

# The sip:provider PRO Handbook mr5.4.2

Sipwise GmbH <support@sipwise.com>

# Contents

# 1 Introduction

	1.1	About this	s Handbook	1
	1.2	What is th	ne sip:provider PRO?	1
	1.3	The Adva	ntages of the sip:provider PRO	1
	1.4	Who is the	e sip:provider PRO for?	2
	1.5	Getting H	elp	2
		1.5.1 Ph	none Support	2
		1.5.2 Tic	cket System	2
2	Arcl	hitecture		3
	2.1	SIP Signa	aling and Media Relay	4
		2.1.1 SI	P Load-Balancer	4
		2.1.2 SI	P Proxy/Registrar	5
		2.1.3 SI	P Back-to-Back User-Agent (B2BUA)	5
		2.1.4 SI	P App-Server	6
		2.1.5 Me	edia Relay	7
	2.2	MySQL D	Patabase	7
	2.3	Redis Dat	tabase	8
	2.4	High Avai	lability and Fail-Over	8
		2.4.1 Ov	verview	8
		2.4.2 Co	pre Concepts and Configuration	8
		2.4.3 Ac	dministration	9
3	Plat	form Depl	ovment	10
	3.1		Specifications	10
	3.1			
			mensions and Weight	
		3.1.2 Fro	ont View	11
		3.1.3 Re	ear View	12

1

		3.1.4 Power Supply Units (PSU)	13
	3.2	Installation Prerequisites	14
	3.3	Rack-Mount Installation	15
	3.4	Power Supply Cabling	16
	3.5	Network Cabling	17
		3.5.1 Internal Communication	17
		3.5.2 External Communication	18
4	VolF	P Service Configuration Scenario	19
	4.1	Creating a Customer	19
	4.2	Creating a Subscriber	24
	4.3	Domain Preferences	29
	4.4	Subscriber Preferences	31
	4.5	Creating Peerings	32
		4.5.1 Creating Peering Groups	32
		4.5.2 Creating Peering Servers	34
		4.5.3 Authenticating and Registering against Peering Servers	43
	4.6	Configuring Rewrite Rule Sets	46
		4.6.1 Inbound Rewrite Rules for Caller	49
		4.6.2 Inbound Rewrite Rules for Callee	51
		4.6.3 Outbound Rewrite Rules for Caller	52
		4.6.4 Outbound Rewrite Rules for Callee	53
		4.6.5 Emergency Number Handling	53
		4.6.6 Assigning Rewrite Rule Sets to Domains and Subscribers	56
		4.6.7 Creating Dialplans for Peering Servers	57
5	Feat	tures	58
	5.1	Managing System Administrators	58
		5.1.1 Configuring Administrators	58
		5.1.2 Access Rights of Administrators	59

5.2	Acces	s Control for SIP Calls	62
	5.2.1	Block Lists	62
	5.2.2	NCOS Levels	64
	5.2.3	IP Address Restriction	69
5.3	Call Fo	prwarding and Call Hunting	70
	5.3.1	Setting a simple Call Forward	70
	5.3.2	Advanced Call Hunting	71
5.4	Local I	Number Porting	74
	5.4.1	Local LNP Database	75
	5.4.2	External LNP via LNP API	78
5.5	Emerg	ency Mapping	82
	5.5.1	Emergency Mapping Description	84
	5.5.2	Emergency Mapping Configuration	84
5.6	Heade	r Manipulation	90
	5.6.1	Header Filtering	90
	5.6.2	Codec Filtering	91
	5.6.3	Enable History and Diversion Headers	91
5.7	SIP Tr	unking with SIPconnect	92
	5.7.1	User provisioning	92
	5.7.2	Inbound calls routing	92
	5.7.3	Number manipulations	92
	5.7.4	Registration	95
5.8	Truste	d Subscribers	95
5.9	Fax Se	erver	96
	5.9.1	Fax2Mail Architecture	96
	5.9.2	Sendfax and Mail2Fax Architecture	97
5.10	Voicen	nail System	97
	5.10.1	Accessing the IVR Menu	97

5.10.2 IVR Menu Structure	3
5.10.3 Type Of Messages	)
5.10.4 Folders	)
5.10.5 Flowcharts with Voice Prompts	1
5.11 Configuring Subscriber IVR Language	5
5.12 Sound Sets	5
5.12.1 Configuring Early Reject Sound Sets	7
5.13 Conference System	l
5.13.1 Configuring Call Forward to Conference	1
5.13.2 Configuring Conference Sound Sets	2
5.13.3 Joining the Conference	1
5.13.4 Conference Flowchart with Voice Prompts	1
5.14 Malicious Call Identification (MCID)	5
5.14.1 Setup	5
5.14.2 Usage	7
5.14.3 Advanced configuration	7
5.15 Subscriber Profiles	7
5.15.1 Subscriber Profile Sets	7
5.16 SIP Loop Detection	)
5.17 Call-Through Application	)
5.17.1 Administrative Configuration	1
5.17.2 Call Flow	3
5.18 Calling Card Application	1
5.18.1 Administrative Configuration	5
5.18.2 Call Flow	7
5.19 Invoices and Invoice Templates	3
5.19.1 Invoices Management	3
5.19.2 Invoice Templates	)

		5.19.3 Invoices Generation	140
	5.20	Email Reports and Notifications	142
		5.20.1 Email events	142
		5.20.2 Initial template values and template variables	142
		5.20.3 Password reset email template	142
		5.20.4 New subscriber notification email template	143
		5.20.5 Invoice email template	144
		5.20.6 Email templates management	145
	5.21	The Vertical Service Code Interface	147
		5.21.1 Vertical Service Codes for PBX customers	147
		5.21.2 Configuration of Vertical Service Codes	148
		5.21.3 Voice Prompts for Vertical Service Code Configuration	148
	5.22	2 Handling WebRTC Clients	149
	5.23	3 XMPP and Instant Messaging	150
6	Cus	stomer Self-Care Interface and Menus	151
6	<b>Cus</b> 6.1	The Customer Self-Care Web Interface	-
6			151
6		The Customer Self-Care Web Interface	151 151
6	6.1	The Customer Self-Care Web Interface	151 151 151
	6.1	The Customer Self-Care Web Interface	151 151 151
	6.1	The Customer Self-Care Web Interface	<ol> <li>151</li> <li>151</li> <li>151</li> <li>157</li> <li>158</li> </ol>
	6.1 6.2 Billi	The Customer Self-Care Web Interface   6.1.1 Login Procedure   6.1.2 Site Customization   The Voicemail Menu	<ol> <li>151</li> <li>151</li> <li>151</li> <li>157</li> <li>158</li> <li>158</li> </ol>
	6.1 6.2 Billi	The Customer Self-Care Web Interface   6.1.1 Login Procedure   6.1.2 Site Customization   The Voicemail Menu	151 151 151 157 <b>158</b> 158 158
	6.1 6.2 Billi	The Customer Self-Care Web Interface   6.1.1 Login Procedure   6.1.2 Site Customization   The Voicemail Menu     Ing Configuration   Billing Profiles   7.1.1 Creating Billing Profiles	151 151 151 157 <b>158</b> 158 158 160
	6.1 6.2 Billi	The Customer Self-Care Web Interface   6.1.1 Login Procedure   6.1.2 Site Customization   The Voicemail Menu   Ing Configuration   Billing Profiles   7.1.1 Creating Billing Profiles   7.1.2 Creating Billing Fees	<ul> <li>151</li> <li>151</li> <li>151</li> <li>151</li> <li>157</li> <li>158</li> <li>158</li> <li>160</li> <li>162</li> </ul>
	<ul><li>6.1</li><li>6.2</li><li>Billi</li><li>7.1</li></ul>	The Customer Self-Care Web Interface   6.1.1   Login Procedure   6.1.2   Site Customization   The Voicemail Menu   Ing Configuration   Billing Profiles   7.1.1   Creating Billing Profiles   7.1.2   Creating Billing Fees   7.1.3   Creating Off-Peak Times	151 151 151 157 <b>158</b> 158 158 160 162 164
	<ul> <li>6.1</li> <li>6.2</li> <li>Billi</li> <li>7.1</li> <li>7.2</li> </ul>	The Customer Self-Care Web Interface   6.1.1 Login Procedure   6.1.2 Site Customization   The Voicemail Menu   Ing Configuration   Billing Profiles   7.1.1 Creating Billing Profiles   7.1.2 Creating Billing Fees   7.1.3 Creating Off-Peak Times   Prepaid Accounting	<ol> <li>151</li> <li>151</li> <li>151</li> <li>157</li> <li>158</li> <li>158</li> <li>158</li> <li>160</li> <li>162</li> <li>164</li> <li>165</li> </ol>
	<ul> <li>6.1</li> <li>6.2</li> <li>Billi</li> <li>7.1</li> <li>7.2</li> </ul>	The Customer Self-Care Web Interface   6.1.1   Login Procedure   6.1.2   Site Customization   The Voicemail Menu   Ing Configuration   Billing Profiles   7.1.1   Creating Billing Profiles   7.1.2   Creating Billing Fees   7.1.3   Creating Off-Peak Times   Prepaid Accounting   Fraud Detection and Locking	<ol> <li>151</li> <li>151</li> <li>151</li> <li>157</li> <li>158</li> <li>158</li> <li>158</li> <li>160</li> <li>162</li> <li>164</li> <li>165</li> <li>165</li> </ol>

		7.4.1	Billing Networks	67
		7.4.2	Profile Mapping Schedule	69
		7.4.3	Profile Packages	72
		7.4.4	Vouchers	83
		7.4.5	Тор-ир	86
		7.4.6	Balance Overviews	87
		7.4.7	Usage Examples	91
	7.5	Billing [	Data Export	93
		7.5.1	Glossary of Terms	94
		7.5.2	File Name Format	94
		7.5.3	File Format	95
		7.5.4	File Transfer	207
8	Prov	visionin	g REST API Interface 2	08
	8.1	API Wo	rkflows for Customer and Subscriber Management	208
9		-		14
9	<b>Con</b> 9.1	Configu	rration templates	214
9		Configu		214
9		Configu 9.1.1	rration templates	214
9		Configu 9.1.1 9.1.2	Iration templates	214 214 215
9		Configu 9.1.1 9.1.2 9.1.3	uration templates       2         .tt2 and .customtt.tt2 files       2         .prebuild and .postbuild files       2	214 214 215 216
9	9.1	Configu 9.1.1 9.1.2 9.1.3 config.y	uration templates       2         .tt2 and .customtt.tt2 files       2         .prebuild and .postbuild files       2         .services files       2	214 214 215 216 217
9	9.1	Configu 9.1.1 9.1.2 9.1.3 config.y ngcpcfg	uration templates       2         .tt2 and .customtt.tt2 files       2         .prebuild and .postbuild files       2         .services files       2         .ml, constants.yml and network.yml files       2	214 214 215 216 217 217
9	9.1	Configu 9.1.1 9.1.2 9.1.3 config.y ngcpcfg 9.3.1	uration templates       2         .tt2 and .customtt.tt2 files       2         .prebuild and .postbuild files       2         .services files       2         vml, constants.yml and network.yml files       2         .g and its command line options       2	214 214 215 216 217 217
9	9.1	Configu 9.1.1 9.1.2 9.1.3 config.y ngcpcfg 9.3.1 9.3.2	arration templates       2         .tt2 and .customtt.tt2 files       2         .prebuild and .postbuild files       2         .services files       2         .ml, constants.yml and network.yml files       2         .g and its command line options       2         .apply       2	214 214 215 216 217 217 217
9	9.1	Configu 9.1.1 9.1.2 9.1.3 config.y ngcpcfg 9.3.1 9.3.2 9.3.3	uration templates       2         .tt2 and .customtt.tt2 files       2         .prebuild and .postbuild files       2         .services files       2         rml, constants.yml and network.yml files       2         g and its command line options       2         apply       2         build       2	214 214 215 216 217 217 217 217
9	9.1	Configu 9.1.1 9.1.2 9.1.3 config.y ngcpcfg 9.3.1 9.3.2 9.3.3 9.3.4	uration templates       2         .tt2 and .customtt.tt2 files       2         .prebuild and .postbuild files       2         .services files       2         rml, constants.yml and network.yml files       2         g and its command line options       2         apply       2         build       2         commit       2	214 214 215 216 217 217 217 217 217 218
9	9.1	Configu 9.1.1 9.1.2 9.1.3 config.y ngcpcfg 9.3.1 9.3.2 9.3.3 9.3.4 9.3.5	uration templates       2         .tt2 and .customtt.tt2 files       2         .prebuild and .postbuild files       2         .services files       2         ml, constants.yml and network.yml files       2         g and its command line options       2         apply       2         build       2         commit       2         decrypt       2	214 214 215 216 217 217 217 217 217 217 218 218

9	9.3.8 initialise	18
9	9.3.9 pull	18
9	9.3.10 push	18
9	9.3.11 services	18
9	9.3.12 status	19
10 Netw	ork Configuration 22	20
10.1	General Structure	20
	10.1.1 Available Host Options	21
	10.1.2 Interface Parameters	22
10.2	Advanced Network Configuration	23
	10.2.1 Extra SIP Sockets	23
	10.2.2 Extra SIP and RTP Sockets	24
11 Softw	vare Upgrade 22	26
11.1	Release Notes	26
11.2	Preparing to a Software Upgrade	26
11.3	Jpgrading the sip:provider PRO	28
	11.3.1 Preparing for maintenance mode	28
	11.3.2 Switch to new repositories	30
	11.3.3 Upgrade the first PRO node	30
	11.3.4 The customtt files handling (if necessary)	31
	11.3.5 Promote inactive node to active	31
	11.3.6 Upgrade the second PRO node	31
11.4	Post-upgrade tasks	31
	11.4.1 Disabling maintenance mode	31
	11.4.2 Post-upgrade checks	32
12 Back	up, Recovery and Database Maintenance 23	34
12.1 :	sip:provider PRO Backup	34

		12.1.1 What data to back up	234
		12.1.2 The built-in backup solution	234
	12.2	Recovery	235
	12.3	Reset Database	235
	12.4	Synchronize database	235
	12.5	Accounting Data (CDR) Cleanup	237
		12.5.1 Cleanuptools Configuration	237
		12.5.2 Accounting Database Cleanup	238
		12.5.3 Exported CDR Cleanup	241
13	Plat	form Security, Performance and Troubleshooting	242
	13.1	Sipwise SSH access to sip:provider PRO	242
	13.2	Firewalling	242
		13.2.1 Firewall framework	242
		13.2.2 NGCP firewall configuration	244
		13.2.3 IPv4 System rules	244
		13.2.4 Custom rules	247
		13.2.5 Example firewall configuration section	248
	13.3	Password management	248
	13.4	SSL certificates.	249
	13.5	Securing your sip:provider PRO against SIP attacks	250
		13.5.1 Denial of Service	250
		13.5.2 Bruteforcing SIP credentials	251
	13.6	System Requirements and Performance	251
	13.7	Troubleshooting	254
		13.7.1 Collecting call information from logs	256
		13.7.2 Collecting SIP traces	257
14	Mon	itoring and Alerting	259
	14.1	Internal Monitoring	259

	14.1.1 Process monitoring via monit	259
	14.1.2 System monitoring via Telegraf	259
	14.1.3 NGCP-specific monitoring via ngcp-witnessd	259
	14.1.4 Monitoring data in InfluxDB	260
14	.2 Monitoring data in Redis	260
14	.3 Statistics Dashboard	261
14	.4 External Monitoring Using SNMP	261
	14.4.1 Overview and Initial Setup	261
	14.4.2 Details	261
15 Ex	ttensions and Additional Modules	267
15	6.1 Cloud PBX	267
10	15.1.1 Configuring the Device Management	
	15.1.2 Preparing PBX Rewrite Rules	
	15.1.3 Creating Customers and Pilot Subscribers	
	15.1.4 Creating Regular PBX Subscribers	290
	15.1.5 Assigning Subscribers to Devices	296
	15.1.6 Configuring Sound Sets for the Customer PBX	302
	15.1.7 Configuring Auto Attendant	304
	15.1.8 Configuring Call Queues	309
	15.1.9 Device Auto-Provisioning Security	311
	15.1.10Device Bootstrap and Resync Workflows	313
	15.1.11Device Provisioning and Deployment Workflows	321
	15.1.12List of available pre-configured devices	324
	15.1.13Phone features	328
15	i.2 Sipwise sip:phone App (SIP client)	359
	15.2.1 Zero Config Launcher	360
	15.2.2 Mobile Push Notification	364
15	5.3 Lawful Interception	385

		15.3.1 Introduction	385
		15.3.2 Architecture and Configuration of LI Service	387
		15.3.3 X1, X2 and X3 Interface Specification	393
A	Bas	ic Call Flows	408
	A.1	General Call Setup	408
	A.2	Endpoint Registration	409
	A.3	Basic Call	412
	A.4	Session Keep-Alive	413
	A.5	Voicebox Calls	414
в	NGC	CP configs overview	416
	B.1	config.yml Overview	416
		B.1.1 apps	
		B.1.2 asterisk	
		B.1.3 autoprov	
		B.1.4 backuptools	
		B.1.5 cdrexport	
		B.1.6 checktools	
		B.1.7 cleanuptools	
			423
			-
		B.1.13 intercept	
		B.1.17 modules	
		Ditty modulos	-100

С

	B.1.18 nginx	430
	B.1.19 ntp	431
	B.1.20 ossbss	431
	B.1.21 pbx (only with additional cloud PBX module installed)	433
	B.1.22 prosody	433
	B.1.23 pushd	433
	B.1.24 qos	434
	B.1.25 rate-o-mat	435
	B.1.26 redis	435
	B.1.27 reminder	435
	B.1.28 rsyslog	436
	B.1.29 rtpproxy	436
	B.1.30 security	438
	B.1.31 sems	439
	B.1.32 snmpagent	440
	B.1.33 sshd	440
	B.1.34 voisniff	440
	B.1.35 www_admin	441
B.2	constants.yml Overview	443
B.3	network.yml Overview	444
NGC	CP-Faxserver Configuration	446
C.1	Faxserver Components	446
C.2	Enabling Faxserver	446
C.3	Fax Templates Configuration	446
C.4	Fax Services Configuration per Subscriber	447
	Fax2Mail and SendFax Settings	
C.6	Mail2Fax Settings	449
C.7	Sending Fax from Web Panel	450

	C.8	Faxserver Mail2Fax Configuration    451
	C.9	Sending Fax Using E-mail Clients
	C.10	Managing Faxes via the REST API
		C.10.1 Configuring Fax Settings
		C.10.2 Sending a Fax
		C.10.3 Receiving a Fax
		C.10.4 Configuring Mail2Fax Settings
		C.10.5 Using Advanced Faxserver and Mail2Fax Settings via the REST API
	C.11	Troubleshooting
		C.11.1 Session ID (SID)
		C.11.2 Fax Storage Location
D	RTC	tengine 458
	D.1	Overview
	D.2	RTC:engine enabling
		D.2.1 Enabling services via CLI
		D.2.2 Enabling via Panel for resellers and subscribers
		D.2.3 Create RTC:engine session
	D.3	RTC:engine protocol details
		D.3.1 Terminology
		D.3.2 Messages
		D.3.3 Account
		D.3.4 Call
		D.3.5 Session
E	com	ax-fileshare-service 475
	E.1	Overview
	E.2	Configuration and Usage
		E.2.1 Change authentication method
		E.2.2 Database Structure

E.3	Activation of Filesharing Service on NGCP	1
E.4	Message Sequence Chart	3
	E.4.1 Simple Message Sequence	3
	E.4.2 Detailed Message Sequence	)
E.5	API of Filesharing Service	)
	E.5.1 HTTP Authentication	¢
	E.5.2 Upload and Download with Simple Identification	)
	E.5.3 Upload and Download with Session Identification	)
	E.5.4 Curl Example for Simple Upload Request	)
	E.5.5 Upload Parameters	)
	E.5.6 Number of Possible Downloads	<u>)</u>
F NG	CP Internals 483	2
1 110		-
F.1	Pending reboot marker	;
F.2	Redis id constants	;
	F.2.1 Redis monitoring keys	ł
F.3	Enum preferences	5

# 1 Introduction

# 1.1 About this Handbook

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

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

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

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

# 1.2 What is the sip:provider PRO?

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

The sip:provider PRO comes pre-installed on two servers. Apart from your product specific configuration, there is no initial configuration or installation to be done to get started.

# 1.3 The Advantages of the sip:provider PRO

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

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

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

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

# 1.4 Who is the sip:provider PRO for?

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

# 1.5 Getting Help

# 1.5.1 Phone Support

Depending on your support contract, you are eligible to contact our Support Team by phone either in business hours or around the clock. Business hours refer to the UTC+1 time zone (Europe/Vienna). Please check your support contract to find out the type of support you've purchased.

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

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

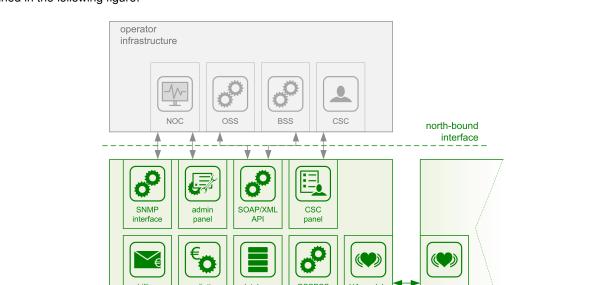
# 1.5.2 Ticket System

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

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

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

# 2 Architecture



OSSBSS

۵)

SIP

signalling

HA module

sip:provider node 2

SIP peerings

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

Figure 1: Architecture Overview

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

billing

sip:provider node 1

SIP endpoints mediation

database

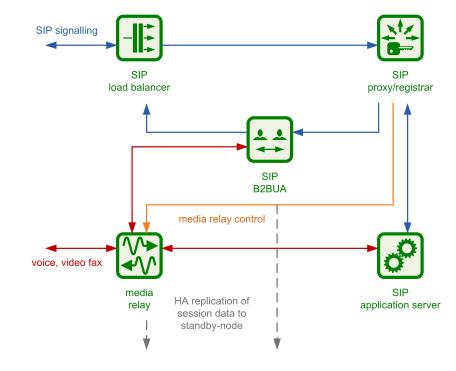
**√**∕

media relav

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

# 2.1 SIP Signaling and Media Relay

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



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

Figure 2: SIP and Media Relay Components

# 2.1.1 SIP Load-Balancer

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

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

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

On the public interface, the load-balancer listens on port 5060 for UDP and TCP, as well as on 5061 for TLS connections. On the

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*.

# Тір

The SIP load-balancer can be managed via the commands monit start lb, monit stop lb and monit restart lb. Its status can be queried by executing monit summary | grep 'lb'. Also ngcp-kamctl lb and ngcp-sercmd lb are provided for querying kamailio functions, for example: ngcp-sercmd lb htable.dump ipban.

# 2.1.2 SIP Proxy/Registrar

The SIP proxy/registrar (or short *proxy*) is the work-horse of the sip:provider PRO. It's also a separate Kamailio instance running in the switch-internal network and is connected to the provisioning database via MySQL, authenticates the endpoints, handles their registrations on the system and does the call routing based on the provisioning data. For each call, the proxy looks up the provisioned features of both the calling and the called party (either subscriber or domain features if it's a local caller and/or callee, or peering features if it's from/to an external endpoint) and acts accordingly, e.g. by checking if the call is blocked, by placing call-forwards if applicable and by normalizing numbers into the appropriate format, depending on the source and destination of a call.

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

If the endpoints indicate negotiation of one or more media streams, the proxy also interacts with the *Media Relay* to open, change and close port pairs for relaying media streams over the sip:provider PRO, 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.

# Тір

The SIP proxy can be controlled via the commands monit start proxy, monit stop proxy and monit restart proxy. Its status can be queried by executing monit summary | grep 'proxy'. Also ngcp-kamctl proxy and ngcp-sercmd proxy are provided for querying kamailio functions, for example: ngcp-kamctl proxy ul show.

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

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

The software part used for this element is a commercial version of SEMS, with the main difference to the open-source version that

it includes a replication module to share its call states with the stand-by node.

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

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

The SIP B2BUA only binds to the system-internal network and listens on UDP port 5080 for SIP messages from the load-balancer or the proxy, on UDP port 5040 for control messages from the cli tool and on TCP port 8090 for XMLRPC connections from the OSSBSS to control the daemon.

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

# Тір

The SIP B2BUA can be controlled via the commands monit start sbc, monit stop sbc and monit restart sbc. Its status can be queried by executing monit summary | grep 'sbc'

# 2.1.4 SIP App-Server

The SIP App-Server is an Asterisk instance used for voice applications like Voicemail and Reminder Calls. It is also used in the software-based Faxserver solution to transcode SIP and RTP into the IAX protocol and vice versa, in order to talk to the Software Fax Modems. Asterisk uses the MySQL database as a message spool for voicemail, so it doesn't directly access the file system for user data. The voicemail plugin is a slightly patched version based on Asterisk 1.4 to make Asterisk aware of the sip:provider PRO internal UUIDs for each subscriber. That way a SIP subscriber can have multiple E164 phone numbers, but all of them terminate in the same voicebox.

The App-Server listens on the internal interface on UDP port 5070 for SIP messages and by default uses media ports in the range from UDP port 10000 to 20000.

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

#### Tip

The SIP App-Server can be controlled via the commands monit start asterisk, monit stop asterisk and monit restart asterisk. Its status can be queried by executing monit summary | grep 'asterisk'

#### 2.1.5 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 40000. 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 rtpproxy. The QoS/ToS value can be changed via the key qos.tos\_rtp.

#### Tip

The Media Relay can be controlled via the commands monit start rtpengine, monit stop rtpengine and monit restart rtpengine. Its status can be queried by executing monit summary | grep 'rtpengine'

# 2.2 MySQL Database

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

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

# 2.3 Redis Database

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

The active-standby replication ensures that the data is immediately copied from the active node to the standby one. As all sensitive call information is held in the shared storage, sip:provider PRO allows to switch the operational state from active to standby on one physical node and from standby to active on the other node without any call interruptions. Your subscribers will never notice that their calls being established on one physical server, were successfully moved to another one and successfully completed there.

# 2.4 High Availability and Fail-Over

# 2.4.1 Overview

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

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

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

# Тір

The login banner at the SSH shell provides information about whether the local system is currently the active one or the standby one. See Section 2.4.3 for other ways to differentiate between the active and the standby node.

# 2.4.2 Core Concepts and Configuration

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

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

The main resource that *heartbeat* manages is the shared service IP address. Each node has its own static IP address configured on its first Ethernet interface (eth0), which is done outside of the sip:provider PRO configuration framework (i.e. in the Debianspecific config file /etc/network/interfaces). The shared service IP is specified in network.yml at the key hosts. sp1|sp2.eth0.shared\_ip. *Heartbeat* will configure it as a secondary IP address on the first Ethernet interface (eth0:0) on the active node and will deconfigure it on the standby node. Thus, all network communications with this IP address will always go only to the currently active node.

# 2.4.3 Administration

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

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

# 3 Platform Deployment

This chapter will provide the step by step instructions on how to put the sip:provider PRO into operations.

# 3.1 Hardware Specifications

Sipwise provides the sip:provider PRO platform fully pre-installed on two Dell PowerEdge R330 servers. Their most important characteristics are:

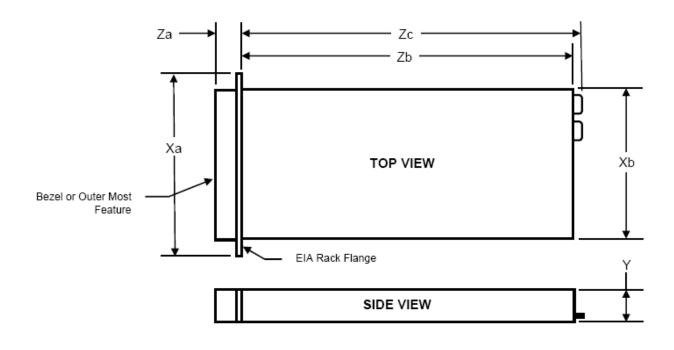
- Up to 8 pcs. of 2.5" storage drives (HDD or SSD); shipped with 4 drives installed and configured as RAID10 array
- · Gbit Ethernet ports: 2 on-board and 2 additional ports (optional)
- iDRAC module for remote maintenance

# Note

Please be aware that prior to Q3 2016 Sipwise used to provide its sip:provider PRO platform on older Dell PowerEdge server models: R310 and R320.

# 3.1.1 Dimensions and Weight

The hardware **dimensions** are defined in the following figure:

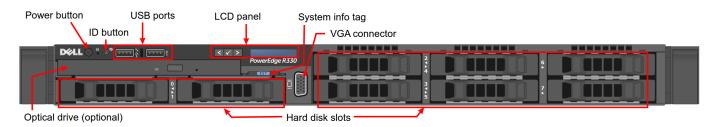


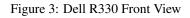
Ха	Xb (Width)	Y (Height)	Za w/ bezel	Za w/o bezel	Zb (Depth)	Zc
482.4mm	434mm	42.8mm	35mm	21mm	610mm	639.5mm

Weight of the server with storage drives and internal components installed: 13.4kg

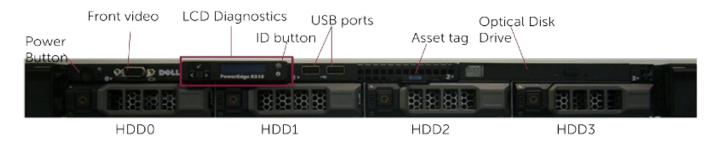
# 3.1.2 Front View

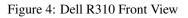
The front view of a current sip:provider PRO Dell R330 server:





# The front view of a former sip:provider PRO Dell R310...:





... and Dell R320 server:

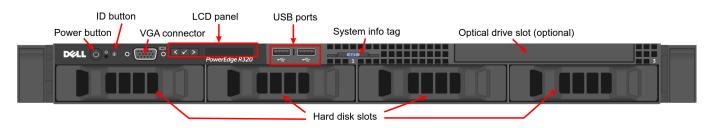
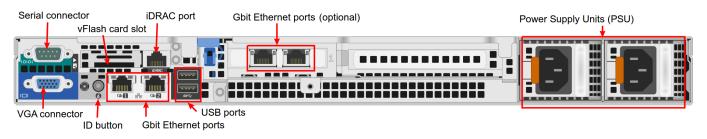
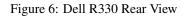


Figure 5: Dell R320 Front View

# 3.1.3 Rear View

The rear view of a current sip:provider PRO Dell R330 server:





The rear view of a former sip:provider PRO Dell R310...:

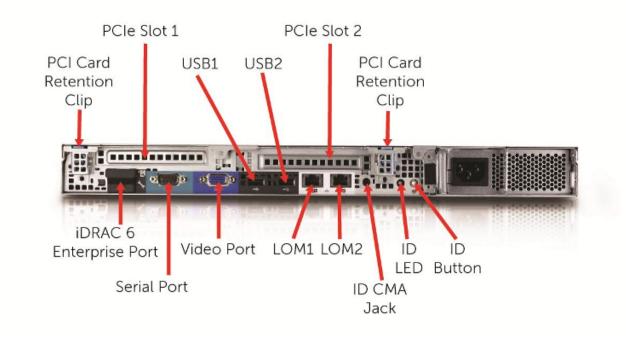


Figure 7: Dell R310 Rear View

... and Dell R320 server:

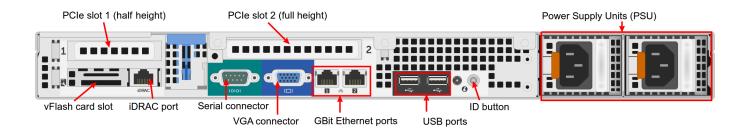


Figure 8: Dell R320 Rear View

# 3.1.4 Power Supply Units (PSU)

The servers are equipped with 2 redundant, hot-swappable PSUs, which are accessible from the rear side and located on the right of the chassis:



# Figure 9: Redundant PSUs

The redundant PSUs include LEDs that indicate the status of the PSU:

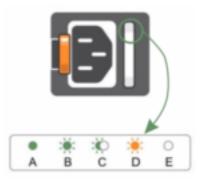


Figure 10: PSU Indicators

- A. The indicator is solidly lit green: A valid power source is connected to the PSU and the PSU is operational.
- B. The indicator is flashing green: The PSU firmware is being updated.

# Caution

Do not disconnect the power cord or unplug the PSU when updating the firmware. If a firmware update is interrupted, the PSUs will not function. You must roll back the PSU firmware by using Dell Lifecycle Controller. For more information, see Dell Lifecycle Controller User's Guide at Dell.com/idracmanuals.

- C. The indicator is flashing green and turns off: When hot-adding a PSU, the PSU handle flashes green five times at 4 Hz rate and turns off. This indicates that there is a PSU mismatch with respect to efficiency, feature set, health status, and supported voltage. Ensure that both the PSUs are the same.
- D. The indicator is flashing amber: Indicates a problem with the PSU.

When correcting a PSU mismatch, replace only the PSU with the flashing indicator. Swapping the other PSU to make a matched pair can result in an error condition and unexpected system shutdown. To change from a High Output configuration to a Low Output configuration or vice versa, you must turn off the system.

# Caution

Caution

AC PSUs support both 220 V and 110 V input voltages with the exception of Titanium PSUs, which support only 220 V. When two identical PSUs receive different input voltages, they can output different wattages, and trigger a mismatch.

# Caution

If two PSUs are used, they must be of the same type and have the same maximum output power.



Combining AC and DC PSUs is not supported and triggers a mismatch.

E. The indicator is not lit: Power is not connected.

# 3.2 Installation Prerequisites

In order to put the sip:provider PRO into operations, you need to rack-mount it into 19" racks.

You will find the following equipment in the box:

2 servers

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

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

• 4 power cables

#### Note

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

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

# 3.3 Rack-Mount Installation

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

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

The following figure shows the mounted rails:

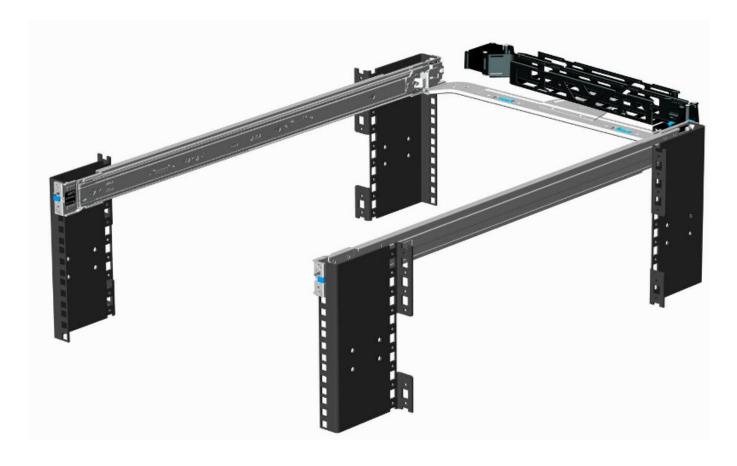


Figure 11: Rack-mounted Rails

# 3.4 Power Supply Cabling

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

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



Figure 12: Proper PSU Cabling

# 3.5 Network Cabling

# **Internal Communication**

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

# **External Communication**

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

# 3.5.1 Internal Communication

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

# Important

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

# Note

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

# 3.5.2 External Communication

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

# 4 VoIP Service Configuration Scenario

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

# Тір

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



# Warning

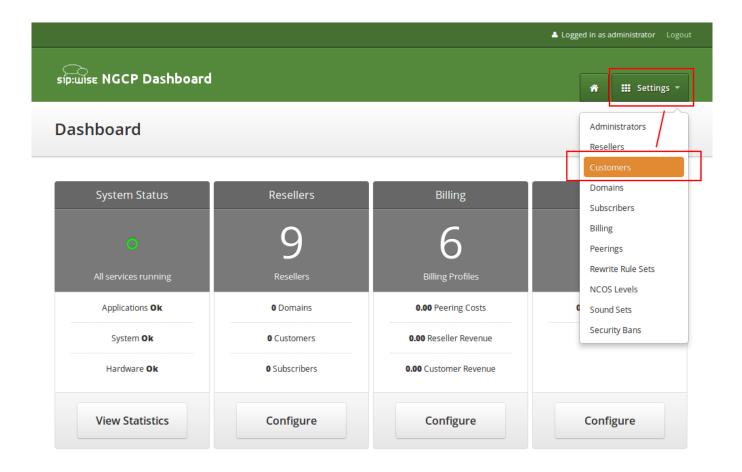
Once you started adding subscribers to a SIP domain, and later decide to change the domain, e.g. from 1.2.3.4 to sip.yourdomain.com, you'll need to recreate all your subscribers in this new domain. It's currently not possible to easily change the domain part of a subscriber.

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

# 4.1 Creating a Customer

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

To create a Customer, got to Settings  $\rightarrow$  Customers.



Click on Create Customer.

				Logged in as administrator Logout
sip:wise NGCP Dashbo	ard			★ Settings ▼
Customers				
← Back ← Create	Customer		Search:	
# 🔶 External #	Reseller	Contact Email	Billing Profile	Status
No data available in table				
Showing 0 to 0 of 0 entries				$\Leftarrow  \leftarrow  \rightarrow  \Rightarrow$
© 2013 Sipwise GmbH, all rights re	served.			

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

Cre	eate Contract						× tings •
ontr	Contact				Sear	ch:	
	contact	#	Reseller	First Name	Last Name	Email	
		1	default	Contact first name	Contact last name	default- customer@default.invalid.contact	
Back		Showing	1 to 1 of 1 entrie	s		← ← 1 → = Create Contact	
-	Billing Profile		Search:				
o data a		#	# Reseller			Profile	
		1		default		Default Billing Profile	
owing 01		Showing	1 to 1 of 1 entrie	S		⇐ ← 1 → =	•

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

					🚢 Logg	ed in as administr	ator Logout
æ _							
síp:wíse Cr	eate Contact					×	tings 👻
Conta	Reseller			Search:			<u>^</u>
		#	Name	Contract #	Status		
		1	default	1	active	1 💌	
← Back		Showing 1 to 1 o	of 1 entries		← ← 1	$\rightarrow$ $\Rightarrow$	
					Create	e Reseller	
# -	First Name						Ξ
1 (	Last Name						
Showing 1	2 Email	myfirstcontact	@example.org				
Showing Th	Company						
© 2013 Sip					3	B Save	]

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

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

					Logged in as administrator Logout					
	P Dashboard				🔒 🇮 Settings 👻					
Customers										
← Back ★ Create Customer Contract successfully created										
# 🔶 External i	Reseller	Contact Email	Billing Profile	Status	Search:					
20	default	myfirstcontact@example.org	Default Billing Profile	active	🛛 Edit 🗙 Terminate 🗮 Details					
Showing 1 to 1 of 1 e	ntries				← 1 → ⇒					

# 4.2 Creating a Subscriber

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

	▲ Logged in as administrator Li								
	sip:wise NGCP Dash	board		🕈 🏭 Settings 🔹					
	Customer Detai	ls							
	← Back								
	Reseller								
	Contact Details								
	Billing Profiles								
1	Subscribers								
	★ Create Subscrib	per 2							
	SIP URI	Primary Number	Registered Devices						
	Contract Balance								

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

						*		
síp:ωise	Create Sub	scriber						× tings -
Custo	Web	Username						
	Web Password							
← Back	E164 Number							=
Reseller	SIP	Username						
Contact	1	Domain			Search:			
Billing			#	Reseller	C	Domain		
Subscri			No data available in table					
			Showing 0 to 0 of 0 entries			4	$\leftarrow  \rightarrow  \Rightarrow $	
*0						2	Create Domain	
SIP UR								
Contrac							Save	
Contrac								

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

<u>_</u>							
sip:ພາ່ຣະ	Create	Domain					🗙 tings 👻
Doma		Reseller			Search:		
			#	Name	Contract #	Status	
			1	default	1	active 1	
🗲 Back			Showing 1 to 1 of 1 en	ntries		← ← 1	→ ⇒
						Create	Reseller
#	2	SIP Domain	1.2.3.4				
No data a						3	Save -
© 2013 Sipwi	ise GmbH, a	ll rights reserved.					

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

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



## Caution

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

- Web Password: This is the password for the subscriber to log into her *Customer Self Care Interface*. It must be at least 6 characters long.
- E164 Number: This is the telephone number mapped to the subscriber, separated into *Country Code (CC)*, *Area Code (AC)* and *Subscriber Number (SN)*. For the first tests, you can set a made-up number here and change it later when you get number blocks assigned by your PSTN interconnect partner. So in our example, we'll use 43 as CC, 99 as AC and 1001 as SN to form the phantasy number +43 99 1001.

## Тір

This number can actually be used to place calls between local subscribers, even if you don't have any PSTN interconnection. This comes in handy if you use phones instead of soft-clients for your tests. The format in which this number can be dialled so the subscriber is reached is defined in Section 4.6.

## Important

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

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

sip:wise NGCP Dashboard								Settings 👻	
Custo	Create	e Subscriber					*		
		Web Password							
← Back	1	E164 Number	43 99	1001					
Domain suc	2	SIP Username	testuser1						
Reseller				Search:	Search:		=		
Reserve			#	Reseller	Domain				
Contact			6	default	1.2.3.4	3			
Billing			Showing 1 to 1 of 1 entrie	25		← 1	$\rightarrow$ $\Rightarrow$		
						Create	Domain		
*0	4	SIP Password	mysecretpassword						
SIP UR		<b>-</b> ···						•	
Contract	ванансе					5	Save		

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

#### Tip

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

You should now revise the Domain and Subscriber Preferences.

## 4.3 Domain Preferences

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

			Logged in as administrator Logout			
sip:wise N	GCP Dashboard		흄 🇮 Settings 👻			
Domair	าร					
← Back	★ Create Domain		Search:			
#	* Reseller	Domain				
6	default	1.2.3.4	Delete			
Showing 1 to 1 o	of 1 entries		$\leftarrow$ $1 \rightarrow \Rightarrow$			
© 2013 Sipwise	e GmbH, all rights reserved.					

The most important settings are in the Number Manipulations group.

Here you can configure the following:

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

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

To assign a *Rewrite Rule Set* to a *Domain*, create a set first as described in Section 4.6, then assign it to the domain by editing the *rewrite\_rule\_set* preference.

	Domain "1.2.3.4" – Preferences										
	← Back										
	Call Blockings										
	Access Restrictions										
1	Number	Number Manipulations									
		Name	Value								
	0	rewrite_rule_set	•	2 Edit							
	0	extension_in_npn									
	0	inbound_upn	From-Username •								
	0	outbound_from_user	User-Provided-Number								
	0	outbound_from_display	None								

Select the Rewrite Rule Set and press Save.

s	ip:wise	Edit Preference rewrite_ru	e_set	× tings +						
	oma Back	1 rewrite_rule_set test		2 Save						
	Call Blockings Access Restrictions Number Manipulations									
		Name	Value							
	0	rewrite_rule_set	•							
	0	extension_in_npn								
	0	inbound_upn	From-Username							

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.

## 4.4 Subscriber Preferences

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

To configure your *Subscriber*, go to *Settings* $\rightarrow$ *Subscribers* and click *Details* on the row of your subscriber. There, click on the *Preferences* button on top.

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

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

#### Note

Subscribers preference *allowed\_clis* will be synchronized with subscribers primary number and aliases if *oss-bss*→*provisioning*→*auto\_allow\_cli* is set to **1** in */etc/ngcp-config/config.yml*.

## Note

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

## 4.5 Creating Peerings

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

## 4.5.1 Creating Peering Groups

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

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

cre	ate SIP Peering (	Groups			×	tings 👻
P Pe	Contract			Search:		
		#	Status	Billing Profile		
		No data available in table				
Back		Showing 0 to 0 of 0 entries			$\Leftarrow \qquad \leftarrow \qquad \rightarrow \qquad \Rightarrow$	
					Create Contract	]
	Name					
) data a	Priority	1			T	
wing 01	Description					-
					Save	

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

Logged in as administrator Logout								
Å.								
síp:wise	Create Contract						× tings -	
Contr	Contact				Searc	ch:		
		#	Reseller	First Name	Last Name	Email		
		1	default	Contact first name	Contact last name	default- 1.1 customer@default.invalid.contact	8	
🗲 Back		17	default			myfirstcontact@example.org		
		Showing 1 to 2 of 2 entries					⇒ <sup>≡</sup>	
# -						or 1.2 Create Contac		
21	Billing Profile				Search:			
		#		Reseller		Profile		
Showing 1		1		default		Default Billing Profile 2	→ <b>⇒</b>	
		Showing 1	I to 1 of 1 entries	;		⇐ ← 1 → =	⇒	
© 2013 Sip	nas anna feir i Sue i Sae i sei					3	ave	

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

							Logged in as administra	ator Logout
sip:wise	Create	SIP Pee	ring G	iroups			×	tings 👻
SIP Pe		Con	ntract			Search:		
				#	Status	Billing Profile		
				21	active	Default Billing Profile	1 💌	
← Back				Showing 1 to 1 of 1 entrie	s		← ← 1 → ⇒ Create Contract	
#		2	Name	test group				
No data a		3 Pr	iority	1			•	
Showing 01	4	Descri	ption	peering to a test carrie	er			-
							Save	
© 2013 Sipv	vise GmbH <b>, al</b>	l rights rese	erved.					

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

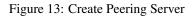
Then click Save to create the group.

## 4.5.2 Creating Peering Servers

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

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

Peering Servers					
🗧 Back 🛛 🖈 Crea	te Peering Serv	ver			
Show 5	- entrie	25			Search:
# 🔺 Name	IP Address	Hostname	Port Protocol	Weight Via Route S	et Enabled
No data available in table					
Showing 0 to 0 of 0 entries					$\leftarrow  \rightarrow$
Outbound Peering	Rules				
ANY of the rules must match	to choose the peering	group for outbound calls.			
Create Outbound	Peering Rule				
Show 5	- entri	25			Search:
# 🔶 Callee Prefix	:	Callee Pattern	Caller Pattern	Description	Enabled
No data available in table					
Showing 0 to 0 of 0 entries					
Inbound Peering R	ules				
ALL of the rules must match	to choose the peering ;	group for inbound calls.			
Create Inbound P	eering Rule				
Show 5	- entri	25			Search:
Priority 🔺	# Field	Pattern	Reject Code	Reject Reason	Enabled
No data available in table					



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

- Name: test-gw-1
- IP Address: 2.3.4.5
- Hostname: leave empty
- Port: 5060
- Protocol: UDP
- Weight: 1
- Via Route: None

Create Peering Serv	er 🛛
Name	test-gw-1
IP Address	2.3.4.5
Hostname	
Port	5060
Protocol	UDP -
Weight	1
Via Route	None
Enabled	
	$\sim$
	Save

## Figure 14: 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 the request will always be sent to the specified IP address, no matter what you put into the *hostname* field.

## Тір

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

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

## kamailio: lb:

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

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

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

## 4.5.2.1 Outbound Peering Rules



## Important

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

#### **Peering Servers**

← Back ★ Ci	reate Peering S	erver							
Show 5	• e	ntries						Search:	
# 🗕 Name	IP Address	Hostname	Port	Protocol	Weight	Via Route Set	Enabled		
29 test-gw-1	2.3.4.5		5060	1	1		1		
Showing 1 to 1 of 1 entrie	es								$\leftarrow$ 1 $\rightarrow$ $\Rightarrow$
ANY of the rules must mathematical strength of	nd Peering Rul		nd calls.					Search:	
r									
# 🔶 Callee Pr	efix	Callee Patter	n		Caller Pattern		Description		Enabled
# Callee Pr		Callee Patter	n		Caller Pattern		Description		Enabled
	ble	Callee Patter	n		Caller Pattern		Description		Enabled



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

Create Outbound Pe	×
Callee prefix	
Callee pattern	
Caller pattern	
Description	Default rule
Enabled	
	Save

Figure 16: Outbound Peering Rule Properties

Then click Save to add the rule to your group.

## Тір

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

## Тір

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>.

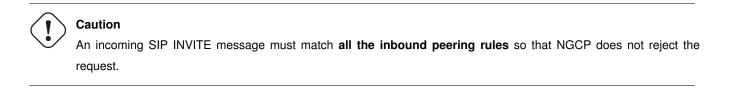
#### Tip

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 <code>@examplel.com</code> into this field.

## 4.5.2.2 Inbound Peering Rules

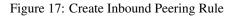
Starting from *mr5.0* release, Sipwise NGCP supports filtering SIP INVITE requests sent by SIP peers. The system administrator may define one or more matching rules for SIP URIs that are present in the headers of SIP INVITE requests, and select which SIP header (or part of the header) must match the pattern declared in the rule.

