1=unknown on standby node, 0=unknown, 1=admin-down, 2=admin-up, 3=pending, 4=down, 5=up; contains the host **id** label.
ngcp_peer_host_cc_inout	Peer host concurrent call (in/out) counter; contains the host **id** label.
ngcp_peer_host_cc_out	Peer host concurrent call (in) counter; contains the host **id** label.

Copyright © Sipwise GmbH - All rights reserved

The 'ngcp_sip_' namespaces consists of the following metrics:

ngcp_sip_answer_seizure_ratio	Current ASR (Answer Seizure Ratio).
ngcp_sip_concurrent_calls	Number of concurrent calls.
ngcp_sip_dialog_active	Number of calls currently in active dialog stage.
ngcp_sip_dialog_early	Number of calls currently in early media dialog stage.
ngcp_sip_dialog_incoming	Number of calls currently in incoming dialog stage.
ngcp_sip_dialog_local	Number of calls currently in local dialog stage.
ngcp_sip_dialog_outgoing	Number of calls currently in outgoing dialog stage.
ngcp_sip_dialog_relay	Number of calls currently in relay dialog stage.
ngcp_sip_network_efficiency_ratio	Current NER (Network Efficiency Ratio).
ngcp_sip_packets_total	Total number of SIP packets in storage.
ngcp_sip_packets_total_perday	Total number of SIP packets since 00:00 today.
ngcp_sip_partition_bytes	Size of packets partition.
ngcp_sip_provisioned_subscribers	Provisioned subscribers.
ngcp_sip_registered_devices	Registered devices.
ngcp_sip_registered_subscribers	Registered subscribers.
ngcp_sip_responsiveness_seconds	SIP server responsiveness in seconds.

The 'ngcp_cdr_' namespaces consists of the following metrics:

ngcp_cdr_total	Total number of generated CDRs.
ngcp_cdr_rated_total	Total number of rated CDRs.

The 'node_' namespace is provided by the *prometheus-node-exporter*.

The 'namedprocess_' namespace is provided by the *prometheus-process-exporter*.

The 'redis_' namespace is provided by the *prometheus-redis-exporter*.

The 'mysql_' and 'mysqld_' namespaces are provided by the *prometheus-mysqld-exporter*.

The 'nginx_' and 'nginxexporter_' namespaces are provided by the *prometheus-nginx-exporter*.

The 'haproxy_' namespace is provided by *haproxy* itself.

The 'rtpengine_' namespace is provided by *rtpengine* itself.

The 'go_', 'process_' and 'promhttp_' namespaces are provided by exporters based on the Prometheus Golang libraries and export introspecting metrics for the exporters themselves.

The 'flag_', 'up_', 'promscrape_', 'scrape_', 'vm_', 'vminsert_' and 'vmselect_' namespaces are provided by *victoria-metrics*.

Copyright © Sipwise GmbH - All rights reserved

NGCP Preferences

Tables

Currently available tables for preferences are

provisioning.voip_preferences	contains all available preferences, do not contain user data.
provisioning.voip_preference_group	contains preference group names, so the preferences can be put into groups.
provisioning.voip_preferences_enum	contains enum values for preferences, do not contain user data.

The following tables contain user data and depend on `voip_preferences` and optionally on `voip_preferences_enum`:

provisioning.voip_dev_preferences	PBX device model preferences
provisioning.voip_devprof_preferences	PBX device profile preferences
provisioning.voip_dom_preferences	domain preferences, replicated by triggers to `kamailio.dom_preferences`
provisioning.voip_contract_preferences	customer preferences, replicated by triggers to `kamailio.contract_preferences`
provisioning.voip_peer_preferences	peering server preferences, replicated by triggers to `kamailio.peer_preferences`
provisioning.voip_prof_preferences	subscriber profile preferences
provisioning.voip_reseller_preferences	reseller preferences
provisioning.voip_usr_preferences	subscriber preferences, replicated by triggers to `kamailio.usr_preferences`

Columns

Columns for table 'provisioning.voip_preferences'

id	primary key, used in user tables as the foreign key
voip_preference_groups_id	preference group id
attribute	preference name
label	tooltip that can be used as a mouseover tooltip on the UI
type	0 - 'string', 1 - 'integer'/'boolean'
max_occur	how many preferences with the name are allowed 0: list, 1: only one
usr_pref	defines if the preference can be used in subscribers
prof_pref	defines if the preference can be used in subscriber profiles

Copyright © Sipwise GmbH - All rights reserved

dom_pref	defines if the preference can be used in domains
peer_pref	defines if the preference can be used in peering servers
contract_pref	defines if the preference can be used in customers
contract_location_pref	defines if the preference can be used in customer locations
dev_pref	defines if the preference can be used in PBX device models
devprof_pref	defines if the preference can be used in PBX device profiles
fielddev_pref	defines if the preference can be used in PBX devices that are assigned to a subscriber
modify_timestamp	preference last modification time
internal	the preference is for internal use only and not shown in the UI/API
expose_to_customer	indicates that the preference can be read/modified by 'subscriberadmin' role
data_type	data type 'enum', 'boolean', 'int', 'string'
read_only	read only flag
description	long description of the preference
dynamic	set to 1 if it is a custom preference that is created by a user (usually for a PBX device model that requires specific preferences) but can be used for all preferences when needed
reseller_pref	defines if the preference can be used in resellers
expose_to_subscriber	indicates that the preference can be read/modified by 'subscriber' role

Enum

All tables are in database "provisioning".

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

Setting the attribute "data_type" of table "voip_preferences" to "enum" marks a preferences as an enum. The list of possible options is stored in table "voip_preferences_enum".

Columns for table 'provisioning.voip_preferences_enum' are:

id	primary key
preference_id	Reference to table voip_preferences.
label	A label to be displayed in frontends.