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



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

Peeri	ng Servers											
← Bac	k 🛛 🖈 Cr	eate Peering	Server									
Show	5		- entries			_				Search:		
# -	Name	IP Address	Hostname	Port	Protocol	Weight	Via Ro	ute Set	Enabled			
29	test-gw-1	2.3.4.5		5060	1	1			1			
Showing	; 1 to 1 of 1 entrie	S									← ←	
Outb	ound Peerii	ng Rules										
ANY of t	he rules must ma	itch to choose the p	peering group for out	bound calls.								
\star Crea	ate Outbou	nd Peering R	ule									
Show	5	]	- entries							Search:		
#	▲ Callee Pref	ïx	Callee Pattern		Caller Patte	rn	D	escription		Enabled		
1							D	efault rule		1		
Showing	g 1 to 1 of 1 entrie	'S									← ←	
Inbou	und Peering	Rules										
ALL of th	ne rules must mat	tch to choose the p	eering group for inbo	ound calls.								
🖈 Crea	ate Inbound	Peering Rul	e									
Show	5		- entries							Search:		
Priori	ty	<b>^</b> #	Field	Pattern	Reje	ct Code		Reject Re	ason		Enabled	
No dat	a available in tab	le										
Showing	g 0 to 0 of 0 entrie	5									¢	=



An inbound peering rule has the following properties:

Create Inbound Pee	ring Rule ×
Match Field	To-Domain 🔹
Pattern	example\.org
Reject code	403
Reject reason	Invalid called party domain
Enabled	
	Save

## Figure 18: Inbound Peering Rule Properties

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

## Note

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

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

#### **Peering Servers**

Bac	k 📩 🛧 Cr	reate Peering S	Server							
how	5	•	entries						Search:	
# ^	Name	IP Address	Hostname	Port P	Protocol	Weight	Via Route Set	Enabled		
29	test-gw-1	2.3.4.5		5060 1		1		1		
howing	1 to 1 of 1 entrie	25								← ← 1 → =
Dutbo	ound Peeri	ng Rules								
NY of t	he rules must ma	atch to choose the pe	ering group for outbour	nd calls.						
Crea	ate Outbou	nd Peering Ru	le							
how	5	•	entries						Search:	
#	<ul> <li>Callee Pret</li> </ul>	fix	Callee Pattern		Caller Patte	ern	Description		Enabled	
1							Default rule		1	
howing	1 to 1 of 1 entrie	25								
nbou	und Peering	g Rules								
LL of th	ne rules must ma	tch to choose the pee	ring group for inbound	calls.						
r Crea	ate Inbound	d Peering Rule								
how	5	•	entries						Search:	
Priori	ty 🔶 #	Field	Pattern	Reject Code	e Re	eject Reason		Enabled		
50	1	to_domain	example\.org	403	Inv	valid called party d	omain	1		
	1 to 1 of 1 entrie	25								⇐ ← 1 → =

## Figure 19: Peering Servers Overview

## 4.5.2.3 Peering Group Selection Procedure

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

- 1. All peering groups are selected as candidates that meet the following criteria:
  - · Callee's username matches callee prefix
  - Callee's URI matches callee pattern
  - Caller's URI matches caller pattern

of the outbound peering rule.

- 2. From the peering group candidates those are considered for further selection where the longest match with *callee prefix* occurs.
- 3. *Priority* of the peering group will decide the order in which peering groups will be tried for routing the outbound call.

Important

A lower priority value means higher effective priority.

The valid range of values is: from "0" for highest priority to "255" for lowest priority.

4. All peering servers in the group with the highest priority (i.e. with the lowest priority value) are tried one by one. The *weight* of the peering servers in the selected peering group will influence the order in which the servers will be tried for routing the outbound call. The weight of a server is an integer value in the range of "1" to "254".



#### Important

A server with higher weight value does *not* always take precedence over a server with lower weight, although the former one has a higher chance to be the first. The weight of a peering server just defines the probability that it will get a call first.

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

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

```
// This script can be used to find out actual probabilities
  // that correspond to a list of peering weights.
if ($argc < 2) {
 echo "Usage: lcr_weight_test.php <list of weights (integers 1-254)>\n";
 exit;
 }
$iters = 10000;
$rands = array();
for ($i = 1; $i <= $iters; $i++) {</pre>
  $elem = array();
  for ($j = 1; $j < $argc; $j++) {</pre>
    $elem["$j"] = $argv[$j] * (rand() >> 8);
  }
  $rands[] = $elem;
 }
$sorted = array();
foreach ($rands as $rand) {
 asort($rand);
  $sorted[] = $rand;
 }
```

```
$counts = array();
for ($j = 1; $j < $argc; $j++) {
   $counts["$j"] = 0;
}
foreach ($sorted as $rand) {
   end($rand);
   $counts[key($rand)]++;
}
for ($j = 1; $j < $argc; $j++) {
   echo "Peer with weight " . $argv[$j] . " has probability " . $counts["$j"]/$iters . "\n";
}
?>
```

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

```
# lcr_weight_test.php 1 2
Peer with weight 1 has probability 0.2522
Peer with weight 2 has probability 0.7478
```

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

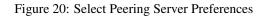
#### 4.5.3 Authenticating and Registering against Peering Servers

#### 4.5.3.1 Proxy-Authentication for outbound calls

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

#### **Peering Servers**

Show		reate Peering S	entries						Search:	
# -	Name	IP Address	Hostname	Port	Protocol	Weight	Via Route Set	Enabled		
29	test-gw-1	2.3.4.5		5060	1	1		1	🖾 Edit 🏦 Delete 🔚 Prei	ferences
Showinį	g 1 to 1 of 1 entrie	25							← ← 1	$\rightarrow$ $\Rightarrow$
Outb	ound Peeri	ng Rules								
ANY of	the rules must ma	atch to choose the pee	ering group for outbou	ind calls.						
Cre	ate Outbou	nd Deering Dui								
_		nd Peering Rul	entries						Search:	
_		•			Caller Patte	rn	Description		Search:	
Show	5	•	entries		Caller Patte	rn	Description Default rule			
Show # 1	5	fix	entries		Caller Patte	rn			Enabled	→ <b>3</b>
Show # 1 Showing	5 Callee Pre	fix 25	entries		Caller Patte	rn			Enabled	→ <b>3</b>
Show # 1 Showing Inbou	5 Callee Pret g1 to 1 of 1 entric und Peering	fix 25	Callee Pattern	d calls.	Caller Patte	rn			Enabled	



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

- peer\_auth\_user: <username for peer auth>
- peer\_auth\_pass: cpassword for peer auth>
- peer\_auth\_realm: <domain for peer auth>

## 🗲 Back

#### Preference peer\_auth\_realm successfully updated.

Access R	Access Restrictions								
Number	Number Manipulations								
NAT and	l Media Flow Control								
Remote	Authentication								
	Name	Value							
0	peer_auth_user 1	peeruser1							
0	peer_auth_pass 2	peerpass1							
0	peer_auth_realm 3	testpeering.com							
0	peer_auth_register								
0	find_subscriber_by_uuid								

## Important

If you do NOT authenticate against a peer host, then the caller CLI is put into the From and P-Asserted-Iden tity 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.

#### Тір

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

## Тір

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

## 4.5.3.2 Registering at a Peering Server

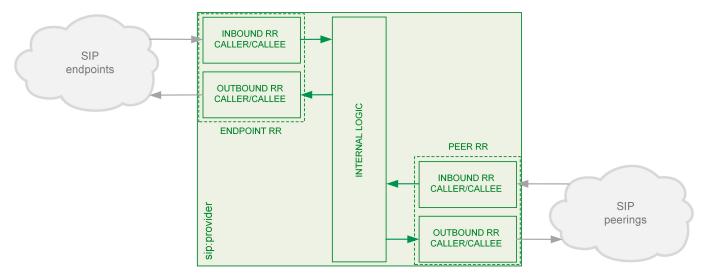
Unfortunately, the credentials configured above are not yet automatically used to register the sip:provider PRO at your peer hosts. There is however an easy manual way to do so, until this is addressed.

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

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

## 4.6 Configuring Rewrite Rule Sets

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



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

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

You would normally begin with creating a Rewrite Rule Set for your SIP domains. This is used to control what an end user can dial

for outbound calls, and what is displayed as the calling party on inbound calls. The subscribers within a domain inherit Rewrite Rule Sets of that domain, unless this is overridden by a subscriber Rewrite Rule Set preference.

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

- \${caller\_cc}: This is the value taken from the subscriber's preference CC value under Number Manipulation
- \${caller\_ac}: This is the value taken from the subscriber's preference AC value under Number Manipulation
- \${caller\_emergency\_cli}: This is the value taken from the subscriber's preference emergency\_cli value under Number Manipulation
- \${caller\_emergency\_prefix} : This is the value taken from the subscriber's preference emergency\_prefix value under
  Number Manipulation
- \${caller\_emergency\_suffix} : This is the value taken from the subscriber's preference emergency\_suffix value under Number Manipulation
- \${caller\_cloud\_pbx\_base\_cli} : This is the value taken from the Primary Number field from section Details → Master Data of the Pilot Subscriber for a particular PBX customer.

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

				Logged in a	<b>s administrator</b> Logout
sip:wise l	NGCP Dashboard			ń	🗰 Settings 🔻
Rewrit	e Rule Sets				
← Back	★ Create Rewrite R	ule Set	Searc	:h:	
# -	Reseller	Name	Description		
1	default	defaultdom	Default Domain		
Showing 1 to	1 of 1 entries			¢.	← <b>1</b> → ⇒
© 2013 Sipwi:	se GmbH, all rights reserved.				

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

se Create Rev	vrite Rule	Sets				×
	Reseller			Search:		
		#	Name	Contract #	Status	
		1	default	1	active <mark>1</mark>	
		Showing 1 to 1 of 1 en	tries		← ← 1 - Create R	→ →
	Name	domain-dialplan	2			
D	escription	Dialplan for Domain	is 3			
					4	Save

Press the Save button to create the set.

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

				Logged in as	administrator Logou
sip:wise N	IGCP Dashboard			*	📰 Settings 👻
Rewrite	e Rule Sets				
← Back	★ Create Rewrite Ru	ule Set			
ewrite rule set	successfully created				
	r	r	Searc	:n:	
# ^	Reseller	Name	Description		
1	default	defaultdom	Default Domain		
2	default	domain-dialplan	Dialplan for Domains	🖸 Edit 🚺 🗖 D	elete 🔚 Rules
Showing 1 to 2	of 2 entries			⇐	- 1 → ⇒

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

## Тір

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.

## 4.6.1 Inbound Rewrite Rules for Caller

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

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

Strip leading 00 or +

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

• Field: Caller

Replace 0 by Caller's country code:

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

## NORMALIZE LOCAL CALLS:

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

síp:wise	eate Rule	× tings -
Rewri	Match pattern ^(00]\+)([1-9][0-9]+)\$ 1	
	Replacement Pattern 2 2	
Back	Description International to E.164 3	
Inboun	Direction Inbound 4	
Inboun	Field Caller 5	
Outbou		6 Save
2013 Sipwise G	imbH, all rights reserved.	

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

## Important

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

## Тір

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

## **Rewrite Rules for domain-dialplan**



Rewrite rule successfully created

	Inbound Rewrite Rules for Caller							
		Match Pattern	Replacement Pattern	Description				
1	^ ↓	^(00 \+)([1-9][0-9]+)\$	\2	International to E.164				
	↑₩2	^0([1-9][0-9]+)\$	{caller_cc}\1	National to E.164				
	<b>↑ ↓</b>	^([1-9][0-9]+)\$	{caller_cc}\${caller_ac}\1	Local to E.164				
	Inbound Re	write Rules for Callee						
Outbound Rewrite Rules for Caller								
	Outbound Rewrite Rules for Callee							

#### 4.6.2 Inbound Rewrite Rules for Callee

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

Strip leading 00 or +

- Match Pattern: ^ (00 | \+) ([1-9][0-9]+) \$
- Replacement Pattern: \2

- Description: International to E.164
- Direction: Inbound
- Field: Callee

Replace 0 by Caller's country code:

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

NORMALIZE AREACODE-INTERNAL CALLS:

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

#### Tip

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

#### 4.6.3 Outbound Rewrite Rules for Caller

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

Replace Austrian country code 43 by 0  $\,$ 

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

- Direction: Outbound
- Field: Caller

Prefix 00 for international caller

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

## Tip

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

## 4.6.4 Outbound Rewrite Rules for Callee

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

PREFIX SIPSP # FOR ALL CALLS

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

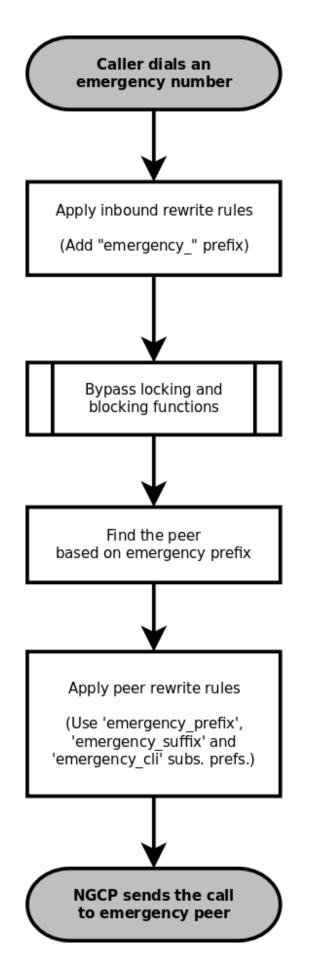
## 4.6.5 Emergency Number Handling

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

- Simple emergency number handling: inbound rewrite rules append an emergency tag to the called number, this will be recognised by NGCP's call routing logic and the call is routed directly to a peer. Please read the next section for details of simple emergency number handling.
- An emergency *number mapping* is applied: a dedicated emergency number mapping database is consulted in order to obtain the most appropriate routing number of emergency services. This logic ensures that the caller will contact the geographically closest emergency service. Please visit the Emergency Mapping Section 5.5 section of the handbook for more details.

## 4.6.5.1 Simple Emergency Number Handling Overview

The overview of emergency call processing is as follows:



Configuring Emergency Numbers is also done via Rewrite Rules.

## 4.6.5.2 Tagging Inbound Emergency Calls

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

TAG AN EMERGENCY CALL

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

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

## 4.6.5.3 Normalize Emergency Calls for Peers

In order to normalize the emergency number to a valid format accepted by the peer, you need to assign an *Outbound Rewrite Rule For Callee*, which strips off the emergency\_prefix. You can also use the variables  $caller_emergency_cli$ ,  $caller_emergency_cli$ ,  $caller_emergency_prefix$  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

## 4.6.6 Assigning Rewrite Rule Sets to Domains and Subscribers

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

## sip:wise NGCP Dashboard Settings 🛪 ñ Domain "demo.sipwise.com" - Preferences 🗲 Back **Call Blockings** Access Restrictions 1 Number Manipulations 0 rewrite\_rule\_set 2 • 3 defaultdom 0 extension\_in\_npn 0 inbound\_upn From-Username •

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

٠

User-Provided-Number

## 4.6.7 Creating Dialplans for Peering Servers

outbound\_from\_user

0

For each peering server, you can use one of the Rewrite Rule Sets that was created previously as explained in Section 4.6 (keep in mind that special variables  $\{caller_ac\}$  and  $\{caller_cc\}$  can not be used when the call comes from a peer). To do so, click on the name of the peering server, look for the preference called *Rewrite Rule Sets*.

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

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

# 5 Features

The sip:provider PRO 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.

## 5.1 Managing System Administrators

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

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

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

## 5.1.1 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*  $\rightarrow$  *Administrators*.

## Administrators



how	5		- entries	• entries S				arch:		
#	Reseller	Login	Master	Active	Read Only	Show Passwords	Show CDRs	Show Billing Info	Lawful Intercept	
1	default	administrator	1	1	0	1	1	1	1	
3	Demo Reseller	demoadmin	1	1	0	1	1	0	0	🖸 Edit 🏦 Delete 🔒 API ke

Showing 1 to 2 of 2 entries

## Figure 22: List of System Administrators

You have 2 options:

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

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

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

Edit Administrator						
Reseller				Search:		
	#	Name		Contract #	Status	
	16	Demo Reseller		200	active	×
	1	default		1	active	
				137	active	
	Showing 1 to 3 of 3 entries				← ← 1	$\rightarrow$ $\Rightarrow$
Login	demoad	min			Create	e Reseller
Password						
ls superuser						
						Save

Figure 23: Generic System Administrator Attributes

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

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

#### 5.1.2 Access Rights of Administrators

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

Edit Administrator		×
		Τ
ls superuser		
ls master	×	
ls active		
Read only		ſ
Show passwords		
Call data		
Billing data		
Lawful intercept		
	Save	

# Figure 24: Access Rights of System Administrators

# Table 1: Access Rights of System Administrators

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

60 / 486

### Table 1: (continued)

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 <b>web interface</b> this means that <i>Create</i> and <i>Edit</i> buttons
		will be hidden or disabled.
		• For the <b>REST API</b> this means that only GET, HEAD, OPTIONS
		HTTP request methods are accepted, and NGCP 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 are stored in an unreadable way (cryp-
		tographic hash digest) in the database, while subscriber passwords
		are basically always stored in plain text. The latter happens on pur-
		pose, e.g. to make subscriber data migration possible.
Show CDRs	Call data	This privilege has effect on 2 items that will be displayed on admin
SHOW CDHS	Call Gala	panel of NGCP, when Subscriber $\rightarrow$ Details is selected:
		parter of redor, when oubscriber 7 betails is selected.
		1. PBX <i>Groups</i> list
		2. Captured Dialogs list
Show Billing Info	Billing data	Some REST API resources that are related to billing are disabled:
		HTTP requests on /api/vouchers, /api/topupcash and /
		api/topupvoucher resources are rejected.
Lawful Intercept	Lawful	If the privilege is selected then the REST API for interceptions (that is:
	intercept	/api/interceptions) is enabled; if the privilege is not selected
		then the interceptions API is disabled.
		Note
		This means that besides enabling LI in config.yml configura-
		tion file one also needs to enable the API via the LI privilege of an
		administrator user, so that NGCP can really provide LI service.

# 5.2 Access Control for SIP Calls

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

#### 5.2.1 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.

Truste	Trusted Sources					
Call Bl	Call Blockings					
	Name		Value			
0	block_in_mode					
0	block_in_list					
0	block_in_clir					
0	block_out_mode					
0	block_out_list					
0	adm_block_in_mode					
0	adm_block_in_list					
0	adm_block_in_clir					
0	adm_block_out_mode					
0	adm_block_out_list					
0	ncos		T			

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.

#### 5.2.1.1 Block Modes

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

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

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

#### 5.2.1.2 Block Lists

The list contents are controlled by the User Preferences *block\_in\_list*, *block\_out\_list* and their administrative counterparts. Click on the *Edit* button in the *Preferences* view to define the list entries.

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

sip:wise Edit Preference block_out_lis	t 3 × tings -
Subs( 1* 1 431234567 2 ← Back	Image: Constraint of the second secon
Call Forwards Voicemail and Voicebox	
Fax2Mail and Sendfax	
Speed Dial	
Reminder	
Trusted Sources	
Call Blockings	
Name	Value

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

#### 5.2.1.3 Block Anonymous Numbers

For incoming call, the User Preference *block\_in\_clir* and *adm\_block\_in\_clir* controls whether or not to reject incoming calls with number supression (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.

#### 5.2.2 NCOS Levels

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

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

NCOS levels can either be whitelists or blacklists.

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

#### 5.2.2.1 Creating NCOS Levels

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

			🐣 Logged	in as administrator Logout
ຣາ໌p:ເພາຣຣ NGCP Dashboard				🕈 🗰 Settings 👻
NCOS Levels				
← Back ← Create NCOS Lev	/el	:	Search:	
# A Reseller	Level Name	Mode	Description	
No data available in table				
Showing 0 to 0 of 0 entries				$\leftarrow \rightarrow \Rightarrow$
© 2013 Sipwise GmbH, all rights reserved.				

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

~						
sip:wise Cre	ate NCOS Levels					× tings -
NCOS	Reseller			Search	:	
		#	Name	Contract #	Status	
		1	default	1	active <mark>1</mark>	
🗲 Back		Showing 1 to 1 of 1 ent	ries		∈ ← 1 →	⇒
					Create Reselle	er
#	Level Name	test 2				
No data a	Mode	blacklist 3				•
Showing 01	Description	NCOS Test Level	4			→⇒
					5 Sav	re
© 2013 Sipwise Gm	nbH, all rights reserved.					

### 5.2.2.2 Creating Rules per NCOS Level

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

© 2013 Sipwise GmbH, all rights reserved.

sip:wise	■ NGCP Dashboa	ard			🐐 🇱 Settings 👻
NCOS	S Levels				
← Bacl	Create N	COS Level		Se	arch:
# -	Reseller	Level Name	Mode	Description	
1	default	test	blacklist	NCOS Test Level	🖾 Edit 🔹 Delete 🗮 Patterns
Showing 1	to 1 of 1 entries				

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

sip:wise Crea	e Number Pattern		× tings -
Numi	Pattern <sup>439</sup> 1 Description Austrian Premium Nur	mbers 2	
← Back			3 Save
# ^	Pattern	Description	
2	^439	Austrian Premium Numbers	
Showing 1 to 1 of 1 er			

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

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

#### 5.2.2.3 Assigning NCOS Levels to Subscribers/Domains

Once you've defined your NCOS Levels, you can assign them to local subscribers. To do so, navigate to *Settings* $\rightarrow$ *Subscribers*, search for the subscriber you want to edit, press the *Details* button and go to the *Preferences* View. There, press the *Edit* button on either the *ncos* or *adm\_ncos* setting in the *Call Blockings* section.

Call Blo	Call Blockings					
	Name	Value				
0	block_in_mode					
0	block_in_list					
0	block_in_clir					
0	block_out_mode	0				
0	block_out_list	1* 431234567				
0	adm_block_in_mode					
0	adm_block_in_list					
0	adm_block_in_clir					
0	adm_block_out_mode					
0	adm_block_out_list					
0	ncos 2	•	3 B Edit			

You can assign the NCOS level to all subscribers within a particular domain. To do so, navigate to *Settings* $\rightarrow$ *Domains*, select the domain you want to edit and click *Preferences*. There, press the *Edit* button on either *ncos* or *admin\_ncos* in the *Call Blockings* section.

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

#### 5.2.2.4 Assigning NCOS Level for Forwarded Calls to Subscribers/Domains

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

#### 5.2.3 IP Address Restriction

The sip:provider PRO provides subscriber preference *allowed\_ips* to restrict the IP addresses that subscriber is allowed to use the service from. If the REGISTER or INVITE request comes from an IP address that is not in the allowed list, the sip:provider PRO will reject it with a 403 message. Also a voice message can be played when the call attempt is rejected (if configured).

By default, allowed\_ips is an empty list which means that subscriber is not restricted. If you want to configure a restriction, navigate

to *Settings*  $\rightarrow$  *Subscribers*, search for the subscriber you want to edit, press *Details* and then *Preferences* and press *Edit* for the *allowed\_ips* preference in the *Access Restrictions* section.

Call Blockir	Call Blockings						
Access Rest	Access Restrictions						
1	Name	Value					
0	lock						
0	concurrent_max						
0	concurrent_max_out						
0	allowed_clis						
0	reject_emergency						
0	concurrent_max_per_account						
0	concurrent_max_out_per_account						
0	allowed_ips 2		3 Edit				
0	man_allowed_ips						
0	ignore_allowed_ips						
0	allow_out_foreign_domain						

Press the Edit button to the right of empty drop-down list.

You can enter multiple allowed IP addresses or IP address ranges one after another. Click the *Add* button to save each entry in the list. Click the *Delete* button if you want to remove some entry.

# 5.3 Call Forwarding and Call Hunting

The sip:provider PRO provides the capabilities for normal *call forwarding* (deflecting a call for a local subscriber to another party immediately or based on events like the called party being busy or doesn't answer the phone for a certain number of seconds) and *serial call hunting* (sequentially executing a group of deflection targets until one of them succeeds). Targets can be stacked, which means if a target is also a local subscriber, it can have another call forward or hunt group which is executed accordingly.

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

#### 5.3.1 Setting a simple Call Forward

Go to your Subscriber Preferences and click Edit on the Call Forward Type you want to set (e.g. Call Forward Unconditional).

71	/	486

sip:wise Edit Call Forward U	nconditional				× tings -
Subsc Destination	<ul> <li>Voicemail</li> <li>Conference</li> <li>URI/Number</li> </ul>				
← Back 2 URI/Number	4312345				
Type Call Fo				3 Advanced View	Save
Call Fo		-			
Call Forward Unavailable					
Voicemail and Voicebox					

If you select *URI/Number* in the *Destination* field, you also have to set a *URI/Number*. The timeout defines for how long this destination should be tried to ring.

#### 5.3.2 Advanced Call Hunting

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

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

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

#### 5.3.2.1 Configuring Destination Sets

Click on *Manage Destination Sets* to see a list of available sets. The *quickset\_cfu* has been implicitly created during our creation of a simple call forward. You can edit it to add more destinations, or you can create a new destination set.

		Logged in as administrat	or Logout
síp:ພາ່ຣະ		estination Set	tings 👻
Subsc		1 Name my test set	
← Back		Destination Voicemail 2 URI/Number	
Call For	3	URI/Number 12345	
Туре		for (seconds) 300	
Call Fo		4 Priority 1	
Call Fo		Remove	
Call Fo			
Call Fo		optional 5 Add another destination	
Voicem		6 Save	

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

s1p:wise						×	tings 🔻
	Edit	Call Forward Un	conditiona	l			tings
Subso	1	Destination Set	quickset_cfu				
		during Time Set	<always></always>			T	
← Back Call For	c	optional 2 via "Add	more"			Remove	
Туре		Destination Set	my test set			•	
Call Fo		during Time Set	<always></always>			¥	
Call Fo						Remove	
Call Fo Call Fo						Add more	-
Voicem				Manage Destination Sets	Manage Time Sets	3 Simple View Save	

Press Save to store your settings.

#### 5.3.2.2 Configuring Time Sets

Click on *Manage Time Sets* in the advanced call-forward menu to see a list of available time sets. By default there are none, so you have to create one.

sip:wise	dit Time Se							,	tings -
Subso	ait fille se	Name	my test time se	at					
	'	Period	Year	Month	Day	Weekday	Hour	Minute	
← Back		2	2013 ▼ through	April • through	▼ through	Monday • through	▼ trough	▼ through	
Call For			•	Septembe •	•	Friday •	•	Remove	
Type Call Fo							optional 3		
Call Fo							Add	another period	
Call Fo								4 Save	
Voicemail a	nd Voicebox								

You need to provide a *Name*, and a list of *Periods* where this set is active. If you only set the top setting of a date field (like the *Year* setting in our example above), then it's valid for just this setting (like the full year of *2013* in our case). If you provide the bottom setting as well, it defines a period (like our *Month* setting, which means from beginning of April to end of September). For example, if a CF is set with the following timeset: "hour { 10-12 } minute { 20-30 }", the CF will be matched within the following time ranges:

- from 10.20am to 10:30am
- from 11.20am to 11:30am
- from 12.20am to 12:30am

#### ↘ Important

the period is a *through* definition, so it covers the full range. If you define an *Hour* definition *8-16*, then this means from *08:00* to *16:59:59* (unless you filter the *Minutes* down to something else).

If you close the Time Sets management, you can assign your new time set to the call forwards you're configuring.

# 5.4 Local Number Porting

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

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

#### Note

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

#### 5.4.1 Local LNP Database

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

#### 5.4.1.1 LNP Carriers

LNP Carriers are defined by an arbitrary *Name* for proper identification (e.g. *British Telecom*) and contain a *Prefix* which can be used as routing prefix in LNP Rewrite Rules and subsequently in Peering Rules to route calls to the proper carriers. The LNP prefix is written to CDRs to identify the selected carrier for post processing and analytics purposes of CDRs. LNP Carrier entries also have an *Authoritative* flag indicating that the numbers in this block belong to the carrier operating the sip:provider PRO. 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.

#### 5.4.1.2 LNP Numbers

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

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

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

#### 5.4.1.3 Enabling local LNP support

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

#### 5.4.1.4 LNP Routing Procedure

#### Calls to non-authoritative Carriers

When a call arrives at the system, the calling and called party numbers are first normalized using the *Inbound Rewrite Rules for Caller* and *Inbound Rewrite Rules for Callee* within the rewrite rule set assigned to the calling party (a local subscriber or a peer).

If the called party number is not assigned to a local subscriber, or if the called party is a local subscriber and has the subscriber/domain preference *Inp\_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\_Inp\_lookup* is set for this peer, then an LNP lookup is performed using the normalized calling party number. The purpose for that is solely to find the LNP prefix of the calling peer, which is then stored as *source\_Inp\_prefix* in the CDR. If the LNP lookup does not return a result (e.g. the calling party number is not populated in the local LNP database), but the peer preference *default\_Inp\_prefix* is set for the originating peer, then the value of this preference is stored in *source\_Inp\_prefix* of the CDR.

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

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

#### Important

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

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

- \${callee\_lnp\_prefix}: The prefix stored in the LNP Carrier
- \${callee\_lnp\_basenumber}: The actual number entry causing the match (may be shorter than the called party number
  due to longest match)

Typically, you would create a rewrite rule to prefix the called party number with the *callee\_lnp\_prefix* by matching ([0-9]+) and replacing it by  $fcallee_lnp_prefix$ .

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

Finally, the *destination\_lnp\_prefix* in the CDR table is populated either by the prefix defined in the Carrier of the LNP database if a match was found, or by the *default\_lnp\_prefix* prefrence of the destination peer or subscriber/domain.

#### 5.4.1.5 Transit Calls using LNP

If a call originated from a peer and the peer preference *force\_outbound\_calls\_to\_peer* is set to *force\_nonlocal\_lnp* (the *if callee is not local and is ported* selection in the panel), the call is routed back to a peer selected via the peering rules.

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

#### 5.4.1.6 CSV Format

The LNP database can be exported to CSV, and in the same format imported back to the system. On import, you can decide whether to drop existing data prior to applying the data from the CSV.

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

carrier\_name carrier\_prefix number routing\_number start end authoritative skip\_rewrite

Table 2:	LNP	CSV	Format
----------	-----	-----	--------

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

#### 5.4.2 External LNP via LNP API

External LNP relies on the *Sipwise LNP Daemon (Inpd)* which kamailio-proxy is talking to via a defined JSONRPC protocol. The proxy sends the A and B number to *Inpd*, which in the current release translates it to a SIP Message sent to an external server (typically a Squire SIP-to-INAP gateway. This external gateway is performing an SS7 INAP request to fetch the LNP result, which is passed back as a binary blob in a 3xx response to the *Inpd*. The *Inpd* extracts the TCAP body of the response and returns the information back to the proxy.

#### 5.4.2.1 Enabling LNP lookup via API

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

- kamailio→proxy→lnp→enabled:*yes*
- kamailio→proxy→lnp→type:*api*
- lnpd→enabled:*yes*

There is a possibility to explicitly allow (whitelist) or deny (blacklist) certain number ranges for which an LNP lookup may be done. The relevant configuration parameters are at kamailio  $\rightarrow$  proxy  $\rightarrow$  lnp  $\rightarrow$  lnp\_request\_whitelist and kamailio  $\rightarrow$  proxy  $\rightarrow$  lnp  $\rightarrow$  lnp\_request\_blacklist. For each entry in the list a POSIX regex expression may be used, see the following example:

Interpretation of the above lists (that are based on numbers represented in national format):

- whitelist: do LNP lookup for any called number that starts with 9 or 800
- blacklist: do not perform LNP lookup for any called number that starts with 1, 900, 110 or 112

Important

If both whitelist and blacklist are defined, the LNP lookup is only performed when the called number matches any of the whitelist patterns and does not match any of the blacklist patterns.

#### 5.4.2.2 The Redundancy Feature

It is possible to set up *LNP daemon* to provide a kind of redundant service to the Proxy. This means the *LNP daemon* will send its LNP query to more LNP serving nodes that are predefined in a list. (See Configuration of LNP daemon Section 5.4.2.3 chapter for details.) The LNP query may happen in 2 ways:

- **round-robin**: *LNP daemon* sends the query to one of the serving nodes then waits for the response for a configurable timeout. If it does not get the response in time, it sends the LNP query to the next serving node.
- parallel: LNP daemon sends the query to all of the serving nodes then waits for the response, and will accept the first response that it receives.

#### 5.4.2.3 Configuration of Sipwise LNP Daemon

LNP daemon takes its active configuration from /etc/ngcp-lnpd/config.yml file. The file is generated automatically when a new NGCP configuration is applied (ngcpcfg apply...)—from the main Sipwise NGCP configuration file: /etc/ ngcp-config/config.yml and a template: /etc/ngcp-config/template/etc/ngcp-lnpd/config.yml. tt2. System administrators are only expected to modify the lnpd.config section of main configuration file /etc/ngcpconfig/config.yml.

A sample LNP daemon configuration file (/etc/ngcp-lnpd/config.yml) looks like:

```
daemon:
        json-rpc:
                ports:
                         - 54321
                         - 12345
                interfaces:
                         - 127.0.0.1
                         - 192.168.1.90
                         - ::1
        sip:
                port: 5095
                address: 0.0.0.0
        threads: 4
        foreground: false
        pidfile: /tmp/lnpd.pid
        loglevel: 7
instances:
        default:
                module: sigtran
                destination: 192.168.1.99
                from-domain: test.example.com
                headers:
```

```
- header: INAP-Service-Key
                  value: 2
        reply:
                tcap: raw-tcap
redundant:
       module: sigtran
        destinations:
                - 192.168.1.99
                - 192.168.1.95
                - 192.168.1.90
        mechanism: round-robin
       retry-time: 30
        timeout: 5
        from-domain: test.example.com
        headers:
                - header: INAP-Service-Key
                 value: 2
        reply:
                tcap: raw-tcap
parallel:
       module: sigtran
        destinations:
                - 192.168.1.99
                - 192.168.1.95
                - 192.168.1.90
        mechanism: parallel
       retry-time: 30
        timeout: 10
        from-domain: test.example.com
       headers:
                - header: INAP-Service-Key
                 value: 2
        reply:
                tcap: raw-tcap
mock1:
       module: mock-tcap
        numbers:
                - number: '4311003'
                 routing-number: '4318881003'
        reply:
               tcap: raw-tcap
```

The corresponding NGCP main configuration file contains:

```
daemon:
  foreground: 'false'
  json-rpc:
    ports:
```

```
- '54321'
- '12345'
loglevel: '7'
sip:
   port: '5095'
threads: '4'
instances:
<< These are the same entries as in /etc/ngcp-lnpd/config.yml file >>
```

#### Description of configuration parameters in /etc/ngcp-config/config.yml file

- daemon section:
  - foreground: determines if the LNP daemon runs as foreground or background process
  - json-rpc.ports: port numbers where LNP daemon listens for incoming JSONRPC requests from NGCP Proxy
  - loglevel: how detailed information LNP daemon writes in its log file
  - sip.port: listening port number used for SIP sessions with LNP serving nodes; LNP daemon will listen on first available (shared) IP address that is taken from /etc/ngcp-config/network.yml file
  - threads: number of threads LNP daemon will use internally; this value determines how many requests the daemon can serve in parallel
- instances section: at least one default instance must be defined here. Others are also useful for providing redundancy, please check redundant and parallel entries above.
  - module: only sigtran is used for normal operations

#### Important

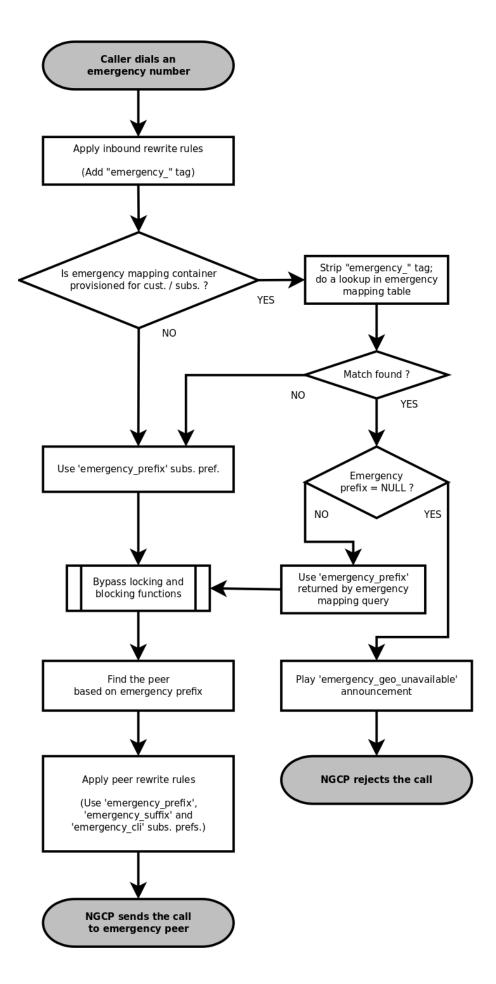
The module mock-tcap is only meant for developers. In this case the LNP daemon does not produce a SIP request that it sends to LNP serving nodes, but instead it uses the numbers parameter to match a called number with a routing number. The numbers parameter contains a list of number — routing-number pairs and is used as a database for number lookups. Finally LNP daemon returns the routing number as a response on LNP query.

- destinations: list of nodes to which LNP daemon sends the LNP query
- mechanism: either parallel or round-robin, defining the method of redundant queries
- retry-time: a period of time in seconds while LNP daemon considers an LNP serving node being unreachable after an LNP query timeout
- timeout: the period of time while LNP daemon waits for a response on an LNP query from one of the LNP serving nodes *PLEASE NOTE*: retry-time and timeout are used with both the parallel and the round-robin redundancy methods
- from-domain: the domain that will be used in SIP From header when LNP daemon sends the LNP query
- headers: this is a list of header name—value pairs; these custom headers will be included in SIP request that LNP daemon sends to an LNP serving node
- reply.tcap: determines the format of reply sent to NGCP Proxy; currently only raw-tcap is supported, which means LNP daemon will not decode the TCAP response it gets from an LNP serving node but it forwards the raw TCAP message body

# 5.5 Emergency Mapping

As opposed to the Simple Emergency Number Handling Section 4.6.5.1 solution, the Sipwise NGCP supports an advanced emergency call handling method, called *emergency mapping*. The main idea is: instead of obtaining a statically assigned emergency prefix / suffix from subscriber preferences, NGCP retrieves an emergency routing prefix from a central emergency call routing table, according to the current location of the calling subscriber.

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



#### 5.5.1 Emergency Mapping Description

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

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

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

#### Note

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

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

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

#### Note

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

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

#### 5.5.2 Emergency Mapping Configuration

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

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

Edit Emergency Con	taine	rs			\$
Reseller			Search:		
	#	Name	Contract #	Status	
	1	default	1	active	×
	3	api_test test reseller	137	active	
	5	patched name 1494894408	179	active	
	7	test reseller 1494894408 2	181	active	
	9	test reseller 1494894408 3	183	active	
	Showir	ng 1 to 4 of 8 entries	¢		te Reseller
Name	Emer	gCont_1			
					Save



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

#### Тір

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

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

Edit Emergency Map	opings			×
Emergency Mapping Container			Search:	
	#	Reseller	Name	
	1	default	EmergCont_1	
	3	default	EmergCont_2	
	Showing	1 to 2 of 2 entries	Create Emerg	$\leftarrow$ $\leftarrow$ 1 $\rightarrow$ $\Rightarrow$ ency Mapping Container
Code	133			
Prefix	E1_133	}_		
				Save

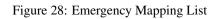
Figure 27: Creating an Emergency Mapping Entry

The following parameters must be set:

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

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

Emerg	gency Mappings						
← Back Emerge	★ Download CSV	★ Upload CSV					
★ Create	Emergency Containe						
Show 5	~	entries				Search:	
#	* Reseller			Name			
1	default			EmergCont_1			
3	default			EmergCont_2			
Showing 1 to	o 2 of 2 entries						$\leftarrow  1  \rightarrow  \Rightarrow $
Emerge	ncy Mappings						
★ Create	Emergency Mapping						
Show 5		entries				Search:	
# ^	Container	Reseller	Emergency Numb	er	Emergency Prefix		
1	EmergCont_1	default	133		E1_133_		
3	EmergCont_1	default	144		E1_144_		
5	EmergCont_2	default	133		E2_133_		



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

	comer #205 - Preferer			
Back				🖌 Expand
Call B	Blockings			
Acces	as Restrictions			
Numb	ber Manipulations			
	Attribute	Name	Value	
0	emergency_prefix	Emergency Prefix variable		
0	emergency_suffix	Emergency Suffix variable		
	emergency_cli	Emergency CLI		
0				
0	emergency_mapping_container	Emergency Mapping Container	EmergCont_2	

#### Figure 29: Assigning an Emergency Mapping Container

#### Rewrite Rules for Emergency Mapping

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

#### Important

Please don't forget to define the rewrite rules for peers—particularly: *Outbound Rewrite Rules for Callee*—as described in Normalize Emergency Calls for Peers Section 4.6.5.3 section of the handbook.

#### 5.5.2.1 Emergency Calls Not Allowed

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

In such a case the emergency mapping lookup will return an emergency prefix, but the value of this will be NULL. Therefore the call is rejected and an announcement is played. The announcement is a newly defined sound file referred as <code>emergency\_geo\_unavailable</code>.

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\_r ejects.emergency\_invalid.announce\_reason.

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

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

#### Downloading Emergency Mapping List

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

EmergCont\_1,133,E1\_133\_ EmergCont\_1,144,E1\_144\_ EmergCont\_2,133,E2\_133\_

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

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

### Uploading Emergency Mapping List

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

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

**Create Emer** 

Upload m

· ·	Itali	ners			
oping	Choo	ose File (None)			
seller			Search:		
#	#	Name	Contract #	Status	
1	1	default	1	active	
З	3	api_test test reseller	137	active	
	5	patched name 1494894408	179	active	
5					



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

#### 5.6 Header Manipulation

Purge existing

#### 5.6.1 Header Filtering

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

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

By default the system will remove from the second leg all the SIP headers which are not in the above list. If you want to keep some additional/custom SIP headers, coming from the first leg, into the second leg you just need to add them at the end of the *header\_list=* list. After that, as usual, you need to apply and push the changes. In this way the system will keep your headers in the INVITE sent to the destination subscriber/peer.

# Warning

DO NOT TOUCH the list if you don't know what you are doing.

#### 5.6.2 Codec Filtering

Sometimes you may need to filter some audio CODEC from the SDP payload, for example if you want to force your subscribers to do not talk a certain codecs or force them to talk a particular one. To achieve that you just need to change the /etc/ngcp-config/config.yml, in the following section:

```
sdp_filter:
    codecs: PCMA,PCMU,telephone-event
    enable: yes
    mode: whitelist
```

In the example above, the system is removing all the audio CODECS from the initial INVITE except G711 alaw, ulaw and telephoneevent. In this way the callee will be notified that the caller is able to talk only PCMA. Another example is the blacklist mode:

```
sdp_filter:
    codecs: G729,G722
    enable: yes
    mode: blacklist
```

In this way the G729 and G722 will be removed from the SDP payload. In order to apply the changes, as usual, you need to run ngcpcfg apply *Enable CODEC filtering* and push the changes.

#### 5.6.3 Enable History and Diversion Headers

It may be useful and mandatory - specially with NGN interconnection - to enable SIP History header and/or Diversion header for outbound requests to a peer or even for on-net calls. In order to do so, you should enable the following preferences in Domain's and Peer's Preferences:

- Domain's Prefererences: inbound\_uprn = Forwarder's NPN
- Peer's Prefererences: outbound\_history\_info = UPRN
- Peer's Prefererences: 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 Prefererences: outbound\_diversion = UPRN (if you want to allow Diversion Header for on-net call as well)

#### 5.7 SIP Trunking with SIPconnect

#### 5.7.1 User provisioning

For the purpose of external SIP-PBX interconnect with sip:provider PRO the platform admin should create a subscriber with multiple aliases representing the numbers and number ranges served by the SIP-PBX.

- · Subscriber username any SIP username that forms an "email-style" SIP URI.
- · Subscriber Aliases numbers in the global E.164 format without leading plus.

To configure the Subscriber, go to *Settings* $\rightarrow$ *Subscribers* and click *Details* on the row of your subscriber. There, click on the *Preferences* button on top.

You should look into the *Number Manipulations* and *Access Restrictions* sections in particular, which control the calling and called number presentation.

#### 5.7.2 Inbound calls routing

Enable preference Number Manipulations  $\rightarrow e164\_to\_ruri$  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).

#### 5.7.3 Number manipulations

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

#### 5.7.3.1 Rewrite rules

The SIP PBX by default inherits the domain dialplan which usually has rewrite rules applied to normal Class 5 subscribers with inbound rewrite rules normalizing the dialed number to the E.164 standard. If most users of this domain are Class 5 subscribers the dialplan may supply calling number in national format - see Section 4.6. While the SIP-PBX trunk configuration can be sometimes amended it is a good idea in sense of SIPconnect recommendation to send only the global E.164 numbers.

Moreover, in mixed environments with the sip:provider PRO Cloud PBX sharing the same domain with SIP trunking (SIP-PBX) customers the subscribers may have different rewrite rules sets assigned to them. The difference is caused by the fact that the dialplan for Cloud PBX is fundamentally different from the dialplan for SIP trunks due to extension dialing, where the Cloud PBX subscribers use the break-out code (see Section 15.1.2) to dial numbers outside of this PBX.

The SIPconnect compliant numbering plan can be accommodated by assigning Rewrite Rules Set to the SIP-PBX subscriber. Below is a sample Rewrite Rule Set for using the global E.164 numbers with plus required for the calling and called number format compliant to the recommendation.

INBOUND REWRITE RULE FOR CALLER

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

#### INBOUND REWRITE RULE FOR CALLEE

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

#### OUTBOUND REWRITE RULE FOR CALLER

- Match Pattern: ^ ( [1-9] [0-9] +) \$
- Replacement Pattern: + \1
- Description: For the calls to SIP-PBX add plus to E.164
- **Direction:** Outbound
- Field: Caller

#### OUTBOUND REWRITE RULE FOR CALLEE

- Match Pattern: ^ ( [1-9] [0-9] +) \$
- Replacement Pattern: +\1
- Description: For the calls to SIP-PBX add plus to E.164
- Direction: Outbound
- Field: Callee

Assign the aforementioned Rewrite Rule Set to the SIP-PBX subscribers.



# Warning

Outbound Rewrite Rules for Callee shall NOT be applied to the calls to normal SIP UAs like IP phones since the number with plus does not correspond to their SIP username.

#### 5.7.3.2 User parameter

The following configuration is needed for your platform to populate the From and To headers and Request-URI of the INVITE request with "user=phone" parameter as per RFC 3261 Section 19.1.1 (if the user part of the URI contains telephone number formatted as a telephone-subscriber).

- Domain's Prefererences: *outbound\_from\_user\_is\_phone* = Y
- Domain's Prefererences: *outbound\_to\_user\_is\_phone* = Y

#### 5.7.3.3 Forwarding number

The following is our common configuration that covers the calling number presentation in a variety of use-cases, including the incoming calls, on-net calls and Call Forward by the platform:

- Domain's Preferences: *inbound\_uprn* = Forwarder's NPN
- Domain's Preferences: *outbound\_from\_user* = UPRN (if set) or User-Provided Number
- Domain's Preferences: *outbound\_pai\_user* = **UPRN (if set) or Network-Provided Number**
- Domain's Preferences: outbound\_history\_info = UPRN (if the called user expects History-Info header)
- Domain's Preferences: *outbound\_diversion* = **UPRN** (if the called user expects Diversion header)
- Domain's Preferences: *outbound\_to\_user* = **Original (Forwarding) called user** if the callee expects the number of the subscriber forwarding the call, otherwise leave default.

The above parameters can be tuned to operator specifics as required. You can of course override these settings in the Subscriber Preferences if particular subscribers need special settings.

#### Тір

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 Prefererences: inbound\_uprn = Forwarder's NPN / Received Diversion
- using History-Info method (RFC 7044): NGCP platform extends the History-Info header received from the PBX by adding another level of indexing according to the specification RFC 7044.