Copyright © Sipwise GmbH - All rights reserved

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 unambiguously (e.g. *"use domain default"*). (Note: Therefore will also not be written to any kamailio table.)
usr_pref	Flag if this is to be used for usr preferences.
dom_pref	Flag if this is to be used for dom preferences.
peer_pref	Flag if this is to be used for peer preferences.
default_val	Flag indicating if this should be used as a default value when creating new entities or introducing new enum preferences (both done via triggers). (Note: For this to work, value must also be set.)

Relevant triggers:

enum_update	Propagates changes of voip_preferences_enum. value to voip_[usr	dom	peer]_preferences.value
enum_set_default	Will create entries for default values when adding a new enum preference. The default value is the tuple from voip_preferences_enum WHERE default_val=1 AND value NOT NULL.	voip_dom_crepl_trig	The trigger will set possible default values (same condition as for enum_set_default) when creating new domains.
voip_phost_crepl_trig	The trigger will set possible default values (same condition as for enum_set_default) when creating new peers.	voip_sub_crepl_trig	The trigger will set possible default values (same condition as for enum_set_default) when creating new subscribers.

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

Copyright © Sipwise GmbH - All rights reserved

Appendix G: Kamailio pv_headers Module

This chapter documents the kamailio "pv_headers" module.

Module Overview

This module enables storing all headers in XAVP to freely modify them in the kamailio logic and only apply them once when it is time for the packet to be routed outside. The main goal of the module is to offload the intermediate header processing into the XAVP dynamic container as well as provide with high level methods and pseudovariables to simplify SIP message header modifications.

In few words:

- as soon as a SIP message enters the proxy, kamailio reads all the headers (using the function "pv_collect_headers()") and stores them in an XAVP called "headers".
- starting from this point all the header changes are directly performed on the "headers" XAVP. For example the From header is available at '$xavp(headers[0]From[0])'.
- right before the SIP message leaves the proxy, kamailio writes back all the headers changes (using the function "pv_apply_headers()").

RURI and the headers listed in the module parameter "skip_headers" are left untouched and not saved in the XAVP. Therefore they should be handled in the usual way.

Template Changes

As described before in the upgrade procedures, the module is enabled by default in kamailio proxy and all the templates have been already updated to use this new logic. Before proceeding with the upgrade, it is essential that the customtt/patchtt files you have in place are updated to this new format.

Here some few examples of what has been changed in the proxy templates:

- variables $fu, $fU, $fd, $fn, $ft have been substituted by $x_fu, $x_fU, $x_fd, $x_fn, $x_ft
- variables $tu, $tU, $td, $tn, $tt have been substituted by $x_tu, $x_tU, $x_td, $x_tn, $x_tt
- variables $rr, $rs have been substituted by $x_rr, $x_rs
- variables $ua have been substituted by $x_hdr(User-Agent)
- variables $ai have been substituted by $x_hdr(P-Asserted-Identity)
- variables $pU, $pd have been substituted by $x_hdr(P-Preferred-Identity)
- variables $re have been substituted by $x_hdr(Remote-Party-ID)
- variables $di have been substituted by $x_hdr(Diversion)
- variables $ct have been substituted by $x_hdr(Contact)
- $hdr("name") has been substituted by $x_hdr("name")
- is_present_hf("name") has been substituted by $x_hdr(name)!= $null
- remove_hf("name") has been substituted by pv_remove_header("name") function or $(x_hdr(name)[*]) = $null
- append_hf("name: value\r\n") has been substituted by pv_append_header("name", "value") / pv_modify_header("name", "value") functions or $(x_hdr(name)[*]) = value

Copyright © Sipwise GmbH - All rights reserved

- t_check_status(code) has been substituted by $T_reply_code == code
- save("location") has been updated in save("location", "0x00", "$x_tu")
- sd_lookup("speed_dial") has been updated in sd_lookup("speed_dial", $x_fu)
- added pv_collect_headers() and pv_reset_headers() functions in the dedicated ROUTE_COLLECT_HDR route
- added pv_apply_headers() function in the dedicated ROUTE_APPLY_HDR route
- added pv_reset_headers() function in the following routing sections

Module Documentation

Parameters

xavp_name (string)

Name of the XAVP where the collected headers are stored.

Default: headers

```
modparam("pv_headers", "xavp_name", "headers")

Result:
  $xavp(headers[0]=>From)
  $xavp(headers[0]=>To)
  $xavp(headers[0]=>Call-ID)
  ....
```

skip_headers (string)

A comma separated headers list that must be excluded from processing (they are skipped when pv_apply_headers() changes the sip message headers). If the parameter is not set then the "Default" list is used. If the parameter is set to an empty string then all the sip message headers are processed.

Default: Record-Route,Via,Route,Content-Length,Max-Forwards

split_headers (string)

A comma separated headers list that must be split into multi headers if their value is a comma separated list. If the parameter is not set then the "Default" is used. If the parameter is set to an empty string then no headers are split.

Default: None

```
modparam("pv_headers", "split_headers", "Diversion")

    Result:
        Received Diversion header:
            Diversion:
<user1@test.local>,<user2@test.local>,<user3@test.local>
```


Copyright © Sipwise GmbH - All rights reserved

```
        After split:
            Diversion: <user1@test.local>
            Diversion: <user2@test.local>
            Diversion: <user3@test.local>
        Becomes handy if used together with pv_modify_header() or
pv_remove_header()
        to change or remove value 2 for instance.
```

Functions

pv_collect_headers()

This function collects all headers from the message into the XAVP. It should be used preferably just when the sip message is received by kamailio.

Returns:

- 1 - on success
- -1 - if there were errors

pv_apply_headers()

This function applies the current XAVP headers state to the real headers and should be called only once per branch when the message is about to leave kamailio.