### 5.7.3.4 Allowed CLIs

- For correct calling number presentation on outgoing calls, you should include the pattern matching all the alias numbers of SIP-PBX or each individual alias number under the *allowed\_clis* preference.
- If the signalling calling number (usually taken from From user-part, see *inbound\_upn* preferences) does not match the *al-lowed\_clis* pattern, the *user\_cli* or *cli* preference (Network-Provided Number) will be used for calling number presentation.

### 5.7.4 Registration

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

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

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

# Important

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

### 5.7.4.1 Trusted sources

If a SIP-PBX can not be configured to perform digest authentication, you can enable IP-based authentication in sip:provider PRO for it. For this go to the subscriber's preferences (*Details* $\rightarrow$ *Preferences* $\rightarrow$ *Trusted Sources*) and specify the IP address of the SIP-PBX in the *Source IP* field.

When this feature is configured, the sip:provider PRO will authenticate all calls of this user that arrive from the specified IP without sending a challenge.



## Important

If the same IP address is mistakenly specified as trusted for different subscribers, the sip:provider PRO will not know which subscriber should be responsible for the call and will randomly select one.

# 5.8 Trusted Subscribers

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

messages from sip:provider PRO. In order to create a Trusted Subscriber you just need to create a normal subscriber, then Create a Permanent Registration via (*Subscribers* $\rightarrow$ *Details* $\rightarrow$ *Registered Devices* $\rightarrow$ *Create Permanent Registration*) and also you need to add the devices IP as Trusted Source in your subscriber's preferences (*Details* $\rightarrow$ *Preferences* $\rightarrow$ *Trusted Sources*). In this way, all messages coming from your device IP will be trusted (and authenticate just via the source IP), on the other side all the SIP messages to your devices will be sent to the SIP URI specified in the Permanent Registration.

# 5.9 Fax Server

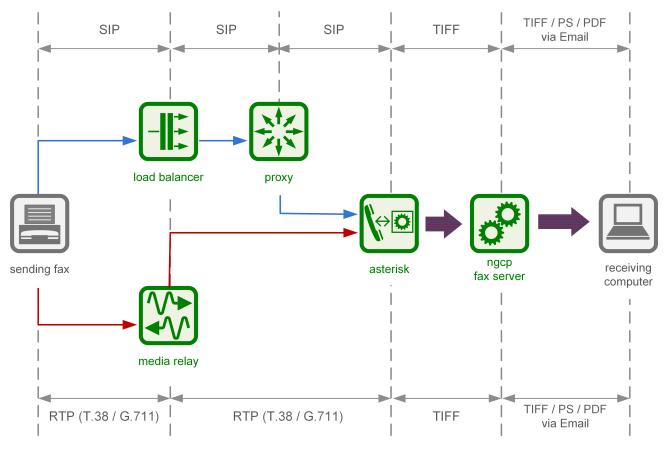
There is a Fax Server included in the sip:provider PRO. The following sections describe its architecture.

The Fax Server is included on the platform and requires no additional hardware. It supports both T38 and G711 codecs and provides a cost-effective paper-free office solution.

For the details of Fax Server configuration options, please see Faxserver Configuration Appendix C chapter in this handbook.

## 5.9.1 Fax2Mail Architecture

To receive faxes via email, a phone call from a sender is connected to the fax application module (Asterisk + NGCP Fax Server) on the sip:provider PRO. The received fax document is converted to the format the receiver has configured (either PS, PDF or TIFF) via the components outlined in the figure below. The email is delivered to one or more configured addresses.

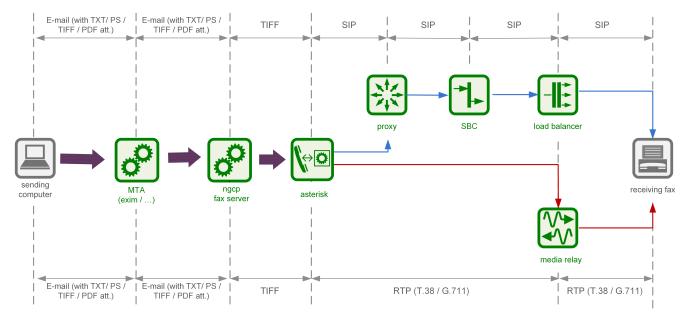


### 5.9.2 Sendfax and Mail2Fax Architecture

To send faxes via the sip:provider PRO a sender can use any email client or an interface such as Webfax or REST API.

Currently, supported formats are TXT, PS, TIFF and PDF.

The document is sent to the NGCP Fax Server instance on the sip:provider PRO. Once successfully queued by the fax server, it is converted to an internal TIFF format and is sent via the components outlined in the below figure to the specified phone number. Of course, a fax device that can receive the document must be connected on the destination side.



# 5.10 Voicemail System

### 5.10.1 Accessing the IVR Menu

For a subscriber to manage his voicebox via IVR, there are two ways to access the voicebox. One is to call the URI voicebox@ yourdomain from the subscriber itself, allowing password-less access to the IVR, as the authentication is already done on SIP level. The second is to call the URI voiceboxpass@yourdomain from any number, causing the system to prompt for a mailbox and the PIN. The PIN can be set in the *Voicemail and Voicebox* section of the *Subscriber Preferences*.

### 5.10.1.1 Mapping numbers and codes to IVR access

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

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

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

### 5.10.1.2 External IVR access

When reaching voiceboxpass, the subscriber is prompted for her mailbox number and a password. All numbers assigned to a subscriber are valid input (primary number and any alias number). By default, the required format is in E.164, so the subscriber needs to enter the full number including country code, for example 4912345 if she got assigned a German number.

You can globally configure a rewrite rule in config.yml using asterisk.voicemail.normalize\_match and aster isk.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.

### 5.10.2 IVR Menu Structure

The following list shows you how the voicebox menu is structured.

- 1 Read voicemail messages
  - 3 Advanced options
    - \* 3 To Hear messages Envelope
    - \* \* Return to the main menu
  - 4 Play previous message
  - 5 Repeat current message
  - 6 Play next message
  - 7 Delete current message
  - 9 Save message in a folder
    - \* 0 Save in new Messages
    - \* 1 Save in old Messages
    - \* 2 Save in Work Messages
    - \* 3 Save in Family Messages
    - \* 4 Save in Friends Messages
    - \* # Return to the main menu
- 2 Change folders
  - 0 Switch to new Messages
  - 1 Switch to old Messages
  - 2 Switch to Work Messages
  - 3 Switch to Family Messages
  - 4 Switch to Friends Messages
  - # Get Back
- 3 Advanced Options

- \* To return to the main menu
- 0 Mailbox options
  - 1 Record your unavailable message
    - \* 1 accept it
    - \* 2 Listen to it
    - \* 3 Rerecord it
  - 2 Record your busy message
    - \* 1 accept it
    - \* 2 Listen to it
    - \* 3 Rerecord it
  - 3 Record your name
    - \* 1 accept it
    - \* 2 Listen to it
    - \* 3 Rerecord it
  - 4 Record your temporary greetings
    - \* 1 accept it / or re-record if one already exist
    - \* 2 Listen to it / or delete if one already exist
    - \* 3 Rerecord it
  - 5 Change your password
  - \* To return to the main menu
- \* Help
- # Exit

### 5.10.3 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.

### 5.10.3.1 Unavailable Message

The standard voice mail greeting is the "unavailable" greeting. This is used if you don't answer the phone and so the call is directed to your voice mailbox.

· You can record a custom unavailable greeting.

- If you have not recorded your unavailable greeting but have recorded your name, the system will play a generic message like: "Recorded name is unavailable."
- If you have not recorded your unavailable greeting, the phone system will play a generic message like: "Digits-of-num ber-dialed is unavailable".

### 5.10.3.2 Busy Message

If you wish, you can record a custom greeting used when someone calls you and you are currently on the phone. This is called your "Busy" greeting.

- You can record a custom busy greeting.
- If you have not recorded your busy greeting but have recorded your name, the phone system will play a generic message: "Recorded name is busy."
- If you have not recorded your busy greeting and have not recorded your name (see below), the phone system will play a generic message: "Digits-of-number-dialed is busy."

### 5.10.3.3 Temporary Greeting

You can also record a temporary greeting. If it exists, a temporary greeting will always be played instead of your "busy" or "unavailable" greetings. This could be used, for example, if you are going on vacation or will be out of the office for a while and want to inform people not to expect a return call anytime soon. Using a temporary greeting avoids having to change your normal unavailable greeting when you leave and when you come back.

### 5.10.4 Folders

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

### 5.10.4.1 The Default Folder List

- 0 New Messages
- 1 Old Messages
- 2 Work Messages
- · 3 Family Messages
- 4 Friends Messages

When a caller leaves a message for you, the system will put the message into the "New Messages" folder. If you listen to the message, but do not delete the message or save the message to a different folder, it will automatically move the message to the "Old Messages" folder. When you first log into your mailbox, the Voicemail System will make the "New Messages" folder the current folder if you have any new messages. If you do not have any new messages the it will make the "Old Messages" folder the current folder.

# 5.10.5 Flowcharts with Voice Prompts

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

#### 5.10.5.1 Listening to New Messages

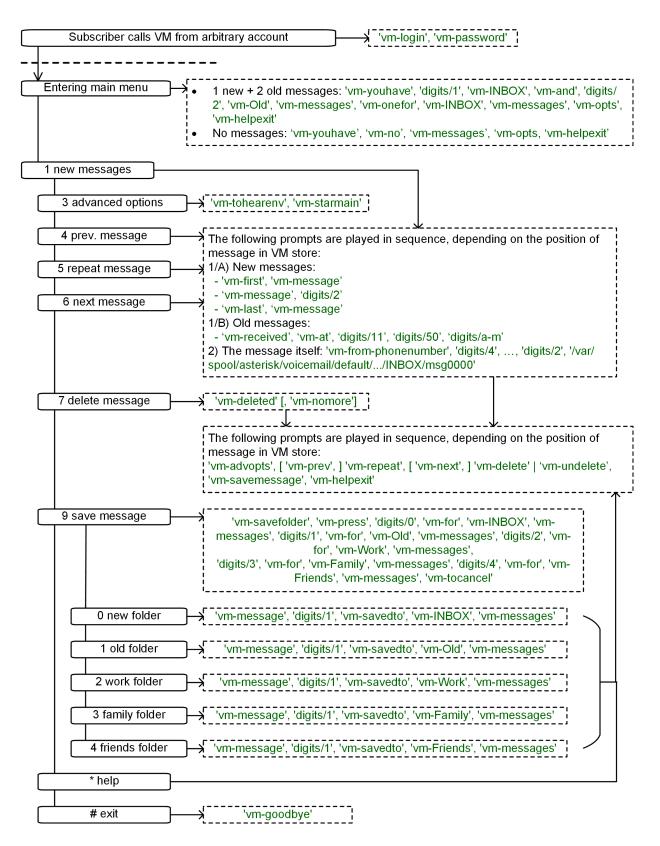


Figure 31: Flowchart of Listening to New Messages

### 5.10.5.2 Changing Voicemail Folders

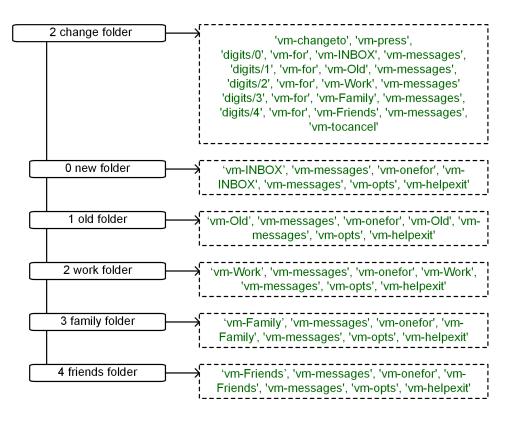


Figure 32: Flowchart of Changing Voicemail Folders

### 5.10.5.3 Mailbox Options

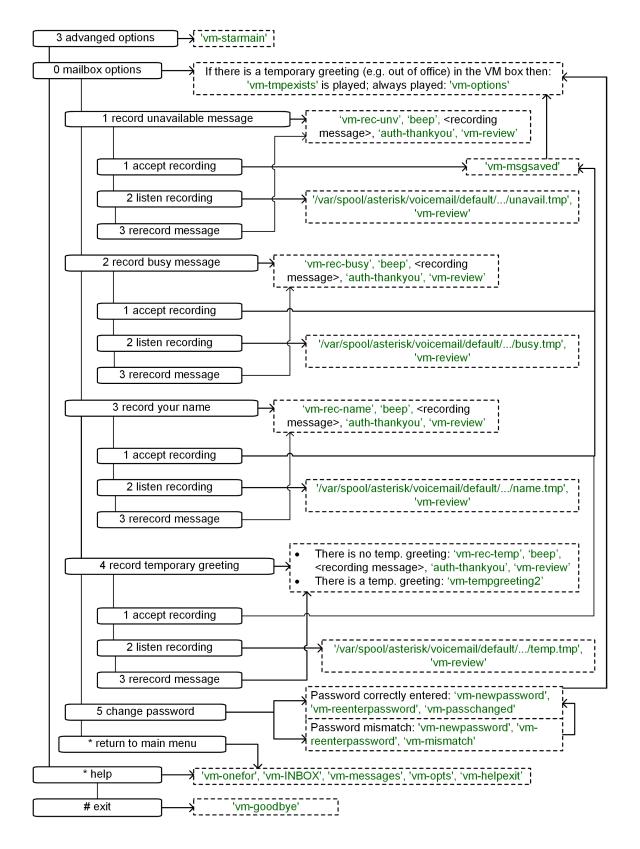


Figure 33: Flowchart of Changing Mailbox Options

## 5.10.5.4 Leaving a Message

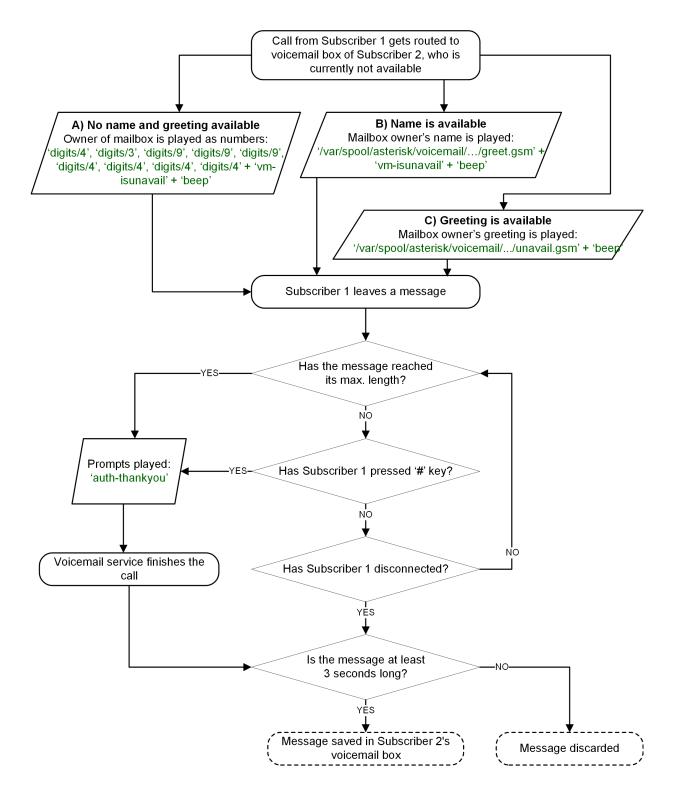


Figure 34: Flowchart of Leaving a Voice Message

# 5.11 Configuring Subscriber IVR Language

The language for the Voicemail system IVR or Vertical Service Codes (VSC) IVRs may be set using the subscriber or domain preference *language*.

0	Edit Preference 'Language for voicemail and app server'								
0	g language g	Spanish		•					
0	8			Save					
0	gpp8	General Purpose Parameter 8							
0	gpp9	General Purpose Parameter 9							
0	conference_pin	PIN for access to pin-protected conference	29165						
0	ua_header_mode	User-Agent header passing mode	Strip						
0	force_outbound_calls_to_peer	Force outbound calls from user or peer to peer	use domain defalut						
0	language	Language for voicemail and app server	use domain default						

The sip:provider PRO provides the pre-installed prompts for the Voicemail in the English, Spanish, French and Italian languages and the pre-installed prompts for the Vertical Service Codes IVRs in English only.

The other IVRs such as the Conference system and the error announcements use the Sound Sets configured in NGCP Panel and uploaded by the administrator in his language of choice.

# 5.12 Sound Sets

The sip:provider PRO provides the administrator with ability to upload the voice prompts such as conference prompts or call error announcements on the *Sound Sets page*. There is a preference *sound\_set* in the *NAT and Media Flow Control* section on Domain and Subscriber levels to link subscribers to the sound set that they should hear (as usual the subscriber preference overrides the domain one). Sound Sets can be defined in *Settings* $\rightarrow$ *Sound Sets*. To create a new Sound Set, click *Create Sound Set*. Then click the *Files* button.

					Logged in as administrate	or 🥹 Language 👻 Logout
sip:wise	NGCP Dashbo	ard		Â		cs 👻 🏭 Settings 👻
Soun	d Sets					
← Back	Create So	ound Set				
Show 5		• entries			Search:	
# ^	Reseller	Customer	Name	Description		
1	default		Conference		🖂 Edit	🔒 Delete 🛛 📰 Files
2	default		Early media rejects	Failed call attempt announcements		
Showing 11	to 2 of 2 entries				e	: ← 1 → ⇒

# Note

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

### 5.12.1 Configuring Early Reject Sound Sets

The call error announcements are grouped under *Early Rejects* section. Unfold the section and click *Upload* next to the sound handles (Names) that you want to use. Choose a WAV file from your file system, and click the Loopplay setting if you want to play the file in a loop instead of just once. Click Save to upload the file.

early_rejects			
Name	Filename	Loop	
block_in			🖾 Uploa
block_out		-	
block_ncos			
block_override_pin_wrong		-	
locked_in			
locked_out		-	
max_calls_in			
max_calls_out		-	
max_calls_peer			
unauth_caller_ip		-	

The call error announcements are played to the user in early media hence the name "Early Reject". If you don't provide the sound files for any handles they will not be used and the sip:provider PRO will fallback to sending the error response code back to the user.

Handle	Description	Message played
block_in	This is what the calling party hears	Your call is blocked by the number you
	when a call is made from a number	are trying to reach.
	that is blocked by the incoming block	
	list (adm_block_in_list, block_in_list	
	subscriber preferences)	
block_out	This is what the calling party hears	Your call to the number you are trying
	when a call is made to a number that	to reach is blocked.
	is blocked by the outgoing block list	
	(adm_block_out_list, block_out_list	
	subscriber preferences)	
block_ncos	This is what the calling party hears	Your call to the number you are trying
	when a call is made to a number that	to reach is not permitted.
	is blocked by the NCOS level assigned	
	to the subscriber or domain (the	
	NCOS level chosen in ncos and	
	adm_ncos preferences)	

Handle	Description	Message played
block_override_pin_wrong	Announcement played to calling party	The PIN code you have entered is not
	if it used wrong PIN code to override	correct.
	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)	
locked_in	Announcement played on incoming	The number you are trying to reach is
	call to a subscriber that is locked for	currently not permitted to receive calls.
	incoming calls	
locked_out	Announcement played on outgoing call	You are currently not allowed to place
	to subscriber that is locked for	outbound calls.
	outgoing calls	
max_calls_in	Announcement played on incoming	The number you are trying to reach is
	call to a subscriber who has exceeded	currently busy. Please try again later.
	the <i>concurrent_max</i> limit by sum of	
	incoming and outgoing calls or whose	
	customer has exceeded the	
	concurrent_max_per_account limit by	
	sum of incoming and outgoing calls	
max_calls_out	Announcement played on outgoing call	All outgoing lines are currently in use.
	to a subscriber who has exceeded	Please try again later.
	the <i>concurrent_max</i> (total limit) or	
	<i>concurrent_max_out</i> (limit on number	
	of outbound calls) or whose customer	
	has exceeded the	
	concurrent_max_per_account or	
	concurrent_max_out_per_account	
	limit	
max_calls_peer	Announcement played on calls from	The network you are trying to reach is
	the peering if that peer has reached	currently busy. Please try again later.
	the maximum number of concurrent	currently busy. Thease try again later.
	calls (configured by admin in	
	concurrent_max preference of peering	
	server)	
unauth_caller_ip	This is what the calling party hears	You are not allowed to place calls from
	when it tries to make a call from	your current network location.
	unauthorized IP address or network	
	(allowed_ips, man_allowed_ips	
	preferences)	

# Table 3: (continued)

Handle	Description	Message played
relaying_denied	Announcement played on inbound call	The network you are trying to reach is
	from trusted IP (e.g. external PBX)	not available.
	with non-local Request-URI domain	
invalid_speeddial	This is what the calling party hears	The speed dial slot you are trying to
	when it calls an empty speed-dial slot	use is not available.
cf_loop	Announcement played when the called	The number you are trying to reach is
	subscriber has the call forwarding	forwarded to an invalid destination.
	configured to itself	
callee_offline	Announcement played on incoming	The number you are trying to reach is
—	call to the subscriber which is currently	currently not available. Please try
	not registered	again later.
callee busy	Announcement played on incoming	The number you are trying to reach is
	call to the subscriber which is currently	currently busy. Please try again later.
	busy (486 response from the UAS)	
callee_unknown	Announcement that is played on call to	The number you are trying to reach is
	unknown or invalid number (not	not in use.
	associated with any of our	not in use.
	subscribers/hunt groups)	
		The number you are trying to reach is
callee_tmp_unavailable	Announcement played on incoming	The number you are trying to reach is
	call to the subscriber which is currently	currently not available. Please try
	unavailable (408, other 4xx or no	again later.
	response code or 30x with malformed	
	contact)	
peering_unavailable	Announcement played in case of	The network you are trying to reach is
	outgoing off-net call when there is no	not available.
	peering rule matching this destination	
	and/or source	
voicebox_unavailable	Announcement played on call to	The voicemail of the number you are
	voicebox if the voicemail server is not	trying to reach is currently not
	configured (system operation is	available. Please try again later.
	impaired)	
emergency_unsupported	Announcement played when	You are not allowed to place
	emergency destination is dialed but	emergency calls from this line. Please
	the emergency calls are	use a different phone.
	administratively prohibited for this user	
	or domain ( <i>reject_emergency</i>	
	preference is enabled)	
emergency_geo_unavailable	Announcement played when	The emergency number you have
	emergency destination is dialed but	dialed is not available in your region.
	the destination is not provisioned for	
	the location of the user.	

# Table 3: (continued)

# Table 3: (continued)

Handle	Description	Message played
no_credit	Announcement played when prepaid	You don't have sufficient credit balance
	account has insufficient balance to	for the number you are trying to reach.
	make a call to this destination	

# 5.13 Conference System

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

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

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

## 5.13.1 Configuring Call Forward to Conference

Go to your Subscriber Preferences and click Edit on the Call Forward Type you want to set (e.g. Call Forward Unconditional).

				🚢 Logg	ed in as administrat	or 🥹 Languag	e 💌 Logout
sip:wise	Edit Call Forward U	nconditi	ional			×	
510.0136	Destination	voicen	nail				ttings 👻
Subso	2 connection	<ul> <li>Confer</li> <li>URI/Nu</li> </ul>	rence				
	URI/Number						
<b>←</b> Back	for (seconds)	300					d Groups
Call For							
Туре					Advanced Viev	w Save	
Call For				_			
Call For	ward Busy						
Call For	ward Timeout						
Call For	ward Unavailable						

You should select *Conference* option in the *Destination* field and leave the *URI/Number* empty. The timeout defines for how long this destination should be tried to ring.

# 5.13.2 Configuring Conference Sound Sets

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

anage Sound	Ser	omerence			
Edit confer	rence_g	reeting		×	$\vdash$
← Bac	Loopplay				d Group
confere	Soundfile	Browse conference_greeting.	wav		
Name					
confere				Save	
conference_pin_wrong		conference_pin_wrong.wav			
conference_joined		conference_joined.wav			
conference_join		conference_join.wav			
conference_leave		conference_leave.wav			
goodbye		goodbye.wav			

Upload the following files:

### Table 4: Conference Sound Sets

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

### Note

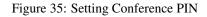
You may use 8 or 16 bit mono WAV audio files.

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

### 5.13.3 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*  $\rightarrow$  *Details*  $\rightarrow$  *Preferences*  $\rightarrow$  *Internals*  $\rightarrow$  *conference\_pin* field.

0	Edit Preference 'I	PIN for access to pin-protec	cted conference'	×
0	conference_pin	29165		
0	g			Save
0	gpho	General Fulpose Farameter o		
0	gpp9	General Purpose Parameter 9		
0	conference_pin	PIN for access to pin-protected conference	29165	
Ð	ua_header_mode	User-Agent header passing mode	Strip	
Ø	force_outbound_calls_to_peer	Force outbound calls from user or peer to peer	use domain defalut	
0	language	Language for voicemail and app server	use domain default	



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.

### 5.13.4 Conference Flowchart with Voice Prompts

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

### 5.13.4.1 Conference Flowchart with PIN Validation

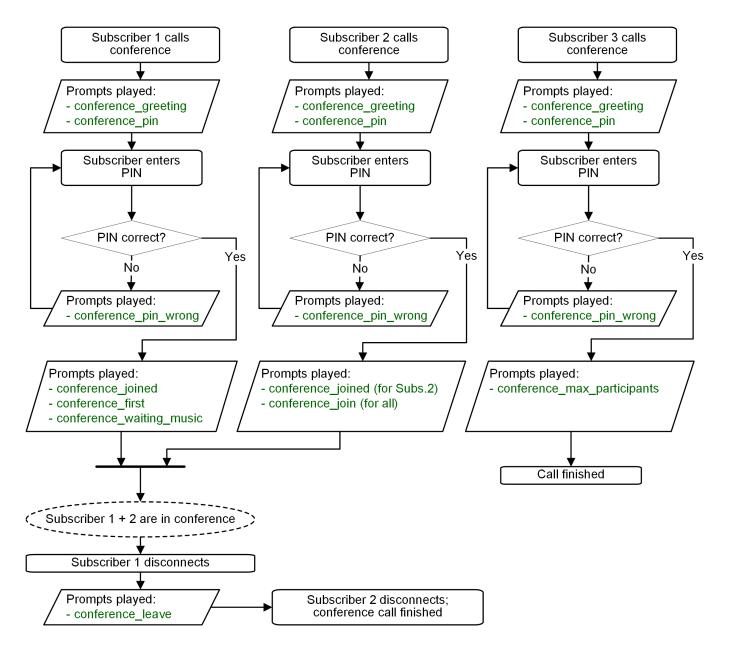


Figure 36: Flowchart of Conference with PIN Validation

### 5.13.4.2 Conference Flowchart without PIN

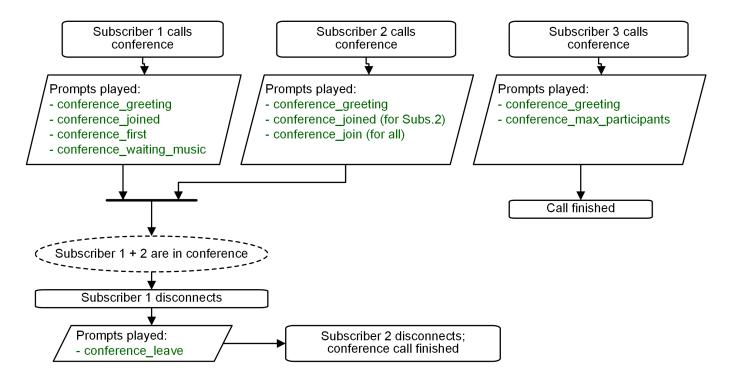


Figure 37: Flowchart of Conference without PIN

# 5.14 Malicious Call Identification (MCID)

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

### 5.14.1 Setup

To enable the feature first edit config.yml and enable there apps:malicious\_call:yes and kamailio:store\_re centcalls:yes. The latter option enables kamailio to store recent calls per subscrbriber UUID in the redis DB (the amount of stored recent calls will not exceed the amount of provisionined subscribers).

Next step is to create a system sound set for the feature. In *Settings* $\rightarrow$ *Sound Sets* either use your already existing *Sound Set* or create a new *Sound Set* and then assign it to your domain or subscribers. In the *Sound Set* there is a fileset *malicious\_call\_identification* $\rightarrow$  for that purpose.

Once the *Sound Set* is created the Subscriber's Preferences *Malicious Call Identification* must be enabled under *Subcriber*  $\rightarrow$  *Preferences*  $\rightarrow$  *Applications* menu. The same parameter can be set in the Customer's preferences to enable this feature for all its subscribers.

The final step is to create a new *Rewrite Rule* and to route calls to, for instance  $*123 \rightarrow MCID$  application. For that you create a *Calee Inbound* rewrite rule  $((*123) \Rightarrow malicious_call)$ 

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

### 5.14.2 Usage

As a subscriber, to report a malicious call you call to either *malicious\_call* or to your custom number assigned for that purpose. Please note that you can report only your last received call. You will hear the media reply from the *Sound Set* you have previosuly configured.

To check reported malicious calls as the plafrom operator open *Settings* $\rightarrow$ *Malicious Calls* tab where you will see a list of registered calls. You can selectively delete records from the list and alternatively you can manage the reported calls by using the REST API.

### 5.14.3 Advanced configuration

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

# 5.15 Subscriber Profiles

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

### 5.15.1 Subscriber Profile Sets

Profile sets define containers for profiles. The idea is to define profile sets with different profiles by the administrator (or the reseller, if he is permitted to do so). Then, a subscriber with administrative privileges can re-assign profiles within his profile sets for the subscribers of his customer account.

Profile Sets can be defined in Settings -> Subscriber Profiles. To create a new Profile Set, click Create Subscriber Profile Set.

						Å Logged	in as administrator	2 Language	e 🔻 Logout
sip:wise N	GCP Dashboard				ñ	aal Monit	oring & Statistics 👻	III Se	ettings 👻
Subsc	Create Subscriber Pı	rofile Sets						×	
	Reseller				Sear	rch:			
← Back		#	Name	Contract	:#		Status		
Dack		1	default	1			active		
Show 5		2	test	2			active		
#		Showing 1 to 2 of 2 entr	ies				← ← 1 -	→ ⇒	
1							Create R	eseller	
2	Name	testset							
3	Description	Test Set							
Showing 1			_			_	l	Save	-

You need to provide a reseller, name and description.

To create Profiles within a Profile Set, hover over the Profile Set and click the Profiles button.

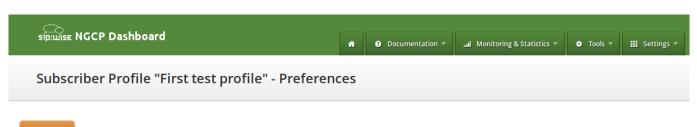
Profiles within a Profile Set can be created by clicking the Create Subscriber Profile button.

	CP Dashboard		• 	
	eate Subscriber P	rofile		× tings -
Subsc	Name	test		
	Description	Test Profile		
🗲 Back	Default Profile	۶		Ξ
Show 5	block_in_mode			
#	block_in_list			
No data a	block_in_clir			
Showing 0	block_out_mode			$\rightarrow$
	block_out_list			
	cfu			<b></b>
				Save

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

### Note

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



🗲 Back

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

ip:wise I	NGCP Dashboar	ł	*	Documentation •	📶 Monitoring & Statistics 👻	Tools -	🗰 Settings 🕶
Subscriber Profile "Test profile 1 for NCOS" - Preferences							
Back							a* Expand Groups
Call Block	ings						
	Attribute	Name	Value				
0	ncos	NCOS Level	Test NCOS for blocking	outca -			

# 5.16 SIP Loop Detection

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

# 5.17 Call-Through Application

Call-through allows telephony client to dial into an IVR system and specify (in two-stage dialing fashion) a new destination number which is then dialed by the sip:provider PRO to connect the client to the destination. As the call-through system needs to be protected from unauthorized use, a list of CLIs which are allowed to use the call-through system is stored in the sip:provider PRO platform.

## Table 5: Call-Through Mappings

Column	Description
uuid	The internal UUID of the call-through subscriber
auth_key	Authentication key (CLI)
source_uuid	The internal UUID of the subscriber that is authorized for
	outgoing call leg (same as uuid in call-through scenario)

### 5.17.1 Administrative Configuration

### 5.17.1.1 Subscriber provisioning

In order to manage the call-through CLIs for subscriber, navigate to *Settings*  $\rightarrow$  *Subscribers*, search for the subscriber you want to edit, press *Details* and then *Preferences*, scroll down to the *Callthrough CLIs* section and press *Edit Callthrough CLIs* button.

					Logged in as administrator
<u> </u>					
sip:wise NGCP Dashboard			ñ	🛛 Documentation 👻	l Monitoring & Statistics 👻
Subscriber Preference	es for 439930	006@10.15.20.133			
← Back					
Successfully updated ccmappings					
Call Forwards					
Voicemail and Voicebox					
Fax Features					
Speed Dial					
Reminder					
Callthrough CLIs					
★ Edit Callthrough CLIs					
Show 5	• entries				Search:
# ^ CLI		Source UUID			
1 431234567890		8bb71bde-ab29-42cc-85dd-dff55e8b33d4			
Showing 1 to 1 of 1 entries					

Using the NGCP Panel the user then creates Call Forward to destination Call Through.

## 5.17.1.2 Forward to local user

If the subscriber has a Call Forward to the call-through application but caller's CLI is not in the authorized CLIs list for call-through, sems responds with error back to proxy and proxy advances to the next number in the Call Forward destinations set. User can enter special destination *Local Subscriber* as next target after *Call Through* in the destinations set in order to terminate the call to the subscriber as if the subscriber didn't exist. This way the user may reach the call-through application from his authorized CLI (e.g. mobile number) and all other callers would reach the SIP subscriber's registered phone as usual.

							nguage 👻
sip:wise NGCP Dashl	board		*	<ul> <li>Documentation *</li> </ul>	d Monitoring & Stati	istics 👻	🗰 Settin
Subscriber Pref	Edit Destination Set				×		
← Back Call Forwards	Name Destination	quickset_cfu Voicemail Conference Fax2Mail					• Expand G
Type Call Forward Unconditional		<ul> <li>Calling Card</li> <li>Call Through</li> <li>Local Subscriber</li> <li>URI/Number</li> </ul>					
Call Forward Busy Call Forward Timeout	URI/Number for (seconds)	300					
Call Forward Unavailable Voicemail and Voicebox	Priority	1			Remove		
Fax Features	Destination	<ul> <li>Voicemail</li> </ul>			Keniove		
Reminder Callthrough CLIs		<ul> <li>Conference</li> <li>Fax2Mail</li> <li>Calling Card</li> <li>Call Through</li> </ul>					
Trusted Sources	[	Local Subscriber					

### 5.17.1.3 Sound Set provisioning

In order for the Callthrough application to work a Sound Set must be created and associated with the Domain or Subscriber.

Sound Sets can be defined in *Settings*  $\rightarrow$  *Sound Sets*. To create a new Sound Set, click *Create Sound Set*. Then click the *Files* button. Administrator can upload the default sounds in one of supported languages or uploaded by the administrator manually in his language of choice.

There is a preference *sound\_set* on Domain and Subscriber levels to link subscribers to the sound set that they should hear (as usual the subscriber preference overrides the domain one).



# Manage Sound Set Calling Card and Call-through

# 🗲 Back

★ Load Default Files

#### Sound set successfully loaded with default files.

# calling\_card

Name	Filename	Loop	
and	and.wav		
busy_ringback_tone			
calling_card_not_found	calling_card_not_found.wav		
connecting	connecting.wav		
could_not_connect	could_not_connect.wav		

### Note

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

## 5.17.2 Call Flow

The call arrives at sems application server with Request-URI user callthrough.

## 5.17.2.1 Internal Header Parameters

The INVITE contains an extra SIP header P-App-Param with the following parameters:

### Table 6: SIP Header parameters for call-through application

Name	Meaning
uuid	The internal UUID of the call-through subscriber

### Table 6: (continued)

Name	Meaning
srcnumber	Caller's CLI for the authentication
outgoing_cli	New CLI to be used by sems application for the outgoing
	call leg

### 5.17.2.2 Caller authorization

Caller is authorized using mapping shown in table above: select source\_uuid from provisioning.voip\_cc\_ma pping where uuid=\$uuid and auth\_key=\$srcnumber;

If the check fails return the configured error response code. Then proceed with the call setup as follows.

### 5.17.2.3 Outgoing call

Sems requests the user to enter destination and starts digit collection. Digit collection process is terminated after 5 seconds (configurable in sems config file) or by pressing the # key. User can start entering destination while the voice prompt is being played.

Sems sends INVITE to the proxy with Request-URI: sip:\$number@\$outboundproxy;sw\_domain=\$subscriber.domain

### From:\$outgoing\_cli

On receiving the 401 or 407 response from the proxy the application authenticates using the digest credentials retrieved for the callthrough subscriber from the voip\_subscribers table:select s.username, s.password, d.domain from pr ovisioning.voip\_subscribers s, provisioning.voip\_domains d where s.uuid=\$source\_uuid a nd s.domain\_id=d.id;

If the call setup fails the application plays back the "could\_not\_connect" sound file. If successful the application acts transparently and does not provide any voice announcements or DTMF detection.

### 5.17.2.4 CLI configuration

The CLI on the outgoing call from the call-through module is set to the Network-Provided Number (NPN) of the call-through subscriber. There is nothing to configure.

# 5.18 Calling Card Application

Calling card application uses a similar concept to call-through except that authorization process operates on the PIN code entered by user using DTMF instead of the CLI. The sip:provider PRO maps incoming UUID of the pilot subscriber to the list of PINs for

calling card application with their corresponding subscriber UUIDs for outbound call leg using table provisioning.voip\_cc \_mapping table {"uuid", "auth\_key", "source\_uuid"}

### Table 7: Calling Cards

Column	Description
uuid	The internal UUID of the pilot subscriber
auth_key	Authentication key (PIN)
source_uuid	The internal UUID of the subscriber that is authorized for
	outgoing call leg

### 5.18.1 Administrative Configuration

### 5.18.1.1 Subscriber provisioning

In order to use the calling cards service the user creates a Call Forward to destination *Calling Card* for the designated subscriber that will be used as access number for this service.

### 5.18.1.2 Sound Set provisioning

In order for the Calling Card application to work a Sound Set must be created and associated with the Domain or Subscriber.

Sound Sets can be defined in Settings  $\rightarrow$  Sound Sets. To create a new Sound Set, click Create Sound Set. Then click the Files button. Administrator can upload the default sounds in one of supported languages or uploaded by the administrator manually in his language of choice.

There is a preference *sound\_set* on Domain and Subscriber levels to link subscribers to the sound set that they should hear (as usual the subscriber preference overrides the domain one).



# Manage Sound Set Calling Card and Call-through

# 🗲 Back

★ Load Default Files

Sound set successfully loaded with default files.

calling_card				
Filename	Loop			
and.wav	•			
	-			
calling_card_not_found.wav	•			
connecting.wav	•			
could_not_connect.wav	-			
	and.wav calling_card_not_found.wav connecting.wav			

### Note

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

## 5.18.1.3 CLI configuration

The CLI on the outgoing call from the calling card app can be configured in one of the following ways using subscriber preferences:

1) Show original caller's CLI: the calling card subscriber shall have allowed\_clis:\* (any). Sems application sends the original caller's CLI in the From header, it is validated by the SIP proxy and sent to outside.

2) Show number of the pilot (calling card) subscriber: the calling card subscriber shall have an empty allowed\_clis and desired number set as value of user\_cli preference. The SIP proxy overrides the original caller's CLI in UPN with the value of the user\_cli preference. The peer must have set outbound\_from\_user, outbound\_from\_display:User-Provided Number (UPN).

### 5.18.2 Call Flow

The call arrives at sems application server with Request-URI user callingcard.

#### 5.18.2.1 Internal Header Parameters

The INVITE contains an extra SIP header P-App-Param with the following parameters:

#### Table 8: SIP Header parameters for calling card application

Name	Meaning
uuid	The internal UUID of the pilot subscriber
outgoing_cli	New CLI to be used by sems application for the outgoing
	call leg

### 5.18.2.2 Caller authorization

- Sems requests the user to enter PIN and starts digit collection. Digit collection process is terminated after 5 seconds (configurable in sems config file) or by pressing the # key. User can start entering destination while the voice prompt is being played.
- Sems checks that PIN is valid and belongs to the pilot subscriber using mapping as shown in the table. It fetches UUID of the subscriber to be used for outgoing call leg: select source\_uuid from provisioning.voip\_cc\_mapping where uuid=\$uuid and auth\_key=\$pin;
- If the check fails sems will request the user to re-enter PIN up to the configured number of times.
- If successful proceed with the call setup making call on behalf of subscriber determined by the source\_uuid key as follows.

### 5.18.2.3 Outgoing call

Sems application plays back the available balance of the customer. Sems requests the user to enter destination and starts digit collection. Digit collection process is terminated after 5 seconds (configurable in sems config file) or by pressing the # key. User can start entering destination while the voice prompt is being played.

Sems sends INVITE to the proxy with Request-URI: sip:\$number@\$outboundproxy;sw\_domain=\$subscriber.domain

From:\$outgoing\_cli

On receiving the 401 or 407 response from the proxy the application authenticates using the digest credentials retrieved for the subscriber for outgoing call leg from the voip\_subscribers table: select s.username, s.password, d.domain f rom provisioning.voip\_subscribers s, provisioning.voip\_domains d where s.uuid=\$source\_ uuid and s.domain\_id=d.id;

### 5.18.2.4 Voucher recharge

During the destination collection phase in calling card application user can enter special code \*1\*<pin># (configurable in sems config file) to transfer balance from other calling card customer to the currently authorized customer. Sems transfers all remaining balance from that customer to the current customer.

### 5.18.2.5 Billing

The call via calling card application as well as call-through generates three CDRs:

- A to B: The incoming call from any source to the call-through subscriber.
- B to callingcard@app.local or callthrough@app.local: The call forward to the sems application.
- B to C: The outgoing call to the final destination. The three CDRs are handled by the billing process as usual, exported and shown in all call lists. .

# 5.19 Invoices and Invoice Templates

Content and vision of the invoices are customizable by invoice templates Section 5.19.2.

### Note

The sip:provider PRO generates invoices in pdf format.

### 5.19.1 Invoices Management

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

sipuise NGCP Dashboard
A Monitoring & Statistics * III Setting
Invoices
← Back ← Create Invoice Show 5 ← entries Search:
# ^ Customer # Customer Email Serial
1 9 ipeshinskaya@sipwise.com INV2014070000001
2 9 ipeshinskaya@sipwise.com INV2014080000002
4 143 ipeshinskaya@sipwise.com INV2014080000004
Showing 1 to 3 of 3 entries = 🛏 🔳 🛁

#### © 2013 Sipwise GmbH, all rights reserved

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

- Template: any of existent invoice template can be selected for the invoice generation.
- Customer: owner of the billing account, recipient of the invoice.
- **Invoice period**: billing period. Can be specified only as one calendar month. Calls with start time between first and last second of the period will be considered for the invoice

All form fields are mandatory.

			👗 Logged in as admir	istrator 🔍 Language 🛩 Logout
sipculse NGCP Dash	board			
			🚓 🔐 "atl Monitoring & St	atistics - 🎬 Settings -
Invoices	Create Invoice		×	
	Template		Search:	
	#	Reseller	Name	
← Back ★ Create	1	default	Default	
Show 5	Showing 1 t	to 1 of 1 entries	⇐ ← 1 → ≈	
	Customer		Search: 175	
# ^ Customer	#	Reseller Contact Email	External # Status	
1 5	175	default sipwise@example.org	active	
Showing 1 to 1 of 1 entries	Showing 1 t	to 1 of 1 entries (filtered from 20 total entries)		
		yy-mm	Create Contract	
	Invoice Period 2015-04			
	• Ap	or 🗸 2015 🖌 🖸		
	Today	Done	Save	

Generated invoice can be downloaded as pdf file.

1 🕨 Page:	1 of 2	- + Automatic Zoom :	53	θ Ι	Bi M	»
		Vour Monthly Statement Vour Customer M. Provided Market Provided Market				
		Recurring Fees				
		Name Quantity Unit Price Total Price in				
		Default Billing Profile 1 0.00 0.00				
		Total 0.00				
		Call Summary				
		Zone Quantity Usage Total Price in				
		Total 0.00				
						~

To do it press button "Download" against invoice in the invoice management interface.

Respectively press on the button "Delete" to delete invoice.

### 5.19.2 Invoice Templates

Invoice template defines structure and look of the generated invoices. The sip:provider PRO makes it possible to create some invoice templates. Multiple invoice templates can be used to send invoices to the different customers using different languages.



## Important

At least one invoice template should be created to enable invoice generation. Each customer has to be associated to one of the existent invoice template, otherwise invoices will be not generated for this customer.

Customer can be linked to the invoice template in the customer interface.

#### 5.19.2.1 Invoice Templates Management

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

sipulise NGCP Dashboard	sip:wise NGCP Dashboard 🖷 🎿 Monitoring & Statistics *
← Back       ★ Create Invoice Template         Search:	
Search: # A Reseller Name Type 1 default Default Svg C Edit Meta C Edit Content X Delete	Invoice Templates
1 default Default Svg 🛛 Edit Meta 📿 Edit Content 🗙 Delete	
	# ^ Reseller Name Type
	1 default Default svg 🛛 Edir Meta 📿 Edir Content 🗙 Delete
Showing I to 1 of 1 entries	Showing 1 to 1 of 1 entries

#### © 2013 Sipwise GmbH, all rights reserve

Invoice template creation is separated on two steps:

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

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

On the invoice template meta information form following parameters can be specified:

- **Reseller**: reseller who owns this invoice template. Please note, that it doesn't mean that the template will be used for the reseller customers by default. After creation, invoice template still need to be linked to the reseller customers.
- Name: unique invoice template name to differentiate invoice templates if there are some.
- Type: currently sip:provider PRO supports only svg format of the invoice templates.

#### All form fields are mandatory.

				👗 Logged in as adminis	trator 🛛 Language 👻 Logout
síp:wise NGCP Dasht	ooard			😤  "ıl Monitoring & Sta	
Invoice Templat	Create Invoice Templa	ate		×	
	Reseller		Search	424093730 3	
	- I	# Name	Contract #	Status	
← Back ★ Create		7 test reseller 1424093730 3	131	active	
Search:	S	Showing 1 to 1 of 1 entries (filtered from	8 total entries)		
# ^ Reselle				Create Reseller	
1 default	Name	English invoice template The	invoice template type (only sv now).		
Showing 1 to 1 of 1 entries	Туре	SVG		~	
				Save	
© 2013 Sipwise GmbH, all rights r	eserved.				

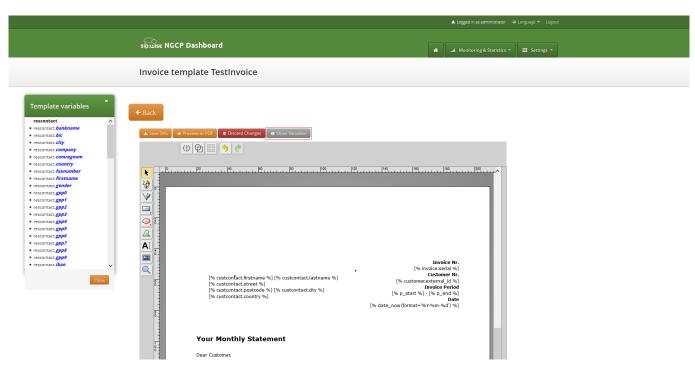
After registering new invoice template you can change invoice template structure in WYSIWYG SVG editor and preview result of the invoice generation based on the template.

#### 5.19.2.2 Invoice Template Content

Invoice template is a XML SVG source, which describes content, look and position of the text lines, images or other invoice template elements. The sip:provider PRO provides embedded WYSIWYG SVG editor svg-edit 2.6 to customize default template. The sip:provider PRO svg-edit has some changes in layers management, image edit, user interface, but this basic introduction still may be useful.

Template refers to the owner reseller contact ("rescontact"), customer contract ("customer"), customer contact ("custcontact"), billing profile ("billprof"), invoice ("invoice") data as variables in the "[%%]" mark-up with detailed information accessed as field name after point e.g. [%invoice.serial%]. During invoice generation all variables or other special tokens in the "[% %]" mark-ups will be replaced by their database values.

Press on "Show variables" button on invoice template content page to see full list of variables with the fields:



You can add/change/remove embedded variables references directly in main svg-edit window. To edit text line in svg-edit main window double click on the text and place cursor on desired position in the text.