The following rules apply:

- all headers in the XAVP except for ones provided in the "skip_headers" parameter and From/To are recreated in the sip message.
- From/To headers are processed by the uac module if it is loaded.
- From/To headers are not changed in the reply messages.
- headers with NULL value are removed if exist in the sip message.
- the initial order of the sip headers is preserved.

Usage:

```
if (pv_apply_headers())
{
    "success"
}
else
{
    "errors"
}
```

pv_reset_headers()

This function resets the current XAVP headers list and enables pv_collect_headers() and pv_apply_headers() to be called again in the same branch.

Copyright © Sipwise GmbH - All rights reserved

Usage:

```
if (pv_reset_headers())
{
    "success"
}
else
{
    "errors"
}
```

pv_check_header(hname)

This function checks if the header already exists in the XAVP. It can be freely called from anywhere, but only after pv_collect_headers().

Usage:

```
if (pv_check_header(hname))
{
    "exists"
}
else
{
    "does not exist"
}
```

pv_append_header(hname, hvalue)

This function appends a new header into the XAVP. It can be freely called from anywhere, but only after pv_collect_headers(). Please note that subsequent "pv_append_header" calls will result in multiple headers. If the provided "hvalue" is $null then the header is added into the XAVP but it is not going to be added into the message.

Usage:

```
if (pv_append_header(hname, hvalue))
{
    "appended"
}
else
{
    "errors"
}
```

pv_modify_header(hname, hvalue)

This function modifies an existing header in the XAVP. It can be freely called from anywhere, but only after pv_collect_headers(). Please note that if the header does not exist it will be explicitly appended. If

Copyright © Sipwise GmbH - All rights reserved

there are multiple headers with the same name only the first one will be affected. If the provided header value is $null then the header is modified in the XAVP and then it is removed from the sip message when pv_apply_headers() is called.

Usage:

```
if (pv_modify_header(hname, hvalue))
{
    "modified"
}
else
{
    "errors"
}
```

pv_modify_header(hname, idx, hvalue)

This function works similar to pv_modify_header(hname, hvalue) but should be used when there are multiple headers with the same name one of them to be modified. Index order is top to bottom.

Usage:

```
if (pv_modify_header(hname, idx, hvalue))
{
    "modified"
}
else
{
    "errors"
}
```

pv_remove_header(hname)

This function removes an existing header from the XAVP. It can be freely called from anywhere, but only after pv_collect_headers(). If there are multiple headers with the same name all of them are removed. It returns -1 if the header does not exist.

Usage:

```
if (pv_remove_header(hname, hvalue))
{
    "removed"
}
else
{
    "does not exist or errors"
}
```

pv_remove_header(hname, idx, hvalue)

Copyright © Sipwise GmbH - All rights reserved

This function works similar to pv_remove_header(hname, hvalue) but should be used when there are multiple headers with the same name one of them to be removed. Index order is top to bottom.

Usage:

```
if (pv_remove_header(hname, idx, hvalue))
{
    "removed"
}
else
{
    "does not exist or errors"
}
```

Pseudovariables

$x_hdr

This pseudovariable is used to append/modify/remove headers by their name and can be used instead of the pv_append_header(), pv_modify_header(), pv_remove_header() functions.

Usage:

- append header "X-Header" with value "example". NOTE: It always appends a header, even there is already one with the same name

```
$x_hdr(X-Header) = "example";
```

- modify header "X-Header" with index 0. Returns an error if there is no such index

```
$(x_hdr(X-Header)[0]) = "example";
```

- remove all occurrences of header "X-Header" and append one with value "example"

```
$(x_hdr(X-Header)[*]) = "example";
```

- remove header "X-Header" with index 2 (if there are multiple headers). Returns an error if there is no such index

```
$(x_hdr(X-Header)[2]) = $null;
```

- remove all occurrences of the header. Does not produce an error if there is no such header

```
$(x_hdr(X-Header)[*]) = $null;
```


Copyright © Sipwise GmbH - All rights reserved

- retrieve a value of header "X-Header" with index 0, otherwise $null

```
$var(test) = $x_hdr(X-Header);
```

- retrieve a value of header "X-Header" with index 0 otherwise $null

```
$var(test) = $x_hdr(X-Header)[*]);
```

- retrieve a value of header "X-Header" with index 2 otherwise $null

```
$var(test) = $(x_hdr(X-Header)[2]);
```

$x_fu, $x_tu

These pseudovariables are used to modify/retrieve the "From" and "To" headers.

Usage:

- modify the header

```
$x_fu = "User1 <440001@example.local>";
```

- retrieve a value of the header

```
$var(test) = $x_fu;
```

- $x_tu usage is the same

$x_fU, $x_tU

These pseudovariables are used to modify/retrieve the username part of the "From" and "To" headers.

Usage:

- modify the username part

```
$x_fU = "440001";
```

- retrieve the username part

```
$var(test) = $x_fU;
```

- $x_tU usage is the same

Copyright © Sipwise GmbH - All rights reserved

$x_fd, $x_td

These pseudovariables are used to modify/retrieve the domain part of the "From" and "To" headers.

Usage:

- modify the domain part

```
$x_fd = "example.local";
```

- retrieve the domain part

```
$var(test) = $x_fd;
```

- $x_td usage is the same

$x_fn, $x_tn

These pseudovariables are used to modify/retrieve the display part of the "From" and "To" headers.

Usage:

- modify the username part

```
$x_fn = "User1";
```

- retrieve the domain part

```
$var(test) = $x_fn;
```

- $x_tn usage is the same

$x_ft, $x_tt

These pseudovariables are used to retrieve the tag part of the "From" and "To" headers.