After implementation of the desired template changes, invoice template should be saved Section 5.19.2.3.

To return to the sip:provider PRO invoice template default content you can press on the "Discard changes" button.

# ) Important

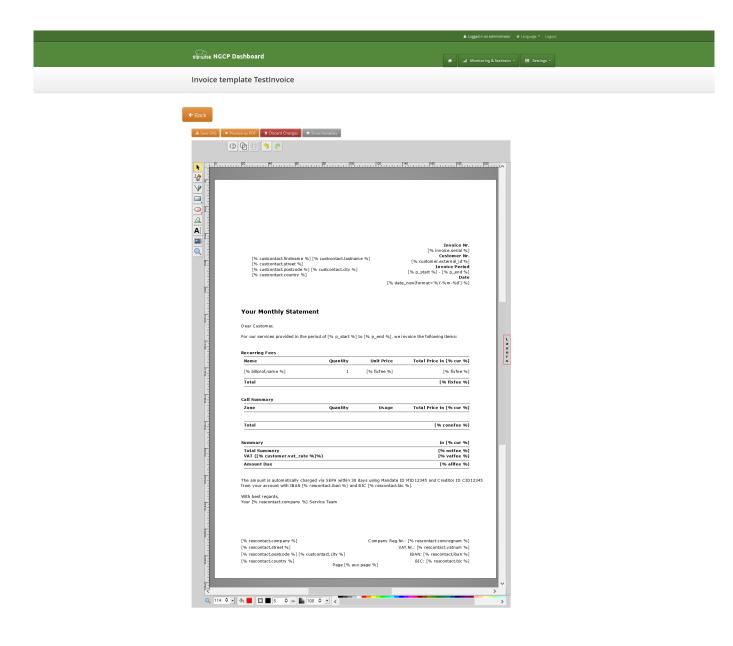
"Discard changes" operation can't be undone.

#### Layers

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

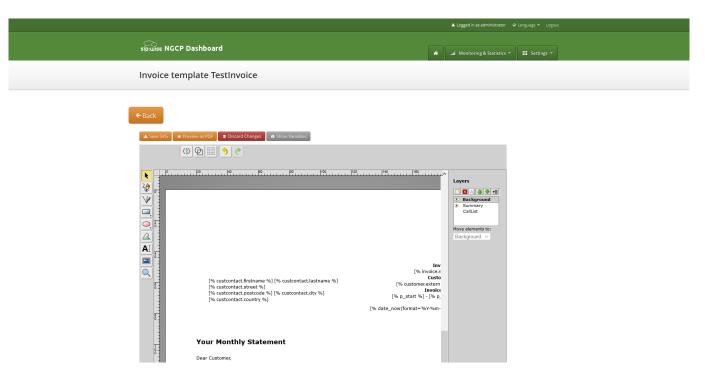
- Background: special layer, which will be repeated as background for every other page of the invoice.
- Summary: page with a invoice summary.
- · CallList: page with calls made in a invoice period. Is invisible by default.

To see all invoice template layers, press on "Layers" vertical sign on right side of the svg-edit interface:



© 2013 Sipwise GmbH, all rights reserved.

Side panel with layers list will be shown.

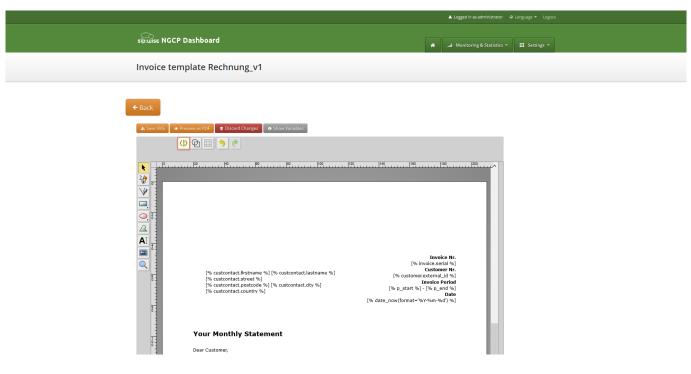


One of the layers is active, and its element can be edited in the main svg-edit window. Currently active layer's name is **bold** in the layers list. The layers may be visible or invisible. Visible layers have "eye" icon left of their names in the layers list.

To make a layer active, click on its name in the layers list. If the layer was invisible, its elements became visible on activation. Thus you can see mixed elements of some layers, then you can switch off visibility of other layers by click on their "eye" icons. It is good idea to keep visibility of the "Background" layer on, so look of the generated page will be seen.

## Edit SVG XML source

Sometimes it may be convenient to edit svg source directly and svg-edit makes it possible to do it. After press on the <svg> icon in the top left corner of the svg-edit interface:



SVG XML source of the invoice template will be shown.

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

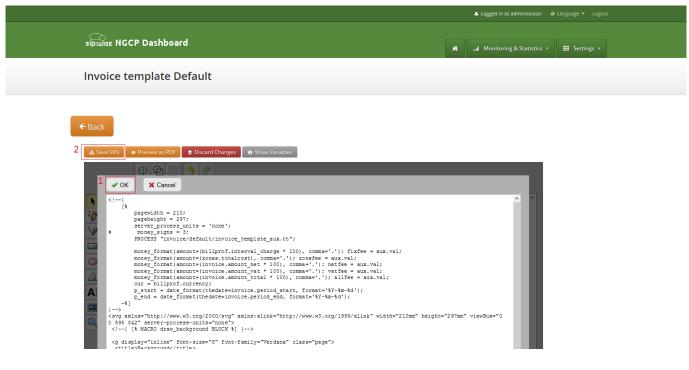
## Note

Template keeps sizes and distances in pixels.

#### Important

When edit svg xml source, please change very carefully and thinkfully things inside special comment mark-up "<!--{ }- $\rightarrow$ ". Otherwise invoice generation may be broken. Please be sure that document structure repeats default invoice template: has the same groups (<g/>>) elements on the top level, text inside special comments mark-up "<!--{ }- $\rightarrow$ " preserved or changed appropriately, svg xml structure is correct.

To save your changes in the svg xml source, first press "OK" button on the top left corner of the source page:



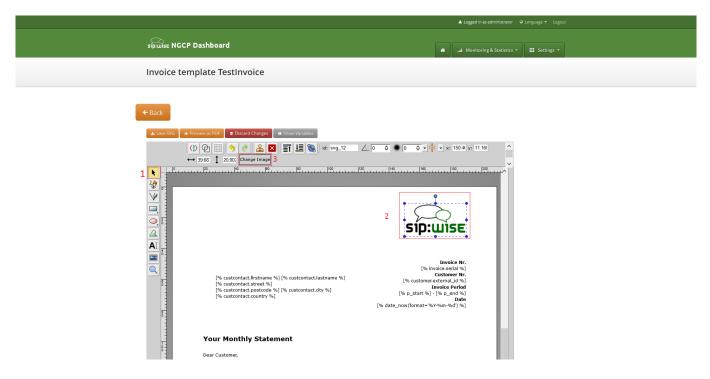
And then save invoice template changes Section 5.19.2.3.

#### Note

You can copy and keep the svg source of your template as a file on the disk before start experimenting with the template. Later you will be able to return to this version replacing svg source.

#### Change logo image

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



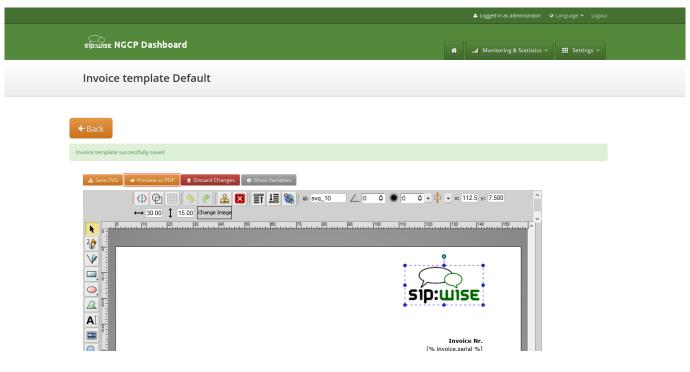
After image uploaded save invoice template changes Section 5.19.2.3.

#### 5.19.2.3 Save and preview invoice template content

To save invoice template content changes press button "Save SVG".

	Logged in as administrator	
sip:Lilise NGCP Dashboard	AI Monitoring & Statistics 👻	₩ Settings +
Invoice template Default		
← Back		
▲ Save SVG		
Image: Constraint of the second se	LIISE Invoice Nr. oice.serial %] Justomer Nr.	

You will see message about successfully saved template. You can preview your invoice look in PDF format. Press on "Preview as PDF" button.



Invoice preview will be opened in the new window.

## Note

Example fake data will be used for preview generation.

: 1 of 4		- + Autor	naticZoom 🗧		2 0 D
				SIP:WISE	
	Customerfirst Customerlast Customerstreet 12/3 12345 Customercity Customercountry			Invoice Nr. 1234567 Customer Nr. Resext1234567800 Invoice Period 2015-04-01 - 2015-04-30 Date 2015-04-05	
You	ur Monthly Statemen	ıt			
Dear	Customer,				
For c	ur services provided in the period	of 2015-04-01 to 2015-	04-30, we invoice the foll	owing items:	
Rec	Irring Fees				
Na	me	Quantity	Unit Price	Total Price in EUR	
Tes	Billing Profile	1	29.90	29.90	
Tot	al			29.90	

## 5.19.3 Invoices Generation

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

## Warning

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

Script is located at: /usr/share/ngcp-panel/tools/generate\_invoices.pl

In short:

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

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

· To generate invoices for the previous month without sending:

perl /usr/share/ngcp-panel/tools/generate\_invoices.pl --prevmonth

· To send already generated invoices for the previous month:

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

· Regenerate invoices for the specified period:

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

Some not obvious options:

- \*--allow\_terminated\* Generates invoices for the terminated contracts too.
- \*--force\_unrated\* Generate invoices despite unrated calls existence in the specified generation period.
- \*--no\_empty\* Skip invoices for the contracts without calls in the specified period and with null permanent fee for the billing profile.

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

/usr/share/ngcp-panel/tools/generate\_invoices.pl --man

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

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

After applying your configuration cron file will be located at:

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

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

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

• **RESELLER\_ID** *1,2,3,...N* 

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

• CLIENT\_CONTRACT\_ID 1,2,3,...N

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

• STIME YYYY-mm-DD HH:MM:SS

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

#### • ETIME YYYY-mm-DD HH:MM:SS

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

• SEND [0/1]

Generated invoices will be immediately sent to the customers.

#### • **RESEND** [0/1]

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

## • REGENERATE [0|1]

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

#### • ALLOW\_TERMINATED [0/1]

Generate invoices for the already terminated customers too.

#### • ADMIN\_EMAIL your@email.com

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

All generated invoices can be seen in the invoice management interface Section 5.19.1.

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

## 5.20 Email Reports and Notifications

#### 5.20.1 Email events

The sip:provider PRO makes it possible to customize content of the emails sent on the following actions:

- Web password reset requested. Email will be sent to the subscriber, whom password was requested for resetting. If the subscriber doesn't have own email, letter will be sent to the customer, who owns the subscriber.
- New subscriber created. Email will be sent to the newly created subscriber or to the customer, who owns new subscriber.
- Letter with the invoice. Letter will be sent to the customer.

#### 5.20.2 Initial template values and template variables

Default email templates for each of the email events are inserted on the initial sip:provider PRO database creation. Content of the default template is described in the corresponding sections. Default email templates aren't linked to any reseller and can't be changed through sip:provider PRO Panel. They will be used to initialize default templates for the newly created reseller.

Each email template refers to the values from the database using special mark-ups "[%" and "%]". Each email template has fixed set of the variables. Variables can't be added or changed without changes in the sip:provider PRO Panel code.

#### 5.20.3 Password reset email template

Email will be sent after subscriber or subscriber administrator requested password reset for the subscriber account. Letter will be sent to the subscriber. If subscriber doesn't have own email, letter will be sent to the customer owning the subscriber.

Template name	passreset_default_email
From	default@sipwise.com
Subject	Password reset email

Default content of the password reset email template is:

Body	
	Dear Customer,
	Please go to [%url%] to set your password and log into your self-care $\leftrightarrow$ interface.
	Your faithful Sipwise system
	 This is an automatically generated message. Do not reply.

Following variables will be provided to the email template:

- [%url%]: specially generated url where subscriber can define his new password.
- [%subscriber%]: username@domain of the subscriber, which password was requested for reset.

#### 5.20.4 New subscriber notification email template

Email will be sent on the new subscriber creation. Letter will be sent to the newly created subscriber if it has an email. Otherwise, letter will be sent to the customer who owns the subscriber.

#### Note

By default email content template is addressed to the customer. Please consider this when create the subscriber with an email.

Template name	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.

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.

#### 5.20.5 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,	
	<pre>Please find your invoice #[%invoice.serial%] for [%invoice. ↔     period_start_obj.month_name%], [%invoice.period_start_obj.year%] in attac     letter.</pre>	chmen <sup>.</sup>
	Your faithful Sipwise system	
	 This is an automatically generated message. Do not reply.	

Variables passed to the email template:

• [%invoice%]: container variable for the invoice information.

## Invoice fields

- [%invoice.serial%]
- [%invoice.amount\_net%]
- [%invoice.amount\_vat%]
- [%invoice.amount\_total%]
- [%invoice.period\_start\_obj%]
- [%invoice.period\_end\_obj%]

The fields [%invoice.period\_start\_obj%] and [%invoice.period\_end\_obj%] provide methods of the perl package DateTime for the invoice start date and end date. Further information about DateTime can be obtained from the package documentation: man DateTime

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

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

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

## 5.20.6 Email templates management

Email templates linked to the resellers can be customized in the email templates management interface. For the administrative account email templates of all the resellers will be shown. Respectively for the reseller account only owned email templates will be shown.

sip:wise NGCP Dashb				
	oard		者 🛄 Monitor	ing & Statistics * 🗰 Settings *
Email Templates				
← Back ★ Create E	mail Template			
Show 5	✓ entries		Searc	h:
# AReseller	Name	From	Subject	
10 New Reseller	subscriber_default_email	default@sipwise.com	Subscriber created	🗷 Edit 💼 Delete
11 New Reseller	passreset_default_email	default@sipwise.com	Password reset email	
12 New Reseller	invoice_default_email	default@sipwise.com	Invoice #[%invoice.serial%] fro [%invoice.period_start_obj.ym [%invoice.period_end_obj.ymd	1%] to
Showing 1 to 3 of 3 entries				

## To create create new email template press button "Create Email Template".

Image: Section Sect	Email Template     Create Email Template     Reseller     Reseller     * Create Email Template     * Create Email Template <th></th>										
Create Email Template         Reseller       Search: New Reseller         Image: Show's       Name       Contract #       Status         Show's       Showing 1 to 1 of 1 entries (filtered from 9 total entries)       Image: Create Reseller         Image: Now Reselt       Name       Create Reseller         Image: Now Reselt       Name       Image: Create Reseller         Image: Now Reselt       Subject       Image: Create Reseller       Image: Create Reseller         Image: Now Reselt	Create Email Template         Reseller         Search: New Reseller         Image: Show 5       Image: Show 1 to 1 of 1 entries (filtered from 9 total entries)       Image: Show 1 to 1 of 1 entries (filtered from 9 total entries)         Image: Show 6       Name       Image: Show 1 to 1 of 1 entries (filtered from 9 total entries)       Image: Show 1 to 1 of 1 entries (filtered from 9 total entries)         Image: Show 7       Image: Show 1 to 1 of 1 entries (filtered from 9 total entries)       Image: Show 1 to 1 of 1 entries (filtered from 9 total entries)       Image: Show 1 to 1 of 1 entries (filtered from 9 total entries)         Image: Show 7       Image: Show 1 to 1 of 1 entries (filtered from 9 total entries)       Image: Show 1 to 1 of 1 entries (filtered from 9 total entries)         Image: Show 7       Image: Show 1 to 3 of 3 entries       Subject       Image: Show 1 to 3 of 3 entries       Image: Show 1 to 3 of 3 entries		GCP Dashb	oard				ń	.al Monitorin		
Reseller     Name     Contract #   Status      10     Name     Showing 1 to 1 of 1 entries (filtered from 9 total entries)     In Reseller     In Resel	Reseller     Show 5     Reseller     Show 5     Reseller     Name     Contract #     Status     21     New Reseller     10     New Resell     Name     11     New Resell     Subject     Body Template        Showing 1 to 3 of 3 entries     Reseller     Subject     A new subscriber %i has been created for you	Email Te	emplate	Create Email Template	e					×	
Create     Show 5     Image: Create Reseller     Image: Create R	21 New Reseller   10 New Resell   11 New Resell   12 New Resell   Subject   12 New Resell   Subject   Body Template   Dear Customer, A new subscriber % las been created for you			Reseller			Search:	New Reseller			
Back Create      Showing 1 to 1 of 1 entries (filtered from 9 total entries)      Reseller      Reseller      Name      Reseller      Name      Reseller      Dear Customer,      A new subscriber % has been created for you	Create     Show 5     Bow 5        Show 5     Show 5     Show 6     Show 6     Show 7     S										
Brown 5     Diswing 1 to 1 of 1 Endles (indeed indees of table endles)       #     Reseller       10     New Reself       11     New Reself       12     New Reself       Showing 1 to 3 of 3 entries     A new subscriber 1% subscriber %1 has been created for you	Brown 5     Create Reseller       #     • Resell       10     New Resell       11     New Resell       12     New Resell       Showing 1 to 3 of 3 entries         Showing 1 to 3 of 3 entries	← Back	🖈 Create		21	New Reseller	184	active	✓	- 60	
10       New Reself       Name         11       New Reself       From Email Address         12       New Reself       Subject         Body Template       Dear Customer,         Showing 1 to 3 of 3 entries       A new subscriber % has been created for you	10       New Reself       Name         11       New Reself       From Email Address         12       New Reself       Subject         Body Template       Dear Customer,         Showing 1 to 3 of 3 entries       A new subscriber % lass been created for you	Show 5		SI	showing 1 to 1 of 1	entries (filtered from 9	total entries)	←	1 → ⇒		
11       New Reself       From Email Address         12       New Reself       Subject         Body Template       Dear Customer,         Showing 1 to 3 of 3 entries       A new subscriber 1% subscriber %1 has been created for you	11       New Resell       From Email Address         12       New Resell       Subject         Body Template       Dear Customer,         Showing 1 to 3 of 3 entries       A new subscriber % lass been created for you	#	A Reseller						Create Reseller		
12       New Reseli       Subject         Body Template       Dear Customer,         A new subscriber % bas been created for you       ✓	12     New Resell     Subject       Body Template     Dear Customer,       A new subscriber % lass been created for you	10	New Reselle	Name							
Subject Body Template Dear Customer, A new subscriber % has been created for you	Body Template     Dear Customer,       Showing 1 to 3 of 3 entries     A new subscriber 1% subscriber %1 has been created for you	11	New Reselle	From Email Address							
Showing 1 to 3 of 3 entries A new subscriber 1% subscriber %1 has been created for you	Showing 1 to 3 of 3 entries A new subscriber 1% subscriber % has been created for you	12	New Reselle	Subject							
A new subschiel to subschiel on has been cleared to you	A new subschilder Pa subschilder av has been created on you			Body Template	Dear Customer,						
Save	Save	Showing 1 to 3 o	of 3 entries		A new subscrib	er f% subscriber %1 h;	as been created for vo	ווכ		~ <u> </u>	- 1 - =
									Sa	re	
										_	

On the email template form all fields are mandatory:

- Reseller: reseller who owns this email template.
- Name: currently only email template with the following names will be considered by the sip:provider PRO on the appropriate event Section 5.20.1 :
  - passreset\_default\_email;
  - subscriber\_default\_email;

- invoice\_default\_email;
- From Email Address: email address which will be used in the From field in the letter sent by the sip:provider PRO.
- · 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.

## 5.21 The Vertical Service Code Interface

*Vertical Service Codes* (VSC) are codes a user can dial on his phone to provision specific features for his subscriber account. The format is \*<code>\*<value> to activate a specific feature, and #<code> or #<code># to deactivate it. The *code* parameter is a two-digit code, e.g. 72. The *value* parameter is the value being set for the corresponding feature.



## Important

The value user input is normalized using the Rewrite Rules Sets assigned to domain as described in Section 4.6.

By default, the following codes are configured for setting features. The examples below assume that there is a domain rewrite rule normalizing the number format 0 < ac > < sn > using 43 as country code.

- 72 enable *Call Forward Unconditional* e.g. to 431000 by dialing \*72\*01000, and disable it by dialing #72.
- 90 enable Call Forward on Busy e.g. to 431000 by dialing \*90\*01000, and disable it by dialing #90.
- 92 enable *Call Forward on Timeout* e.g. after 30 seconds of ringing to 431000 by dialing \*92\*30\*01000, and disable it by dialing #92.
- 93 enable Call Forward on Not Available e.g. to 431000 by dialing \*93\*01000, and disable it by dialing #93.
- 50 set Speed Dial Slot, e.g. set slot 1 to 431000 by dialing \*50\*101000, which then can be used by dialing \*1.
- 55 set One-Shot Reminder Call e.g. to 08:30 by dialing \*55\*0830.
- 31 set Calling Line Identification Restriction for one call, e.g. to call 431000 anonymously dial \*31\*01000.
- 32 enable *Block Incoming Anonymous Calls* by dialing \*32\*, and disable it by dialing #32.
- 80 call using Call Block Override PIN, number should be prefixed with a block override PIN configured in admin panel to disable the outgoing user/admin block list and NCOS level for a call. For example, when override PIN is set to 7890, dial \*80\*789001000 to call 431000 bypassing block lists.

## 5.21.1 Vertical Service Codes for PBX customers

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

NGCP provides the following PBX-specific VSCs:

• 97 - *Call Parking:* during a conversation the subscriber can park the call with his phone to a "parking slot" and later on continue the conversation from another phone. To do that, a destination must be dialled as follows: \*97\*3; this will park the call to slot no. 3.

PLEASE NOTE:

- Cisco IP phones provide a softkey for Call Parking, that means the subscriber must only dial the parking slot number after pressing "Park" softkey on the phone.
- Other IP phones can perform Call Parking as a *blind transfer*, where the destination of the transfer must be dialled in the format described above.
- Both the caller and the callee can park the call.
- 98 *Call Unparking:* if a call has been parked, a subscriber may continue the conversation from any extension (phone) under the same PBX customer. To do that, the subscriber must dial the following sequence: \*98\*3; this will pick up the call that was parked at slot no. 3.
- 99 Directed Call Pickup: if a subscriber's phone is ringing (e.g. extension 23) and another subscriber wants to answer the call instead of the original callee, he may pick up the call by dialling \*99\*23 on his phone.

## 5.21.2 Configuration of Vertical Service Codes

You can change any of the codes (but not the format) in /etc/ngcp-config/config.yml in the section sems  $\rightarrow$  vsc. After the changes, execute ngcpcfg apply 'changed VSC codes'.

## Caution

If you have the EMTAs under your control, make sure that the specified VSCs don't overlap with EMTA-internal VSCs, because the VSC calls must be sent to the NGCP via SIP like normal telephone calls.

## 5.21.3 Voice Prompts for Vertical Service Code Configuration

#### Table 9: 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.

Prompt Handle	Related VSC	Message
vsc_cfb_off	90 (Call Forward Busy)	Your call forward on busy has
		successfully been deactivated.
vsc_cft_on	92 (Call Forward on Timeout)	Your call forward on ring timeout has
		successfully been activated.
vsc_cft_off	92 (Call Forward on Timeout)	Your call forward on ring timeout has
		successfully been deactivated.
vsc_cfna_on	93 (Call Forward on Not Available)	Your call forward while not reachable
		has successfully been activated.
vsc_cfna_off	93 (Call Forward on Not Available)	Your call forward while not reachable
		has successfully been deactivated.
vsc_speeddial	50 (Speed Dial Slot)	Your speed dial slot has successfully
		been stored.
vsc_reminder_on	55 (One-Shot Reminder Call)	Your reminder has successfully been
		activated.
vsc_reminder_off	55 (One-Shot Reminder Call)	Your reminder has successfully been
		deactivated.
vsc_blockinclir_on	32 (Block Incoming Anonymous Calls)	Your rejection of anonymous calls has
		successfully been activated.
vsc_blockinclir_off	32 (Block Incoming Anonymous Calls)	Your rejection of anonymous calls has
		successfully been deactivated.

Table 9: (continued)

## 5.22 Handling WebRTC Clients

WebRTC is an open project providing browsers and mobile applications with Real-Time Communications (RTC) capabilities. Configuring your platform to offer WebRTC is quite easy and straightforward. This allows you to have a SIP-WebRTC bridge in place and make audio/video call towards normal SIP users from WebRTC clients and vice versa. Sip Provider listens, by default, on the following WebSockets and WebSocket Secure: ws://your-ip:5060/ws,wss://your-ip:5061/ws and wss: //your-ip:1443/wss/sip/.

The WebRTC subscriber is just a normal subscriber which has just a different configuration in his Preferences. You need to change the following preferences under Subscribers  $\rightarrow$  Details  $\rightarrow$  Preferences  $\rightarrow$  NAT and Media Flow Control:

- use\_rtpproxy: Always with rtpproxy as additional ICE candidate
- transport\_protocol: RTP/SAVPF (encrypted SRTP with RTCP feedback)

The transport\_protocol setting may change, depending on your WebRTC client/browser configuration. Supported protocols are the following:

- Transparent (Pass through using the client's transport protocol)
- RTP/AVP (Plain RTP)
- RTP/SAVP (encrypted SRTP)
- RTP/AVPF (RTP with RTCP feedback)
- RTP/SAVPF (encrypted SRTP with RTCP feedback)
- UDP/TLS/RTP/SAVP (Encrypted SRTP using DTLS)
- UDP/TLS/RTP/SAVPF (Encrypted SRTP using DTLS with RTCP feedback)

# Warning

The below configuration is enough to handle a WebRTC client/browser. As mentioned, you may need to tune a little bit your transport\_protocol configuration, depending on your client/browser settings.

In order to have a bridge between normal SIP clients (using plain RTP for example) and WebRTC client, the normal SIP clients' preferences have to have the following configuration:

## transport\_protocol: RTP/AVP (Plain RTP)

This will teach Sip Provider to translate between Plain RTP and RTP/SAVPF when you have calls between normal SIP clients and WebRTC clients.

## 5.23 XMPP and Instant Messaging

Instant Messaging (IM) based on XMPP comes with sip:provider PRO out of the box. sip:provider PRO uses prosody as internal XMPP server. Each subscriber created on the platform have assigned a XMPP user, reachable already - out of the box - by using the same SIP credentials. You can easily open an XMPP client (e.g. Pidgin) and login with your SIP username@domain and your SIP password. Then, using the XMPP client options, you can create your buddy list by adding your buddies in the format user@domain.

# 6 Customer Self-Care Interface and Menus

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

## 6.1 The Customer Self-Care Web Interface

The NGCP provides a web panel for end users (CSC panel) to maintain their subscriber accounts, which is running on *https://<ngcp-ip>*. Every subscriber can log in there, change subscriber feature settings, view their call lists, retrieve voicemail messages and trigger calls using the click-to-dial feature.

## 6.1.1 Login Procedure

To log into the CSC panel, the end user has to provide his full web username (e.g. user1@1.2.3.4) and the web password defined in Section 4.2. Once logged in, he can change his web password in the *Account* section. This will NOT change his SIP password, so if you control the end user devices, you can auto-provision the SIP password into the device and keep it secret, and just hand over the web password to the customer. This way, the end user will only be able to place calls with this auto-provisioned device and not with an arbitrary soft-phone, but can nonetheless manage his account via the CSC panel.

## 6.1.2 Site Customization

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

## 6.1.2.1 Changing the Logo

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

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

The web panel customisation happens as follows:

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

Edit Reseller Brandin	g	×
Logo	Browse mouse.jpg	
CSS		
	Save	

Figure 38: CSC Customisation Step 1: Select an image

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

Reseller branding successfully updated
🖸 Edit Branding 🗴 💼 Delete Logo
Custom Logo
You can use the logo by adding the following CEE to the Eastorn CEE below.
<pre>#header .brand {     background: url(https://10.15.18.227:1443/reseller/3/css/logo/download) no-repeat 0 0;     background-size: 280px 32px; /</pre>
Custom CSS

Figure 39: CSC Customisation Step 2: Copy CSS code

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

Keseller Branding	•
Logo Browse No file selected.	
CSS #header .brand { background: url(https://10.15.18.227:1443/reseller/3/css/logo/download) no-repeat 0 0; background-size: 330px 230px; }	
Save	

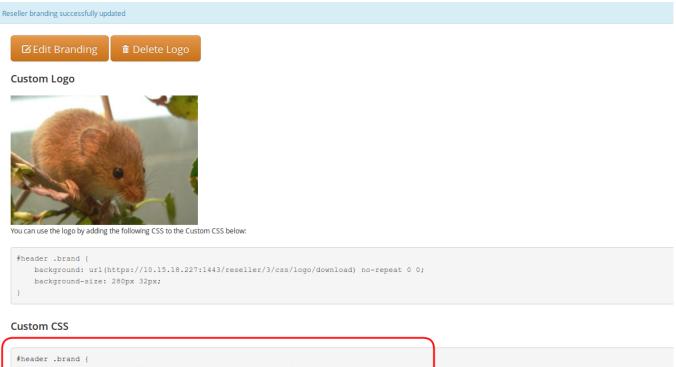
Figure 40: CSC Customisation Step 3: Paste CSS code

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

Edit Reseller Brandi	ng	×
Logo	Browse No file selected.	
CSS	<pre>#header .brand {     background: url(https://10.15.18.227:1443/reseller/3/css/logo/download) no-repeat 0 0;     background-size: 330px 230px; } #footer &gt; div.container { visibility: hidden }</pre>	
	Sav	re

Figure 41: CSC Customisation: Hide copyright notice

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



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

Figure 42: CSC Customisation: Custom data on panel

## 6.1.2.2 Other Website Customisations

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

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

# Warning

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

## 6.1.2.3 Selecting Available Languages

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

You can select the *default language* provided by CSC by changing the parameter www\_admin.force\_language in /etc/ ngcp-config/config.yml file. An example to set the English language as default: force\_language:en

## 6.2 The Voicemail Menu

NGCP offers several ways to access the Voicemail box.

The CSC panel allows your users to listen to voicemail messages from the web browser, delete them and call back the user who left the voice message. User can setup voicemail forwarding to the external email and the PIN code needed to access the voicebox from any telephone also from the CSC panel.

To manage the voice messages from SIP phone: simply dial internal voicemail access number 2000.

To change the access number: look for the parameter *voicemail\_number* in */etc/ngcp-config/config.yml* in the section  $sems \rightarrow vsc$ . After the changes, execute *ngcpcfg apply 'changed voicebox number'*.

## Тір

To let the callers leave a voice message when user is not available he should enable Call Forward to Voicebox. The Call Forward can be provisioned from the CSC panel as well as by dialing Call Forward VSC with the voicemail number. E.g. when parameter *voicemail\_number* is set to *9999*, a Call Forward on Not Available to the Voicebox is set if the user dials \*93\*9999. As a result, all calls will be redirected to the Voicebox if SIP phone is not registered.

#### To manage the voice messages from any phone:

- As an operator, you can setup some DID number as external voicemail access number: for that, you should add a special rewrite rule (Inbound Rewrite Rule for Callee, see Section 4.6.) on the incoming peer, to rewrite that DID to "voiceboxpass". Now when user calls this number the call will be forwarded to the voicemail server and he will be prompted for mailbox and password. The mailbox is the full E.164 number of the subscriber account and the password is the PIN set in the CSC panel.
- The user can also dial his own number from PSTN, if he setup Call Forward on Not Available to the Voicebox, and when reaching the voicemail server he can interrupt the "user is unavailable" message by pressing \* key and then be prompted for the PIN. After entering PIN and confirming with # key he will enter own voicemail menu. PIN is random by default and must be kept secret for that reason.

# 7 Billing Configuration

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

## 7.1 Billing Profiles

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

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

## 7.1.1 Creating Billing Profiles

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

					💄 Logged i	n as administrator Logout
sip:wise Cre	eate Billing Profile	25			_	× tings -
Billin	Reseller			Search:		
		#	Name	Contract #	Status	
		1	default	1	active 1	
← Back		Showing 1 to 1 of 1 eni	ries		⇐ ← 1 · Create R	
#	Handle	mytestprofile	2			
1	Name	My Test Profile	3			
Showing 1 t	Prepaid					-
	Interval charge	0				
© 2013 Sip					4	Save

The fields Reseller, Handle and Name are mandatory.

- Reseller: The reseller this billing profile belongs to.
- · Handle: A unique, permanently fixed string which is used to attach the billing profile to a customer or SIP peering contract.
- Name: A free form string used to identify the billing profile in the Admin Panel. This may be changed at any time.
- Prepaid: Enables prepaid accounting for this profile as opposed to normal post-paid mode.
- Interval charge: A base fee for the billing interval, specifying a monetary amount (represented as a floating point number) in whatever currency you want to use.
- Interval free time: If you want to include free calling time in your billing profile, you may specify the number of seconds that are available every billing interval. See *Creating Billing Fees* below on how to select destinations which may be called using the free time.
- Interval free cash: Same as for *interval free time* above, but specifies a monetary amount which may be spent on outgoing calls. This may be used for example to implement a minimum turnover for a contract, by setting the *interval charge* and *interval free cash* to the same values.
- Fraud monthly limit: The monthly fraud detection limit (in Cent) for accounts with this billing profile. If the call fees of an account reach this limit within a billing interval, an action can be triggered.
- Fraud monthly lock: a choice of *none, foreign, outgoing, incoming, global.* Specifies a lock level which will be used to lock the account and his subscribers when *fraud monthly limit* is exceeded.

- Fraud monthly notify: An email address or comma-separated list of email addresses that will receive notifications when *fraud* monthly limit is exceeded.
- Fraud daily limit: The fraud detection limit (in Cent) for accounts with this billing profile. If the call fees of an account reach this limit within a calendar day, an action can be triggered.
- Fraud daily lock: a choice of *none, foreign, outgoing, incoming, global*. Specifies a lock level which will be used to lock the account and his subscribers when *fraud daily limit* is exceeded.
- Fraud daily notify: An email address or comma-separated list of email addresses that will receive notifications when *fraud daily limit* is exceeded.
- Currency: The currency symbol for your currency. Any UTF-8 character may be used and will be printed in web interfaces.
- VAT rate: The percentage of value added tax for all fees in the billing profile. Currently for informational purpose only and not used further.
- VAT included: Whether VAT is included in the fees entered in web forms or uploaded to the platform. Currently for informational purpose only and not used further.

## 7.1.2 Creating Billing Fees

Each Billing Profile holds multiple Billing Fees.

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

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

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

						Logged in as administrator Logout
sip:wise						× tings -
	Create	Billing Fees				uiigo
Billin		Zone			Search:	
			#	Zone	Zone Detail	
			2	test	test zone	2
← Back			Showing 1 to 1 of 1 entri	es	create "Crea	te Zone" button below ■ ← button below
Billing Zone						1 Create Zone
		Source				
	3	Destination	*			
# 🔶	4	Direction	outbound			
Showing 01	O	npeak init rate	0			
						Save

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

- **Zone**: A zone for a group of destinations. May be used to group destinations for simplified display, e.g. on invoices. (e.g. foreign zone 1)
- Source: The source pattern. This is a POSIX regular expression matching the complete source URI (e.g. `.\*@sip\. example\.org\$ or `someone@sip\.sipwise\.com\$ or just . to match everything). If you leave this field empty, the default pattern . matching everything will be set implicitly. Internally, this pattern will be matched against the <source\_cli>@ <source\_domain> fields of the CDR.
- Destination: The destination pattern. This is a POSIX regular expression matching the complete destination URI (e.g. some one@sip\.example\.org or ^43). This field must be set.
- Direction: Outbound for standard origination fees (applies to callers placing a call and getting billed for that) or Inbound for termination fees (applies to callees if you want to charge them for receiving various calls, e.g. for 800-numbers). *If in doubt, use Outbound*. If you upload fees via CSV files, use out or in, respectively.

## ✓ Important

The {source, destination, direction} combination needs to be unique for a billing profile. The system will return an error if such a set is specified twice, both for the file upload and the input via the web interface.

#### Important

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

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

## 7.1.3 Creating Off-Peak Times

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

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

		👗 Logged in as administrator 🛛 Logout
sip:wise		
sip:wise Edit Monday		× tings -
Off-p		
00:00:00	07:59:59	
18:00:00	23:59:59	3 Add
← Back	2	
Weekdays		
Weekday	Start - End	
Monday	00:00:00 - 07:59:59	
Tuesday		
Wednesday		
Thursday		
Friday		
Saturday		

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

síp:wise	
Create Date Definitions	× tings -
Off-p 1 Start Date/Time 2013-12-24 (	00:00:00
2 End Date/Time 2013-12-24 2	23:59:59
← Back	
Weekda	3 Save
Weekday	Start - End
Monday	00:00:00 - 07:59:59 18:00:00 - 23:59:59
Tuesday	
Wednesday	
Thursday	
Friday	

## 7.2 Prepaid Accounting

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

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

#### Important

For technical reasons, the system can make the distinction between on-peak and off-peak times only at call establishment time. In other words, if the currently active call fee at the moment when the call is established is an off-peak fee, then the same off-peak fee will remain active for the whole length of this call, even if the call actually transitions into an on-peak fee (and vice versa).



## Important

For technical reasons, prepaid billing can't charge local endpoint calls to Voicebox, VSC calls or calls to a Conference Room.

The Sipwise NGCP platform offers advanced billing features which are especially designed for pre-paid billing scenarios. For details please visit Billing Customizations Section 7.4 section of the handbook.

## 7.3 Fraud Detection and Locking

The NGCP supports a fraud detection feature, which is designed to detect accounts causing unusually high customer costs, and then to perform one of several actions upon those accounts. This feature can be enabled and configured through two sets of billing profile options described in Section 7.1.1, namely the monthly (*fraud monthly limit, fraud monthly lock* and *fraud monthly notify*) and daily limits (*fraud daily limit, fraud daily lock* and *fraud daily notify*). Either monthly/daily limits or both of them can be active at the same time.

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

- If **fraud lock** is set to anything other than *none*, it will lock the account accordingly (e.g. if **fraud lock** is set to *outgoing*, the account will be locked for all outgoing calls).
- If anything is listed in **fraud notify**, an email will be sent to the email addresses configured. The email will contain information about which account is affected, which subscribers within that account are affected, the current account balance and the configured fraud limit, and also whether or not the account was locked in accordance with the **fraud lock** setting. It should be noted that this email is meant for the administrators or accountants etc., and not for the customer.

## 7.3.1 Fraud Lock Levels

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

The following lock levels are available:

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



## Important

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



## Caution

Accounts that were automatically locked by the fraud detection feature will **not** be automatically unlocked when the next billing interval starts. This has to be done manually through the administration panel or through the provisioning interface.



#### Important

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

#### Note

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

#### Note

Apart from the daily fraud detection check service, NGCP also provides instant, "hard" locking for prepaid use cases, by means of billing profile packages. See Billing Profile Packages Section 7.4.3 for reference.

## 7.4 Billing Customizations

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

The sip:provider PRO supports — starting from its mr4.2.1 version — alternate ways of defining billing profiles and intervals which are especially worthy for pre-paid scenarios. New functionality is covered by the following titles:

- 1. Billing Networks Section 7.4.1
- 2. Profile Mappings Schedule Section 7.4.2
- 3. Profile Packages Section 7.4.3
- 4. Vouchers Section 7.4.4
- 5. Top-up Section 7.4.5
- 6. Balance Overviews Section 7.4.6

## 7. Usage Examples Section 7.4.7

Subsequent sections will provide an introduction and configuration instructions to these advanced features of sip:provider PRO.

## 7.4.1 Billing Networks

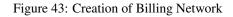
The idea is to dynamically select billing profiles (including fees) depending on the IP network the caller's SIP client is using to connect. The caller's IP is populated in a call's CDR, and effectively processed by:

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

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

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

Create Billing Netwo	ork					
Reseller Billing Network Name Description Billing Network Block				Search:		
	#	Name		Contract #	Status	
	16	Demo Reseller		200	active	×
	Showir	g 9 to 9 of 9 entries		<b>~</b>	1 2 3	→ ⇒
					Create	Reseller
Billing Network Name	Demo	Billing Net 1				
Description	Some	text				
Billing Network Block	10.0.	1.0	/ 24			
						Remove
						Save



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

Create Billing Netwo	ork *	
e Billing Network Name	Demo Billing Net 3	
Description	description net 3	l
b Billing Network Block	[10.0.1.0 / 28 Remove	Į
i; Billing Network Block	10.0.1.10 / 26 Remove	
Block '10.0.1.10/26' overlaps wi	th block(s) '10.0.1.0/28'	
	Add another billing network block	
	Save	

Figure 44: Overlapping Block Prevention

## 7.4.2 Profile Mapping Schedule

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

## 7.4.2.1 Definition of Profile Mapping Schedules

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

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

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

Mapping intervals can be of type:

- open: no start time + no end time
- half-open:
  - left-open: no start time + definite end time
  - right-open: definite start time + no end time
- · closed: definite start time + definite end time

## 7.4.2.2 Schedule Example

id	Billing Profile Interval Schedule Example	٨	Mai 2015			Jun 2015									
Ĩŭ	bining Projne interval schedule example	29	30	31	1	2	8	4	5	6	7	8	9	10	11
1	open: base/fallback (profile 1, no/any network)														
2	closed: (profile 2 , network 1) from June, 2nd. – 4th.														
3	right open: (profile 3 , network 1) starting on June, 1st.														
4	right open: (profile 4 , network 2) starting on June, 1st.														
5	closed: (profile 5, no/any network) from June, 3rd. – 10th.														

Figure 45: Profile Mapping Schedule Example

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

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

## Table 10: Active Billing Profiles

## 7.4.2.3 Configuration of Schedules

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

Edit Customer #202								×	
Set billing profiles	schedule (	billing mappi	ng intervals)					•	
Billing Profiles	actual	Date		g Profile		Bil	ing Network Name	e	
	×	NULL - NULL	- Demo	Billing Pr	ofile				
		Star	rt 2016-11	-01					
		En	d 2016-12	-31					
		Profil	e		Network				
	Search:			Search:					
	# F	teseller	Profile		#	Reseller	Network		
		)emo eseller	Demo Billi	×	31	Demo Reseller	Demo Billi	×	
	Showing 49 t	o 49 of 49 entr	ies		33	Demo Reseller	Demo Billi		
								Save	
	_	_	_	-	-	_	(=	-	

Figure 46: Profile Mapping Schedule Creation

### Тір

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

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

## Customer Details for #202 (Cloud PBX Account)

Reseller	■Preferences			✓ Expand Gro
Contact Deta				
actual	Date	Billing Profile Name	Prepaid	Billing Network Name
X	NULL-NULL	Demo Billing Profile		
	2016-11-01T00:00:00 - 2016-12-31T00:00:00	Demo Billing Profile		Demo Billing Net 1
	2017-01-01T00:00:00 - 2017-12-31T00:00:00	Demo Billing Profile		

### Figure 47: Profile Mapping Schedule List

## Note

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

The currently active mapping is depicted by a checked box.

### 7.4.2.4 REST API for Profile Mapping Schedules

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

- billing\_profiles field: explicitly declare profile mappings in form of (billing profile, billing netwo rk, start time, stop time) tuples

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

## Mapping Intervals

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

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

Table 11: Allowed Mapping Intervals

## Example Profile Mapping

An example JSON structure for definition of profile mapping schedules shown in Billing Profile Schedule List Figure 47 :

## 7.4.3 Profile Packages

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

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

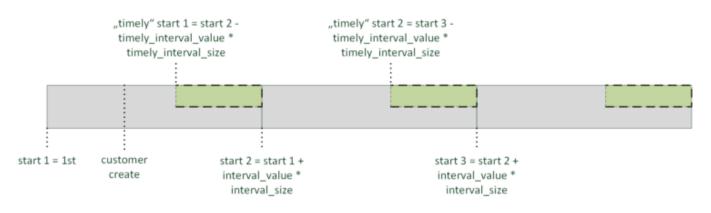
*Profile Packages* are fundamental for pre-paid billing scenarios, since in such a billing scheme the traditional, fixed monthly periods prove to be insufficient to cover the business needs of the NGCP platform operator. As an example: pre-paid subscribers typically have their "billing periods" between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

## 7.4.3.1 Elements of Profile Packages

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

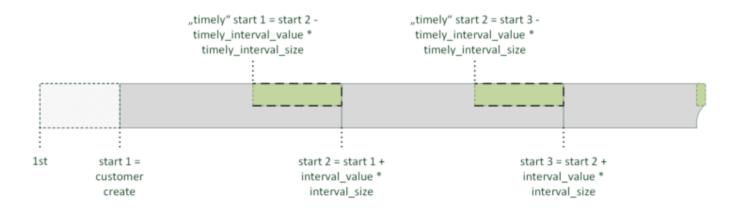
## Basic Balance Intervals Setup

- Interval duration (n hours, days, weeks, months)
- Interval start mode:
  - 1st of month (1st): billing interval is 1 calendar month; this is the default for each Customer created on Sipwise NGCP platform



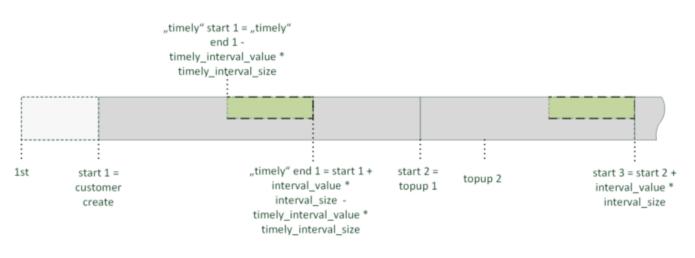
## Figure 48: Interval Start Mode: 1st

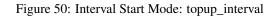
- upon customer creation (create): (the initial) billing interval starts when the Customer is created



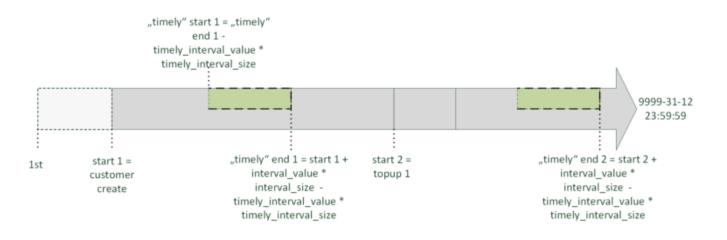


- upon topup (topup\_interval): interval starts at *first topup* event and its length is defined by interval duration parameter of the profile package





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





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

## **Balance Carry Over**

- Carry Over: balance carry over behaviour upon interval transitions:
  - carry-over: always keep balance
  - carry-over only if topped-up timely: keep balance in case of a *timely* top-up only; where **timely** means the topup happens within a pre-defined time span before the end of the balance interval
  - discard: discard balance at the end of each interval
- Timely Duration: duration of the timely period
- · Discard balance after intervals: for how many balance intervals the remaining account balance is kept before its disposal

## **Underrun Settings**

- Underrun lock threshold: when account balance reaches this amount the subscriber will be locked to a restricted set of services
- · Underrun lock level: this level of services will apply when an account balance underruns
  - don't change: no change in the available set of services
  - no lock: all services are available
  - foreign: only calls within subscriber's own domain are allowed
  - outgoing: all outgoing calls are prohibited
  - all calls: all calls (incoming + outgoing) are prohibited
  - global: all calls + access to Customer Self Care web interface are prohibited
  - ported: only automatic call forwarding, due to number porting, is allowed
- Underrun profile threshold: when account balance reaches this amount the Underrun Billing Profile will be applied

## **Basic Top-up Settings**

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

## Profile mappings

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

- Initial Billing Profile: when creating or manually changing the customers package (initial\_profiles)
- Underrun Billing Profile: when the balance underuns a cash threshold (underrun\_profiles)
- Top-up Billing Profile: when the customer tops-up using a voucher associated with the package (topup\_profiles)

### 7.4.3.2 Examples

## Profile Package Configuration

1. Definition of basic profile package parameters

Cı	reate Profile Packa	age							×		
e	Name	Demo	ProfPack1								
	Description	Profile	e package for	r demo 1							
n	Initial Balance	500	500								
ni 1- 1- 1- 1- 	Initial Billing Profile/Network	Profile Search: demo					Network Search:				
n: 1:		#	Reseller	Profile		#	Reseller	Network			
		106	Demo	Demo Billi		1	default	test billi			
ur fil			Reseller			3	default	test billi			
30		107	Demo Reseller	Demo Pre-p	×	7	default	test ipv6			
1		109	Demo Reseller	Demo Pre-p		9	default	test ipv6			
		111 Demo Demo Pre-p				Showing 1 to 4 of 14 entries					
6			· ••						Save		



2. Definition of balance interval and carry-over behaviour

Create Profile Packa	ıge			×
Balance Interval	1	month(s)	•	
Balance Interval Start	1st day of month			•
Carry Over	carry over			•
" "Timely" Duration		minute(s)	•	Π
Discard balance after I. intervals	3			
Underrun lock threshold	100			
Underrun lock level	outgoing			·
Underrun profile threshold	300			
5				Save

Figure 53: Balance Interval and Carry-over

3. Definition of balance underrun parameters

threshold	100									
Jnderrun lock level	outgo	bing								
Underrun profile threshold	300									
Underrun Billing	Profile					N	etwork			
Profile/Network	Search: demo					Search:				
	#	Reseller	Profile		#	Reseller	Network			
	113	Demo	Demo Pre-p		1	default	test billi			
		Reseller			3	default	test billi			
	115	Demo Reseller	Demo Pre-p	×	7	default	test ipv6			
	117	Demo	Demo Pre-p		9	default	test ipv6			
		Reseller		_						

Figure 54: Balance Underrun Parameters

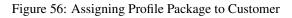
4. Definition of top-up settings

Top-up lock level	no lo	ck (unlock)								
Service Charge	50									
Top-up Billing			Profile			Ne	twork			
Profile/Network	Search: demo				Search:					
	#	Reseller	Profile		#	Reseller	Network			
	106	Demo	Demo Billi		1	default	test billi			
		Reseller			3	default	test billi			
	107	Demo Reseller	Demo Pre-p		7	default	test ipv6			
	109	Demo	Demo Pre-p		9	default	test ipv6			
		Reseller								
	111	Demo Reseller	Demo Pre-p	×	Show	ing 1 to 4 of 14 e ← 1	2 3 4	<b>→</b>		
								Demo		
	Showir	nor 1 to 4 of 7 er	ntries (filtered from s	5 total				Rem		

Figure 55: Balance Top-up Settings

5. Assigning a profile package to a customer

Edit Customer #197						×			
Set billing profiles	package (	initial profiles of a profile pa	ackage)		•				
Package			Search:	demo					
	#	Reseller	Package						
	67	Demo Reseller	DemoProfPack1		×				
r	69	Demo Reseller	DemoProfpack2			U			
	Showing 1 to	nowing 1 to 2 of 2 entries (filtered from 32 total entries)							
Product		ſ	Search:						
	#	Name							
	4	Basic SIP Account		×					
	5	Cloud PBX Account							
					Save				
cust_contact0@custcontact.inv	alid Basic S	IP SILVER_NETWORK_Y	1473815306 active						



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

Profile package setup:

- initial\_balance: 1.0 euro
- balance\_interval: 30 "day(s)"
- interval\_start\_mode: "topup\_interval"
- carry\_over\_mode: "timely"
- timely\_duration: 12 "day(s)"
- underrun\_lock\_threshold: 0.7 euro
- underrun\_profile\_threshold: 5.0 euro
- underrun\_lock\_level:...