Usage:

- retrieve the tag part

```
$var(test) = $x_ft;
```

- $x_tt usage is the same

$x_rs, $x_rr

These pseudovariables are used to modify/retrieve or change "status" and "code" of the SIP reply

Copyright © Sipwise GmbH - All rights reserved

NOTE: Only messages with reply status > 300 can be changed as well as reply status 1xx and 2xx cannot be set

Usage:

- modify the reply status

```
$x_rs = 486
```

- retrieve the reply status

```
$var(test) = $x_rs;
```

- modify the reply reason

```
$x_rr = "Custom Reason"
```

- retrieve the reply reason

```
$var(test) = $x_rr;
```

Copyright © Sipwise GmbH - All rights reserved

Appendix H: Extra Configuration Scenarios

AudioCodes Devices Workaround

Old AudioCodes devices suffer from a problem where they replace `127.0.0.1` address in Record-Route headers (added by Sipwise C5's internal components) with the device's IP address. Supposedly, the whole range of AudioCodes devices with a firmware version below 6.8.X are affected. As a workaround, you may enable the topos feature to stop sending Record-Route headers out. To achieve this, execute the following commands:

```
ngcpcfg set /etc/ngcp-config/config.yml
kamailio.lb.security.topos.enable=yes
ngcpcfg apply 'enable topos for audiocodes devs workaround'
```

"Debug Proxy" for Troubleshooting

IMPORTANT This functionality only makes sense on Sipwise C5 CARRIER appliance environment that has an inactive proxy node available.

In order to troubleshoot/debug/capture a scenario, the "debug proxy" allows defining a list of "debug subscribers" that will be matched against From/To headers for every SIP message on a specific lb node. If matched that SIP message will be delivered to the assigned proxy node. In summary, any call to/from that subscriber will be easily traced since no calls are delivered by default to an inactive "debug proxy" node. Also, you can enable extra debug levels on the "debug proxy" node which will NOT affect production traffic on the platform.

IMPORTANT The subscribers have to be specified in the same format as they are received on Kamailio LB (before all the rewrite rules applied).

IMPORTANT In the following examples both proxy node 'prx99a' and 'prx99b' must be set to **inactive** nodes 'hosts.prx99a.status=inactive' and 'hosts.prx99b.status=inactive' in network.yml!

To enable the feature for INACTIVE 'prx99' proxy pair, execute:

```
ngcpcfg set /etc/ngcp-config/network.yml hosts.prx99a.status=inactive #
for the safety
ngcpcfg set /etc/ngcp-config/network.yml hosts.prx99b.status=inactive #
for the safety
ngcpcfg set /etc/ngcp-config/config.yml kamailio.lb.debug_uri.enable=yes
ngcpcfg apply 'Enable debug proxy on node prx99'
ngcpcfg push-parallel all
```

And make sure 'prx99' active node has this new config applied.

There is a command-line tool available to manage the subscriber list. An example of use:

```
ngcp-debug-subscriber add lb01 +4310001000@example.org
```


Copyright © Sipwise GmbH - All rights reserved

```
sipuser@example.org prx99
ngcp-debug-subscriber delete lb01 +4310001000@example.org
ngcp-debug-subscriber list lb01
```

Be aware that the list of debug subscribers belongs to just one lb pair, the info is kept in REDIS 'local' database, it is necessary to survive LB restarts and/or HA switchovers. To skip saving in REDIS 'local' database, specifying the option '--no-store' (in this case the information will stay in memory only and will be void on Kamailio LB restart):

```
ngcp-debug-subscriber add --no-store lb01 +10001042@example.org prx99
```

See more available options in general and per-action help messages:

```
ngcp-debug-subscriber --help
ngcp-debug-subscriber add --help
```

WARNING It is recommended to keep the amount of "debug subscribers" as small as possible (for performance reasons).

The "debug subscribers" is kept in a kamailio htable. htable index size value can be changed if necessary in config.yml using the option 'kamailio.lb.debug_uri.htable_idx_size'. This is **not** the maximum size. From Kamailio htable documentation:

```
size - number to control how many slots (buckets) to create for the hash
table.
A larger value means more slots with a higher probability for fewer
collisions.
The actual number of slots (or buckets) created for the table is 2^size.
The possible range for this value is from 2 to 31, smaller or larger
values
will be increased to 3 (8 slots) or decreased to 14 (16384 slots).
Note that each slot can store more than one item, when there are
collisions of
hash ids computed for keys. The items in the same slot are stored in a
linked list.
In other words, the size is not setting a limit of how many items can be
stored in a hash table, as long as there is enough free shared memory,
new items can be added.
```

The default value 'kamailio.lb.debug_uri.htable_idx_size=4' is enough for all the use cases in production.

Copyright © Sipwise GmbH - All rights reserved

Appendix I: NGCP CLI Helpers and Tools

Main NGCP Tools

Sipwise C5 provides a list of various scripts, helpers, and tools to successfully maintain the system from the POSIX console. You can access those scripts using ssh or log in to the Unix terminal. All Sipwise C5 scripts start with the prefix 'ngcp-'.

NOTE Currently services and daemon executables namespaced too with 'ngcp-' can be found within PATH, but are not intended for general execution, and might eventually be moved out under /usr/libexec/. These are listed in the table "Internal NGCP component" below.

A must-have list to learn for everyday usage:

- ngcpcfg
- ngcp-loglevel
- ngcp-service
- ngcp-status

Public NGCP Tools

Name	Description
Configuration tools:	
ngcpcfg	main NGCP configuration tool
ngcp-initial-configuration	configure 'bare' node as NGCP (cluster) member
ngcp-network	command line interface to ngcp-config network configuration settings
ngcp-support-access	enable/disable Sipwise Support team access to NGCP
ngcp-toggle-performance-config	switch NGCP performance modes: low/high
Maintenance tools:	
ngcp-approx-cache	update cached APT metadata in Approx
ngcp-approx-snapshots	manage (create/delete/export/import) snapshots of Approx
ngcp-customtt-diff-helper	prepare local customtt/patchtt files for Sipwise analyses
ngcp-loglevel	show/set debug level per component in run-time
ngcp-ppa	handle Sipwise PPA APT repositories for NGCP customization
ngcp-prepare-upgrade	prepare NGCP to upgrade to the new release/build
ngcp-reset-db	reset MariaDB database. NOTE: think twice before calling it!