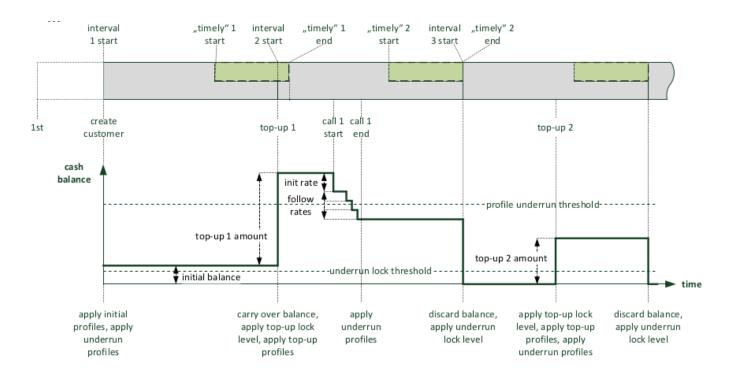


Figure 57: Example: Top-up Interval and Timely Carry-over

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

- initial\_balance: 1.0 euro
- balance\_interval: 30 "day(s)"
- interval\_start\_mode: "topup"
- carry\_over\_mode: "carry-over"
- notopup\_discard\_intervals: 1
- underrun\_lock\_threshold: 0.7 euro
- underrun\_profile\_threshold: 5.0 euro
- underrun\_lock\_level:...

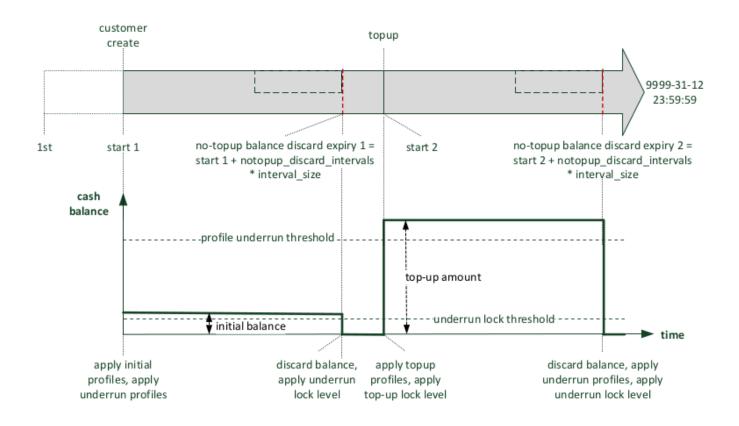


Figure 58: Example: Top-up and Always Carry-over

# 7.4.3.3 REST API

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

A sample JSON structure follows:

```
{
 "reseller_id" : 1,
 "status" : "active",
 "name" : "demo profile package",
 "description" : "package for 10€ ....",
 "balance_interval_start_mode" : "1st",
 "balance_interval_value" : 1,
 "balance_interval_unit" : "month",
 "carry_over_mode" : "carry_over",
 "timely_duration_unit" : null,
 "timely_duration_value" : null,
 "initial_balance" : 0,
 "initial_profiles" : [...], // required default, e.g. same as "topup_profiles"
 "notopup_discard_intervals" : null,
 "underrun_lock_threshold" : 0,
 "underrun_lock_level" : 4,
```

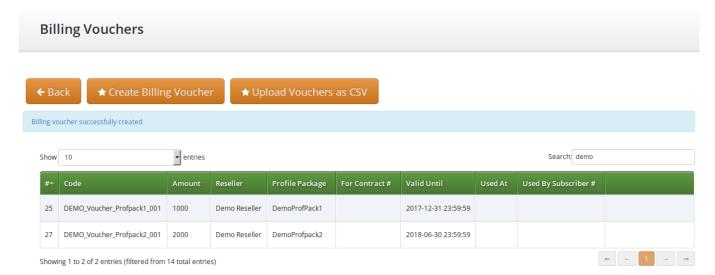
## 7.4.4 Vouchers

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

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

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

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

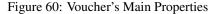




## 7.4.4.1 Properties of Vouchers

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

	Create Billing Vouch	iers							×
te	Reseller				Search: de	mo			) n
		#	Name		Contract #	5	Status		
		16	Demo Reselle	er	200	ä	active	×	
l		Showing	g 1 to 1 of 1 en	tries (filtered from 9 total entries)		⇒	- 1	→	
5							Create F	Reseller	JU
5	Code	DEMO	_Voucher_Pr	rofpack1_001					
53	Amount	1000							
l	Valid until	17-12-	-31						]
I	Customer				Search:		1		
t11		#	Reseller	Contact Email	E	xternal #	Status		
								Save	

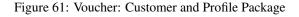


Setting following properties of a voucher is optional:

- · Customer: the Customer whom the voucher will be assigned to; subscribers of other customers can not redeem the voucher
- Package: vouchers may be associated with profile packages; if done so, some changes will be applied to the *Customer* for whom the voucher is redeemed with the top-up event:
  - applying top-up profile mappings starting with the time of the top-up
  - subtracting the new package's service charge from the voucher amount

- resizing the current balance interval for a gapless transition, if the new package has a different interval start mode (e.g. from "create" to "1st")
- if a new balance interval starts with the top-up, the carry-over mode of the customer's previous package applies

Customer				Search:		
	#	Reseller	Cor	ntact Email	External #	Status
	7	default	cus	tomer.test@spce.test		active
	13	default	cus	t_contact0@custcontact.invalid		active
	15	default	cus	t_contact0@custcontact.invalid		active
	17	default	cus	t_contact0@custcontact.invalid		active
Package	SHOWI	ng 1 to 4 of 71 e	enures	Search:	2 3 4 demo	5 Create Co
	#	Reseller		Package		
	69	Demo Reselle	er	DemoProfpack2		
	67	Demo Reselle	er	DemoProfPack1		



### 7.4.4.2 REST API

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

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

## 7.4.5 Top-up

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

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

The NGCP platform provides 2 interfaces to perform top-ups:

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

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

## 7.4.5.1 Top-up Cash

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

Edit Settings				×
Amount	12			
Package			Search:	
	#	Reseller	Package	
	67	Demo Reseller	DemoProfPack1	×
	69	Demo Reseller	DemoProfpack2	
	Showing 1 to	o 2 of 2 entries		← 1 → ⇒
				Create Profile Package
				Perform top-up

Figure 62: Balance Top-up with Cash

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

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

### 7.4.5.2 Top-up Voucher

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

Voucher				Search:			
	#	Code	Amount	Reseller	Package	Used At	
	25	DEMO_Voucher_Profpack1_001	1000	Demo Reseller	DemoProfPack1		
	27	DEMO_Voucher_Profpack2_001	2000	Demo Reseller	DemoProfpack2		
	Show	ing 1 to 2 of 2 entries			← ←	1 →	=



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

### 7.4.6 Balance Overviews

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

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

## 7.4.6.1 Contract Balance

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

Sound Sets									
Contract Balance	Contract Balance								
C Top-up Voucher C Top-up	Cash 🛛 🖾 S	et Cash Balance							
Cash balance	Cash balance			Debit		0.00			
Free time balance		0		Free time spent		0			
Interval from	2016-10-01T00:0	0:00		Interval to	2016-10-31T23	3:59:59			
"Timely" top-ups from				"Timely" top-ups to					
Balance will be discarded, if no tup-up happens un	1			2017-02-01T00:00:00					
Actual profile package		DemoProfPack1	Actual b	illing profile	1	Demo Pre-paid Topup 1			
Balance threshold when underrun profiles get app	ied	1.00	Balance	threshold when subscribers will be locked		1.00			
Balance Intervals									
Top-up Log									

## Figure 64: Contract Balance Status

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

## 7.4.6.2 Balance Intervals

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

Sound Sets	Sound Sets										
Contract Balance											
Balance Intervals											
Show 5	Show 5 entries Search:										
From 🔺	То	Cash	Debit	#Top-ups	#Timely Top-ups	Underrun detected (Profiles)	Underrun detected (Lock)				
2016-09-01 00:00:00	2016-09-30 23:59:59	0.00	0.00	0	0						
2016-10-01 00:00:00	2016-10-31 23:59:59	11.50	0.00	1	0	2016-10-07 15:05:26	2016-10-07 15:05:26				
Showing 1 to 2 of 2 entries											
Top-up Log											
Fraud Limits											

### Figure 65: List of Balance Intervals

#### Content of the balance intervals table is:

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

#### Note

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

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

### 7.4.6.3 Top-up Log

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

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

Contract Balance	iontract Balance										
Balance Intervals											
Top-up Log											
Show 5		- entrie	s		From D	ate:		To Date:		Search:	
Timestamp 🔺	Subscriber	Туре	Outcome	Message	Voucher ID	Amount	Balance	before	Balance after	Package before	Package after
2016-10-07 15:11:29		cash	ok			11.50	0.00		11.50	DemoProfPack1	DemoProfPack1
Showing 1 to 1 of 1 entrie	Showing 1 to 1 of 1 entries										
Fraud Limits	Fraud Limits										
Invoices											

## Figure 66: Balance Top-up Log

### Content of the top-up log table is:

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

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

An example of the response:

```
{
  "_embedded" : {
    "ngcp:topuplogs" : [{
      "_links" : {...},
      "amount" : null,
      "cash_balance_after" : null,
      "cash_balance_before" : null,
```

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

## 7.4.7 Usage Examples

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

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

Initial:

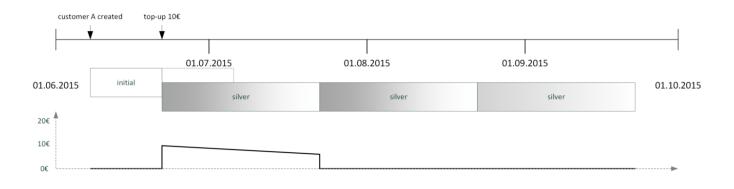
}

- Balance interval: 1 month
- Timely duration: 1 month
- Interval start mode: topup\_interval
- Carry-over mode: carry\_over\_timely
- Silver:
  - Balance interval: 1 month
  - Timely duration: 1 month
  - Interval start mode: "topup\_interval"
  - Carry-over mode: "carry\_over\_timely"
  - Service charge: 2 EUR

- Underrun lock level: "no lock"
- Voucher value: 10 EUR
- Gold:
  - Balance interval: 1 month
  - Interval start mode: "topup\_interval"
  - Carry-over mode: "carry\_over"
  - Service charge: 5 EUR
  - Underrun lock level: "no lock"
  - Voucher value: 20 EUR
- Extension:
  - Balance interval: 1 month
  - Timely duration: 1 month
  - Interval start mode: "topup\_interval"
  - Carry-over mode: "carry\_over\_timely"
  - Service charge: 2 EUR
  - Underrun lock level: "no lock"
  - Voucher value: 2 EUR

## 7.4.7.1 Customer A — Silver Package

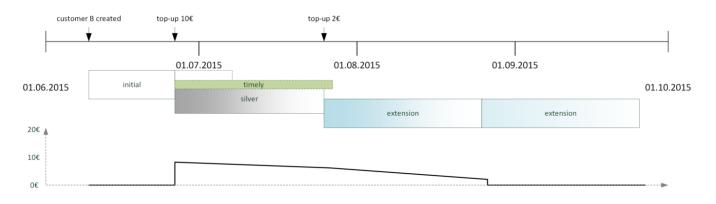
- 1. Customer A tops up 10 EUR with a "silver" voucher. 2 EUR are deducted as service charge. Remaining balance is 8 EUR starting on the date of the top- up.
- 2. Customer A doesn't top-up balance within the next month, so remaining balance is set to 0 after one month, and billing profiles and lock levels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

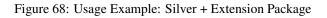


## Figure 67: Usage Example: Silver Package

## 7.4.7.2 Customer B—Silver and Extension Package

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

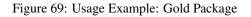




## 7.4.7.3 Customer C—Gold Package

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





# 7.5 Billing Data Export

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

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

## 7.5.1 Glossary of Terms

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

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

## 7.5.2 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><separator><sequence number>< <> suffix>
```

The definition of the specific fields is as follows:

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

#### Table 12: CDR/EDR export file name format

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\_000000042.cdr

### 7.5.3 File Format

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

## 7.5.3.1 File Header Format

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

<version>, <number of records>

The definition of the specific fields is as follows:

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

#### Table 13: CDR/EDR export file header line format

#### A valid example for a Header is:

007,0738

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

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

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

DESTINATION\_ACCOUNT\_ID

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

1-11

uint

# Table 14: Default set of system CDR fields

Internal ID of called party customer.

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

Body Element	Length	Туре	Description
CALL_ID	0-255	string	The SIP Call-ID.
RATING_STATUS	2-7	string	The internal rating status of the CDR -
			one of:
			unrated: not rated
			ok: successfully rated
			failed: error while rating
			Currently always ok or unrated,
			depending on whether rating is enabled
			or not.
RATED_AT	0-19	datetime	Time of rating, including date and time
			(with seconds precision). Empty if CDR
			is not rated.
SOURCE_CARRIER_COST	7-14	fixed	The originating carrier cost that the
		precision (6	carrier (i.e. SIP peer) charges for the
		decimals)	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	The originating customer cost, or empty
		precision (6	if CDR is not rated.
		decimals)	
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 DESTINATION_CARRIER_COST DESTINATION_CUSTOMER_COST	7-14	uint fixed precision (6 decimals)	The number of free time seconds used from the originating customer's account balance, or empty if CDR is not rated. The terminating carrier cost, or empty if CDR is not rated. <i>PLEASE NOTE: Only available in</i>
		precision (6	balance, or empty if CDR is not rated. The terminating carrier cost, or empty if CDR is not rated.
		precision (6	The terminating carrier cost, or empty if CDR is not rated.
		precision (6	CDR is not rated.
DESTINATION_CUSTOMER_COST	7-14		
DESTINATION_CUSTOMER_COST	7-14	decimals)	PLEASE NOTE: Only available in
DESTINATION_CUSTOMER_COST	7-14		I LEADE NOTE. UTILY available III
DESTINATION_CUSTOMER_COST	7-14		system exports, not for resellers.
		fixed	The terminating customer cost, or
		precision (6	empty if CDR is not rated.
		decimals)	
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	The originating reseller cost, or empty if
		precision (6	CDR is not rated.
		decimals)	PLEASE NOTE: Only available in
		· · · · · · · · · · · · · · · · · · ·	system exports, not for resellers.

Body Element	Length	Туре	Description
SOURCE_RESELLER_ZONE	0-127	string	Name of the originating reseller billing
			zone, or empty if CDR is not rated.
			PLEASE NOTE: Only available in
			system exports, not for resellers.
SOURCE_RESELLER_DETAIL	0-127	string	Description of the originating reseller
			billing zone, or empty if CDR is not
			rated.
			PLEASE NOTE: Only available in
			system exports, not for resellers.
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	The terminating reseller cost, or empty
		precision (6	if CDR is not rated.
		decimals)	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></line_terminator>	1	string	Always \n (special char LF - ASCII
			0x0A).

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

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

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

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

### Note

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

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

## 7.5.3.3 Extra fields that can be exported to CDRs

#### Supplementary Data

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

### Note

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

#### Table 15: Supplementary data in CDR fields

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

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

## Account balance details (prepaid calls)

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

#### Tip

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

#### Note

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

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

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

### Examples:

source\_customer\_cash\_balance\_before

· destination\_customer\_profile\_package\_id



## Important

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



# Important

There are some limitations in rating pre-paid calls, please visit Pre-paid Billing Section 7.2 section for details.

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

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

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

Body Element	Length	Туре	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.
CONTRACT_EXTERNAL_ID	0-255	string	The external ID of the customer. (A string value shown
			as "External ID" property of an NGCP customer.)
COMPANY	0-127	string	The company name of the customer's contact.

#### Table 16: Default set of system EDR fields

# Table 16: (continued)

Body Element	Length	Туре	Description
SUBSCRIBER_EXTERNAL_ID	0-255	string	The external ID of the subscriber. (A string value shown
			as "External ID" property of an NGCP subscriber.)
			PLEASE NOTE: This field is empty in case of
			start_huntgroup <b>and</b> end_huntgroup <b>events</b> .
PILOT_PRIMARY_NUMBER	0-64	string	The pilot subscriber's primary number (HPBX
			subscribers). PLEASE NOTE: This is not included in
			default set of EDR fields from NGCP version mr5.0
			upwards.
PRIMARY_NUMBER	0-64	string	The VoIP number of the subscriber with the highest ID
			(DID or primary number).
OLD_PROFILE_NAME	0-255	string	The old status of the event. Depending on the
			event_type:
			start_profile: Empty.
			end_profile: The name of the subscriber profile
			which got removed from the subscriber.
			update_profile: The name of the former
			subscriber profile which got updated.
			start_huntgroup: Empty.
			end_huntgroup: Empty.
			start_ivr: Empty.
			end_ivr: Empty.
NEW_PROFILE_NAME	0-255	string	The new status of the event. Depending on the
			event_type:
			start_profile: The name of the subscriber profile
			which got assigned to the subscriber.
			end_profile: Empty.
			update_profile: The name of the new subscriber
			profile which got applied.
			start_huntgroup: <b>Empty</b> .
			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></line_terminator>	1	string	A fixed character. Always \n (special char LF - ASCII
		_	0x0A).

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

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

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

#### Note

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

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

# 7.5.3.5 Extra fields that can be exported to EDRs

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

Body Element	Length	Туре	Description
SUBSCRIBER_PROFILE_SET	0-255	string	The subscriber's profile set name.
_NAME			
PILOT_SUBSCRIBER_PROFI	0-255	string	The profile set name of the subscriber's pilot subscriber.
LE_SET_NAME			
PILOT_SUBSCRIBER_PROFI	0-255	string	The profile name of the subscriber's pilot subscriber.
LE_NAME			
FIRST_NON_PRIMARY_ALIA	0-255	string	The subscriber's non-primary alias with lowest ID, before
S_USERNAME_BEFORE			number updates during the operation.
FIRST_NON_PRIMARY_ALIA	0-255	string	The subscriber's non-primary alias with lowest ID, after
S_USERNAME_AFTER			number updates during the operation.
PILOT_FIRST_NON_PRIMAR	0-255	string	The non-primary alias with lowest ID of the subscriber's
Y_ALIAS_USERNAME_BEF			pilot subscriber, before number updates during the
ORE			operation.
PILOT_FIRST_NON_PRIMAR	0-255	string	The non-primary alias with lowest ID of the subscriber's
Y_ALIAS_USERNAME_AFTER			pilot subscriber, after number updates during the
			operation.
NON_PRIMARY_ALIAS_USER	0-255	string	The non-primary alias of a subscriber affected by an
NAME			update_profile,start_profile or
			end_profile event to track number changes.

#### Table 17: Supplementary data in EDR fields

Body Element	Length	Туре	Description
PRIMARY_ALIAS_USERNAME	0-255	string	The subscriber's primary alias, before number updates
_BEFORE			during the operation.
PRIMARY_ALIAS_USERNAME	0-255	string	The subscriber's primary alias, after number updates
_AFTER			during the operation.
PILOT_PRIMARY_ALIAS_US	0-255	string	The primary alias of the subscriber's pilot subscriber,
ERNAME_BEFORE			before number updates during the operation.
PILOT_PRIMARY_ALIAS_US	0-255	string	The primary alias of the subscriber's pilot subscriber,
ERNAME_AFTER			after number updates during the operation.
FIRST_NON_PRIMARY_ALIA	0-255	string	Equals FIRST_NON_PRIMARY_ALIAS_USERNAME
S_USERNAME_BEFORE_AF			_BEFORE, if the value is not NULL, otherwise it's the
TER			same as FIRST_NON_PRIMARY_ALIAS_USERNAM
			E_AFTER.
PILOT_FIRST_NON_PRIMAR	0-255	string	Equals PILOT_FIRST_NON_PRIMARY_ALIAS_US
Y_ALIAS_USERNAME_BEFOR			ERNAME_BEFORE, if the value is not NULL, otherwise
E_AFTER			it's the same as PILOT_FIRST_NON_PRIMARY_ALI
			AS_USERNAME_AFTER.

#### Table 17: (continued)

## 7.5.3.6 File Trailer Format

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

<md5 sum>

#!/bin/sh

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

To validate the billing file, one must remove the Trailer before computing the MD5 sum of the file. An example bash script to validate the integrity of the file is given below:

```
error() { echo $0; exit 1; }
test -n "$1" || error "Usage: $0 <cdr-file>"
test -f "$1" || error "File '$1' not found"
TMPFILE="/tmp/$(basename "$1").$$"
MD5="$(sed -rn '$ s/^([a-z0-9]{32}).*$/\1/i p' "$1") $TMPFILE"
sed '$d' "$1" > "$TMPFILE"
echo "$MD5" | md5sum -c -
rm -f "$TMPFILE"
```

Given the script is located in cdr-md5.sh and the CDR-file is sipwise\_001\_20071110123000\_00000004.cdr, the output of the integrity check for an intact CDR file would be:

\$ ./cdr-md5.sh sipwise\_001\_20071110123000\_000000004.cdr /tmp/sipwise\_001\_20071110123000\_000000004.cdr: OK

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

\$ ./cdr-md5.sh sipwise\_001\_20071110123000\_000000004.cdr /tmp/sipwise\_001\_20071110123000\_000000004.cdr: FAILED md5sum: WARNING: 1 of 1 computed checksum did NOT match

#### 7.5.4 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 easy detection of lost billing files on the 3rd party side.

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

If public key authentication is chosen, the public key file has to be stored in the file ~/.ssh/authorized\_keys2 below the home directory of the cdrexport user. Otherwise, a password has to be set for the user.

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

#### Note

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

# 8 Provisioning REST API Interface

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

The sip:provider PRO provides a REST API to provision various functionality of the platform. The entry point - and at the same time the official documentation - is at *https://<your-ip>:1443/api*. It allows both administrators and resellers (in a limited scope) to manage the system.

You can either authenticate via username and password of your administrative account you're using to access the admin panel, or via SSL client certificates. Find out more about client certificate authentication in the online API documentation.

# 8.1 API Workflows for Customer and Subscriber Management

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

The standard life cycle of a customer and subscriber is:

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

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

```
#!/usr/bin/perl -w
use strict;
use v5.10;
use LWP::UserAgent;
use JSON qw();
my $uri = 'https://ngcp.example.com:1443';
my $ua = LWP::UserAgent->new;
my $user = 'myusername';
my $pass = 'mypassword';
$ua->credentials('ngcp.example.com:1443', 'api_admin_http', $user, $pass);
my ($req, $res);
```

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

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

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

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

#### Important

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

# Warning

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

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

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

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

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

```
$req = HTTP::Request->new('POST', "$uri/api/subscribers/");
$req->header('Content-Type' => 'application/json');
$req->content(JSON::to_json({
    status => 'active',
    customer_id => $customer_id,
    primary_number => { cc => 43, ac => 9876, sn => 10001 }, # the main number
    alias_numbers => [ \# as many alias numbers the subscriber can be reached at (or skip \leftrightarrow
       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 \leftrightarrow
    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+)s/s1/; # extract the ID from the Location header
say "Created subscriber, id is $subscriber_id";
```



Important

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

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

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

```
$req = HTTP::Request->new('PUT', "$uri/api/callforwards/$subscriber_id");
$req->header('Content-Type' => 'application/json');
response
$req->content(JSON::to_json({
   cfna => { # set a call-forward if subscriber is not registered
      destinations => [
          { destination => "4366610001", timeout => 10 }, # ring this for 10s
          { destination => "4366710001", timeout => 300 }, # if no answer, ring that for ↔
             300s
       ],
       times => undef # no time-based call-forward, trigger cfna always
   }
}));
$res = $ua->request($req);
if($res->code != 204) { # if return=representation, it's 200
   die "Failed to set cfna for subscriber: ".$res->decoded_content."\n";
}
```

You can set cfu, cfna, cft and cft via this API call, also all at once. Destinations can be hunting lists as described above or just a single number. Also, a time set can be provided to trigger call forwards only during specific time periods.

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

```
$req = HTTP::Request->new('GET', "$uri/api/subscriberpreferences/$subscriber_id");
$res = $ua->request($req);
if($res->code != 200) {
    die "Failed to fetch subscriber preferences: ".$res->decoded_content."\n";
}
my $prefs = JSON::from_json($res->decoded_content);
delete $prefs->{_links}; # not needed in update
$prefs->{prepaid_library} = 'libinewrate'; # switch to inew billing
$prefs->{block_in_clir} = JSON::true; # reject incoming anonymous calls
$prefs->{block_in_list} = [ # reject calls from the following numbers:
    '4366412345', # this particular number
    '431*', # all vienna/austria numbers
```

```
];
$req = HTTP::Request->new('PUT', "$uri/api/subscriberpreferences/$subscriber_id");
$req->header('Content-Type' => 'application/json');
$req->header('Prefer' => "return=minimal"); # use return=representation to get full json ↔
response
$req->content(JSON::to_json($prefs));
$res = $ua->request($req);
if($res->code != 204) {
die "Failed to update subscriber preferences: ".$res->decoded_content."\n";
}
say "Updated subscriber preferences";
```

Modifying numbers assigned to a subscriber, changing the password, locking a subscriber, etc. can be done directly on the subscriber resource.

```
$req = HTTP::Request->new('GET', "$uri/api/subscribers/$subscriber_id");
$res = $ua->request($req);
if($res->code != 200) {
   die "Failed to fetch subscriber: ".$res->decoded_content."\n";
}
my $sub = JSON::from_json($res->decoded_content);
delete $sub->{_links}; # not needed in update
push @{ $sub->{alias_numbers} }, { cc => 1, ac => 5432, sn => $t }; # add this number
push @{ $sub->{alias_numbers} }, { cc => 1, ac => 5433, sn => $t }; # add another number
$req = HTTP::Request->new('PUT', "$uri/api/subscribers/$subscriber_id");
$req->header('Content-Type' => 'application/json');
response
$req->content(JSON::to_json($sub));
$res = $ua->request($req);
if($res->code != 204) {
   die "Failed to update subscriber: ".$res->decoded_content."\n";
}
say "Updated subscriber";
```

At the end of a subscriber life cycle, it can be terminated. Once terminated, you can NOT recover the subscriber anymore.

```
$req = HTTP::Request->new('DELETE', "$uri/api/subscribers/$subscriber_id");
$res = $ua->request($req);
if($res->code != 204) {
    die "Failed to terminate subscriber: ".$res->decoded_content."\n";
}
say "Terminated subscriber";
```

Note that certain information is still available in the internal database to perform billing/rating of calls done by this subscriber. Nevertheless, the data is removed from the operational tables of the database, so the subscriber is not able to connect to the system, login or make calls/chats. Resources modification can be done via the GET/PUT combination. Alternatively, you can add, modify or delete single properties of a resource without actually fetching the whole resource. See an example below where we terminate the status of a customer using the PATCH method.

```
$req = HTTP::Request->new('PATCH', "$uri/api/customers/$customer_id");
$req->header('Content-Type' => 'application/json-patch+json');
$req->header('Prefer' => "return=minimal"); # use return=representation to get full json 
response
$req->content(JSON::to_json([
        { op => 'replace', path => '/status', value => 'terminated' }
]));
$res = $ua->request($req); # this will also terminate all still active subscribers
if($res->code != 204) {
        die "Failed to terminate customer: ".$res->decoded_content."\n";
}
say "Terminated customer";
```

# 9 Configuration Framework

The sip:provider PRO provides a configuration framework for consistent and easy to use low level settings management. A basic usage of the configuration framework only needs two actions already used in previous chapters:

- Edit /etc/ngcp-config/config.yml file.
- Execute ngcpcfg apply 'my commit message' command.

Low level management of the configuration framework might be required by advanced users though. This chapter explains the architecture and usage of the NGCP configuration framework. If the basic usage explained above fits your needs, feel free to skip this chapter and return to it when your requirements change.

A more detailed workflow of the configuration framework for creating a configuration file consists of 7 steps:

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

# 9.1 Configuration templates

The sip:provider PRO provides configuration file templates for most of the services it runs. These templates are stored in the directory /etc/ngcp-config/templates.

Example: Template files for /etc/ngcp-sems/sems.conf are stored in /etc/ngcp-config/templates/etc/ngcp-sems/.

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

# 9.1.1 .tt2 and .customtt.tt2 files

These files are the main template files that will be used to generate the final configuration file for the running service. They contain all the configuration options needed for a running sip:provider PRO system. The configuration framework will combine these files with the values provided by *config.yml*, *constants.yml* and *network.yml* to generate the appropriate configuration file.

Example: Let's say we are changing the IP used by kamailio load balancer on interface *eth0* to IP 1.2.3.4. This will change kamailio's listen IP address, when the configuration file is generated. A quick look to the template file under /*etc/ngcp-config/templates/etc/ka* will show a line like this: listen=udp:[% ip %]:[% kamailio.lb.port %]

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

listen=udp:1.2.3.4:5060

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

Instead of editing .tt2 files, the configuration framework recognises .customtt.tt2 files. These files are the same as .tt2, but they have higher priority when the configuration framework creates the final configuration files. An advanced user should create a .customtt.tt2 file from a copy of the corresponding .tt2 template and leave the .tt2 template untouched. This way, the user will have his personalized configuration and the system will continue providing a working, updated configuration template in .tt2 format.

Example: We'll create /etc/ngcp-config/templates/etc/lb/kamailio.cfg.customtt.tt2 and use it for our personalized configuration. In this example, we'll just append a comment at the end of the template.

```
cd /etc/ngcp-config/templates/etc/kamailio/lb
cp kamailio.cfg.tt2 kamailio.cfg.customtt.tt2
echo '# This is my last line comment' >> kamailio.cfg.customtt.tt2
ngcpcfg apply 'my commit message'
```

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

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

#### Tip

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

#### 9.1.2 .prebuild and .postbuild files

After creating the configuration files, the configuration framework can execute some commands before and after placing that file in its target directory. These commands usually are used for changing the file's owner, groups, or any other attributes. There are some rules these commands need to match:

- They have to be placed in a .prebuild or .postbuild file in the same path as the original .tt2 file.
- The file name must be the same as the configuration file, but having the mentioned suffixes.
- The commands must be *bash* compatible.
- · The commands must return 0 if successful.

• The target configuration file is matched by the environment variable output\_file.

Example: We need *www-data* as owner of the configuration file */etc/ngcp-ossbss/provisioning.conf*. The configuration framework will by default create the configuration files with root:root as owner:group and with the same permissions (rwx) as the original template. For this particular example, we will change the owner of the generated file using the *.postbuild* mechanism.

#### 9.1.3 .services files

*.services* files are pretty similar and might contain commands that will be executed after the *build* process. There are two types of *.services* files:

- The particular one, with the same name as the configuration file it is associated to. Example: /etc/ngcp-config/templates/etc/asterisk/sip.conf.services is associated to /etc/asterisk/sip.conf
- The general one, named *ngcpcfg.services* wich is associated to every file in its target directory. Example: /*etc/ngcp-config/templates/etc/asterisk/ngcpcfg.services* is associated to every file under /*etc/asterisk/*

When the *services* step is triggered all *.services* files associated to a changed configuration file will be executed. In case of the general file, any change to any of the configuration files in the directory will trigger the execution of the commands.

#### Tip

If the service script has the execute flags set (*chmod* +x \$file) it will be invoked directly. If it doesn't have execute flags set it will be invoked under bash. Make sure the script is bash compatible if you do not set execute permissions on the service file.

These commands are usually service reload/restarts to ensure the new configuration has been loaded by running services.

#### Note

The configuration files mentioned in the following example usually already exist on the platform. Please make sure you don't overwrite any existing files if following this example.

#### Example:

In this example we created two *.services* files. Now, each time we trigger a change to */etc/mysql.my.cnf* or to */etc/asterisk/\** we'll see that MySQL or Asterisk services will be restarted by the ngcpcfg system.

# 9.2 config.yml, constants.yml and network.yml files

The /etc/ngcp-config/config.yml file contains all the user-configurable options, using the YAML (YAML Ain't Markup Language) syntax.

The /*etc/ngcp-config/constants.yml* file provides configuration options for the platform that aren't supposed to be edited by the user. Do not manually edit this file unless you really know what you're doing.

The */etc/ngcp-config/network.yml* file provides configuration options for all interfaces and IP addresses on those interfaces. You can use the *ngcp-network* tool for conveniently change settings without having to manually edit this file.

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

# 9.3 ngcpcfg and its command line options

The ngcpcfg utility supports the following command line options:

#### 9.3.1 apply

The *apply* option is a short-cut for the options "check && build && services && commit" and also executes *etckeeper* to record any modified files inside /etc. It is the recommended option to use the ngcpcfg framework unless you want to execute any specific commands as documented below.

#### 9.3.2 build

The *build* option generates (and therefore also updates) configuration files based on their configuration (config.yml) and template files (.tt2). Before the configuration file is generated a present .prebuild will be executed, after generation of the configuration file the according .postbuild script (if present) will be executed. If a *file* or *directory* is specified as argument the build will generate only the specified configuration file/directory instead of running through all present templates.

Example: to generate only the file /etc/nginx/sites-available/ngcp-panel you can execute:

ngcpcfg build /etc/nginx/sites-available/ngcp-panel

Example: to generate all the files located inside the directory /etc/nginx/ you can execute:

ngcpcfg build /etc/nginx/

#### 9.3.3 commit

The *commit* option records any changes done to the configuration tree inside */etc/ngcp-config*. The commit option should be executed when you've modified anything inside the configuration tree.

## 9.3.4 decrypt

Decrypt /etc/ngcp-config-crypted.tgz.gpg and restore configuration files, doing the reverse operation of the *encrypt* option. Note: This feature is only available if the ngcp-ngcpcfg-locker package is installed.

#### 9.3.5 diff

Show uncommitted changes between ngcpcfg's Git repository and the working tree inside /etc/ngcp-config. Iff the tool doesn't report anything it means that there are no uncommitted changes. If the --addremove option is specified then new and removed files (iff present) that are not yet (un)registered to the repository will be reported, no further diff actions will be executed then. Note: This option is available since ngcp-ngcpcfg version 0.11.0.

#### 9.3.6 encrypt

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

### 9.3.7 help

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

#### 9.3.8 initialise

The *initialise* option sets up the ngcpcfg framework. This option is automatically executed by the installer for you, so you shouldn't have to use this option in normal operations mode.

#### 9.3.9 pull

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

#### 9.3.10 push

Push modifications to shared storage and remote systems. After changes have been pushed to the nodes the *build* option will be executed on each remote system to rebuild the configuration files (unless the --nobuild has been specified, then the build step will be skipped). If hostname(s) or IP address(es) is given as argument then the changes will be pushed to the shared storage and to the given hosts only. If no host has been specified then the hosts specified in */etc/ngcp-config/systems.cfg* are used. Note: This option is available in the High Availability setup only.

#### 9.3.11 services

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

## 9.3.12 status

The *status* option provides a human readable interface to check the state of the configuration tree. If you are unsure what should be done as next step or if want to check the current state of the configuration tree just invoke *ngcpcfg status*.

If everything is OK and nothing needs to be done the output should look like:

```
# ngcpcfg status
Checking state of ngcpcfg:
OK: has been initialised already (without shared storage)
Checking state of configuration files:
OK: nothing to commit.
Checking state of /etc files
OK: nothing to commit.
```

If the output doesn't say "OK" just follow the instructions provided by the output of ngcpcfg status.

Further details regarding the ngcpcfg tool are available through man ngcpcfg on the Sipwise Next Generation Platform.

# 10 Network Configuration

Starting with version 2.7, the sip:provider PRO uses a dedicated *network.yml* file to configure the IP addresses of the system. The reason for this is to be able to access all IPs of all nodes for all services from any particular node in case of a distributed system on one hand, and in order to be able the generate */etc/network/interfaces* automatically for all nodes based on this central configuration file.

# 10.1 General Structure

The basic structure of the file looks like this:

```
hosts:
  self:
    role:
      - proxy
      - lb
      - mgmt
    interfaces:
      - eth0
      - 10
    eth0:
      ip: 192.168.51.213
      netmask: 255.255.255.0
      type:
        - sip_ext
        - rtp_ext
        - web_ext
        - web_int
    lo:
      ip: 127.0.0.1
      netmask: 255.255.255.0
      type:
        - sip_int
        - ha_int
```

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

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

In sip:provider PRO systems all hosts of the system are defined, and the names are the actual host names instead of *self*, like this:

```
sp1:
peer: sp2
role: ...
```

hosts:

```
interfaces: ...
sp2:
   peer: sp1
   role: ...
   interfaces: ...
```

#### 10.1.1 Available Host Options

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

- role: The role setting is an array defining which logical roles a node will act as. Possible entries for this setting are:
  - mgmt: This entry means the host is acting as management node for the platform. In a sip:provider PRO 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.
  - Ib: This entry means the host is acting as SIP load-balancer for the platform. In a sip:provider PRO system this option must always be set. The SIP load-balancer acts as an ingress and egress point for all SIP traffic to and from the platform.
  - proxy: This entry means the host is acting as SIP proxy for the platform. In a sip:provider PRO system this option must always be set. The SIP proxy acts as registrar, proxy and application server and media relay, and is responsible for providing the features for all subscribers provisioned on it.
  - db: This entry means the host is acting as the database node for the platform. In a sip:provider PRO system this option must always be set. The database node exposes the MySQL and Redis databases.
  - *rtp*: This entry means the host is acting as the RTP relay node for the platform. In a sip:provider PRO system this option must always be set. The RTP relay node runs the *rtpengine* NGCP component.
- *interfaces*: The interfaces setting is an array defining all interface names in the system. The actual interface details are set in the actual interface settings below. It typically includes lo, eth0, eth1 physical and a number of virtual interfaces, like: bond0, vlanXXX
- <interface name>: After the interfaces are defined in the interfaces setting, each of those interfaces needs to be specified as a separate set of parameters.

Addtional main parameters of a node:

- · dbnode: the sequence number (unique ID) of the node in the database cluster; not used in sip:provider PRO system
- *peer*: the hostname of the peer node within the pair of nodes (e.g. "sp2" for *sp1* host). The purpose of that: each node knows its companion for providing high availability, data replication etc.
- *status*: one of *online*, *offline*, *inactive*. *inactive* means that the node is up but is not ready to work in the cluster (installing process). *offline* means that the node is not reachable. *online* is a normal working node.

## 10.1.2 Interface Parameters

- hwaddr: MAC address of the interface
- ip: IPv4 address of the node
- v6ip: IPv6 address of the node; optional
- netmask: IPv4 netmask
- shared\_ip: shared IPv4 address of the pair of nodes; this is a list of addresses
- shared\_v6ip: shared IPv6 address of the pair of nodes; optional; this is a list of addresses
- type: type of services that the node provides; these are usually the VLANs defined for a particular NGCP system.

#### Note

You can assign a type only once per node.

#### Available types are:

- api\_int: internal, API-based communication interface
- aux\_ext: interface for potentially insecure external components like remote system log collection service.

#### Note

For example the *CloudPBX* module can use it to provide time services and remote logging facilities to end customer devices. The type *aux\_ext* is assigned to *lo* interface by default. If it is needed to expose this type to the public, it is recommended to assign the type *aux\_ext* to a separate VLAN interface to be able to limit or even block the incoming traffic easily via firewalling in case of emergency, like a (D)DoS attack on external services.

- mon\_ext: remote monitoring interface (e.g. SNMP)
- rtp\_ext: main (external) interface for media traffic
- sip\_ext: main (external) interface for SIP signalling traffic between NGCP and other SIP endpoints
- sip\_ext\_incoming: additional, optional interface for incoming SIP signalling traffic
- sip\_int: internal SIP interface used by NGCP components (Ib, 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
- li\_int: used for LI (Lawful Interception) traffic routing
- ha\_int: HA (High Availability) communication interface between the nodes

#### Note

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

- vlan\_raw\_device: tells which physical interface is used by the particular VLAN
- post\_up: routes can be defined here (interface-based routing)
- · bond\_XY: specific to "bond0" interface only; these contain Ethernet bonding properties

# 10.2 Advanced Network Configuration

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

#### 10.2.1 Extra SIP Sockets

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

The correct format consists of a label and value like this:

```
extra_sockets:
    port_5064: udp:10.15.20.108:5064
    test: udp:10.15.20.108:6060
```

The label is shown in the outbound\_socket peer preference (if you want to route calls to the specific peer out via specific socket); the value must contain a transport specification as in example above (udp, tcp or tls). After adding execute ngcpcfg apply:

```
ngcpcfg apply 'added extra socket' && ngcpcfg push
```

The direction of communication through this SIP extra socket is incoming+outgoing. The sip:provider PRO will answer the incoming client registrations and other methods sent to the extra socket. For such incoming communication no configuration is needed. For the outgoing communication the new socket must be selected in the outbound\_socket peer preference. For more details read the next section Section 10.2.2 that covers peer configuration for SIP and RTP in greater detail.

# Important

In this section you have just added an extra SIP socket. RTP traffic will still use your rtp\_ext IP address.

## 10.2.2 Extra SIP and RTP Sockets

If you want to use an additional interface (with a different IP address) for SIP signalling and RTP traffic you need to add your new interface in the */etc/network/interfaces* file. Also the interface must be declared in */etc/ngcp-config/network.yml*.

Suppose we need to add a new SIP socket and a new RTP socket on VLAN 100. You can use the *ngcp-network* tool for adding interfaces without having to manually edit this file:

```
ngcp-network --set-interface=eth0.100 --host=sp1 --ip=auto --netmask=auto --type= ↔
sip_ext_incoming --type=rtp_int_100
ngcp-network --set-interface=eth0.100 --host=sp2 --ip=auto --netmask=auto --type= ↔
sip_ext_incoming --type=rtp_int_100
```

The generated file should look like the following:

```
sp1:
. .
. .
    eth0.100:
     hwaddr: ff:ff:ff:ff:ff
      ip: 192.168.1.2
      netmask: 255.255.255.0
      shared_ip:
        - 192.168.1.3
      shared_v6ip: ~
      type:
        - sip_ext_incoming
        - rtp_int_100
• •
. .
    interfaces:
     - lo
      - eth0
      - eth0.100
      - eth1
. .
. .
sp2:
. .
. .
    eth0.100:
      hwaddr: ff:ff:ff:ff:ff
      ip: 192.168.1.4
      netmask: 255.255.255.0
      shared_ip:
        - 192.168.1.3
      shared_v6ip: ~
      type:

    sip_ext_incoming
```

```
- rtp_int_100
...
...
interfaces:
    - lo
    - eth0
    - eth0.100
    - eth1
```

As you can see from the above example, extra SIP interfaces must have type *sip\_ext\_incoming*. While *sip\_ext* should be listed only once per host, there can be multiple *sip\_ext\_incoming* interfaces. The direction of communication through this SIP interface is incoming only. The sip:provider PRO will answer the incoming client registrations and other methods sent to this address and remember the interfaces used for clients' registrations to be able to send incoming calls to him from the same interface.

In order to use the interface for the outbound SIP communication it is necessary to add it to extra\_sockets section in */etc/ngcp-config/config.yml* and select in the outbound\_socket peer preference. So if using the above example we want to use the vlan100 IP as source interface towards a peer, the corresponding section may look like the following:

```
extra_sockets:
    port_5064: udp:10.15.20.108:5064
    test: udp:10.15.20.108:6060
    int_100: udp:192.168.1.3:5060
```

#### The changes have to be applied:

ngcpcfg apply 'added extra SIP and RTP socket' && ngcpcfg push

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.

# 11 Software Upgrade

# 11.1 Release Notes

The sip:provider PRO version mr5.4.2 has several important changes comparing to the previous release:

- [PRO/Carrier] Implement new Apple push protocol for iOS SIP:Phone application [TT#14952]
- Introduce maintenance mode for ngcp-upgrade [TT#16751]
- Switch from the YAML and YAML::Tiny Perl modules to the modern YAML::XS [TT#17650]
- Implement new network.yml dynamic validator program. [TT#14817]
- [Carrier] Introduce node status at network.yml to be able to add more nodes to the cluster without affecting a running system [TT#16316]
- The ngcp-snmp-agent daemon now exposes OIDs and emits traps for peer host status. [TT#16530, TT#17602]
- The ngcp-witnessd daemon now collects extended Kamailio statistics. [TT#2399]
- The ngcp-sercmd program has gained a new memdbg command. [TT#16548]
- The ngcp-check\_active has been renamed to ngcp-check-active, and has gained a new -v option. [TT#16850]
- New ngcp-type, ngcp-make-active and ngcp-make-standby scripts. [TT#16850]
- The ngcpcfg program gained new set and del commands. [TT#16903]
- added option kamailio.lb.remove\_isup\_body\_from\_replies [TT#17453]
- kamailio: control debug per module via config.yml [TT#17200]
- [PRO/Carrier] added huntgroup display-name support for PBX hunt indicator [TT#14840]

Please find the complete changelog in our release notes on our WEB site.

# 11.2 Preparing to a Software Upgrade

#### Warning

Make sure that all the SIP domains and peering servers have appropriate rtp\_interface option (e.g. *ext*) selected in the NAT and Media Flow Control section. If you leave *default* there, then incorrect network interface may be used for sending and receiving RTP media after the software upgrade.

Make sure you're prepared to spend two hours or so upgrading the system. There can be service interruptions, so also notify the customer and get their approval.

Check the system overal status:

ngcp-status --all

Check the system for locally modified files (move them to appropriate customtt.tt2 files if necessary):