Copyright © Sipwise GmbH - All rights reserved

Name	Description
ngcp-update	wrapper for the latest hotfixes installations inside the same release
ngcp-update-cfg-schema	update ngcp-config YML files. Safe to re-execute
ngcp-update-db-schema	upgrade MariaDB. Safe to re-execute
ngcp-upgrade	Sipwise NGCP platform upgrade framework to upgrade on new release/build. Execute with caution!
ngcp-upgrade-pre-checks	check possible issues/misconfigurations before the upgrade
Operational tools:	
ngcp-active-calls	show/drop active calls matching search criteria
ngcp-cdr-md5	validate the integrity of exported CDR files
ngcp-check-active	show HA node status (active/standby)
ngcp-check-redis-dialogs	script to check stalled Kamailio dialogs in Redis DB
ngcp-disk-usage	disk usage statistic tool (symlink to ncdu)
ngcp-kamcmd	Kamailio console debug tool
ngcp-kamctl	Kamailio command-line interface
ngcp-location-cleanup	Delete expired location entries from Redis
ngcp-log-flow	create a call flow of a single Call-ID taking NGCP logs as input
ngcp-logs	search a string in NGCP logs. Useful to retrieve Kamailio and SEMS logs given a call-id
ngcp-make-active	make HA node active (revoke standby HA state from the peer)
ngcp-make-standby	make HA node standby (revoke active HA state from the peer)
ngcp-memory-usage	report processes with the biggest RAM/SWAP usage
ngcp-mysql-compare-dbs	fast compare MariaDB schemas between two hosts
ngcp-redis-helper	easier various data requests from Redis DB
ngcp-service	manage NGCP system services. See 'man ngcp-service' for more details
ngcp-snmpd-adduser	add or replace SNMPv3 users
ngcp-snmpd-deluser	delete SNMPv3 users
ngcp-status	show general overall NGCP status. It is a wrapper for many NGCP tools
ngcp-support-upload	upload files to the ticket on support.sipwise.com

Copyright © Sipwise GmbH - All rights reserved

Name	Description
ngcp-system-tests	main self-check tool. Safe to execute in production
ngcp-usr-location	show the correct VoIP registrations (from Redis DB). See 'ngcp-usr-location --help'
REST API tools:	
ngcp-api-admins	manage NGCP Administrators
ngcp-api-delete	send arbitrary DELETE request to NGCP REST API
ngcp-api-get	send arbitrary GET request to NGCP REST API
ngcp-api-patch	send arbitrary PATCH request to NGCP REST API
ngcp-api-ping	fast check for NGCP REST API availability
ngcp-api-post	send arbitrary POST request to NGCP REST API
ngcp-api-put	send arbitrary PUT request to NGCP REST API
ngcp-create-customer	create NGCP Customer
ngcp-create-domain	create NGCP Domain
ngcp-create-subscriber	create NGCP Subscriber
ngcp-delete-domain	delete an NGCP Domain. Execute with caution (subscribers are deleted with a domain)!
ngcp-get-customer	get customer info by the customer ID
ngcp-terminate-customer	terminate an NGCP Customer
ngcp-terminate-subscriber	terminate an NGCP Subscriber
Deprecated tools (will be removed soon):	
ngcp-delete-subscriber	backward compatibility symlink to ngcp-terminate-subscriber
ngcp-delete-voip-account	backward compatibility symlink to ngcp-terminate-customer
ngcp-get-voip-account	backward compatibility symlink to ngcp-get-customer
ngcp-voicemail-table-cleanup	backward compatibility symlink to ngcp-cleanup-voicemail-table

Internal NGCP Tools

TIP they can be used, but they might be changed without the notification.

Name	Description
ngcp-bcrypt-webpassword	migration encrypt subscribers' WEB passwords using bcrypt
ngcp-check-rev-applied	check which DB/config revisions have been executed

Copyright © Sipwise GmbH - All rights reserved

Name	Description
ngcp-check-sip-option	monitoring tool that sends an OPTIONS request to a SIP server
ngcp-check-ssh-keys	check system and user SSH keys
ngcp-chroot-shell	a suid root program that will take care of securely opening a shell inside a jail
ngcp-create-testusers	developers generate a batch of customers/subscribers
ngcp-dlgcnt-check	check Kamailio dialogs counters in Redis DB
ngcp-dlgcnt-clean	remove Kamailio dialogs from Redis DB
ngcp-dlglist-clean	remove Kamailio queue dialogs from Redis DB
ngcp-ha-host-state	keep HA peers in sync (used by ngcp-config)
ngcp-ha-proc-state	keep HA peers in sync (used by ngcp-config)
ngcp-ha-crm	print current CRM in use
ngcp-io-scheduler	systemd helper to set proper IO scheduler on the system boot (HDD related only)
ngcp-kamailio-shm-usage	developers generate Kamailio shared memory usage report
ngcp-location-sync	fixes pseudo-indexes used in Redis to make location lookups faster
ngcp-memdbg-csv	developers generate Kamailio modules memory usage
ngcp-network-validator	dynamically validates the network.yml file (used by ngcp-config)
ngcp-nodename	print the current NGCP HA node name
ngcp-panel-create-keys	generate encryption keys for ngcp-panel
ngcp-peerprobe-status	internal NGCP monitoring tool
ngcp-prepare-translations	developers tool for NGCP localization files. Available on 'trunk' systems only
ngcp-screen-check	check if the current session is running inside a screen/tmux
ngcp-ssh	NGCP wrapper for SSH. Used inside various NGCP scripts to access neighbors in the cluster
ngcp-sync-constants	backwards compatibility symlink for ngcp-sync-db-creds
ngcp-sync-db-creds	sync MariaDB credentials with /etc/ngcp-config/constants.yml (used by ngcp-config)
ngcp-sync-db-grants	sync MariaDB grants with /etc/mysql/grants.yml (used by ngcp-config)