```
ngcp-status --integrity
```

Try to find local changes to the template files by issuing:

find /etc/ngcp-config -name \\*customtt.tt2

You will also need to find the dpkg-dist files under the templates files because people sometimes forget about creating customtt files and edit tt2 files directly. That makes upgrades not to replace the tt2 files. If so, you need to treat the tt2 files as if they were customtt files and make sure you merge the new templates with the changes of the old ones.

find /etc/ngcp-config -name \\*.tt2.dpkg-dist

Also, please check/clean old dpkg backup files (just in case if previous person did the previous step not carefully enough). Normally the list should be empty:

find /etc/ngcp-config -name \\*.tt2.dpkg\\*

You will have to understand why the changes are there and if they are still needed after the upgrade. You should create a ticket in the bugtracker if there isn't one yet.

Log into the two servers. Use their real IPs so you can switch the cluster forth and back later on. Make sure the cluster status is ok - on **both** nodes issue:

- monit summary one should be running all services, the other all but rtpengine, lb, proxy, sbc, mediator and rate-o-mat
- ngcp-check-active -v one (with all services running) should print "active", the other "none"
- mysql -e "show slave status\G" look for the following:

Slave\_IO\_Running: Yes
Slave\_SQL\_Running: Yes
Seconds\_Behind\_Master: 0

- · ngcpcfg status should print OK all the times
- ngcp-collective-check should not report any problems.

A cluster fail-over could be a good idea to see if everything works on the second node too. On the standby node issue:

ngcp-make-active

Afterwards again check monit, ngcp-check-active and ngcp-collective-check.

Create two test subscribers, or retrieve the credentials for two of them. Register a client to the platform and perform a test call between the two to ensure call routing works.

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

# Warning

Installation may use locally specified mirrors. Discuss with a customer possibility to switch on Sipwise APT repositories (at least for the time of upgrades), the public Debian mirrors may not provide packages for old Releases anymore or be at least outdated!

# 11.3 Upgrading the sip:provider PRO

The sip:provider PRO system upgrade to mr5.4.2 will perform a couple of fundamental tasks:

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

So assuming you have a running sip:provider PRO system and want to upgrade it, start on the inactive node by upgrading software, then take over from the other node and then upgrade the other (now inactive) node, as detailed in the steps below.

#### 11.3.1 Preparing for maintenance mode

Sipwise NGCP introduces **Maintenance Mode** with its mr5.4.1 release. The maintenance mode of NGCP will disable some background services (for instance: *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.

#### Important

For sip:provider PRO systems running NGCP release earlier than mr5.4.1: As upgrading from an earlier release to mr5.4.1 or later will result in a system being aware and making use of the maintenance mode, it is necessary to prepare the NGCP for it *before* the software upgrade.

Enable maintenance mode:

• Pull ngcpcfg outstanding changes (if any):

ngcpcfg pull

- Edit /etc/ngcp-config/config.yml file:
  - For systems running NGCP release earlier than mr5.4.1: insert a new line: "maintenance:yes" in general section of the configuration file
  - For systems running NGCP release mr5.4.1 or later: set the general.maintenance parameter to yes

```
general:
  maintenance: yes
```

#### **Disabling background services**

*These tasks are for systems currently running NGCP release earlier than mr5.4.1.* The *maintenance mode* of NGCP is not available in your system yet, before the software upgrade. Therefore it is necessary to manually disable the following background services to avoid potential inconsistency in system state:

- · Heartbeat watchdog
- · Mediator
- Rate-o-mat
- · CDR Exporter

To do:

1. Stop HB watchdog on all nodes:

```
monit stop hb_watchdog
```

2. Stop mediator and rate-o-mat services on both nodes :

```
monit stop mediator
monit stop rate-o-mat
```

3. Prevent background services from (re)starting. You need to change configuration in /etc/ngcp-config/config. yml file: set the enable parameter to "no" as shown below:

```
heartbeat:
   hb_watchdog:
      enable: no
...
mediator:
   enabled: no
...
rateomat:
   enable: no
```

4. In order to **disable** *CDR* exporter process you need to add a new custom template file because there is no such file by default:

```
config="/etc/cron.d/ngcp-cdr-exporter"
customtt="/etc/ngcp-config/templates/etc/cron.d/ngcp-cdr-exporter.customtt.tt2"
[ -f "${customtt}" ] || cp "${config}" "${customtt}"
vim "${customtt}"
```

Comment out the single line in the file:

```
# 25,55 * * * * root . /etc/default/ngcp-roles; if ...
```

*Please don't forget* to reverse the manual changes after the upgrade, as described in Enabling background services section of the handbook!

#### For all NGCP systems, regardless of their version:

Apply configuration changes, execute commands:

```
ngcpcfg apply 'Enable maintenance mode before the upgrade to mr5.4.2' ngcpcfg push
```

#### 11.3.2 Switch to new repositories

For upgrading the sip:provider PRO to the latest mr5.4.2 release, execute the following commands on **both** nodes:

```
NGCP_CURRENT_VERSION=$(cat /etc/ngcp_version)
sed -i "s/$NGCP_CURRENT_VERSION/mr5.4.2/" /etc/apt/sources.list.d/sipwise.list
ngcp-approx-cache-helper --auto --node localhost
apt-get update
apt-get install ngcp-upgrade-pro
```

#### 11.3.3 Upgrade the first PRO node

Execute ngcp-upgrade in inactive node as root:

ngcp-upgrade

#### Note

sip:provider PRO can be upgraded to mr5.4.2 from previous release or previous build only. The script ngcp-upgrade will find all the possible destination releases for the upgrade and allow to choose the proper one.

#### Note

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

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

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

#### 11.3.4 The customtt files handling (if necessary)

Merge/add the custom configuration templates if needed. Apply the changes to configuration templates if any and send them to the shared storage and the other node:

```
ngcpcfg apply 'upgrade node'
ngcpcfg push --nobuild --noapply
```

#### 11.3.5 Promote inactive node to active.

Execute on inactive node as root:

ngcp-make-active

#### 11.3.6 Upgrade the second PRO node

Go to the new inactive node. Run ngcp-upgrade, as root:

ngcp-upgrade

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

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

# 11.4 Post-upgrade tasks

#### 11.4.1 Disabling maintenance mode

In order to disable maintenance mode do the following:

• Pull ngcpcfg outstanding changes (if any):

ngcpcfg pull

• Edit /etc/ngcp-config/config.yml file:

```
general:
   maintenance: no
```

#### **Enabling background services**

If you upgraded from NGCP release earlier than mr5.4.1 you have to manually reverse the changes you applied before the upgrade. (Disabling background services)

1. Enable background services to start. You need to change configuration in /etc/ngcp-config/config.yml file: set the enable parameter to "yes" as shown below:

```
heartbeat:
   hb_watchdog:
      enable: yes
...
mediator:
   enabled: yes
...
rateomat:
   enable: yes
```

2. In order to enable CDR exporter process you need to remove customtt file and edit cron config file:

```
rm /etc/ngcp-config/templates/etc/cron.d/ngcp-cdr-exporter.customtt.tt2
vim /etc/cron.d/ngcp-cdr-exporter
```

Remove comment at the beginning of the single line of file on ALL nodes:

25,55 \* \* \* \* root . /etc/default/ngcp-roles; if ...

For all NGCP systems, regardless of their previous version:

· Execute commands:

```
ngcpcfg apply 'Disable maintenance mode after the upgrade to mr5.4.2' ngcpcfg push
```

#### 11.4.2 Post-upgrade checks

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

#### Note

You can find a backup of some important configuration files of your existing installation under /var/backup/ngcp-mr5.4.2-\* (where \* is a place holder for a timestamp) in case you need to roll back something at any time. A log file of the upgrade procedure is available at /var/backup/ngcp-mr5.4.2-\*/upgrade.log.

# 12 Backup, Recovery and Database Maintenance

# 12.1 sip:provider PRO Backup

For any service provider it is important to maintain a reliable backup policy as it enables prompt services restoration after any force majeure event. Although the design of sip:provider PRO implies data duplication and high availability of services, we still strongly suggest you to configure a backup procedure. The sip:provider PRO has a built-in solution that can help you back up the most crucial data. Alternatively, it can be integrated with any Debian compatible backup software.

#### 12.1.1 What data to back up

The database

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

System configuration

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

· Exported CDRs (optional)

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

#### 12.1.2 The built-in backup solution

The sip:provider PRO comes with an easy-to-use solution that creates everyday backups of the most important data:

- The system configuration files. The whole /etc directory is backed up.
- Exported CDRs. The /home/jail/home/cdrexport directory with csv files.
- · All required databases on corresponding servers.

This functionality is disabled by default and can be enabled and configured in the backuptools subsection in the config.yml file. Please, refer to the "C.1.3 backup tools" section of the "NGCP configs overview" chapter for the backup configuration options.

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

ngcpcfg apply 'enabled the backup feature'
ngcpcfg push

Once you activate the feature, the sip:provider PRO will create backups in the off-peak time on the standby nodes and put them to the */var/backup/ngcp\_backup* directory. You can copy these files to your backup server using scp or ftp.

#### Note

make sure that you have enough free disk space to store the backups for the specified number of days.

## 12.2 Recovery

In the worst case scenario, when the system needs to be recovered from a total loss, you only need 4 steps to get the services back online:

- Install the sip:provider PRO as explained in chapter 2.
- Restore the /etc/ngcp-config/ directory and the /etc/mysql/sipwise.cnf file from the backup, overwriting your local files.
- · Restore the database from the latest MySQL dump.
- · Apply the changes to bring the original configuration into effect:

```
ngcpcfg apply 'restored the system from the backup' ngcpcfg push
```

# 12.3 Reset Database



#### Important

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

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

# 12.4 Synchronize database

In case of unresolvable database replication issues or to copy mysql data between a pair of hosts (usually a pair of sp1 and sp2 nodes).

There is a script for that: ngcp-sync-db.

To synchronize databases you need to run the script on your target host.

• Definitions:

- master remote/master host (the database is dumped from there)
- local target/local host (the database is imported onto)
- Usage:

# ✓ Important

Your existing database on *local* will be completely wiped. The script provides a possibility to backup both *master* and *local* databases during the procedure.

You can run the script with -h or --help to check its options or use man ngcp-sync\_db

If you run it without any options it automatically calculates *master* hostname (e.g. if you run it on *sp2* then *sp2==local* and sp1==master).

The script also requires mysql credentials and if none provided it uses username=*sipwise* and the password is picked from /etc/mysql/sipwise.cnf. You can specify user and/or password for both *master* and *local*.

Before the actual start it produces a summary with settings used to the procedure and a confirmation prompt to prevent accidental usage. Making use of --force option" however suppresses the confirmation prompt. By default no messages are printed on STDOUT (compliant to be integrated into another tools) and with -v or --verbose options you enable debugging where all the ongoing steps will be printed to STDOUT.

There are 2 modes available for synchronization, *online* and *backup*. By default *online* is used where the procedure does not create any backups and everything goes on the fly. That is useful for large databases where creating backups would require solid amounts of available free disk space. With the *backup* mode *master* db is dumped into a backup file on *local* first (default directory: */var/backup/ngcp-sync-db*) and imported upon the backup completion.

Mysql database connection to the *master* db and the *local* db is the essential part and by default the script tries to establish direct mysql connection however that may not be possible due to the access restrictions. To overcome that you can use --ssh-tunnel option and specifying there a local custom free port (e.g. --ssh-tunnel=33125) in this case an ssh tunnel will be created to *master* and used to establish the db connection on the *localhost* behalf (NOTE: Public key based ssh negotiation is required for the tunnel as the script does not suppot ssh credentials for security reasons).

Backups may be a subject to create during synchornization for possible rollbacks. To create the *local* db backup you should add – -local-backup. The *master* db backup is automatically created only using --sync-mode=backup. Upon completion all those created backups are deleted and if you need to keep them please use --keep-backups option (NOTE: In case of errors during synchronization and when backups are created they are NOT automatically deleted. Therefore, if the script had failed with an error and afterwards completed successully you may want to manually remove the remaining backups from */var/backup/ngcp-sync-db*).

· Examples:

#### Normal online mode synchronization $sp1 \rightarrow sp2$ .

sp2> ngcp-sync-db

#### Normal backup mode synchronization $sp1 \rightarrow sp2$ .

sp2> ngcp-sync-db --sync-mode=backup

Forced online mode synchronization *sp1*  $\rightarrow$  *sp2*. USE WITH CARE as there will be no confirmation prompts.

sp2> ngcp-sync-db --force

Direct mysql db access is not possible. SSH tunnel is initialised to local port 33125 and forwards all connections 127.0.0.1:33125  $\rightarrow$  sp1:3306.

```
sp2> ngcp-sync-db --ssh-tunnel=33125
```

Custom mysql credentials for the master db connection (by default: sipwise:/etc/mysql/sipwise.cnf)

sp2> ngcp-sync-db --master-user=frank --master-pass=dbconnect

Normal online mode synchronization  $sp1 \rightarrow sp2$  with the *local* db backup and retaining the backup. (no *master* backup in this case as it is only available with -sync-mode=backup).

sp2> ngcp-sync-db --local-backup --keep-backups

Normal online mode synchronization *custom-node*  $\rightarrow$  *sp2* with ssh tunnel

sp2> ngcp-sync-db --master-host=custom-node --ssh-tunnel=45001

Forced syncrhonization *custom-node*  $\rightarrow$  *sp2* with ssh tunnel, backup sync mode, local backup, custom *master* and *local* db credentials and ports as well as a different backup dir

#### 12.5 Accounting Data (CDR) Cleanup

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

#### 12.5.1 Cleanuptools Configuration

The configuration parameters of *cleanuptools* are located in the main NGCP configuration file: /etc/ngcp-config/con fig.yml. Please refer to the config.yml file description: Cleanuptools Configuration Data Section B.1.7 for configuration parameter details.

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

> ngcpcfg apply 'Modified cleanuptools config'
> ngcpcfg push

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

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

### 12.5.2 Accounting Database Cleanup

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

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

An extract from a sample configuration file is provided here:

```
############
batch = 10000
archive-target = /var/backup/cdr
compress = gzip
username = dbcleaner
password = rcKamRdHhx7saYRbkJfP
host = localhost
connect accounting
time-column = from_unixtime(start_time)
backup-months = 2
backup-retro = 2
backup cdr
connect accounting
archive-months = 2
archive cdr
connect kamailio
time-column = time
cleanup-days = 90
cleanup acc
# Clean up after mediator by deleting old leftover acc entries and deleting
# old entries out of acc_trash and acc_backup
connect kamailio
time-column = time
```

cleanup-days = 30
cleanup acc\_trash
cleanup acc\_backup

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

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

- backup CDRs
- archive CDRs
- cleanup CDRs

These actions are discussed in following sections.

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

## 12.5.2.1 Backup CDRs

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

Configuration values that govern the backup procedure are:

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

## Important

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

• backup-retro: How many months to process for backups, going backwards in time. Using the example above, with this value set to "3", the months October, September and August would be backed up, while any older records would be left untouched.

## 12.5.2.2 Archive CDRs

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

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

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

## Important

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

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

## 12.5.2.3 Cleanup CDRs

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

- cleanup acc
- cleanup acc\_trash
- cleanup acc\_backup

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

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

## 12.5.3 Exported CDR Cleanup

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

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

## A sample configuration file is provided here:

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

Configuration values that define the files to be deleted:

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

# 13 Platform Security, Performance and Troubleshooting

Once the sip:provider PRO is in production, security and maintenance becomes really important. In this chapter, we'll go through a set of best practices for any production system.

## 13.1 Sipwise SSH access to sip:provider PRO

The sip:provider PRO provides SSH access to the system for Sipwise operational team for debugging and final tuning. Operational team uses user *sipwise* which can be logged in through SSH key only (password access is disabled) from dedicated access server *jump.sipwise.com* only.

To completely remove Sipwise access to your system, please execute as user root:

```
root@myserver:~# ngcp-support-access --disable && apt-get install ngcp-support-noaccess
```

### Note

you have to execute the command above on each node of your sip:provider PRO system!



Warning

please ensure that the script complete successfully:

\* Support access successfully disabled.

If you need to restore Sipwise access to the system, please execute as user root:

root@myserver:~# apt-get install ngcp-support-access && ngcp-support-access --enable



Warning please ensure that the script complete successfully:

\* Support access successfully enabled.

## 13.2 Firewalling

## 13.2.1 Firewall framework

The sip:provider PRO runs a wide range of services. In order to secure the platfrom while allowing access to the sip:provider PRO the NGCP 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 the NGCP configuration framework provides a save setup for sip:provider PRO 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 konwledge of firewalling principles in general and the *netfilter* facility in particular.

Zone name	Description
ha_int	Internal cluster interface providing internal cluster communications between cluster
	pairs (heartbeat) and synchronization of data and configuration
mon_ext	Interface to conect external monitoring appliances (SNMP)
rtp_ext	Interface for external RTP media relay between sip:provider PRO and endpoints (e.g.
	user agents, peers)
sip_ext	Interface for external SIP signalling between sip:provider PRO and endpoints (e.g.
	user agents, peers)
sip_int	Interface for internal signalling, e.g. between load-balancers, proxies and applications
	servers
ssh_ext	Interface providing external access to the sip:provider PRO 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

### Table 18: NGCP network zones

### Note

Additional custom zones may be configured, but will not be automatically integrated into the firewall configuration.

To facilitate firewall functionality sip:provider PRO 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:  $table \rightarrow chain \rightarrow rule$ . Default firewall setups of sip:provider PRO 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 *security* $\rightarrow$ *firewall* $\rightarrow$ *nat\_rules4* and *security* $\rightarrow$ *firewall* $\rightarrow$ *nat\_rules6*.

Each chain deploys a default policy handling packets which did not trigger and rule in a prticular chain.

Chain	Default	Description
	policy	
INPUT	DROP	Handling all packets directly destined for a sip:provider PRO node (only
		packets matching a rule are allowed)
FORWARD	DROP	Handling all packets received by a sip:provider PRO node and destined
		for another, non-local IP destination (no default rules added)
OUTPUT	ACCEPT	Handling all packets originating on a sip:provider PRO node (no default
		rules added)
rtpengine	N/A	Container for rptengine rule to allow the rule to persist even when the
		Kernel module is unloaded (e.g. during upgrades)

## Table 19: NGCP netfilter default policies

The default firewall setup provided by sip:provider PRO:

- · adds rules to INPUT to secure access to platform and services
- · blocks all traffic from and to FORWARD
- allows all OUTPUT traffic

## 13.2.2 NGCP firewall configuration

The sip:provider PRO comes with a preconfigured set of firewall rules, which can be enabled and configured in /etc/ngcpconfig/config.yml in section security  $\rightarrow$  firewall. Refer to Section B.1.30 for available configuration options.

Firewall configuration is applied by running ngcpcfg apply. However, this will not activate new rules automatically to avoid inadvertent self-lockout. To finally activate new firewall rules run iptables-apply. This will prompt for another system logon to verify access remains available. If the prompt is not confirmed, firewall rules will automatically be reverted to the previous state re-enabling access to the command line.

### Caution

The NGCP 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/ngcpconfig/config.yml*.

## 13.2.3 IPv4 System rules

The following set of rules is added by the system upon activation of the firewall subsystem. Individual system rules are configured in /etc/ngcp-config/templates/etc/iptables/rules.v4.tt2 and /etc/ngcp-config/templates/etc/iptables/rules.v6.tt2

## Table 20: Firewall system rules

Zone	Chain	Target	Rule	Description
all	INPUT	rtpengine	-p udp -j rtpengine	Redirects all incoming UDP
				packets to chain <i>rtpengine</i> (putting
				RTPENGINE rule into a dedicated
				chain allows for the rule to persist
				even when the Kernel module gets
				unloaded, e.g. during upgrades)
all	rtpengine	RTPENGINE	-p udp -j RTPENGINE	Feeds all RTP packets to
			id 0	RTPENGINE Kernel module
n/a	INPUT	ACCEPT	-i lo -j ACCEPT	Accept all packets received by
				local loopback interface
all	INPUT	ACCEPT	-m statestate	Accept all incoming packets tied to
			RELATED,ESTABLISHED -j	related or established connections
			ACCEPT	
all	INPUT (IPv4)	ACCEPT	-p icmp -m icmp	Accept all ICMP echo messages
			icmp-type 8 -j ACCEPT	
all	INPUT (IPv4)	ACCEPT	-p icmp -m icmp	Accept all ICMP echo reply
			icmp-type 0 -j ACCEPT	messages
all	INPUT (IPv6)	ACCEPT	-A INPUT -p ipv6-icmp	Accept all ICMPv6 messages
			-j ACCEPT	
all	INPUT	cluster	-j cluster	Divert all incoming packets to the
				<i>cluster</i> chain
all	cluster	ACCEPT	-s <node_ip> -j ACCEPT</node_ip>	Set of rules white-listing all
				IP-addresses owned by the NGCP
				platform for incoming traffic
api_int	INPUT	ACCEPT	-p tcpdport	Set of rules for all api_int
			<ossbss.port> -j</ossbss.port>	interfaces accepting all incoming
			ACCEPT	packets for API port defined in
				/etc/ngcp-config/config.yml with
				key ossbss.port
mon_ext	INPUT	ACCEPT	+-p udp -s <i><snmpclient_ip></snmpclient_ip></i>	Set of rules for all mon_ext
			dport 161 -j ACCEPT	interfaces based on a list of IPs for
				all SNMP communities configured
				in checktools.snmpd.communities
rtp_ext	INPUT	ACCEPT	-p udpdport	Set of rules for all <i>rtp_ext</i>
			<rtpproxy.minport>:</rtpproxy.minport>	interfaces accepting all incoming
			<pre>' <rtpproxy.maxport>' -</rtpproxy.maxport></pre>	packets for RTP port range
			j ACCEPT	defined in
				/etc/ngcp-config/config.yml with
				keys <i>rtpproxy.minport</i> and
				rtpproxy.maxport

## Table 20: (continued)

Zone	Chain	Target	Rule	Description
sip_ext	INPUT	ACCEPT	-p udpdport	Set of rules for all <i>sip_ext</i>
			<kamailio.lb.port> -j</kamailio.lb.port>	interfaces accepting all packets on
			ACCEPT	the loda balancer's SIP signalling
				port defined in
				/etc/ngcp-config/config.yml with
				key kamailio.lb.port (UDP)
sip_ext	INPUT	ACCEPT	-p tcpdport	Set of rules for all <i>sip_ext</i>
			<kamailio.lb.port> -j</kamailio.lb.port>	interfaces accepting all packets on
			ACCEPT	the loda balancer's SIP signalling
				port defined in
				/etc/ngcp-config/config.yml with
				key kamailio.lb.port (TCP)
sip_ext	INPUT	ACCEPT	-p tcpdport	Set of rules for all <i>sip_ext</i>
			<kamailio.lb.tls.port></kamailio.lb.tls.port>	interfaces accepting all packets on
			-j ACCEPT	the loda balancer's SIP signalling
				port defined in
				/etc/ngcp-config/config.yml with
				key kamailio.lb.tls.port (TCP/TLS)
sip_ext	INPUT	ACCEPT	-p tcpdport 5222 -j	Set of rules for all <i>sip_ext</i>
			ACCEPT	interfaces accepting all packets on
				TCP port 5222 (XMPP client)
sip_ext	INPUT	ACCEPT	-p tcpdport 5269 -j	Set of rules for all <i>sip_ext</i>
			ACCEPT	interfaces accepting all packets on
				TCP port 5269 (XMPP server)
sip_ext	INPUT	ACCEPT	-p tcpdport <pushd.< td=""><td>Set of rules for all <i>sip_ext</i></td></pushd.<>	Set of rules for all <i>sip_ext</i>
			port> -j ACCEPT	interfaces accepting all packets
				incoming for the <i>pushd</i> server port
				configured in
				/etc/ngcp-config/config.yml with
				key pushd.port
ssh_ext	INPUT	ACCEPT	-A INPUT -i	List of rules to accept incoming
			<ssh_ext_interface> -p</ssh_ext_interface>	packets for SSH on all ssh_ext
			tcp -s < <i>sshd</i> .	interfaces from hosts configured in
			permit_support_from> -	/etc/ngcp-config/config.yml with
			-dport <i>sshd.port</i> -j	key sshd.permit_support_from
			ACCEPT	

Zone	Chain	Target	Rule	Description
web_ext	INPUT	ACCEPT	-p tcpdport	List of rules to accept incoming
			<www_admin.http_csc.< td=""><td>packets for the Customer Self</td></www_admin.http_csc.<>	packets for the Customer Self
			port> -j ACCEPT	Care interface defined in
				/etc/ngcp-config/config.yml with
				key www_admin.http_csc.port on
				all web_ext interfaces
web_int	INPUT	ACCEPT	-p tcpdport	List of rules to accept incoming
			<www_admin.http_admin.< td=""><td>packets for the Admin Panel</td></www_admin.http_admin.<>	packets for the Admin Panel
			port> -j ACCEPT	interface defined in
				/etc/ngcp-config/config.yml with
				key www_admin.http_admin.port
				on all web_int interfaces

### Table 20: (continued)

## Caution

To function correctly, the *rtpengine* requires an additional *iptables* rule installed. This rule (with a target of RTPENGINE) is automatically installed and removed when the rtpengine starts and stops, so normally you don't need to worry about it. However, any 3rd party firewall solution can potentially flush out all existing iptables rules before installing its own, which would leave the system without the required RTPENGINE rule and this would lead to decreased performance. It is imperative that any 3rd party firewall solution either leaves this rule untouched, or installs it back into place after flushing all rules out. The complete parameters to install this rule (which needs to go into the INPUT chain of the filter table) are: -p udp -j RTPENGINE --id 0

## Note

Some of the parameters used to populate the firewall rules automatically may contain hostnames instead of IP addresses. Since firewall rules need to be configured based on IP addresses by design, the NGCP configuration framework will lookup such hostnames during *ngcpcfg apply* and expand them to the IP addresses as returnd 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 reperform the lookup and update the rules to reflect chages in DNS. If this step is omitted, clients may be locked out of the system.

## 13.2.4 Custom rules

The NGCP configuration framework allows to add custom rules to the firewall setup in */etc/ngcp-config/config.yml*. The custom rules are added after the system rules. Hence, they apply for packets not matched by the systems rules only.

Example custom rule to whitelist all IPv4 traffic from network interface eth1.301 effectively making VLAN 301 a trusted network:

```
rules4:
- '-A INPUT -i eth1.301 -j ACCEPT'
```

Example custom rule to accept incoming traffic from monitoring station 203.0.113.93 for an optionally installed check\_mk agent:

rules4: - '-A INPUT -p tcp -s 203.0.113.93 --dport 6556 -j ACCEPT'

To add hosts or networks to the SSH whitelist they can be either added to key *sshd.permit\_support\_from* in */etc/ngcp-config/config.yml* or a custom rule may be used:

```
rules4:

- '-A INPUT -s 198.51.100.0/24 --dport 22 - j ACCEPT'

- '-A INPUT -s 203.0.113.93 --dport 22 -j ACCEPT'
```

#### Note

In custom rules keys from /etc/ngcp-config/config.yml cannot be referenced. Thus, the values need to be manually looked up, hard coded, and kept in sync manually. This is by design of YAML.

### 13.2.5 Example firewall configuration section

An example for NGCP firewall configuration in /etc/ngcp-config/config.yml enabling both the firewall subsystem and the logging facility may look like:

```
security:
firewall:
    enable: 'yes'
    logging:
        enable: 'yes'
        file: '/var/log/firewall.log'
        tag: 'NGCPFW'
    policies:
        input: 'DROP'
        forward: 'DROP'
        output: 'ACCEPT'
    rules4:
        - '-A INPUT -i eth0 -j ACCEPT'
```

### 13.3 Password management

The sip:provider PRO comes with some default passwords the user should change during the deployment of the system. They have been explained in the previous chapters of this handbook.

- The root user in MySQL has no default password. A password should be set using the mysqladmin password command.
- The administrative web interface has a default user *administrator* with password *administrator*. It should be changed within this interface.
- Generate new password for user *ngcpsoap* to access the provisioning interfaces, see the details in Section 8.

## Important

Many NGCP services use MySQL backend. Users and passwords for these services are created during the installation. These passwords are unique for each installation, and the connections are restricted to localhost. You should not change these users and passwords.

## 13.4 SSL certificates.

The sip:provider PRO provides default, self-signed SSL certificates for SSL connections. These certificates are common for every installation. Before going to production state, the system administrator should provide SSL certificates for the web services. These certificates can either be shared by all web interfaces (*provisioning*, *administrator interface* and *customer self care interface*), or separate ones for each them can be used.

- Generate the certificates. The *customer self care interface* certificate should be signed by a certification authority to avoid browser warnings.
- · Upload the certificates to the system
- Set the path to the new certificates in /etc/ngcp-config/config.yml:
  - $ossbss \rightarrow apache \rightarrow autoprov \rightarrow sslcert file and <math>ossbss \rightarrow apache \rightarrow autoprov \rightarrow sslcert key file for the provisioning interface.$
  - $ossbss \rightarrow apache \rightarrow restapi \rightarrow sslcert file and <math>ossbss \rightarrow apache \rightarrow restapi \rightarrow sslcert key file for the REST interface.$
  - www\_admin $\rightarrow$  http\_admin $\rightarrow$  sslcertfile and www\_admin $\rightarrow$  http\_admin $\rightarrow$  sslcertkeyfile for the admin interface.
  - www\_admin $\rightarrow$  http\_csc $\rightarrow$ sslcertfile and www\_admin $\rightarrow$  http\_csc $\rightarrow$ sslcertkeyfile for the customer self care interface.
- Apply the configuration changes with ngcpcfg apply 'added web ssl certs'.

The sip:provider PRO 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 (*ka-mailio* $\rightarrow$ *lb* $\rightarrow$ *tls* $\rightarrow$ *enable* (disabled by default)).

- Generate the certificates.
- · Upload the certificates to the system
- Set the path to the new certificates in /etc/ngcp-config/config.yml:

- $kamailio \rightarrow lb \rightarrow tls \rightarrow sslcert file and kamailio \rightarrow lb \rightarrow tls \rightarrow sslcert key file .$
- Apply the configuration changes with ngcpcfg apply 'added kamailio certs'.

## 13.5 Securing your sip:provider PRO against SIP attacks

The sip:provider PRO allows you to protect your VoIP system against SIP attacks, in particular **Denial of Service** and **brute-force attacks**. Let's go through each of those attacks and let's see how to configure your system in order to face such situations and react against them.

### 13.5.1 Denial of Service

As soon as you have packets arriving on your sip:provider PRO server, it will require a bit of time of your CPU. Denial of Service attacks are aimed to break down your system by sending floods of SIP messages in a very short period of time and keep your system busy to handle such huge amount of requests. sip:provider PRO allows you to block such kind of attacks quite easily, by configuring the following section in your /etc/ngcp-config/config.yml:

```
security:
    dos_ban_enable: 'yes'
    dos_ban_time: 3600
    dos_reqs_density_per_unit: 50
    dos_sampling_time_unit: 2
    dos_whitelisted_ips: []
    dos_whitelisted_subnets: []
```

Basically, as soon as sip:provider PRO 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 the kamailio-lb.log a line saying:

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

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

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

### Excluding SIP endpoints from banning

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



## Caution

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

The NGCP platform administrator may also add whitelisted IP addresses manually in /etc/ngcp-config/config.yml at kamailio.lb.security.dos\_whitelisted\_ips and kamailio.lb.security.dos\_whitelisted\_sub nets parameters.

#### 13.5.2 Bruteforcing SIP credentials

This is a very common attack you can easily detect checking your /var/log/ngcp/kamailio-proxy.log. You will see INVITE/REGISTER messages coming in with strange usernames. Attackers is trying to spoof/guess subscriber's credentials, which allow them to call out. The very first protection against these attacks is: **ALWAYS USE STRONG PASSWORD**. Nevertheless sip:provider PRO allow you to detect and block such attacks quite easily, by configuring the following /etc/ngcp-config/config.yml section:

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

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

```
Nov 9 13:31:56 sp1 lb[41952]: WARNING: <script>: Consecutive Authentication Failure for ' ↔
sipvicous@mydomain.com' UA='sipvicous-client' IP='1.2.3.4' - R=<null> ID ↔
=313793-3624525116-589163@testlab.local
```

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

# ngcp-kamctl lb fifo sht\_dump auth

### 13.6 System Requirements and Performance

The sip:provider PRO is a very flexible system, capable of serving from hundreds to several tens of thousands of subscribers in a single node. The system comes with a default configuration, capable of serving up to 50.000 subscribers in a *normal* environment. But there is no such thing as a *normal* environment. And the sip:provider PRO has sometimes to be tunned for special environments, special hardware requirements or just growing traffic.

#### Note

If you have performance issues with regards to disk I/O please consider enabling the *noatime* mount option for the root filesystem. Sipwise recommends the usage of *noatime*, though remove it if you use software which conflicts with its presence.

In this section some parameters will be explained to allow the sip:provider PRO administrator tune the system requirements for optimum performance.

Option	Default value	Requirement impact
cleanuptools $\rightarrow$ binlog_days	15	Heavy impact on the harddisk storage needed for mysql logs. It can help
		to restore the database from backups or restore broken replication.
database→bufferpoolsize	1/2 * Total	The installer will calculate the total system RAM and dedicate 50% to the
	system RAM	mysql innodb buffer. This value won't be changed in case the system
		RAM changes so it's up to the administrator to adjust it. For test systems
		or low RAM systems, lowering this setting is one of the most effective
		ways of releasing RAM. The administrator can check the innodb buffer hit
		rate on production systems; a hit rate over 99% is desired to avoid
		bottlenecks.
kamailio→lb→pkg_mem	16	This setting affects the amount of RAM the system will use. Each
		kamailio-lb worker will have this amount of RAM reserved. Lowering this
		setting up to 8 will help to release some memory depending on the
		number of kamailio-lb workers running. This can be a dangerous setting
		as the lb process could run out of memory. Use with caution.
kamailio→lb→shm_mem	1/16 * Total	The installer will set this value to 1/16 of the total system RAM. This
	System RAM	setting does not change even if the system RAM does so it's up to the
		administrator to tune it. It has been calculated that 1024 (1GB) is a good
		value for 50K subscriber environment. For a test environment, setting the
		value to 64 should be enough. "Out of memory" messages in the
		kamailio log can indicate that this value needs to be raised.
kamailio→lb→tcp_children	8	Number of TCP workers kamailio-lb will spawn per listening socket. The
		value should be fine for a mixed UDP-TCP 50K subscriber system.
		Lowering this setting can free some RAM as the number of kamailio
		processes would decrease. For a test system or a pure UDP subscriber
		system 2 is a good value. 1 or 2 TCP workers are always needed.
kamailio→lb→tls→enable	yes	Enable or not TLS signaling on the system. Setting this value to "no" will
		prevent kamailio to spawn TLS listening workers and free some RAM.
kamailio→lb→udp_children	8	See <i>kamailio</i> → <i>lb</i> → <i>tcp_children</i> explanation
kamailio→proxy→children	8	See $kamailio \rightarrow lb \rightarrow tcp\_children$ 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.
kamailio→proxy→*_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.
kamailio-proxy-natping_inte	rval 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 <i>kamailio→proxy→natping_processes</i> explanation.

## Table 21: Requirement\_options

Option	Default value	Requirement impact
kamailio->proxy->natping_processes 7		Kamailio-proxy will spawn this number of processes to send keepalive
		OPTIONS to the nated subscribers. Each worker can handle about 250
		messages/second (depends on the hardware). Depending the number of
		nated subscribers and the <i>kamailio</i> → <i>proxy</i> → <i>natping_interval</i> 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.
kamailio->proxy->shm_mem	1/16 * Total	See kamailio $\rightarrow$ lb $\rightarrow$ shm_mem explanation.
	System RAM	
rateomat→enable	yes	Set this to no if the system shouldn't perform rating on the CDRs. This
		will save CPU usage.
rsyslog→external_log	0	If enabled, the system will send the log messages to an external server.
		Depending on the <i>rsyslog</i> → <i>external_loglevel</i> parameter this can
		increase dramatically the network traffic.
rsyslog→ngcp_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.

## Table 21: (continued)

## Тір

In case of using virtualized environment with limited amount of hardware resources, you can use the script *ngcp-toggle-performance-config* to adjust sip:provider PRO configuration for high/low performance:

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

```
/usr/sbin/ngcp-toggle-performance-config - tool to adjust sip:provider configuration for ↔
    low/high performance
    --help Display this usage information
    --high-performance Adjust configuration for system with normal/high performance
    --low-performance Adjust configuration for system with low performance (e.g. VMs)
```

root@spce:~#

## 13.7 Troubleshooting

The sip:provider PRO platform provides detailed logging and log files for each component included in the system via rsyslog. The main folder for log files is /var/log/ngcp/, it contains a list of self explanatory log files named by component name.

The sip:provider PRO is a high performance system which requires compromise between traceability (maximum amount of debug information being written to hard drive) and productivity (minimum load on IO subsystem). This is the reason why different log levels are configured for the provided components by default.

Most log files are designed for debugging sip:provider PRO 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	medium
	providing type	
	and content of	
	API requests	
	and	
	responses as	
	well as	
	potential	
	errors	
/var/log/ngcp/panel.log	Admin Web UI	medium
/var/log/ngcp/panel-	logs when	
debug.log	performing	
	operational	
	tasks on the	
	ngcp-panel	
/var/log/ngcp/cdr.log	mediation and	medium
	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	

Log file	Content	Estimated size
/var/log/ngcp/ha.log	fail-over	small
	related logs in	
	case a node	
	in a pair loses	
	connection to	
	the other side,	
	when a	
	standby node	
	takes over or	
	an active	
	node goes	
	standby due	
	to intra-node	
	communica-	
	tion issues or	
	external ping	
	node	
	connection	
	issues	
/var/log/ngcp/kamailio-	Overview of	huge
proxy.log	SIP requests	
	and replies	
	between lb,	
	proxy and	
	sems	
	processes. It's	
	the main log	
	file for SIP	
	overview	
/var/log/ngcp/kamailio-lb.log	Overview of	huge
	SIP requests	
	and replies	
	along with	
	network	
	source and	
	destination	
	information	
	flowing	
	through the	
	platform	

Log file	Content	Estimated size
/var/log/ngcp/sems.log	Overview of	small
	SIP requests	
	and replies	
	between lb,	
	proxy and	
	sems	
	processes	
/var/log/ngcp/rtp.log	rtpengine	small
	related log,	
	showing	
	information	
	about RTP	
	communica-	
	tion	



## 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.

## 13.7.1 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 CalIID (a unique identifier for a SIP dialog). The call ID is used as call marker in almost all the voip related log file, such as /var/log/ngcp/kamailio-lb.log , /var/log/ngcp/kamailio-proxy.log , /var/log/ngcp/sems.log or /var/log/ngcp/rtp.log. Example of kamailio-proxy.log line:

```
Nov 19 00:35:56 sp1 proxy[7475]: NOTICE: <script>: New request on proxy - M=REGISTER R=sip: ↔
    sipwise.local
F=sip:jdoe@sipwise.local T=sip:jdoe@sipwise.local IP=10.10.1.10:5060 (127.0.0.1:5060) ID ↔
    =364e4676776621034977934e055d19ea@127.0.0.1 UA='SIP-UA 1.2.3.4'
```

The above line shows the SIP information you can find in a general line contained in /var/log/ngcp/kamailio-\*:

- M=REGISTER : The SIP Method
- R=sip:sipwise.local : The SIP Request URI
- F=sip:jdoe@sipwise.local : The SIP From header

- T=sip:jdoe@sipwise.local : The SIP To header
- IP=10.10.1.10:5060 (127.0.0.1:5060) : The source IP where the message is coming from. Between brackets it is shown the local internal IP where the message come from (in this case Load Balancer)
- ID=364e4676776621034977934e055d19ea@127.0.0.1 : The SIP CallID.
- UAIP=10.10.1.10 : The User Agent source IP
- UA=SIP-UA 1.2.3.4 : The SIP User Agent header

In order to collect the full log related to a single call, it's necessary to "grep" the /var/log/ngcp/kamailio-proxy.log using the ID= string, for example:

# grep "364e4676776621034977934e055d19ea@127.0.0.1" /var/log/ngcp/kamailio-proxy.log

### 13.7.2 Collecting SIP traces

The sip:provider PRO platform provides several tools to collect SIP traces. It can be used the sip:provider PRO *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 -0 /tmp/SIP\_trace\_file\_name.pcap port 5062 or port 5060

The sngrep debian graphic tool as well can be used to visualize SIP trace and save them in a .pcap file :

```
# sngrep
```

The sip:provider PRO platform provides also the native Voip sniffer, called *voisniff-ng*, which provide a graphic view of all the calls passing through the platform. It can be enabled via \_\_/etc/ngcp-config/config.yml:

```
voisniff:
admin_panel: 'yes'
daemon:
bpf: 'port 5060 or 5062 or ip6 proto 44 or ip[6:2] & 0x1fff != 0'
external_interfaces: 'eth0 eth1'
filter:
    exclude:
    -
    active: 1
    case_insensitive: 1
    pattern: '\ncseq: *\d+ +(register|notify|options|subscribe)'
```