Copyright © Sipwise GmbH - All rights reserved

Name	Description
ngcp-sync-grants	backwards compatibility symlink for ngcp-sync-db-grants
ngcp-type	reports back NGCP type: spce/sppro/carrier
ngcp-virt-identify	check hardware/virtual installation type. See 'man ngcp-virt-identify' for more details

Internal NGCP Component

WARNING do NOT execute them directly. Use `ngcp-service` to start/stop service.

Name	Description
ngcp-backup	main executable binary file for backup component
ngcp-cdr-exporter	main executable binary file for CDR exporter component
ngcp-cleanup-acc	script responsible for cleaning up accounting database by cron
ngcp-cleanup-cdr-files	script responsible for cleaning up exported CDR files by cron
ngcp-cleanup-sems	script cleans up SEMS calling card tokens in Redis by cron
ngcp-cleanup-voicemail-table	script cleans up voicemail records on a monthly basis by cron
ngcp-credit-warning	main executable binary file for check for contract balances above credit warning thresholds
ngcp-emergency-mode	main executable binary file for emergency mode component
ngcp-event-exporter	main executable binary file for events exporter component
ngcp-fraud-notifier	sends fraud notifications for customers that exceed the threshold
ngcp-installer	main executable binary file for installer component
ngcp-int-cdr-exporter	main executable binary file for intermediate CDRs exporter component
ngcp-licensed	main executable binary file for licensed component
ngcp-mediator	main executable binary file for mediator component
ngcp-provisioning-template	create subscribers with detailed settings according to provisioning templates
ngcp-rate-o-mat	main executable binary file for rate-o-mat component

Copyright © Sipwise GmbH - All rights reserved

Name	Description
ngcp-reminder	main executable binary file for reminder component
ngcp-rtpengine-iptables-setup	systemd related helper for rtpengine
ngcp-rtpengine-recording-nfs-setup	systemd related helper for rtpengine-recording
ngcp-sems	main executable binary file for B2BUA component
ngcp-task-agent	enables other components to communicate with each other asynchronously
ngcp-vmnotify	an Asterisk VoiceMail compatible MWI notification script
ngcp-vmsmsnotify	same as ngcp-vmnotify, but for SMS notifications
ngcp-witnessd	main executable binary file for witnessd component

Copyright © Sipwise GmbH - All rights reserved

Appendix J: 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 calls towards normal SIP users from WebRTC clients and vice versa. Sipwise C5 listens, by default, on the following WebSockets and WebSocket Secure: `ws://your-ip:5060/ws`, `wss://your-ip:5061/ws` and `wss://your-ip:1443/wss/sip/`.

The WebRTC subscriber is a normal subscriber which has just a different configuration in his Preferences. You need to change the following preferences under *SubscribersDetailsPreferencesNAT and Media Flow Control*:

- **use_rtpproxy**: Always with rtpproxy as additional ICE candidate
- **transport_protocol**: RTP/SAVPF (encrypted SRTP with RTCP feedback)

The `transport_protocol` setting may change, depending on your WebRTC client/browser configuration. Supported protocols are the following:

- Transparent (Pass through using the client's transport protocol)
- RTP/AVP (Plain RTP)
- RTP/SAVP (encrypted SRTP)
- RTP/AVPF (RTP with RTCP feedback)
- RTP/SAVPF (encrypted SRTP with RTCP feedback)
- UDP/TLS/RTP/SAVP (Encrypted SRTP using DTLS)
- UDP/TLS/RTP/SAVPF (Encrypted SRTP using DTLS with RTCP feedback)

WARNING The below configuration is enough to handle a WebRTC client/browser. As mentioned, you may need to tune a little bit your `transport_protocol` configuration, depending on your client/browser settings.

In order to have a bridge between normal SIP clients (using plain RTP for example) and WebRTC client, the normal SIP clients' preferences have to have the following configuration:

transport_protocol: RTP/AVP (Plain RTP)

This will teach Sipwise C5 to translate between Plain RTP and RTP/SAVPF when you have calls between normal SIP clients and WebRTC clients.

Copyright © Sipwise GmbH - All rights reserved

Appendix K: Batch Provisioning Extras

Built-in Template

NGCP comes with a built-in template with basic settings. It is shown by default when creating a new template via Admin Panel, which can be edited for advanced configurations. It is also possible to define a list of templates in the */etc/ngcp-config/config.yml* file.

NOTE

By default, the built-in template will set the subscriber's SIP URI as `[CC][AC][SN]@[DOMAIN]`, where `[DOMAIN]` is defined within the template as a static value (in the `subscriber.domain` attribute). An alpha-numeric string will be generated automatically for the SIP password. Additionally, the `contract_contact.reseller` and `contract.billing_profile` attributes (among others) may need to be adjusted to particular provisioning needs.

Below, are detailed:

a. The built-in template available in Admin Panel.

b. Configurations for the built-in template in *config.yml* file.

for both JavaScript and Perl languages, respectively.

(a.1) Built-in Admin Panel Template (JavaScript):

```
fields:
  - name: first_name
    label: "First Name:"
    type: Text
    required: 1
  - name: last_name
    label: "Last Name:"
    type: Text
    required: 1
  - name: cc
    label: "Country Code:"
    type: Text
    required: 1
  - name: ac
    label: "Area Code:"
    type: Text
    required: 1
  - name: sn
    label: "Subscriber Number:"
    type: Text
    required: 1
  - name: sip_username
    type: calculated
    value_code: "function() { return
row.cc.concat(row.ac).concat(row.sn); }"
  - name: purge
    label: "Terminate subscriber, if exists:"
```

Copyright © Sipwise GmbH - All rights reserved

```
    type: Boolean
contract_contact:
  identifier: "firstname, lastname, status"
  reseller: default
  firstname_code: "function() { return row.first_name; }"
  lastname_code: "function() { return row.last_name; }"
  status: "active"
contract:
  product: "Basic SIP Account"
  billing_profile: "Default Billing Profile"
  identifier: contact_id
  contact_id_code: "function() { return contract_contact.id; }"
subscriber:
  domain: "example.org"
  primary_number:
    cc_code: "function() { return row.cc; }"
    ac_code: "function() { return row.ac; }"
    sn_code: "function() { return row.sn; }"
    username_code: "function() { return row.sip_username; }"
    password_code: "function() { return row.sip_password; }"
subscriber_preferences:
  gpp0: "test"
```


Copyright © Sipwise GmbH - All rights reserved

(a.2) Built-in Admin Panel Template (Perl):

```
fields:
  - name: first_name
    label: "First Name:"
    type: Text
    required: 1
  - name: last_name
    label: "Last Name:"
    type: Text
    required: 1
  - name: cc
    label: "Country Code:"
    type: Text
    required: 1
  - name: ac
    label: "Area Code:"
    type: Text
    required: 1
  - name: sn
    label: "Subscriber Number:"
    type: Text
    required: 1
  - name: sip_username
    type: calculated
    value_code: "sub { return $row{cc}.$row{ac}.$row{sn}; }"
  - name: purge
    label: "Terminate subscriber, if exists:"
    type: Boolean
contract_contact:
  identifier: "firstname, lastname, status"
  reseller: default
  firstname: "sub { return $row{first_name}; }"
  lastname: "sub { return $row{last_name}; }"
  status: "active"
contract:
  product: "Basic SIP Account"
  billing_profile: "Default Billing Profile"
  identifier: contact_id
  contact_id_code: "sub { return $contract_contact{id}; }"
subscriber:
  domain: "example.org"
  primary_number:
    cc_code: "sub { return $row{cc}; }"
    ac_code: "sub { return $row{ac}; }"
    sn_code: "sub { return $row{sn}; }"
    username_code: "sub { return $row{sip_username}; }"
    password_code: "sub { return $row{sip_password}; }"
subscriber_preferences:
  gpp0: "test"
```

Copyright © Sipwise GmbH - All rights reserved

(b.1) Config.yml Template Configuration (JavaScript):

A template can be defined at system level by using the `www_admin.provisioning_templates` property in the *config.yml* file. This template will also be displayed on Admin Panel.

```
www_admin:
  batch_provisioning_features: 1
  provisioning_templates:
    "My First Provisioning Template":
      description: "Create a contract including contact with firstname
and lastname for a single subscriber."
      lang: js
      fields:
        - name: first_name
          label: "First Name:"
          type: Text
          required: 1
        - name: last_name
          label: "Last Name:"
          type: Text
          required: 1
        - name: cc
          label: "Country Code:"
          type: Text
          required: 1
        - name: ac
          label: "Area Code:"
          type: Text
          required: 1
        - name: sn
          label: "Subscriber Number:"
          type: Text
          required: 1
        - name: sip_username
          type: calculated
          value_code: "function() { return
row.cc.concat(row.ac).concat(row.sn); }"
        - name: purge
          label: "Terminate subscriber, if exists:"
          type: Boolean
      contract_contact:
        identifier: "firstname, lastname, status"
        reseller: default
        firstname_code: "function() { return row.first_name; }"
        lastname_code: "function() { return row.last_name; }"
        status: "active"
      contract:
        product: "Basic SIP Account"
        billing_profile: "Default Billing Profile"
        identifier: contact_id
        contact_id_code: "function() { return contract_contact.id; }"
      subscriber:
```


Copyright © Sipwise GmbH - All rights reserved

```
        domain: "example.org"
        primary_number:
          cc_code: "function() { return row.cc; }"
          ac_code: "function() { return row.ac; }"
          sn_code: "function() { return row.sn; }"
        username_code: "function() { return row.sip_username; }"
        password_code: "function() { return row.sip_password; }"
      subscriber_preferences:
        gpp0: "test"
```

Copyright © Sipwise GmbH - All rights reserved

(b.2) Config.yml Template Configuration (Perl):

A template can be defined at system level by using the `www_admin.provisioning_templates` property in the *config.yml* file. This template will also be displayed on Admin Panel.

```
www_admin:
  batch_provisioning_features: 1
  provisioning_templates:
    "My First Provisioning Template":
      description: "Create a contract including contact with firstname
and lastname for a single subscriber."
      fields:
        - name: first_name
          label: "First Name:"
          type: Text
          required: 1
        - name: last_name
          label: "Last Name:"
          type: Text
          required: 1
        - name: cc
          label: "Country Code:"
          type: Text
          required: 1
        - name: ac
          label: "Area Code:"
          type: Text
          required: 1
        - name: sn
          label: "Subscriber Number:"
          type: Text
          required: 1
        - name: sip_username
          type: calculated
          value_code: "sub { return $row{cc}.$row{ac}.$row{sn}; }"
        - name: purge
          label: "Terminate subscriber, if exists:"
          type: Boolean
      contract_contact:
        identifier: "firstname, lastname, status"
        reseller: default
        firstname: "sub { return $row{first_name}; }"
        lastname: "sub { return $row{last_name}; }"
        status: "active"
      contract:
        product: "Basic SIP Account"
        billing_profile: "Default Billing Profile"
        identifier: contact_id
        contact_id_code: "sub { return $contract_contact{id}; }"
      subscriber:
        domain: "example.org"
        primary_number:
```


Copyright © Sipwise GmbH - All rights reserved

```
          cc_code: "sub { return $row{cc}; }"
          ac_code: "sub { return $row{ac}; }"
          sn_code: "sub { return $row{sn}; }"
        username_code: "sub { return $row{sip_username}; }"
        password_code: "sub { return $row{sip_password}; }"
      subscriber_preferences:
        gpp0: "test"
```

Call Forwards Template Example

The following example considers the definition of call forward mappings inside the template. Let us assume that subscribers will be set with the following configuration:

Table 38. Call Forward Mappings Example.

Type	Answer Timeout	Timeset	Sources	To (B-Numbers)	New Destinations	Enabled
Call Forward Busy		always	all sources	any number	123456	Yes
Call Forward Timeout	15s	always	all sources	any number	654321	Yes
Call Forward Unavailable		always	all sources	any number	VoiceMail	Yes

Then, the following `cf_mappings` section can be appended to the batch provisioning template:

```
cf_mappings:
  cfu: []
  cfb:
    - enabled: 1
      destinationset:
        name: "Phone2"
        destinations:
          - destination: "123456"
            priority: 1
            timeout: 300
  cft:
    - enabled: 1
      destinationset:
        name: "Phone3"
        destinations:
          - destination: "654321"
            priority: 1
            timeout: 300
  cft_ringtimeout: 15
  cfna:
    - enabled: 1
      destinationset:
        name: "VoiceMail"
```

Copyright © Sipwise GmbH - All rights reserved

```
        destinations:
          - destination: "voicebox"
            priority: 1
            timeout: 300
  cfs: []
  cfr: []
  cfo: []
```


Copyright © Sipwise GmbH - All rights reserved

Appendix L: Fritzbox Prefix Removal on Sipwise C5

Sipwise C5 offers *fritzbox prefix removal service* to its local subscribers. When activated it will instruct Kamailio proxy to check the request uri username against a list of known prefixes and special numbers; if there is a match, the prefix part is removed. This can be useful in situations where the User Agent Client sends a prefix in front, for example an emergency number. In this case Sipwise C5 can correctly route the emergency call to the emergency peer even if there is a misleading prefix in the incoming request uri.

Configuration

Activation and configuration of the Fritzbox prefix removal can only be done using config.yml:

```
kamailio:
    ...
    proxy:
    ...
    fritzbox:
      enable: yes
      prefixes:
      - 0$avp(caller_ac)
      - $avp(caller_cc)$avp(caller_ac)
      - \+$avp(caller_cc)$avp(caller_ac)
      - 00$avp(caller_cc)$avp(caller_ac)
      - '112'
      - '110'
      - 118[0-9]{2}
      special_numbers:
      - '112'
      - '110'
      - 118[0-9]{2}
      vsc_voice_box_prefix_removal: yes
```

To enable the Fritz box removal service just set the key *enable* to *yes*, then configure *prefixes* and *special_numbers*. In the above example we are asking Sipwise C5 to check the Request Uri username against all combination of prefixes and special numbers. If the regexp matches, the Request Uri is rewritten accordingly, for example:

- incoming request uri: INVITE sip:+431112@voip.test SIP/2.0
- incoming request uri username: +431112
- subscriber country code: 43
- subscriber area code: 1

The following regexp will be built combining *\+$avp(caller_cc)$avp(caller_ac)* from the *prefixes* list and *'112'* from the *special_numbers* list:

```
^(\+431)(112)$
```

Copyright © Sipwise GmbH - All rights reserved

Since the above regexp matches with *+431112* then the prefix is removed. The resulting request uri username will be *INVITE sip:112@voip.test SIP/2.0*.

Notice that the + sign inside prefixes has been escaped with a backslash and this is because this character has a special meaning in regexp. In this way we are asking to match the + sign literally. In other words,the list of prefixes and special numbers expects regexp and not just strings.

The very same method applies to calls to voicebox. To enable this feature the *vsc_voice_box_prefix_removal* must be set to *yes*. When activated, the request uri username is checked against the list of prefix + *voicemail number* , as defined in __. As usual, if there is a match, the prefix part is removed in the resulting request uri, for example:

- Request uri IN: INVITE sip:+4312000@voip.test SIP/2.0
- Request uri OUT: sip:2000@voip.test SIP/2.0

In this scenario typically the request uri domain will change again prefixing with "abc" and setting the domain "voicebox.local".

Copyright © Sipwise GmbH - All rights reserved

Appendix M: Migration from KeyDB to Valkey

Overview

Starting from mr12.5.1 a new noSQL database called Valkey has been introduced. Valkey is an open-source fork of Redis and a powerful drop-in alternative to KeyDB or Redis itself. Valkey introduced a new I/O threading architecture, which improves the parallelism of the system and executes commands more efficiently. All new deployments already have Valkey activated by default, but existing deployments are probably still set to use KeyDB and therefore have to be migrated. The procedure can be done at a later time but it is mandatory because all tests and fixes done for this and future versions will consider Valkey only. To check if your system is already running with Valkey or not, execute the following command:

```
ngcpcfg get "database.key_value.flavor"
```

If the command returns 'valkey' as output, the following steps can be skipped. Otherwise if the command returns nothing or 'keydb' then the migration must be performed.

How to Migrate

Perform the switch:

```
ngcp-redis-flavor valkey
```

Copyright © Sipwise GmbH - All rights reserved