```
include: []
internal_interfaces: lo
mysql_dump_threads: 4
start: 'yes'
threads_per_interface: 10
partitions:
increment: 700000
keep: 10
```

admin\_panel should be set to yes as well as start. Also filter.exclude.active should be set to 1 in order to avoid to sniff REGISTER, NOTIFY, OPTIONS and SUBSCRIBE messages. Then run:

ngcpcfg apply 'enable voisniff' && ngcpcfg push



## Warning

Please notice that enabling voisniff, specially under a huge amount of traffic, may affect the system performance due to the fact that voisniff needs to save all the traffic into the database.

# 14 Monitoring and Alerting

## 14.1 Internal Monitoring

### 14.1.1 Process monitoring via monit

The platform uses the internal *monit* service to monitor all essential services. Since the sip:provider PRO runs in an active/standby mode, not all services are always running on both nodes, some of them will only run on the active node and be stopped on the standby node. The following commands show the most critical services on the platform: \* monit summary - to get the list of services and their current status, \* monit status - to get the list of services with detailed status.



When you perform a stop/start/monitor/unmonitor operation on a service, *monit* affects other services that depend on the initial one. Hence, if you stop or unmonitor a service all services that depend on it will be stopped or unmonitored as well.

For example, monit stop mysql operation will stop kamailio, sbc, asterisk, prosody and some other services. Although the recommended way to operate on services is via the ngcp-services wrapper which will take care of abstracting the underlying process monitoring implementation.

If any service ever fails for whatever reason the *monit* daemon quickly restarts it. When that happens, the daemon will send a notification email to the address specified in the config.yml file under the general.adminmail key. It will also send warning emails to this address under certain abnormal conditions, such as high memory consumption (> 75% is used) or high CPU load.

### Important

In order for *monit* to be able to send emails to the specified address, the local MTA (*exim4*) must be configured correctly. If you haven't done so already, run dpkg-reconfigure exim4-config to do this. The CE edition's handbook contains more information about this in the *Installation* chapter.

## 14.1.2 System monitoring via Telegraf

The platform uses the internal *telegraf* service 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 InfluxDB, in the telegraf database.

### 14.1.3 NGCP-specific monitoring via ngcp-witnessd

The platform uses the internal *ngcp-witnessd* service to monitor NGCP-specific metrics or system metrics currently not tracked by *telegraf*, including memory, process count, Heartbeat, MTA, Kamailio, SIP and MySQL.

The gathered information is stored in *InfluxDB*, in the *ngcp* database.

### 14.1.4 Monitoring data in InfluxDB

The platform uses InfluxDB as a time series database, to store most of the metrics collected in the system.

On a sip:provider PRO each node stores its own metrics and the ones for their peer node. This is done via *influxdb-relay* which listens for *InfluxDB* writes and multiplexes them to the local node and any other node necessary.

The monitoring data is used by various components of the platform, including *ngcp-collective-check*, *ngcp-snmp-agent* and by the statistics dashboard powered by *Grafana*.

The monitoring data can also be accessed directly by various means; by using the *influx* command-line tool in CLI or TUI modes; by using the *ngcp-influxdb-extract* wrapper which provides two convenience commands to run arbitrary queries or to fetch the last value for a measurement's field; or by using the HTTP API with *curl* (or other HTTP fetchers), or with the *Sipwise::InfluxDB::HTTP* perl module.

### Тір

See https://docs.influxdata.com/influxdb/v1.1/query\_language/spec/ for information about InfluxQL, the query language used by *InfluxDB*.

### Tip

To get the list of all measurements for a specific database the following query can be used SHOW MEASUREMENTS.

### Tip

To get the list of fields for a specific measurement the following query can be used SELECT LAST(\*) FROM "measurem ent".

## Тір

To get the list of tags for a specific measurement the following query can be used SHOW TAG KEYS FROM "measureme nt", and for all the current tag values for a tag SHOW TAG VALUES FROM "measurement" WITH KEY = "tag".

## 14.2 Monitoring data in Redis

The platform uses *Redis* to store some of the monitoring data, mostly due to historical reasons, as the previously used RRD files did not make it possible to store anything other than numbers.

See Section F.2.1 for detailed information about the list of data currently stored in the *Redis* monitoring database.

### Note

These keys are being phased out, and will be moved to InfluxDB.

## 14.3 Statistics Dashboard

The platform's administration interface (described in Section 4) 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.

## 14.4 External Monitoring Using SNMP

### 14.4.1 Overview and Initial Setup

The sip:provider PRO exports a variety of cluster health data and statistics over the standard SNMP interface. By default, the SNMP interface can only be accessed locally. To make it possible to provide the SNMP data to an external system, the config. yml file needs to be edited and the list of allowed community names and allowed hosts/IP ranges must be populated. This list can be found under the checktools.snmpd.communities key and it consists of one or more community/source value pairs. The community is the allowed community name, while source is an IP address or an IP block where to allow the requests from.

The public entry with the localhost source is used for local testing of SNMP functionality. It is recommended that you leave this entry in place. Other legal sources can be formed as single IP addresses or IP blocks in IP/prefix notation, for example 192.168.115.0/24.

#### Tip

To locally check if SNMP is working correctly, execute the command snmpwalk -v2c -cpublic localhost . (note the trailing dot). This will generate a long list of raw SNMP OIDs and their values, provided that the default SNMP community key has been left in place.

## Тір

SNMP version 1 and version 2c are supported.

## 14.4.2 Details

There are two types of information that can be retrieved from SNMP. The first one is the native NGCP cluster overview from the Sipwise MIBs (Management Information Bases). The second is the legacy ad-hoc information using the Net-SNMP extension OIDs, and detailed information for the node running the SNMP daemon using standard OIDs (Object Identifiers).

## 14.4.2.1 Sipwise NGCP OIDs

The entire NGCP cluster can be monitored by using the SIPWISE-NGCP-MIB, SIPWISE-NGCP-MONITOR-MIB and SIPW ISE-NGCP-STATS-MIB. These OIDs are rooted at the Sipwise NGCP slot .1.3.6.1.4.1.34274.1.\*.

The MIBs are self-documented, and can be found as part of the *ngcp-snmp-mibs* package (running dpkg -S *SIPWISE\*MIB* will list their pathnames). The NGCP SNMP Agent is a part of the *ngcp-snmp-agent* package, which is installed by default and

works out-of-the-box as long as the snmpd has been properly configured.

The SIPWISE-NGCP-MIB acts as the root MIB and provides information about the cluster licensing and layout (which is mostly static data about each node, such as node name, its IP address, its roles, etc.) and information required to access the OIDs from the other MIBs.

The SIPWISE-NGCP-MONITOR-MIB provides current monitoring information, global health conditions, the number of provisioned and registered subscribers and devices. It also provides per node information (independently of the number of nodes or their names) on their filesystem, processes, databases, system load, memory, heartbeat status, MTA queues, etc.

The SIPWISE-NGCP-STATS-MIB provides accumulated statistics on billing, performance and processed SIP messages.

NOTICE: OIDs under the following trees are not yet implemented: *ngcpMonitorFraud*, *ngcpMonitorPerformance.perfCAPSCurTable* and *ngcpStats*.

### Tip

The NGCP SNMP Agent uses *Redis* and *InfluxDB* as data sources. This data is essential for accurate and complete monitoring data in the SNMP OID tree. In addition, the *Redis* database must be available on a shared IP address, so that *ngcp-witnessd* can always write to it.

## 14.4.2.2 Legacy OIDs

### Note

The following OIDs have been superseded by the Sipwise NGCP OIDs, but they are still provided for backwards compatibility.

All basic system health variables (such as memory, disk, swap, CPU usage, network statistics, process lists, etc.) for the *mgmt* node can be found in standard OID slots from standard MIBs. For example, memory statistics can be found through the *UCD-SNMP-MIB* in OIDs such as memTotalSwap.0, memAvailSwap.0, memTotalReal.0, memAvailReal.0, etc., which translate to numeric OIDs .1.3.6.1.4.1.2021.4.\*. In fact, UCD-SNMP-MIB is the most useful MIB for overall system health checks.

Additionally, there's a list of specially monitored processes, also found through the UCD-SNMP-MIB. UCD-SNMP-MIB::prNa mes (.1.3.6.1.4.1.2021.2.1.2) gives the list of monitored processes, prCount (.1.3.6.1.4.1.2021.2.1.5) is how many of each process are running and prErrorFlag (.1.3.6.1.4.1.2021.2.1.100) gives a 0/1 error indication (with prErrMessage (.1.3.6.1.4.1.2021.2.1.101) providing an explanation of any error).

### Тір

Some of these processes are not supposed to be running on the standby node, so you'll see the error flag raised there. A possible solution is to run these SNMP checks against the shared service IP of the cluster. See in Section 2.4 below for more information.

Furthermore, UCD-SNMP-MIB provides a list of custom external checks. The names of these can be found under the UCD-SNMP-MIB::extNames(.2) tree, with extOutput(.101) providing the output (one line) from each check and extResult

(.100) the exit code from each check.

The first of these external checks called collective\_check provides a combined and overall system health status indicator. It gathers information from both nodes and returns 0 in extResult.1(.100.1) if everything is OK and running as it should. If it finds a problem somewhere, but with the system still operational (e.g. a service is stopped on the inactive node), extResult. 1 will return 1 and extOutput.1 will be set to a string that can be used to diagnose the problem. In case the system is found in a critical and non-operational state, extResult.1 will return 2, again with an error message set. If you want to keep it really simple, you can just monitor this one OID and raise an alarm if it ever goes to non-zero.

Тір

The 0/1/2 status codes allow for easy integration with Nagios.

The remaining external checks simply return statistics on the system, they all return a number in extOutput and have extRe sult always set to zero.

The full list of such checks is below. All of these checks have three modes: the first returns the statistics from sp1 (the first node in the sip:provider PRO pair), the second - from sp2, and the third - from whichever node is being queried (which is useful when querying the shared service IP). For example, the local SIP response time from sp1 is in  $sip_check_sp1$ , from sp2 - is in  $sip_check_sp2$ , and from the host itself - is in  $sip_check_self$ .

The base OID of the Result and Output OIDs is always .1.3.6.1.4.1.2021.8.1, so if you read .100.1, the full OID is .1.3.6.1.4.1.2021.8.1.100.1.

Name in MIB	Result OID	Output OID	Name	Description
UCD-SNMP-	.100.1	.101.1	collective_check	Summarized platform
MIB::extNames.1				check
UCD-SNMP-	.100.2	.101.2	sip_check_sp1	SIP response time in
MIB::extNames.2				seconds on sp1
UCD-SNMP-	.100.3	.101.3	sip_check_sp2	SIP response time in
MIB::extNames.3				seconds on sp2
UCD-SNMP-	.100.4	.101.4	mysql_check_sp1	Average number of
MIB::extNames.4				MySQL queries per
				second on sp1
UCD-SNMP-	.100.5	.101.5	mysql_check_sp2	Average number of
MIB::extNames.5				MySQL queries per
				second on sp2
UCD-SNMP-	.100.6	.101.6	mysql_replication_che	eck_MgSQL replication
MIB::extNames.6				delay in seconds on
				sp1
UCD-SNMP-	.100.7	.101.7	mysql_replication_che	eck_Mg8QL replication
MIB::extNames.7				delay in seconds on
				sp2
UCD-SNMP-	.100.8	.101.8	mpt_check_sp1	RAID status on sp1
MIB::extNames.8				

Name in MIB	Result OID	Output OID	Name	Description
UCD-SNMP-	.100.9	.101.9	mpt_check_sp2	RAID status on sp2
MIB::extNames.9				
UCD-SNMP-	.100.10	.101.10	exim_queue_check_sp	Number of mails
MIB::extNames.10				undelivered in MTA
				queue on sp1
UCD-SNMP-	.100.11	.101.11	exim_queue_check_sp2	2 Number of mails
MIB::extNames.11				undelivered in MTA
				queue on sp2
UCD-SNMP-	.100.12	.101.12	provisioned_subscriber	s_Noberdbe_ispfi
MIB::extNames.12				subscribers
				provisioned on sp1
UCD-SNMP-	.100.13	.101.13	provisioned_subscriber	s_Noberdbe_isp2
MIB::extNames.13				subscribers
				provisioned on sp2
UCD-SNMP-	.100.14	.101.14	kam_dialog_active_che	ck <u>N</u> spnber of active
MIB::extNames.14				calls on sp1
UCD-SNMP-	.100.15	.101.15	kam_dialog_active_che	ckNsp2ber of active
MIB::extNames.15				calls on sp2
UCD-SNMP-	.100.16	.101.16	kam_dialog_early_chec	k <u>N</u> spr1nber of calls in
MIB::extNames.16				Early Media state on
				sp1
UCD-SNMP-	.100.17	.101.17	kam_dialog_early_chec	k <u>N</u> sup2aber of calls in
MIB::extNames.17				Early Media state on
				sp2
UCD-SNMP-	.100.18	.101.18	kam_dialog_type_local	c <b>heckbep</b> of active
MIB::extNames.18				calls local on sp1
UCD-SNMP-	.100.19	.101.19	kam_dialog_type_local	c <b>heckbep</b> Øf active
MIB::extNames.19				calls local on sp2
UCD-SNMP-	.100.20	.101.20	kam_dialog_type_relay	dNeokbespof active
MIB::extNames.20				calls routed via peers
				on sp1
UCD-SNMP-	.100.21	.101.21	kam_dialog_type_relay	dNeokosp@factive
MIB::extNames.21				calls routed via peers
				on sp2
UCD-SNMP-	.100.22	.101.22	kam_dialog_type_incon	nin <u>kur</u> abeck <u>of</u> sip¢oming
MIB::extNames.22				calls on sp1
UCD-SNMP-	.100.23	.101.23	kam_dialog_type_incon	nin <u>kur</u> abeck <u>of</u> sp&oming
MIB::extNames.23				calls on sp2
UCD-SNMP-	.100.24	.101.24	kam_dialog_type_outgo	in <b>l<u>gu</u>ærhæe</b> rk <u>o</u> fs <b>pú</b> tgoing
MIB::extNames.24				calls on sp1
UCD-SNMP-	.100.25	.101.25	kam_dialog_type_outgo	in <b>lo<u>ju</u>orhoee</b> rko <u>f</u> s <b>p</b> 2tgoing
MIB::extNames.25				calls on sp2

Name in MIB	Result OID	Output OID	Name	Description
UCD-SNMP-	.100.26	.101.26	kam_usrloc_reguse	ers_chbekckm_beptiof
MIB::extNames.26				subscribers with at
				least one active
				registration on sp1
UCD-SNMP-	.100.27	.101.27	kam_usrloc_reguse	ers_ch <b>blck<u>m</u>bp</b> 2of
MIB::extNames.27				subscribers with at
				least one active
				registration on sp2
UCD-SNMP-	.100.28	.101.28	kam_usrloc_regdev	rices_dlotextkn_spotber of
MIB::extNames.28				registered end
				devices on sp1
UCD-SNMP-	.100.29	.101.29	kam_usrloc_regdev	rices_dloteedkn_uspo2ber of
MIB::extNames.29				registered end
				devices on sp2
UCD-SNMP-	.100.30	.101.30	mysql_replication_c	liscrephuncies_oheths_op1
MIB::extNames.30				tables not in sync
				between sp1 and sp2
UCD-SNMP-	.100.31	.101.31	mysql_replication_c	liscreptancies_ohetks_op2
MIB::extNames.31				tables not in sync
				between sp1 and sp2
UCD-SNMP-	.100.32	.101.32	sip_check_self	Summarized platform
MIB::extNames.32				check on active node
UCD-SNMP-	.100.33	.101.33	mysql_check_self	Average number of
MIB::extNames.33				MySQL queries per
				second on active
				node
UCD-SNMP-	.100.34	.101.34	mysql_replication_c	heck_MggBQL replication
MIB::extNames.34				delay in seconds on
				active node
UCD-SNMP-	.100.35	.101.35	mpt_check_self	RAID status on active
MIB::extNames.35				node
UCD-SNMP-	.100.36	.101.36	exim_queue_check	_self Number of mails
MIB::extNames.36				undelivered in MTA
				queue on active node
UCD-SNMP-	.100.37	.101.37	provisioned_subscr	ibers_Nobendbe_seff
MIB::extNames.37				subscribers
				provisioned on active
				node
UCD-SNMP-	.100.38	.101.38	kam_dialog_active_	_check/uselfber of active
MIB::extNames.38				calls on active node
UCD-SNMP-	.100.39	.101.39	kam_dialog_early_c	check Neuentriber of calls in
MIB::extNames.39				Early Media state on
				active node

Name in MIB	Result OID	Output OID	Name	Description
UCD-SNMP-	.100.40	.101.40	kam_dialog_type_local_	c <b>heckbeeb</b> f active calls
MIB::extNames.40				local on active node
UCD-SNMP-	.100.41	.101.41	kam_dialog_type_relay	dNeoko_sebf active
MIB::extNames.41				calls routed via peers
				on active node
UCD-SNMP-	.100.42	.101.42	kam_dialog_type_incon	nin <u>tur</u> abeck <u>of</u> setfoming
MIB::extNames.42				calls on active node
UCD-SNMP-	.100.43	.101.43	kam_dialog_type_outgo	in <b>l<u>gu</u>orhoeo</b> ko <u>f</u> s <b>el</b> ftgoing
MIB::extNames.43				calls on active node
UCD-SNMP-	.100.44	.101.44	kam_usrloc_regusers_c	h <b>&amp;kck<u>m</u>belf</b> of
MIB::extNames.44				subscribers with at
				least one active
				registration on active
				node
UCD-SNMP-	.100.45	.101.45	kam_usrloc_regdevices	_d <b>bædk<u>n</u>sælb</b> er of
MIB::extNames.45				registered end
				devices on active
				node
UCD-SNMP-	.100.46	.101.46	mysql_replication_discr	ephanciesr_ohetks_sallf
MIB::extNames.46				tables not in sync
				between sp1 and sp2

## Тір

Some of the checks can be disabled (most are enabled by default) through the config.yml file, and those checks will then return an error message or an empty string in their extOutput. Enable those checks in the config file to get their output in the SNMP OID tree. The enable/disable flags can be found in the checktools section.

# 15 Extensions and Additional Modules

## 15.1 Cloud PBX

The sip:provider PRO comes with a commercial Cloud PBX module to provide B2B features for small and medium sized enterprises. The following chapters describe the configuration of the PBX features.

## 15.1.1 Configuring the Device Management

The *Device Management* is used by admins and resellers to define the list of device models, firmwares and configurations available for end customer usage. These settings are pre-configured for the default reseller up-front by Sipwise and have to be set up for every reseller separately, so a reseller can choose the devices he'd like to serve and potentially tweak the configuration for them. List of available pre-configured devices Section 15.1.12.

End customers choose from a list of *Device Profiles*, which are defined by a specific *Device Model*, a list of *Device Firmwares* and a *Device Configuration*. The following sections describe the setup of these components.

To do so, go to Settings  $\rightarrow$  Device Management.

<b>.</b>	Logged in as	administrator Logout
sip:wise NGCP Dashboard	*	🗰 Settings 👻
Device Management		
← Back		** Expand Groups
Device Models		
Device Firmwares		
Device Configurations		
Device Profiles		
© 2013 Sipwise GmbH, all rights reserved.		



## 15.1.1.1 Setting up Device Models

A *Device Model* defines a specific hardware device, like the vendor, model name, the number of keys and their capabilities. For example a Cisco SPA504G has 4 keys, which can be used for private lines, shared lines (SLA) and busy lamp field (BLF). If you have an additional attendant console, you get 32 more buttons, which can only do BLF.

In this example, we will create a Cisco SPA504G with an additional Attendant Console.

Expand the Device Models row and click Create Device Model.

First, you have to select the reseller this device model belongs to, and define the vendor and model name.

							🛓 Logg	ed in as a	administrator L	.ogout
sip:wise	NGCP Dash	board						A	III Settings	
Devic	Create Devi	ce Mode	l						×	
		Reseller				Searc	h:			
			#		Name	Contract #	Status		d Gro	oups
🗲 Back			1		default	1	active	_		
			3		Demo Reseller	3	active	1		
Device			Showing 1 to	to 2 of 2 entrie	25		← 1	$\rightarrow$	⇒	
*0							Creat	e Resel	ller	
Show	2	Vendor	Cisco							
#	3	Model	SPA504G	+ 1 Attenda	nt Console					
No dat	Line/K	ey Range		Na	me Phone Ke	ys				
Showing			Number	r of Lines/K	eys 4					⇒
Device									Save	
Device Co	onfigurations									
Device Pr	rofiles									



In the *Line/Key Range* section, you can define the first set of keys, which we will label Phone Keys. The name is important, because it is referenced in the configuration file template, which is described in the following sections. The SPA504G internal phone keys support private lines (where the customer can assign a normal subscriber, which is used to place and receive standard phone calls), shared lines (where the customer can assign a subscriber which is shared across multiple people) and busy lamp field (where the customer can assign other subscribers to be monitored when they get a call, and which also acts as speed dial button to the subscriber assigned for BLF), so we enable all 3 of them.

		Logged in as administrator Logout
sip:ເພາຣະ NGCP Dashboard		A III Settings ▼
Devic Create Device Mode		×
Model	SPA504G + 1 Attendant Console	▲ d Groups
Device 5	4     Name     Phone Keys       Number of Lines/Keys     4	=
6	Supports Private Line 🕑	=
Show 8	Supports Shared Line 🕑	
# No dat	Field	Remove
Showing		Save
Device Configurations		
Device Profiles		



In order to also configure the attendant console, press the *Add another Line/Key Range* button to specify the attendant console keys.

Again provide a name for this range, which will be Attendant Console 1 to match our configuration defined later. There are 32 buttons on the attendant console, so set the number accordingly. Those 32 buttons only support BLF, so make sure to **uncheck** the private and shared line options, and only check the busy lamp field option.

		Logged in as administrator Logout
sip:wise NGCP Dashboard		🚔 🇱 Settings 👻
Devic Create Device Model		×
	9 Name Attendant Console 1	d Groups
← Back 10	Number of Lines/Keys 32	
Device 11	Supports Private Line 🔲	
* C	Supports Shared Line 📃	
Show [ 12	Supports Busy Lamp 🗹 Field	=
#		Remove
No dat		
Showing	Add another	r Line/Key Range
Device		Save
Device Configurations Device Profiles		



The last two settings to configure are the *Front Image* and *MAC Address Image* fields. Upload a picture of the phone here in the first field, which is shown to the customer for him to recognize easily how the phone looks like. The MAC image is used to tell the customer where he can read the MAC address from. This could be a picture of the back of the phone with the label where the MAC is printed, or an instruction image how to get the MAC from the phone menu.

The rest of the fields are left at their default values, which are set to work with Cisco SPAs. Their meaning is as follows:

- Bootstrap Sync URI: If a stock phone is plugged in for the first time, it needs to be provisioned somehow to let it know where to fetch its configuration file from. Since the stock phone doesn't know about your server, you have to define an HTTP URI here, where the customer is connected with his web browser to set the according field.
- Bootstrap Sync HTTP Method: This setting defines whether an HTTP GET or POST is sent to the Sync URI.
- Bootstrap Sync Params: This setting defines the parameters appended to the Sync URI in case of a GET, or posted in the request body in case of POST, when the customer presses the Sync button later on.

Finally press Save to create the new device model.

sip:wise NGCP Dashboard		*	
Devic Create Device Mod	91		×
		Kellio	
← Back	Add another	Line/Key Ran	d Groups
Front Image	Choose File SPA504-front.jpg 13		
MAC Address Image	Choose File SPA504-back.jpg 14		
Bootstrap Sync URI	http://[% client.ip %]/admin/resync		
Bootstrap Sync HTTP Method	GET		•
No dat Bootstrap Sync Parameters	[% server.uri %]/\$MA		
Showing		45 -	<b>V I</b>
Device		15	Save
Device Configurations			



## 15.1.1.2 Uploading Device Firmwares

A device model can optionally have one or more device firmware(s). Some devices like the Cisco SPA series don't support direct firmware updates from an arbitrary to the latest one, but need to go over specific firmware steps. In the device configuration discussed next, you can return the *next* supported firmware version, if the phone passes the current version in the firmware URL.

Since a stock phone purchased from any shop can have an arbitrary firmware version, we need to upload all firmwares needed to get from any old one to the latest one. In case of the Cisco SPA3x/SPA5x series, that would be the following versions, if the phone starts off with version 7.4.x:

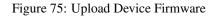
- spa50x-30x-7-5-1a.bin
- spa50x-30x-7-5-2b.bin
- spa50x-30x-7-5-5.bin

So to get an SPA504G with a firmware version 7.4.x to the latest version 7.5.5, we need to upload each firmware file as follows.

Open the Device Firmware row in the Device Management section and press Upload Device Firmware.

Select the device model we're going to upload the firmware for, then specify the firmware version and choose the firmware file, then press *Save*.

Devic <sup>Cr</sup>	reate l	Device Firmv	ware			×
		Device Model			Search:	
			#	Vendor	Model	d Groups
- Back			7	Cisco	SPA504G + 1 Attendant Console	. 1
Device			Showing 1 to 1 of 1 ent	ries	← 1	→ ⇒
Device					Create Devi	ce Model
<b>*</b> U	2	Version	7.5.1a			
Show	3	Firmware File	Choose File spa5	0x-30x-7-5-1a.bin		
# -					4	Save
	3	Firmware File	Choose File spa5	0x-30x-7-5-1a.bin		



Repeat this step for every firmware in the list above (and any new firmware you want to support when it's available).

## 15.1.1.3 Creating Device Configurations

Each customer device needs a configuration file, which defines the URL to perform firmware updates, and most importantly, which defines the subscribers and features configured on each of the lines and keys. Since these settings are different for each physical phone at all the customers, the Cloud PBX module provides a template system to specify the configurations. That way, template variables can be used in the generic configuration, which are filled in by the system individually when a physical device fetches its configuration file.

To upload a configuration template, open the *Device Configuration* row and press *Create Device Configuration*.

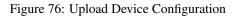
Select the device model and specify a version number for this configuration (it is only for your reference to keep track of different

versions). For Cisco SPA phones, keep the *Content Type* field to text/xml, since the configuration content will be served to the phone as XML file.

For devices other than the Cisco SPA, you might set text/plain if the configuration file is plain text, or application/ octet-stream if the configuration is compiled into some binary form.

Finally paste the configuration	template into the Co	ontent area and press Save.
---------------------------------	----------------------	-----------------------------

evic Cre	eate Device Confi	guration			×	
	Device Model			Search:		
		#	Vendor	Model	d Gr	
Back		7	Cisco	SPA504G + 1 Attendant Console	. 1	Ì
Device		Showing 1 to 1 of 1 entries		← 1		
Device				Create Dev	vice Model	
Device	2 Version	1.0				_
*0	Content Type	text/xml				
Show	3 Content	<flat-profile> [% barge_in = 0 -%] <profile_rule>[% confile</profile_rule></flat-profile>	g.url %]			
#		<upgrade_enable>Yes</upgrade_enable>				



The templates for certified device models are provided by Sipwise, but you can also write your own. The following variables can be used in the template:

- config.url: The URL to the config file, including the device identifier (e.g. http://sip.example.org:1444/dev ice/autoprov/config/001122334455).
- firmware.maxversion: The latest firmware version available on the system for the specific device.
- firmware.baseurl: The base URL to download firmwares (e.g. http://sip.example.org:1444/device/aut oprov/firmware). To fetch the next newer firmware for a Cisco SPA, you can use the template line [% firmware.baseurl %]/\$MA/from/\$SWVER/next.

- phone.stationname: The name of the station (physical device) the customer specifies for this phone. Can be used to show on the display of the phone.
- phone.lineranges: An array of lines/keys as specified for the device model. Each entry in the array has the following keys:
  - name: The name of the line/key range as specified in the Device Model section (e.g. Phone Keys).
  - num\_lines: The number of lines/keys in the line range (e.g. 4 in our Phone Keys example, or 32 in our Attendant Console 1 example).
  - lines: An array of lines (e.g. subscriber definitions) for this line range. Each entry in the array has the following keys:
    - \* keynum: The index of the key in the line range, starting from 0 (e.g. keynum will be 3 for the 4th key of our Phone Keys range).
    - \* rangenum: The index of the line range, starting from 0. The order of line ranges is as you have specified them (e.g. Phone Keys was specified first, so it gets rangenum 0, Auto Attendant 1 gets rangenum 1).
    - \* type: The type of the line/key, on of private, shared or blf.
    - \* username: The SIP username of the line.
    - \* domain: The SIP domain of the line.
    - \* password: The SIP password of the line.
    - \* displayname: The SIP Display Name of the line.

Within the configuration template itself, you can use any Template Toolkit directive and any own variables you like (just make sure to not override any of the ones specified above). For documentation on the syntax, please refer to the Template Toolkit Manual.

#### Тір

In order to change the provisioning base IP and port (default 1444), you have to access /etc/ngcp-config/config. yml and change the value host and port under the autoprov.server section.

# 15.1.1.4 Creating Device Profiles

When the customer configures his own device, he doesn't select a *Device Model* directly, but a *Device Profile*. A device profile specifies which model is going to be used with which configuration version. This allows the operator to create new configuration files and assign them to a profile, while still keeping older configuration files for reference or roll-back scenarios. It also makes it possible to test new firmwares by creating a test device model with the new firmware and a specific configuration, without impacting any existing customer devices.

To create a *Device Profile* for our phone, open the *Device Profile* row in the *Device Management* section and press *Create Device Profile*.

Select the device configuration (which implicitly identifies a device model) and specify a *Profile Name*. This name is what the customer sees when he is selecting a device he wants to provision, so pick a descriptive name which clearly identifies a device. Press *Save* to create the profile.

p:wise l	NGCP Dashboard								1		
evic	Create Device Pro	file								×	
	Device Configuratio	n				Search					
		#		Device Vendor		Device Model		Version			d Groups
Back		7		Cisco		SPA504G + 1 A Console	ttendant	1.0	1		
Device		Sho	wing 1 to 1 of 1	entries			¢	← 1	$\rightarrow$	⇒	
Device							Create	Device Co	nfigurat	ion	
Device	Profile Nam	e Cis	co SPA504G	+ 1 Attendant C	onsole 2						_
*								3	Sa	ave	
Show 5		T	entries				Search:				
# ^	Reseller Name		Device Ven	dor	Device Moo	lel	Configurat	ion Version			
No data a	available in table										



Repeat the steps as needed for every device you want to make available to customers.

#### 15.1.2 Preparing PBX Rewrite Rules

In a PBX environment, the dial-plans usually looks different than for normal SIP subscribers. PBX subscribers should be able to directly dial internal extensions (e.g. 100) instead of the full number to reach another PBX subscriber in the same PBX segment. Therefore, we need to define specific *Rewrite Rules* to make this work.

The PBX dial plans are different from country to country. In the Central European area, you can directly dial an extension (e.g. 100), and if you want to dial an international number like  $0049 \ 1 \ 23456$ , you have to dial a break-out digit first (e.g. 0), so the number to be dialed is  $0 \ 0049 \ 1 \ 23456$ . Other countries are used to other break-out codes (e.g. 9), which then results in  $9 \ 0049 \ 1 \ 23456$ . If you dial a national number like  $01 \ 23456$ , then the number to actually be dialled is  $9 \ 01 \ 23456$ .

Since all numbers must be normalized to E.164 format via inbound rewrite rules, the rules need to be set up accordingly.

Let's assume that the break-out code for the example customers created below is 0, so we have to create a *Rewrite Rule Set* with the following rules.

#### 15.1.2.1 Inbound Rewrite Rules for Caller

- Match Pattern: ^ ([1-9][0-9]+)\$
- Replacement Pattern: \${caller\_cloud\_pbx\_base\_cli}\1
- **Description**: extension to e164
- Direction: Inbound
- Field: Caller

				Logged in as	administrator Logout
ip:wise	NGCP Dashboard			ń	III Settings 🗸
ewrit	e Rules for pb	x-at			
Back vrite rule st	Create Rewrite	e Rule			Expand Groups
Inbound	Rewrite Rules for Caller Match Pattern	Replacement Pattern	Description		
<b>↑</b> ↓	^([1-9][0-9]+)\$	<pre>\${caller_cloud_pbx_base_cli}\1</pre>	extension to e164		
	Rewrite Rules for Callee d Rewrite Rules for Caller				
Outboun	d Rewrite Rules for Callee				



# 15.1.2.2 Inbound Rewrite Rules for Callee

These rules are the most important ones, as they define which number formats the PBX subscribers can dial. For the break-out code of 0, the following rules are necessary e.g. for German dialplans to allow pbx internal extension dialing, local area calls without area codes, national calls with area code, and international calls with country codes.

PBX INTERNAL EXTENSION DIALIN

- Match Pattern: ^ ([1-9][0-9]+)\$
- Replacement Pattern: \${caller\_cloud\_pbx\_base\_cli}\1
- **Description**: extension to e164
- Direction: Inbound
- Field: Callee

LOCAL DIALING WITHOUT AREA CODE (USE BREAK-OUT CODE 0)

- Match Pattern: ^0 ([1-9][0-9]+) \$
- Replacement Pattern: \${caller\_cc}\${caller\_ac}\1
- **Description**: local to e164
- Direction: Inbound
- Field: Callee

National dialing (use break-out code 0 and prefix area code by 0)

- Match Pattern: ^00 ([1-9][0-9]+) \$
- Replacement Pattern: \${caller\_cc}\1
- **Description**: national to e164
- Direction: Inbound
- Field: Callee

International dialing (use break-out code 0 and prefix country code by 00)

- Match Pattern: ^000 ([1-9][0-9]+) \$
- Replacement Pattern: \1
- **Description**: international to e164
- Direction: Inbound
- Field: Callee

				Logged in as administrator Logou
):Wise	NGCP Dashboard			希 🇮 Settings 👻
ewrit	e Rules for pbx	-at		
				** Expand Groups
Back	★ Create Rewrite	Rule		
rite rule su	iccessfully created			
nbound F	Accessfully created Rewrite Rules for Caller Rewrite Rules for Callee			
nbound F	Rewrite Rules for Caller	Replacement Pattern	Description	
nbound F	Rewrite Rules for Caller Rewrite Rules for Callee	Replacement Pattern         \${caller_cloud_pbx_base_cli}\1	Description internal to e164	
nbound F nbound F	Rewrite Rules for Caller Rewrite Rules for Callee Match Pattern			
nbound F nbound F	Rewrite Rules for Caller Rewrite Rules for Callee Match Pattern ^([1-9][0-9]+)\$	<pre>\${caller_cloud_pbx_base_cli}\1</pre>	internal to e164	



# 15.1.2.3 Outbound Rewrite Rules for Caller

When a call goes to a PBX subscriber, it needs to be normalized in a way that it's call-back-able, which means that it needs to have the break-out code prefixed. We create a rule to show the calling number in international format including the break-out code. For PBX-internal calls, the caller name will be shown (this is handled by implicitly setting domain preferences accordingly, so you don't have to worry about that in rewrite rules).

Adding a break-out code (use break-out code 0 and prefix country code by 00)

- Match Pattern: ^ ([1-9][0-9]+)\$
- Replacement Pattern: 000\1
- Description: e164 to full international
- Direction: Outbound
- Field: Caller

#### Displaying the extension in the caller number for $\ensuremath{\mathsf{PBX}}\xspace$ -internal calls

- Match Pattern: ^@{callee\_cloud\_pbx\_account\_cli\_list}\$
- Replacement Pattern: \${caller\_cloud\_pbx\_ext}
- **Description**: e164 to full international
- Direction: Outbound
- Field: Caller

SIP:WISE	NGCP Dashboard	*	<ul> <li>Documentation •</li> </ul>	l Monitoring & Statisti	cs - 🏶	Tools 👻	👥 Settings 🗸
Rewri	te Rules for pbx-at						
							<sup>2</sup> Expand Groups
← Back	★ Create Rewrite Rule						
	Rewrite Rules for Caller						
	Rewrite Rules for Callee						
	Match Pattern	Re	placement Pattern	Description	Enabled		
<b>↑</b> ↓	^@{callee_cloud_pbx_account_cli_list}\$		aller_cloud_pbx_ext}	Intra-PBX to extension	yes		
↑ ↓	^([1-9][0-9]+)\$	000	0\1	e164 to full international	yes		

#### Figure 80: Outbound Rewrite Rule for Caller

Create a new *Rewrite Rule Set* for each dial plan you'd like to support. You can later assign it to customer domains and even to subscribers, if a specific subscriber of a PBX customer would like to have his own dial plan.

#### 15.1.3 Creating Customers and Pilot Subscribers

As with a normal SIP Account, you have to create a *Customer* contract per customer, and one *Subscriber*, which the customer can use to log into the web interface and manage his PBX environment.

## 15.1.3.1 Creating a PBX Customer

Go to *Settings*→*Customers* and click *Create Customer*. We need a *Contact* for the customer, so press *Create Contact*.

SID:MISE	NGCP Dashboard					▲ Logged in		t
Custo	Create Customer						×	
	Contact				Search:			
		#	Reseller	First Name	Last Name	Email		
← Back		1	default			default- customer@default.invali	d =	
Show 5		31		Demo	Reseller	demo@reseller.org		
# ^ Ex		Showing 1	to 2 of 2 entries			← ← 1 → 1 Create Co		
No data a						Cleate Co	ontact	
Showing 01	Billing Profile				Search:			
0		#		Reseller		Profile		
		9		Demo Reseller		Billing Profile for PBX		
		1		default		Default Billing Profile		
		Showing 1	to 2 of 2 entries			⇐ ← 1 →	→ <b>▼</b>	
							Save	
© 2013 Sipw	ise GmbH, all rights reserved.							

Figure 81: Create PBX Customer Part 1

Fill in the desired fields (you need to provide at least the *Email Address*) and press Save.

							🛔 Logg	ed in as a	dministrator Logout
sip:wise	NGCP Da	ashboard						Â	🗰 Settings 👻
Conta	Create C	ontact							×
		Reseller				Search:			
			#	٩	Name	Contract #	Status		
🗧 🗲 Back			1	c	default	1	active		
Show 5			9	C	Demo Reseller	37	active	1	
			Showin	ng 1 to 2 of 2 entries	5		← ← 1	→	⇒
# ^							Create	e Resel	ler
1	2	First Name	John						
31	3	Last Name	Doe						
Showing 1	4	Email	johndo	oe@example.org	1				
		Company							
							5		Save

Figure 82: Create PBX Customer Contact

The new *Contact* will be automatically selected now. Also select a *Billing Profile* you want to use for this customer. If you don't have one defined yet, press *Create Billing Profile*, otherwise select the one you want to use.

	P Dashboard					â	III Settings 👻
usto Cre	ate Customer						×
	Contact				Search:		
		#	Reseller	First Name	Last Name	Email	
Back		33	Demo Reseller	John	Doe	johndoe@example.org <mark>1</mark>	
act suc		1	default			default- customer@default.invalio	d
		31		Demo	Reseller	demo@reseller.org	
w 5 ← Ex		Showing 1	to 3 of 3 entries			⇐ ← 1 →	
						Create Co	ntact
data a	Billing Profile				Search:		
wing 0 t		#		Reseller		Profile	→ [=
		9		Demo Reseller		Billing Profile for PBX 2	
		1		default		Default Billing Profile	• •
							Save



Next, you need to select the *Product* for the PBX customer. Since it's going to be a PBX customer, select the product *Cloud PBX Account*.

Since PBX customers are supposed to manage their subscribers by themselves, they are able to create them via the web interface. To set an upper limit of subscribers a customer can create, define the value in the *Max Subscribers* field.

# Important

As you will see later, both PBX subscribers and PBX groups are normal subscribers, so the value defined here limits the overall amount of subscribers **and** groups. A customer can create an unlimited amount of subscribers if you leave this field empty.

Press Save to create the customer.

					🛎 Log	ged in as	administrator Logout
sip:wise	NGCP Dashboard					A	🗰 Settings 👻
Custo	Create Customer						×
		Showing Flore of the charles			Create Bil		file
🗲 Back	Product			Search:			
Contact suc		#		Name			
		4 5	1	Basic SIP Account Cloud PBX Account			
Show 5		Showing 1 to 2 of 2 entries			← ← 1	→	⇒ -
# ^ Ex	Max Subscribers	20 2					
No data a	Status	active					•
Showing 01	External #						$\rightarrow$
						_	
						3	Save
© 2013 Sipv	wise GmbH, all rights reserved.				_		

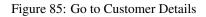


# 15.1.3.2 Creating a PBX Pilot Subscriber

Once the customer is created, you need to create at least one *Subscriber* for the customer, so he can log into the web interface and manage the rest by himself.

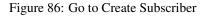
Click the *Details* button on the newly created customer to enter the detailed view.

							💄 Logg	ged in as ac	dministrator	Logout	
	P Dasht	ooard						Â	III Setting	5 🕶	
Customer	S										
← Back Customer #39 succes Show 5		d e Customer				2	Search:				Ξ
# External	Reseller	Contact Email	Product	Billing Profile	Status	Max Number of Subscribers					
39	Demo Reseller	johndoe@example.org	Cloud PBX Account	Billing Profile for PBX	active	20	C Edit ×	Termina	te 🔚 Deta	ails 1	I
Showing 1 to 1 of 1 e	entries							← ←	1 →	⇒	
© 2013 Sipwise Gmb	oH, all rights (	reserved.									



To create the subscriber, open the Subscribers row and click Create Subscriber.

				Logged in	as administrator Logout
		ashboard		•	👬 Settings 👻
	Customer De	etails for #39 (Clo	oud PBX Account)		
	← Back	it			** Expand Groups
	Reseller				
	Contact Details				
_	<b>Billing Profiles</b>				
1	Subscribers				
	0 of maximum 20 su	bscribers (including PBX groups) c	reated		
2	★ Create Sub	oscriber			
	SIP URI	Primary Number	PBX Group	Registered Devices	
	Sound Sets				
	Contract Balance				•



For your pilot subscriber, you need a SIP domain, a pilot number (the main number of the customer PBX), the web credentials for the customer to log into the web interfaces, and the SIP credentials to authenticate via a SIP device.

#### Important



In a PBX environment, customers can create their own subscribers. As a consequence, each PBX customer should have its own SIP domain, in order to not collide with subscribers created by other customers. This is important because two customers are highly likely to create a subscriber (or group, which is also just a subscriber) called office. If they are in the same SIP domain, they'd both have the SIP URI office@pbx.example.org, which is not allowed, and the an end customer will probably not understand why office@pbx.example.org is already taken, because he (for obvious reasons, as it belongs to a different customer) will not see this subscriber in his subscribers list.

### Тір

To handle one domain per customer, you should create a wild-card entry into your DNS server like \*.pbx.example.org, which points to the IP address of pbx.example.org, so you can define SIP domains like customer1.pbx.example.org or customer2.pbx.example.org without having to create a new DNS entry for each of them. For proper secure access to the web interface and to the SIP and XMPP services, you should also obtain a SSL wild-card certificate for \*.pbx.example.org to avoid certification warnings on customers' web browsers and SIP/XMPP clients.

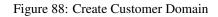
#### So to create a new domain for the customer, click Create Domain.

p:wise NC	GCP Dashboard				ń	🗰 Settings 👻
usto C	reate Subscriber					×
	Domain			Search:		
Deels		#	Reseller	Domain		d Groups
Back		No data available in table				
Reseller		Showing 0 to 0 of 0 entries			$\Leftarrow  \leftarrow  \rightarrow$	⇒
Contact					1 Create Do	main
Billing	E.164 Number					
Subscri	Display Name					
of ma	Web Username					
<b>★</b> C	Web Password					
SIP UR						Save

#### Figure 87: Go to Create Customer Domain

Specify the domain you want to create, and select the PBX Rewrite Rule Set which you created in Section 15.1.2, then click Save.

					🚨 Logge	ed in as a	dministrator Logout
	P Dashboard					â	🗰 Settings 👻
Doma Crea	ate Domain						×
1	SIP Domain	customer1.pbx.example.org					
← Back 2	Rewrite Rule Set	pbx-dialplan					•
Show 5						3	Save
# No data available i	A Reseller	_	ſ	Domain		-	
Showing 0 to 0 of 0 e						Ļ	
	_		_	_			



Finish the subscriber creation by providing an E.164 number, which is going to be the base number for all other subscribers within this customer, the web username and password for the pilot subscriber to log into the web interface, and the sip username and password for a SIP device to connect to the PBX.

The parameters are as follows:

- **Domain**: The domain in which to create the pilot subscriber. Each customer should get his own domain as described above to not collide with SIP usernames between customers.
- E.164 Number: The primary number of the PBX. Calls to this number are routed to the pilot subscriber, and each subsequent subscriber created for this customer will use this number as its base number, suffixed by an individual extension. You can later assign alias numbers also for DID support.
- **Display Name**: This field is used on phones to identify subscribers by their real names instead of their number or extension. On outbound calls, the display name is signalled in the Display-Field of the From header, and it's used as a name in the XMPP contact lists.
- · Web Username: The username for the subscriber to log into the customer self-care web interface. This is optional, if you don't

want a subscriber to have access to the web interface.

- Web Password: The password for the subscriber to log into the customer self-care web interface.
- SIP Username: The username for the subscriber to authenticate on the SIP and XMPP service. It is automatically used for devices, which are auto-provisioned via the *Device Management*, or can be used manually by subscribers to sign into the SIP and XMPP service with any arbitrary clients.
- SIP Password: The password for the subscriber to authenticate on the SIP and XMPP service.

					👗 Logg	ed in as adn	ninistrato	or Logout	
51p:wise	NGCP	Dashboard				A	👥 Set	ttings 👻	
Custo	Creat	e Subscriber					×		
		Domain			Search:				
			#	Reseller	Domain			d Groups	
← Back			21	Demo Reseller	customer1.pbx.example.org	g 🕑	1		Ξ
Domain suc			Showing 1 to 1 of 1 entries		← ← 1	→ =			
Reseller					Creat	te Domain			
Contact	2	E.164 Number	43 1	9999					
Billing	3	Display Name	Administrator						
Subscri	4	Web Username	administrator						
of ma	5	Web Password	adminpassword						
*						Sa			
SIP UR	1	Primary Num	Jor	PBX Group	Registered Devices	Ja	ve		
SIF OK		1 Hinary Num		Tox Group	Register ed Devices				

Figure 89: Create Pilot Subscriber Part 1

ſ							
← Back	Create	Subscriber				×	d Groups
)omain suc		E.164 Number	43 1	9999			
Reseller		Display Name	Administrator				
Contact		Web Username	administrator				
Billing		Web Password	adminpassword				
Subscri	1	SIP Username	administrator				
of ma	2	SIP Password	adminpassword				=
*0		Status	active			T	
SIP UR		External ID					
						3 Save	

#### Figure 90: Create Pilot Subscriber Part 2

Once the subscriber is created, he can log into the customer self-care interface at https://<your-ip>/login/subsc riber and manage his PBX, like creating other users and groups, assigning Devices to subscribers and configure the Auto Attendant and more.

As an administrator, you can also do this for the customer, and we will walk through the typical steps as an administrator to configure the different features.

Go the the *Customer Details* of the PBX customer you want to configure, e.g. by navigating to *Settings* $\rightarrow$ *Customers* and clicking the *Details* button of the customer you want to configure.

# 15.1.4 Creating Regular PBX Subscribers

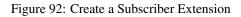
Since we already created a pilot subscriber, more settings now appear on the *Customer Details* view. The sections we are interested in for now are the *Subscribers* and *PBX Groups* sections.

	Logged in as administrator Logout
síp:wise NGCP Dashboard	👚 🎞 Settings 🗸
Customer Details for #39 (Cloud PBX Account)	
← Back	* <sup>a</sup> Expand Groups
Reseller	
Contact Details	
Billing Profiles	
Subscribers	
PBX Groups	
PBX Devices	
Sound Sets	
Contract Balance	
Fraud Limits	

# Figure 91: Subscribers and PBX Groups

To create another subscriber for the customer PBX, open the Subscribers row and click Create Subscriber.

					🚢 Lo	ogged in as a	dministrator Logout	Â
9	ກ່ກະພາຣະ NGCP Dashboard					•	👪 Settings 🔻	
(	Customer Details for #3	9 (Cloud P	BX Acco	ount)				
	Back GEdit					(	* <sup>a</sup> Expand Groups	Ш
	Reseller							
	Contact Details							
	Billing Profiles							
1	Subscribers							
	1 of maximum 20 subscribers (including PB)	(groups) created						
2	★ Create Subscriber							
	SIP URI	Primary Number	PBX Group	Registered Devices				
	administrator@customer1.pbx.example.org	43 1 9999						
								-



When creating another subscriber in the PBX after having the pilot subscriber, some fields are different now, because the *Domain* and *E.164 Number* are already pre-defined at the pilot subscriber level.

What you need to define for a new subscriber is the *Group* the subscriber is supposed to be in. We don't have a group yet, so create one by clicking *Create Group*.

A PBX Group has four settings:

- Name: The name of the group. This is used to identify a group when assigning it to subscribers on one hand, and also subscribers are pushed as server side contact lists to XMPP clients, where they are logically placed into their corresponding groups.
- Extension: The extension of the group, which is appended to the primary number of the pilot subscriber, so you can actually call the group from the outside. If our pilot subscriber number is 43 1 9999 and the extension is 100, you can reach the group from the outside by dialing 43 1 9999 100. Since PBX Groups are actually just normal subscribers in the system, you can assign *Alias Numbers* to it for DID later, e.g. 43 1 9998.
- Hunting Policy: If you call a group, then all members in this group are ringing based on the policy you choose. Serial

Ringing causes each of the subscribers to be tried one after another, until one of them picks up or all subscribers are tried. Parallel Ringing causes all subscribers in the group to be tried in parallel. Note that a subscriber can have a call-forward configured to some external number (e.g. his mobile phone), which will work as well.

• Serial Hunting Timeout: This value defines for how long to ring each member of a group in case of serial hunting until the next subscriber is being tried.

We will only fill in the *Name* and *Extension* for now, as the hunting policy can be changed later if needed. Click *Save* to create the group.

					4	Logged in as a	
sip:ພise NGCP Dashb	oard					ñ	
Custo Create PBX G	iroup						×
	Name ma	arketing					
← Back	tension 10	00 2					d Groups
Hunting Policy		erial Ringing					•
Serial Hunting T	imeout 10	)					
Billing Subscri						3	Save
of maximum 20 subscribers	(including PBX ध्र	groups) created					
★ Create Subscrib	er						
SIP URI		Primary Number	PBX Group	Registered Devices			
administrator@customer1.pb	x.example.org	43 1 9999					



Once the group is created and selected, fill out the rest of the form as needed. Instead of the *E.164 Number*, you can now only choose the *Extension*, which is appended to the primary number of the pilot subscriber and is then used as primary number for this particular subscribers. Again, if your pilot number is  $43\ 1\ 9999$  and you choose extension 101 here, the number of this subscriber is going to be  $43\ 1\ 9999\ 101$ . Also, you can again later assign more alias numbers (e.g.  $43\ 1\ 9997$ ) to this subscriber for DID.

The rest of the fields is as usual, with *Display Name* defining the real name of the user, *Web Username* and *Web Password* allowing the subscriber to log into the customer self-care interface, and the *SIP Username* and *SIP Password* to allow signing into

# the SIP and XMPP services.

sip:wise	NGCP Dashboard						*	🗰 Settings 👻	
Custo	Create Subscriber							×	
_	Group				Search:				
		#	Nam	e	Extension			d Groups	-
🗲 Back		1	marke	eting	100		1 💌		
PBX group s		Showing 1 to 1 of 1 entr	ies			← ← 1	→ C		
Reseller									
Contac	Extension	101 2							
Billing	Display Name	Peter Buck 3							
Subscri	Web Username	peter.buck 4							
of ma	Web Password	somepassword 5							
*0								Save	
SIP UR	શ	Primary Number	PBX Group	Registered Devices					

Figure 94: Finish PBX Subscriber Creation Part 1

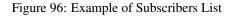
Click Save to create the subscriber.

	to Cale it		× d Group
ack	eate Subscriber		
roup s	Extension	101	<b>^</b>
	Display Name	Peter Buck	
seller	Display Name		
ontact	Web Username	peter.buck	
lling	Web Password	somepassword	
ıbscri	SIP Username	peter.buck 1	
	Sir Ösername		
of ma	SIP Password	somepassword 2	=
<b>★</b> C	Status	active	
	External ID		
IP UR	External ID		
dmini			3 Save

# Figure 95: Finish PBX Subscriber Creation Part 2

Repeat the steps to create all the subscribers and groups as needed. An example of a small company configuration in terms of subscribers and groups might look like this:

eseller									
ontact Details									
illing Profiles									
ubscribers									
7 of maximum 20 subscribers (including PBX groups) created									
★ Create Subscriber									
SIP URI	Primary Number	PBX Group	Registered Devices						
administrator@customer1.pbx.example.org	43 1 9999								
peter.buck@customer1.pbx.example.org	43 1 9999101	marketing							
michelle.miller@customer1.pbx.example.org	43 1 9999102	marketing							
frank.fowler@customer1.pbx.example.org	43 1 9999201	development							
	43 1 9999202	development							



## Тір

The subscribers can be reached via 3 different ways. First, you can call them by their SIP URIs (e.g. by dialing frank. fowler@customer1.pbx.example.org) from both inside and outside the PBX. Second, you can dial by the full number (e.g. 43 1 9999 201; depending on your rewrite rules, you might need to add a leading \+ or 00 or leave out the country code when dialing from the outside, and adding a 0 as break-out digit when dialing from the inside) from both inside and outside the PBX. Third, you can dial just the extension (e.g. 201) from inside the PBX. If the subscriber also has an alias number assigned, you can dial that number also, according to your dial-plan in the rewrite rules.

#### 15.1.5 Assigning Subscribers to Devices

Basically you can register any SIP phone to the system using the SIP credentials of your subscribers. However, the platform supports *Device Provisioning* of certain vendors and models, as described in Section 15.1.1.

To configure a physical device, open the PBX Devices row in the Customer Details view and click Create Device.

You have to set three general parameters for your new device, which are:

- MAC Address/Identifier: The MAC address of the phone to be added. The information can usually either be found on the back of the phone, or in the phone menu itself.
- Station Name: Since you can (depending on the actual device) configure more lines on a phone, you can give it a station name, like Reception or the name of the owner of the device.

In addition to that information, you can configure the lines (subscribers) you want to use on which key, and the mode of operation (e.g. if it's a normal private phone line, or if you want to monitor another subscriber using BLF, or if you want it to act as shared line using SLA).

For example, a *Cisco SPA504G* has 4 keys you can use for private and shared lines as well as BLF on the phone itself, and in our example we have an *Attendant Console* attached to it as well, so you have 32 more keys for BLF.

The settings per key are as follows:

- Subscriber: The subscriber to use (for private/shared lines) or to monitor (for BLF).
- Line/Key: The key where to configure this subscriber to.
- Line/Key Type: The mode of operation for this key, with the following options (depending on which options are enabled in the *Device Model* configuration for this device:
  - **Private Line**: Use the subscriber as a regular SIP phone line. This means that the phone will register the subscriber, and you can place and receive phone calls with/for this subscriber.
  - Shared Line: The subscriber is also registered on the system and you can place and receive calls. If another phone has the same subscriber also configured as shared line, both phones will ring on incoming calls, and you can pick the call up on either of them. You cannot place a call with this subscriber though if the line is already in use by another subscriber. However, you can "steal" a running call by pressing the key where the shared line is configured to barge into a running call. The other party (the other phone where the shared line is configured too) will then be removed from the call (but can steal the call back the same way).
  - BLF Key: The Busy Lamp Field monitors the call state of another subscriber and provides three different functionalities, depending on the actual state:
    - \* Speed Dial: If the monitored subscriber is on-hook, the user can press the button and directly call the monitored subscriber.
    - \* Call Pickup: If the monitored subscriber is ringing, the user can press the button to pick up the call on his own phone.
    - \* State Indication: It the monitored subscriber is on the phone, the key is indicating that the monitored subscriber is currently busy.

In our example, we will first configure a private line on the first key, and BLF for another subscriber on the second key.

				•		dministrator Logout
51p:wise	NGCP	Dashboard			*	🗰 Settings 👻
Custo	Creat	e PBX Device				×
	1	Device Profile	Cisco SI	A504G + 1 Attendant Console		
← Back	2	MAC Address / Identifier	3CCE73	258FD		d Groups
Reseller	3	Station Name	Peter Bu	k		=
Contact		Lines/Keys		Subscriber peter.buck@customer1.pbx.example.org		•
Billing I Subscri				5 Line/Key Phone Keys - Key/Line 0		•
PBX Gro			6	Line/Key Type Private Line		
PBX De					Remov	e
Sound S						<b>T</b>
Contrac Fraud L					s	ave
Fraudi						

# Figure 97: Configuring a PBX Device Part 1

This configures the general options plus the first key. To configure the second key, click *Add another Line/Key* and fill out the second line config accordingly. Click *Save* to save your PBX device configuration.

← Back Crea	te PBX Device						× d Groups
Reseller	Lines/Keys		Subscriber	peter.buck@customer1.pb	x.example.org	•	
Contact			Line/Key	Phone Keys - Key/Line 0		•	
Billing			Line/Key Type	Private Line		Ţ	
Subscri						Remove	
PBX Gro		1	Subscriber	michelle.miller@customer	1 pby oxomplo org		=
Sound S		2	Line/Key	Phone Keys - Key/Line 1	1.pbx.example.org		
Contrac		2		BLF Key			
Fraud L		J	Line/Key Type	DLF Ney		Remove	
							-
						4 Save	

# Figure 98: Configuring a PBX Device Part 2

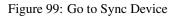
Once the PBX device is saved, you will see it in the list of PBX Devices.

# 15.1.5.1 Synchronizing a PBX Device for Initial Usage

Since a stock device obtained from an arbitrary distributor doesn't know anything about your system, it can't fetch its configuration from there. For that to work, you need to push the URL of where the phone can get the configuration to the phone once.

In order to do so, click the Sync Device button on the device you want to configure for the very first time.

ng Pr	rofiles				
crib	ers				
Grou	ups				
Devi	ices				
<b>☆</b> Cr	eate PB	( Device			
	Station		MAC Address /		
	Name	Subscriber	Identifier	Device Profile	
	Peter Buck	Phone Keys/0: private peter.buck@customer1.pbx.example.org Phone Keys/1: blf michelle.miller@customer1.pbx.example.org	3cce73d258fd	Cisco SPA504G + 1 Attendant Console	X Delete G Edit Sync Device
und Se	ets				
ntract	Balance				
aud Lir	mits				



#### Important

As you will see in the next step, you need the actual IP address of the phone to push the provisioning URL onto it. That implies that you need access to the phone to get the IP, and that your browser is in the same network as the phone in order to be able to connect to it, in case the phone is behind NAT.

Enter the IP Address of the phone (on Cisco SPAs, press Settings 9, where Settings is the paper sheet symbol, and note down the Current IP setting), then click *Push Provisioning URL*.

		🚨 Logg	ed in as	administrator Logout			
51 <b>p:</b> Wise	NGCP D	ashboard				ń	🗰 Settings 👻
Custo	Edit Dep	ployed Device					×
		IP Address 10.15.20.176	1				d Groups
← Back					2 Push Provi	sioning	d Groups
Reseller	_						
Contact	Details						
Billing P	rofiles						
Subscrib	ers						
PBX Gro	ups						
PBX Dev							
× Ci	reate PBX	Device					
	Station Name	Subscriber	MAC Address / Identifier	Device Profile			
n=A	Peter	Phone Keys/0: private	3cce73d258fd	Cisco SPA504G + 1			

Figure 100: Sync Device

You will be redirected directly to the phone, and the Provisioning URL is automatically set. If everything goes right, you will see a confirmation page from the phone that it's going to reboot.

SPA will resync the profile when it is not in use and reboot. You can click <u>here</u> to return to the configuration page.

Figure 101: Device Sync Confirmation from Phone

You can close the browser window/tab and proceed to sync the next subscriber.

#### Tip

You only have to do this step once per phone to tell it the actual provisioning URL, where it can fetch the configuration from. From there, it will regularly sync with the server automatically to check for configuration changes, and applies them automatically.

#### 15.1.6 Configuring Sound Sets for the Customer PBX

In the *Customer Details* view, there is a row *Sound Sets*, where the customer can define his own sound sets for *Auto Attendant*, *Music on Hold* and the *Office Hours Announcement*.

To create a new sound set, open the Sound Sets row and click Create Sound Set.

If you do this as administrator or reseller, the Reseller and/or Customer is pre-selected, so keep it as is. If you do this as customer, you don't see any *Reseller* or *Customer* fields.

So the important settings are:

- Name: The name of the sound set as it will appear in the *Subscriber Preferences*, where you can assign the sound set to a subscriber.
- · Description: A more detailed description of the sound set.
- **Default for Subscribers**: If this setting is enabled, then the sound set is automatically assigned to all already existing subscribers which do NOT have a sound set assigned yet, and also for all newly created subscribers.

Fill in the settings and click Save.

sip:wise	NGCP Dashboard					ñ	🗰 Settings 👻
Soun	Create Sound Sets						×
						Create Resell	er
← Back	Customer		<b>.</b>	Sear			
Show 5		# 39	Reseller Demo Reseller	Contact Email johndoe@example.org	External #	Status active	
#		Showing 1 to 1	of 1 entries		=	← 1 →	-
" No data a						Create Contra	ct
Showing 01	Name	mypbxset	1				
	Description	My Default S	ound Set 2				
	Default for Subscribers	☑ 3					
							<b>_</b>
						4	Save
© 2013 Sipw	vise GmbH, <b>all rights reserved</b> .						

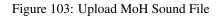


To upload files to your Sound Set, click the Files button for the Sound Set.

#### 15.1.6.1 Uploading a Music-on-Hold File

Open the *music\_on\_hold* row and click *Upload* on the *music\_on\_hold* entry. Choose a WAV file from your file system, and click the *Loopplay* setting if you want to play the file in a loop instead of just once. Click *Save* to upload the file.

	Logged in as administrator Logout
síp:wise NGCP Dashboard	👚 🎞 Settings 🗸
Mana Edit music_on_hold	×
1 Loopplay 🗹	
← Ba 2 Soundfile Choose File manolo_campcoffee.w	d Groups
music_c	3 Save
pbx	



#### 15.1.7 Configuring Auto Attendant

The *Auto Attendant* is a built-in IVR feature that is available to Cloud PBX subscribers. It provides an automatic voice menu that enables the caller to select from a number of destinations—which are other PBX subscribers or groups—when the called subscriber is not available.

The Auto Attendant feature can be activated for any subscriber in the Customer PBX individually. There are three steps involved:

- 1. You have to prepare a *Sound Set* to have Auto Attendant sound files.
- 2. You have to configure the destinations for the various options you provide (e.g. pressing 1 should go to the marketing subscriber, 2 to development and 3 to some external number).
- 3. You have to set a Call Forward to the Auto Attendant.

To do so, go to *Customer Details* and in the *Subscribers* section, click the *Preferences* button of the subscriber, where the Auto Attendant should be set.

# 15.1.7.1 Preparing the Sound Set

Create a Sound Set and upload the Sound Files for it as described below. Afterwards in the *Subscriber Preferences* view, set the *Customer Sound Set* preference to the Sound Set to be used. To do so, click *Edit* on the *Customer Sound Set* preference and assign the set to be used.

## Uploading Auto-Attendant Sound Files

When configuring a Call Forward to the Auto Attendant, it will play the following files:

- aa\_welcome: This is the welcome message (the greeting) which is played when someone calls the Auto Attendant.
- each available pair of aa\_X\_for/aa\_X\_option: Each menu item in the Auto Attendant consists of two parts. The for part, which plays something like *Press One for*, and the option part, which play something like *Marketing*. The Auto Attendant only plays those menu options where both the for part and the option part is present, so if you only have 3 destinations you'd like to offer, and you want them to be on keys 1, 2 and 3, you have to upload files for aa\_1\_for, aa\_1\_option, aa\_2\_for, aa\_2\_option and aa\_3\_for and aa\_3\_option.



# Important

The sound files only define the general structure of what is being played to the caller. The actual destinations behind your options are configured separately in Configuring the Auto Attendant Slots Section 15.1.7.3.

An example configuration could look like this:

← Back			* Expand Groups
ind handle successfully	y uploaded		
music_on_hold			
pbx Name	Filename	Loop	
aa_welcome	welcome.wav		
aa_1_for	press-1.wav		
aa_1_option	for-sales.wav		
aa_2_for	press-2.wav		
aa_2_option	for-service.wav		
aa_3_for	press-3.wav		
aa_3_option	for-tech-support.wav		
aa_4_for			
aa_4_option			
aa_5_for			

# Figure 104: Upload Auto Attendant Sound File

# 15.1.7.2 Auto Attendant Flowchart with Voice Prompts

The illustration below shows the sequence of voice prompts played when Auto Attendant feature is activated and a caller listens the IVR menu.

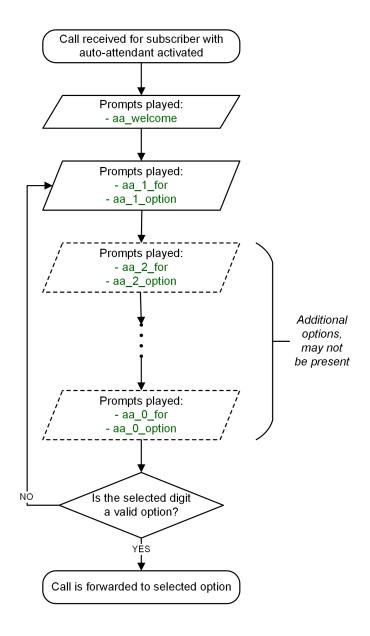


Figure 105: Flowchart of Auto Attendant

# 15.1.7.3 Configuring the Auto Attendant Slots

In the Auto Attendant Slots section, click the Edit Slots button to configure the destination options.

Click Add another Slot to add a destination option, select the Key the destination should be assigned to, and enter a Destination. The destination can be a subscriber username (e.g. marketing), a full SIP URI (e.g. sip:michelle.miller@custom er1.pbx.example.org or any external SIP URI) or a number or extension (e.g. 491234567 or 101).

Repeat the step for every option you want to add, then press Save.

					👗 Lo	ogged in as a	dministrator Logout
sip:wise	NGCP Dashboard					â	🗰 Settings 👻
Subso	Edit Auto Attendant Slot						×
	IVR Slots	Key	1	1			d Groups
← Back		Destination	marketing	2		Domo	
Call For Voicem						Remo	
Fax2Ma		Key Destination	2 491234567				=
Auto Al						Remo	ve
Show					3 Add a	nother Sl	ot
# No dat						4	Save
Showing	0 to 0 of 0 entries					-	

Figure 106: Define the Auto Attendant Slots

#### 15.1.7.4 Activating the Auto Attendant

Once the Sound Set and the Slots are configured, activate the Auto Attendant by setting a Call Forward to Auto Attendant.

To do so, open the *Call Forwards* section in the *Subscriber Preferences* view and press *Edit* on the Call Forward type (e.g. *Call Forward Unconditional* if you want to redirect callers unconditionally to the Auto Attendant).

Select Auto Attendant and click Save to activate the Auto Attendant.

			Logged in as administrator Logout
5ip:wise	NGCP Dashboard		🖌 🏭 Settings 👻
Subso	Edit Call Forward U	conditional	×
← Back Call For	Destination	<ul> <li>Voicemail</li> <li>Conference</li> <li>Fax2Mail</li> <li>Auto Attendant</li> <li>Office Hours Announcement</li> <li>URI/Number</li> </ul>	d Groups
Type Call Fo Call Fo Call Fo	URI/Number for (seconds)	300	2 Advanced View Save
	ail and Voicebox il and Sendfax		
Auto At	tendant Slots		



# Тір

As with any other Call Forward, you can define more complex forwarding rules in the *Advanced View* to only forward the call to the Auto Attendant during specific time periods, or as a fallback if no one picks up the office number.

## 15.1.8 Configuring Call Queues

The sip:provider PRO platform offers call queueing feature for Cloud PBX subscribers. For any subscriber within the PBX the NGCP system administrator or the subscriber himsef may activate the *Call Queue*. This is done individually for each subscriber on demand.

If call queue activation has been done and the subscriber receives more than 1 call at a time, then the second and all further callers will be queued until the subscriber finishes his call with the first caller and gets free.

# 15.1.8.1 Activating the Call Queue

The call queue configuration is available at the path: Subscribers  $\rightarrow$  select one  $\rightarrow$  Details  $\rightarrow$  Preferences  $\rightarrow$  Cloud PBX.

Following configuration parameters may be set for call queueing:

- cloud\_pbx\_callqueue : shows the status of call queueing (enabled / disabled); by default it is disabled
- max\_queue\_length : the length of call queue, i.e. the maximum number of callers in a queue; the default is 5
- queue\_wrap\_up\_time: the delay in seconds between the ending of the previous call and the connection of the next queued
  caller with the subscriber; the default is 10

In order to change the actual setting, press the Edit button in the relevant row.

Cloud PBX					
	Attribute	Name	Value		
0	enable_t38	Enable T38 Fax-over-IP			
0	cloud_pbx_callqueue	PBX Call Queue			
0	max_queue_length	Call Queue length			
0	queue_wrap_up_time	Call Queue wrap-up time, sec			

# Figure 108: Call Queue Configuration

# 15.1.8.2 Call Queue Voice Prompts

Queued callers first hear a greeting message then information about their position in the queue and finally a waiting music / signal.

#### Table 22: Call Queue Voice Prompts

Prompt handle	Prompt content
queue_greeting	All lines are busy at the moment, you are being queued.
queue_prefix	You are currently number
queue_suffix	in the queue, please hold the line.
queue_full	All lines are busy at the moment, please try again later.
queue_waiting_music	<waiting music=""></waiting>

### 15.1.8.3 Call Queue Flowchart with Voice Prompts

The following illustration shows which voice prompts are played to the caller when the call gets into a queue.

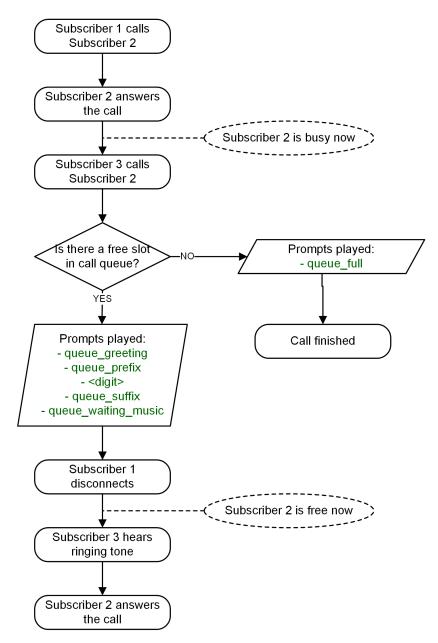


Figure 109: Flowchart of Call Queue

#### 15.1.9 Device Auto-Provisioning Security

### 15.1.9.1 Server Certificate Authentication

The Cisco SPA phones can connect to the provisioning interface of the PBX via HTTP and HTTPS. When perform secure provisioning over HTTPS, the phones validate the server certificate to check if its a legitimate Cisco provisioning server. To pass this check, the provisioning interface must provide a certificate signed by Cisco for that exact purpose.

#### The following steps describe how to obtain such a certificate.

#### First, a new SSL key needs to be generated:

```
$ openssl genrsa -out provisioning.key 2048
Generating RSA private key, 2048 bit long modulus
...+++
e is 65537 (0x10001)
```

Next, a certificate signing request needs to be generated as follows. Provide your company details.

#### Important

The **Common Name (e.g. server FQDN or YOUR name)** field is crucial here. Provide an FQDN which the phones will later use via DNS to connect to the provisioning interface, for example *pbx.example.org*. Cisco does **NOT** support wild-card certificates.



#### Important

Leave the password empty when asked for it (press Enter without entering anything).

```
$ openssl req -new -key provisioning.key -out provisioning.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
```

```
Country Name (2 letter code) [AU]:AT
State or Province Name (full name) [Some-State]:Vienna
Locality Name (eg, city) []:Vienna
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Sipwise GmbH
Organizational Unit Name (eg, section) []:Operations
Common Name (e.g. server FQDN or YOUR name) []:pbx.example.org
Email Address []:office@sipwise.com
```

```
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Finally, compress the provisioning.csr file via ZIP and send it to our Cisco sales representative. If in doubt, you can try to send it directly to ciscosb-certadmin@cisco.com asking them to sign it.



# Important

Only send the CSR file. Do NOT send the key file, as this is your private key!

## ) Important

Ask for both the signed certificate AND a so-called *combinedca.crt* which is needed to perform client authentication via SSL. Otherwise you can not restrict access to Cisco SPAs only.

You will receive a signed CRT file, which Sipwise can use to configure the PBX provisioning interface.

### 15.1.9.2 Client Certificate Authentication

If a client connects via HTTPS, the server also checks for the client certificate in order to validate that the device requesting the configuration is indeed a legitimate Cisco phone, and not a fraudulent user with a browser trying to fetch user credentials.

### 15.1.10 Device Bootstrap and Resync Workflows

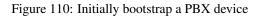
The IP phones supported by the PBX need to initially be configured to fetch their configuration from the system. Since the phones have no initial information about the system and its provisioning URL, they need to be boot-strapped. Furthermore, changes for a specific device might have to be pushed to the device immediately instead of waiting for it to re-fetch the configuration automatically.

The following sections describe the work-flows how this is accomplished without having the customer directly accessing the phone.

#### 15.1.10.1 Cisco SPA Device Bootstrap

Initial Bootstrapping

Customer IP		hone PBX	Panel
	press "Sync Device" button		*
	<	200, serve "IP address" field	
	enter phone IP and submit		*
	•	301, redirect to phone URL	
	automatically push sync URL		
	200, confirm sync ≺		
		fetch device configuration	>
		200, serve device configuration	



### Subsequent Device Resyncs

If one of the subscribers configured on a PBX device is registered via SIP, the system can trigger a re-sync of the phone directly via SIP without having the customer enter the IP of the phone again. This is accomplished by sending a special NOTIFY message to the subscriber:

```
NOTIFY sip:subscriber@domain SIP/2.0
To: <sip:subscriber@domain>
From: <sip:subscriber@domain>;tag=some-random-tag
```

Call-ID: some-random-call-id CSeq: 1 NOTIFY Subscription-State: active Event: check-sync Content-Length: 0

In order to prevent unauthorized re-syncs, the IP phone challenges the request with its own SIP credentials, so the NOTIFY is sent twice, once without authentication, and the second time with the subscriber's own SIP credentials.

Cust	omer IP	Phone	рвх р	Panel	SIP Proxy
	press "Sync Device" button				
	<	200 if regist	ered		
				NOTIFY private subscriber	
				NOTIFY private subsc	riber
		401 Unauthorized			
				401 Unauthor	ized
				NOTIFY private subscriber	
				NOTIFY private subsc	riber
		200 OK			
				200	оок
		fetch device configuration		•	
		200, serve device configura	ation		

Figure 111: Resync a registered PBX device

## 15.1.10.2 Panasonic Device Bootstrap

#### Initial Bootstrapping

Panasonic provides a zero-touch provisioning mechanism in their firmwares, which causes the factory-reset phones to connect to a Panasonic web service at *https://provisioning.e-connecting.net* to check if a custom provisioning URL is configured for the MAC address of the phone. If an association between the MAC and a provisioning URL is found, the web service redirects the phone to the provisioning URL, where the phone connects to in order to obtain the configuration file.

Server

Customer IP Ph		Phone	PBX F	Panel	Panasonic Re	direct
	create PBX device			create MAC/Prov-l	URL association	
	•		200	<	200	
	plug-in factory reset phone	fetch device association				
				301 redired	t to CloudPBX	
		fetch device configuration				
		200, serve device configu	ration			

Figure 112: Initially bootstrap a Panasonic phone

The CloudPBX module ensures that when an end customer creates a Panasonic device, the MAC address is automatically provisioned on the Panasonic web service via an API call, so the customer's phone can use the correct provisioning URL to connect to the auto-provisioning server of the CloudPBX.

As a result, no customer interaction is required to bootstrap Panasonic phones, other than just creating the phone with the proper MAC on the CloudPBX web interface.

## Factory Reset

For already provisioned phones, the end customer might need to perform a factory reset:

- Press Settings or Setup
- Enter #136
- Select Factory Setting and press Enter
- · Select Yes and press Enter
- · Select Yes and press Enter

The default username for factory-reset phones is admin with password adminpass.

#### Subsequent Device Resyncs

The same procedure as with Cisco SPA phones applies, once a subscriber configured on the phone is registered.

#### 15.1.10.3 Yealink Device Bootstrap

#### Initial Bootstrapping

Yealink provides a zero-touch provisioning mechanism in their firmwares, which causes the factory-reset phones to connect to a Yealink web service at *https://rps.yealink.com* to check if a custom provisioning URL is configured for the MAC address of the phone. If an association between the MAC and a provisioning URL is found, the web service redirects the phone to the provisioning URL, where the phone connects to in order to obtain the configuration file.

If both Cisco SPA and Yealink phones are used, an issue with the Cisco-signed server certificate configured on the provisioning port (1444 by default) of the CloudPBX provisioning server arises. Yealink phones by default only connect to trusted server certificates, and the Cisco CA certificate used to sign the server certificate is not trusted by Yealink. Therefore, a two-step approach is used to disable the trusted check via a plain insecure http port (1445 by default) first, where only device-generic config options are served. No user credentials are provided in this case, because no SSL client authentication can be performed. The generic configuration disables the trusted check, and at the same time changes the provisioning URL to the secure port, where the Yealink phone is now able to connect to.

Customer IP Ph		hone PBX F		Panel	Yealink Redire	Yealink Redirect Server	
	create PBX device		,	. create MAC/Prov-	URL association		
			200	<b>~</b>	200		
	plug-in factory reset phone	fetch device association					
				301 redire	ct to CloudPBX		
		fetch bootstrap configurati via http on port 1445	on				
		200, serve generic configur	ation				
		fetch device configuration via https on port 1444	,				
		200, serve device configur	ation				

Figure 113: Initially bootstrap a Yealink phone

The CloudPBX module ensures that when an end customer creates a Yealink device, the MAC address is automatically provisioned on the Yealink web service via an API call, so the customer's phone can use the correct insecure bootstrap provisioning URL to connect to the auto-provisioning server of the CloudPBX for the generic configuration, which in turn provides the information on where to connect to for the secure, full configuration.

As a result, no customer interaction is required to bootstrap Yealink phones, other than just creating the phone with the proper MAC on the CloudPBX web interface.

## Factory Enable Yealink Auto-Provisioning

Older Yealink firmwares don't automatically connect to the Yealink auto-provisioning server on initial boot, so it needs to be enabled manually by the end customer.

- Log in to http://phone-ip/servlet?p=hidden&q=load using admin and admin as user/password when prompted
- Change Redirect Active to Enabled
- Press Confirm and power-cycle phone

#### Subsequent Device Resyncs

The same procedure as with Cisco SPA phones applies, once a subscriber configured on the phone is registered.

#### 15.1.10.4 Audiocodes Mediant Device Bootstrap and Configuration

#### Initial Bootstrapping

An Audiocodes device provides a zero-touch provisioning mechanism in its firmware which causes a factory-reset device to connect to the URL built into the firmware. This URL is pointing to the NGCP provisioning server (in case of NGCP Carrier: web01 node) listening on TCP port 1444 for HTTPS sessions.

The prerequisites for the device provisioning are that the device has a routable IP address and can reach the IP address of the NGCP provisioning interface.

The Audiocodes device should request the firmware file or CLI configuration file from the NGCP platform. The firmware versions and CLI config versions are decoupled from each other; the NGCP can not enforce specific version of the firmware on the device. Instead, it should be requested by the device itself. In other words, provisioning is a *pull* and not a *push* process.

NGCP expects the provisioning request from the Audiocodes device after SSL handshake and serves the requested file to the device if the device provides valid MAC address as the part of the URL. The MAC address is used to identify the device to the NGCP platform. The firmware and CLI config files are provided at the following URLs:

- the base URL to download firmwares: https://<NGCP\_IP>:1444/device/autoprov/firmware/001122334 455/from/0/latest
- the base URL to download CLI config: https://<NGCP\_IP>:1444/device/autoprov/config/001122334455

where 001122334455 should be replaced with the actual device's MAC address and <NGCP\_IP> with IP address of the NGCP provisioning interface.

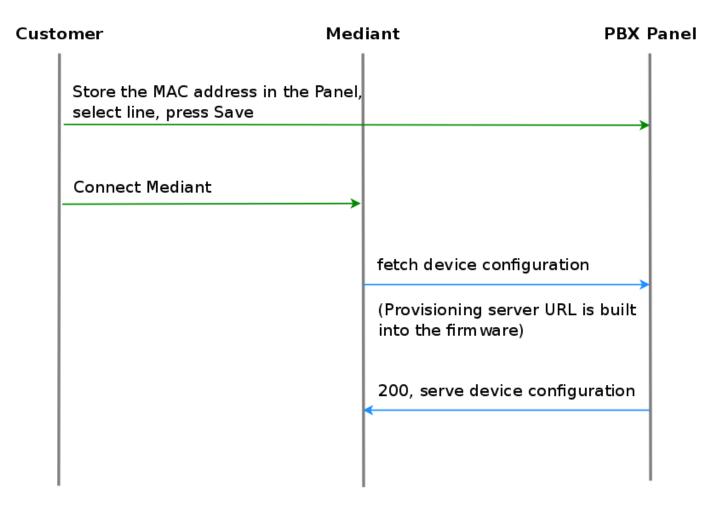


Figure 114: Initially bootstrap a Mediant gateway

#### Device management basics

The list of device models, firmwares and configurations are global to a reseller and are available for end customer. This data is initially provided by Sipwise as bulk upload of all supported phone models. The firmwares and settings are stored in the database on the DB node pair(s). The NGCP leverages the Cloud PBX module with its template system to generate the configurations and firmware files from database on the fly. Please refer to the following chapters in NGCP handbook for the current information on how to perform device management:

- Uploading device firmwares Section 15.1.1.2
- Creating device configuration Section 15.1.1.3
- Creating device profiles Section 15.1.1.4

#### Parameterizing the Device Configuration Template

The device-specific parameters are filled in by the system individually when a physical device fetches its configuration file. Parameters from the NGCP panel:

- <code>username</code> : Subscriber Details  $\rightarrow$  Master Data  $\rightarrow$  SIP Username

- <code>password: Subscriber Details  $\rightarrow$  Master Data  $\rightarrow$  SIP Password</code>
- domain: Subscriber Details  $\rightarrow$  Master Data  $\rightarrow$  Domain
- extension: Subscriber Details  $\rightarrow$  Master Data  $\rightarrow$  Extension
- area code: Subscriber Preferences  $\rightarrow$  Number Manipulations  $\rightarrow$  ac
- country code: Subscriber Preferences  $\rightarrow$  Number Manipulations  $\rightarrow$  cc

The produced **CLI config file** has the following structure:

```
1. SIP account credentials:
```

"sip-definition account 0"

- user-name [username]
- password [password]
- host-name [domain]
- · register reg
- contact-user "[country code][area code][extension]"
- 2. IP Groups:

"voip-network ip-group 1" and "voip-network ip-group 2"

- sip-group-name [domain]
- 3. Proxy and registration settings:

```
"sip-definition proxy-and-registration"
```

```
• set gw-name [domain]
```

- 4. Manipulations:
  - manipulation-name "from trunk domain":
    - "sbc manipulations message-manipulations 3"
    - action-value "[% line.domain %]"
  - manipulation-name "clip no screening":

```
"sbc manipulations message-manipulations 8"
```

- action-value "' < sip:+[country code][area code][extension]@' + param.ipg.dst.host + '
>' "

Specific CLI parameters are:

- [IPPBX\_Hostname]
- [IPPBX\_server\_IP]

which are used at the following configuration parameters:

· Proxy settings:

```
"voip-network proxy-ip 1"
```

- proxy-address [IPPBX\_Hostname]
- · Manipulations:

```
"sbc manipulations message-manipulations 1"
```

- action-value [IPPBX\_Hostname]

#### 15.1.11 Device Provisioning and Deployment Workflows

This chapter provides information and hints for preparing and performing the deployment of certain VoIP devices at customer sites, that have a customer-facing interface which also needs customisation.

#### 15.1.11.1 Audiocodes Mediant Device Provisioning Workflow

Audiocodes ISDN gateways and eSBCs are devices used to connect legacy (ISDN) PBX and IP-PBX to the Sipwise NGCP platform and maintain their operations within the Operator's network. Sipwise NGCP offers a *SipConnect 1.1* compliant signaling and media interface to connect SIP trunks to the platform. In addition to this interface, the Sipwise NGCP provides an auto-provisioning mechanism to configure SIP endpoints like IP phones, media gateways and eSBCs.

### **Provisioning URL**

An Audiocodes device needs to obtain the provisioning URL of the Sipwise NGCP in one way or the other to request its device configuration and subsequently download specific firmwares, obtain SIP credentials to connect to the network facing side, and configure the customer facing side for customer devices to connect either via ISDN or SIP. Typical ways of obtaining the provisioning URL for a SIP endpoint are:

- using DHCP option-66 (in a pre-staging environment or directly at the customer premise) where vendor-specific Redirect Servers
  are configured in the default configuration or firmware
- · getting pre-configured per deployment from the SIP endpoint vendor
- · getting pre-configured per deployment by a 3rd party distributor

The assumption is that Audiocodes devices are supplied with a firmware (and all required SSL certificates) being pre-configured and the provisioning URL pointing to an Operator URL the Sipwise NGCP is serving, before handing the devices over to field service engineers doing the truck rolls.

### Field Configuration

The Sipwise NGCP provides a SipConnect 1.1 compliant interface on the network side for the Audiocodes devices. This interface clearly defines the numbering formats of the calling and called party, the SIP header mechanisms to provide CLI restriction, the RTP codecs, etc.

On the customer facing side, however, those variables might be different from deployment to deployment:

- An IP-PBX might choose to only send its extension as calling party number, or might choose to send the full number in national format.
- It might choose to use the SIP From-header mechanisms to suppress displaying of the CLI, or use the SIP Privacy header.
- The same uncertainty exists to some extent for a legacy PBX connecting via ISDN to the Audiocodes device.

The assumption here is that a field service engineer is NOT supposed to change the Audiocodes configuration in order to make the customer interface work, as this will lead to big issues in maintaining those local changes, especially if a replacement of the device is necessary. Instead, the Audiocodes configuration must ensure that all different kinds of variants in terms of SIP headers, codecs and number formats are translated correctly to the network side and vice versa. If it turns out that there are scenarios in the field which are not handled correctly, temporary local changes might be performed to finish a truck roll, but those changes MUST be communicated to the platform operator, and the server-side configuration templates must be adapted to handle those scenarios gracefully as well.

For deployments with ISDN interfaces on the customer facing side of the Audiocodes, different *Device Profiles* with specific *Device Configurations* per *Device Model* must exist to handle certain scenarios, specifically whether the ISDN interface is operating in Point-to-Point or Point-to-Multipoint mode. Configuration options like which side is providing the clock-rate are to be defined up-front, and the PBX must be reconfigured to adhere to the configuration.

#### Network Configuration

On the network facing side, both the ISDN and eSBC style deployments have to be designed to obtain an IP address via DHCP. The definition of the IP address ranges is up to the Operator. It may or may not be NAT-ed, but it is advised to use a private IP range directly routed in the back-bone to avoid NAT.

On the customer facing side, networking is only relevant for the eSBC deployment. In order to make the IP-PBX configuration as stream-lined as possible, a pre-defined network should be established on the customer interface of the Audiocodes device.

### Тір

The proposal is to define a network 192.168.255.0/24 with the Audiocodes device using the IP 192.168.255.2 (leaving the 192.168.255.1 to a possible gateway). The IP-PBX could obtain its IP address via DHCP from a DHCP server running on the Audiocodes device (e.g. serving IP addresses in the range of 192.168.255.100-254), or could have it configured manually (e.g. in the range of 192.168.255.3-99). Since the Audiocodes device IP on the customer side is always fixed at 192.168.255.2, the IP-PBX for each customer can be configured the same way, pointing the SIP proxy/registrar or outbound proxy always to this IP.

The customer facing side is outside the Sipwise demarcation line, that's why the network configuration mentioned above only serves as proposal and any feedback is highly welcome. However, it must be clearly communicated how the customer facing

network is going to be configured, because the Sipwise NGCP needs to incorporate this configuration into the Audiocodes configuration templates.

### 15.1.11.2 Audiocodes Mediant Device Deployment Workflow

#### Pre-Configuration on Sipwise NGCP platform

- 1. Before connecting a customer to a SIP trunk, it must be clear which Audiocodes *Device Model* is going to be used (depending on if, which and how many ISDN ports are necessary) and which *Device Profile* for the *Device Model* is required (eSBC mode, ISDN P-to-P or P-to-MP mode). Based on that, the correct physical device must be picked.
- 2. Next, the customer has to be created on the Sipwise NGCP. This step requires the creation of the customer, and the creation of a subscriber within this customer. For the subscriber, the proper E.164 numbers or number blocks must be assigned, and the correct subscriber preferences must be set for the network interface to adhere to the SipConnect 1.1 interface. This step is automated by a script provided by Sipwise until the provisioning work-flow is fully integrated with Operator's OSS/BSS systems. *Required parameters are:* 
  - an external customer id to relate the customer entity on the Sipwise NGCP with a customer identifier in Operator's IT systems
  - · a billing profile name
  - · a subscriber username and password, the domain the subscriber is configured for
  - the numbers or number blocks assigned to the subscriber, and the network provided number of the subscriber
  - optional information is geographic location information and IP network information to properly map emergency calls
- 3. Finally, the association between the MAC address of the Audiocodes device and the SIP subscriber to be used on the SIP trunk must be established. This step is also automated by a script provided by Sipwise. *Required parameters are:* 
  - · the subscriber id
  - the Device Profile to be used
  - · and the MAC address of the Audiocodes device

#### Installation

Once the above requirements are fulfilled and the customer is created on the Sipwise NGCP, the Audiocodes device can be installed at the customer premise.

When the Audiocodes device boots, it requests the configuration file from the Sipwise NGCP by issuing a GET request via HTTPS.

For **authentication and authorization** purposes, the Sipwise NGCP requests an SSL client certificate from the device and will check whether it's signed by a Certificate Authority known to the Sipwise NGCP. Therefore, Audiocodes must provide the CA certificate used to sign the devices' client certificates to Sipwise to allow for this process. Also, the Sipwise NGCP will provide an SSL server certificate to the device. The device must validate this certificate in order to prevent man-in-the-middle attacks. Options here are to have:

• Sipwise provide a self-signed certificate to Audiocodes for Audiocodes or a 3rd party distribution partner to configure it as trusted CA in the pre-staging process

• the Operator provide a certificate signed by a CA which is already in the trust store of the Audiocodes devices.

Once the secured HTTPS connection is established, the Sipwise NGCP will provide a CLI style configuration file, with its content depending on the pre-configured *Device Profile* and subscriber association to the device's MAC address.

The configuration includes the firmware version of the latest available firmware configured for the *Device Model*, and a URL defining from where to obtain it. The configuration details on how the Audiocodes devices manage the scheduling of firmware updates are to be provided by Audiocodes or its partners, since this is out of scope for Sipwise. Ideally, firmware updates should only be performed if the device is idle (no calls running), and within a specific time-frame (e.g. between 1 a.m. and 5 a.m. once a certain firmware version is reached, including some random variation to prevent all devices to download a new firmware version at the same time).

#### **Device Replacement**

If a customer requires the replacement of a device, e.g. due to hardware issues or due to changing the number or type of ISDN interfaces, a new association of the new device MAC, its *Device Profile* and the subscriber must be established.

In order to make the change as seamless as possible for the customer, a new device is created for the customer with the new MAC, a proper *Device Profile*, but the same subscriber as used on the old device. Once the new device boots at the customer premise, it will obtain its configuration and will register with the same subscriber as the old device (in case it's still operational). For inbound calls to the customer, this will cause parallel ringing to take place, and it's up to the customer or the field engineer when to re-configure or re-cable the PBX to connect to one or the other device.

Once the old device is decommissioned, the old MAC association can be deleted on the Sipwise NGCP.

15.1.12	List of available pre-configured devices
---------	--

Vendor	Model	Available from release
Audiocodes	Mediant800	mr4.1.1.1
Cisco	ATA112	mr3.4.1.1
Cisco	ATA122	mr3.4.1.1
Cisco	SPA232D	mr3.4.1.1
Cisco	SPA301	mr3.4.1.1
Cisco	SPA303	mr3.4.1.1
Cisco	SPA501G	mr3.4.1.1
Cisco	SPA502G	mr3.4.1.1
Cisco	SPA512G	mr3.4.1.1
Cisco	SPA504G	mr3.4.1.1
Cisco	SPA504G + SPA500S	mr3.7.1.4
Cisco	SPA504G + two SPA500S	mr3.7.1.4
Cisco	SPA514G	mr3.4.1.1
Cisco	SPA508G	mr3.4.1.1
Cisco	SPA509G	mr3.4.1.1
Cisco	SPA525G	mr3.4.1.1
Innovaphone	IP2X2X	mr3.8.3.3

Vendor	Model	Available from release
Innovaphone	IP230-X	mr3.8.3.3
Innovaphone	IP232	mr3.8.3.3
Innovaphone	IP222	mr3.8.3.3
Innovaphone	IP240	mr3.8.3.3
Innovaphone	IP22	mr3.8.3.3
Innovaphone	IP111	mr3.8.3.3
Panasonic	KX-UT113	mr3.7.1.1
Panasonic	KX-UT123	mr3.7.1.1
Panasonic	KX-UT133	mr3.7.1.1
Panasonic	KX-UT136	mr3.7.1.1
Panasonic	KX-UT248	mr3.7.1.1
Yealink	SIP-T19P	mr3.7.1.1
Yealink	SIP-T20P	mr3.7.1.1
Yealink	SIP-T21P	mr3.7.1.1
Yealink	SIP-T22P	mr3.7.1.1
Yealink	SIP-T23P	mr3.7.1.1
Yealink	SIP-T23G	mr3.7.1.1
Yealink	SIP-T26P	mr3.7.1.1
Yealink	SIP-T28P	mr3.7.1.1
Yealink	SIP-T32G	mr3.7.1.1
Yealink	SIP-T38G	mr3.7.1.1
Yealink	SIP-T41P	mr3.7.1.1
Yealink	SIP-T42G	mr3.7.1.1
Yealink	SIP-T46G	mr3.7.1.1
Yealink	SIP-T48G	mr3.7.1.1
Yealink	SIP-T28P + EXP39	mr3.8.1.1
Yealink	SIP-T28P + two EXP39	mr3.8.1.1
Yealink	W52P	mr3.7.1.6

## 15.1.12.1 Cisco Devices

## IP Phones

Model	IPv6	IPv6 TLS		Auto	Private	Shared	Busy	Extension
		1123	Shir	provisioning	Line	Line	Lamp	Boards
SPA301	N	Y	Y	http	1	1	0	Ν
SPA303	N	Y	Y	http	1-3	1-3	1-2	N
SPA501G	N	Y	Y	http	1-8	1-8	1-7	N
SPA502G	N	Y	Y	http	1	1	0	N
SPA512G	N	N	Y	http	1	1	0	N
SPA504G	N	Y	Y	http	1-4	1-4	1-3	2
SPA514G	N	N	Y	http	1-4	1-4	1-3	N
SPA508G	N	Y	Y	http	1-8	1-8	1-7	N

Model	IPv6	TLS	SRTP	Auto	Private	Shared	Busy	Extension
	IFVO	IL5	SHIP	provisioning	Line	Line	Lamp	Boards
SPA509G	N	Y	Y	http	1-12	1-12	1-11	Ν
SPA525G	N	Y	N	http	1-5	1-5	1-4	Ν

## Analog Adapters

Model	IPv6	TLS	SRTP	Auto	Private	Shared	Busy
	IFVO	11.5	SHIP	provisioning	Line	Line	Lamp
SPA232D	Ν	Y	Y	http	1-6	0	0
ATA112	Υ	Y	Y	http	1-2	0	0
ATA122	Υ	Y	Y	http	1-2	0	0

## Extension Boards

Model	Ports	Buttons	Busy Lamp	Supported phones
SPA500S	2	32	1-32	SPA500

# 15.1.12.2 Panasonic Devices

### IP Phones

Model	IPv6 TL	TLS	SRTP	Auto	Private	Shared	Busy	Extension
	IPVO	115	SHIP	provisioning	Line	Line	Lamp	Boards
KX-UT113	N	N	N	redirect	1-2	1-2	0	N
KX-UT123	N	N	N	redirect	1-2	1-2	0	N
KX-UT133	N	N	N	redirect	1-4	1-4	1-23	N
KX-UT136	N	N	N	redirect	1-4	1-4	1-23	N
KX-UT248	N	N	Y	redirect	1-6	1-6	1-23	N

## 15.1.12.3 Yealink Devices

### IP Phones

Model	IPv6	TLS	SRTP	Auto	Private	Shared	Busy	Extension
Model	IFVO	115	JUL	provisioning	Line	Line	Lamp	Boards
SIP-T19P	Y	Y	Υ	redirect	1	1	0	N
SIP-T20P	Y	Y	Y	redirect	1	1	0	N
SIP-T21P	Y	Y	Y	redirect	1-2	1-2	1	N
SIP-T22P	Y	Y	Y	redirect	1-3	1-3	1-2	N
SIP-T23P	Y	Y	Y	redirect	1-3	1-3	1-2	N
SIP-T23G	Y	Y	Y	redirect	1-3	1-3	1-2	N
SIP-T26P	Y	Y	Y	redirect	1-3	1-3	1-12	N

Model	IPv6	TLS	SRTP	Auto	Private	Shared	Busy	Extension
Model	IPVO	IL5	3612	provisioning	Line	Line	Lamp	Boards
SIP-T28P	Y	Y	Y	redirect	1-6	1-6	1-15	2
SIP-T32G	Y	Υ	Y	redirect	1-3	1-3	1-2	N
SIP-T38G	Y	Υ	Y	redirect	1-6	1-6	1-15	N
SIP-T41P	Y	Υ	Y	redirect	1-3	1-3	1-14	N
SIP-T42G	Y	Υ	Y	redirect	1-3	1-3	1-14	N
SIP-T46G	Y	Y	Y	redirect	1-6	1-6	1-26	N
SIP-T48G	Y	Y	Y	redirect	1-6	1-6	1-28	N
W52P	N	Υ	Y	redirect	1-5	1-5	0	N

### 15.1.12.4 Innovaphone Devices

### IP Phones

Madal	ID. C	TIC	ODTD	Auto	Private	Shared	Busy	Extension
Model	IPv6	TLS	SRTP	provisioning	Line	Line	Lamp	Boards
IP232	N	Y	Y	dhcp	1	0	1-16	2
IP222	N	Y	Y	dhcp	1	0	1-16	2
IP240	N	N	N	dhcp	1	0	1-15	2
IP111	N	Y	Y	dhcp	1	0	1-16	0

# Analog Adapters

Model	IPv6	те	ертр	Auto	Private	Shared	Busy
Model	IFVO	TLS SRTP	provisioning	Line	Line	Lamp	
IP22	Ν	Y	Y	dhcp	1	0	0

## Extension Boards

Model	Ports	Buttons	Busy Lamp	Supported phones
IP2X2X	2	64	1-32	IP2x2
IP230-X	2	30	1-30	IP230

# 15.1.12.5 Audiocodes Devices

Model	IPv6	TLS	SRTP	Auto provisioning	Private Line	Shared Line	Busy Lamp	Speed Dial
Mediant800	Y	Y	Y	dhcp	1	0	0	N

# 15.1.13 Phone features

### 15.1.13.1 Cisco phones

### SPA301

# 1) Soft keys

Not available.

## 2) Hard keys

- vm
- hold/unhold

### 3) Line keys

Not available.

### 4) VSC

- · directed pickup
- park/unpark

### SPA303

### 1) Soft keys

### Idle:

redial	lcr	dir	dnd >
< cfwd	unpark		

### Idle with missed calls:

Icr miss
----------

### Call:

hold	endCall	conf	xfer >
< bxfer	park		

### Call on hold:

resume	endCall	newCall	redial >
< dir	cfwd	dnd	

## **Ringing:**

answer ignore
---------------

## 2) Hard keys

- vm
- hold/unhold

# 3) Line keys

- BLF monitoring
- directed pickup

# 4) VSC

• directed pickup

### SPA501G

## 1) Soft keys

## Idle:

redial	lcr	dir	dnd >
< cfwd	unpark		

### Idle with missed calls:

lcr		miss

# Call:

hold/resume	endCall	conf	xfer >
< bxfer	park		

## Call on hold:

resume	endCall	newCall	redial >
< dir	cfwd	dnd	

# **Ringing:**

answer	reject	

### 2) Hard keys

- vm
- hold/unhold

# 3) Line keys

- BLF monitoring
- directed pickup

## 4) VSC

· directed pickup

### SPA502G

## 1) Soft keys

## Idle:

redial	lcr	dir	dnd >
< cfwd	unpark		

### Idle with missed calls:

## Call:

hold/resume	endCall	conf	xfer >
< bxfer	park		

### Call on hold:

resume	endCall	newCall	redial >
< dir	cfwd	dnd	

# Ringing:

answer reject	
---------------	--

## 2) Hard keys

- vm
- hold/unhold

## 3) Line keys

Not available.

## 4) VSC

directed pickup

### SPA504G

### 1) Soft keys

# Idle:

redial	lcr	dir	dnd >
< cfwd	unpark		

# Idle with missed calls:

lcr miss
----------

# Call:

hold/resume	endCall	conf	xfer >
< bxfer	park		

### Call on hold:

resume	endCall	newCall	redial >
< dir	cfwd	dnd	

## Ringing:

answer reject		
---------------	--	--

### 2) Hard keys

- vm
- hold/unhold

## 3) Line keys

- BLF monitoring
- directed pickup

## 4) VSC

directed pickup

### SPA512G

# 1) Soft keys

### Idle:

redial	lcr	dir	dnd >
< cfwd	unpark		

## Idle with missed calls:

miss
------

### Call:

hold/resume	endCall	conf	xfer >
< bxfer	park		

### Call on hold:

resume	endCall	newCall	redial >
< dir	cfwd	dnd	

## **Ringing:**

answer reject	
---------------	--

# 2) Hard keys

- vm
- hold/unhold

# 3) Line keys

Not available.

# 4) VSC

• directed pickup

### SPA514G

### 1) Soft keys

Idle:

redial	lcr	dir	dnd >
< cfwd	unpark		

## Idle with missed calls:

Icr miss
----------

### Call:

hold/resume	endCall	conf	xfer >
< bxfer	park		

### Call on hold:

resume	endCall	newCall	redial >
< dir	cfwd	dnd	

### **Ringing:**

answer	reject		
--------	--------	--	--

### 2) Hard keys

- vm
- hold/unhold

## 3) Line keys

- BLF monitoring
- directed pickup

# 4) VSC

directed pickup

### SPA509G

## 1) Soft keys

#### Idle:

redial	lcr	dir	dnd >
< cfwd	unpark		

## Idle with missed calls:

lcr	miss
-----	------

# Call:

hold/resume	endCall	conf	xfer >
< bxfer	park		

## Call on hold:

resume	endCall	newCall	redial >
< dir	cfwd	dnd	

### **Ringing:**

r		
answer	reject	

## 2) Hard keys

- vm
- hold/unhold

### 3) Line keys

- BLF monitoring
- directed pickup

## 4) VSC

directed pickup

### SPA508G

1) Soft keys

## Idle:

redial	lcr	dir	dnd >
< cfwd	unpark		

## Idle with missed calls:

lcr		miss

## Call:

hold/resume	endCall	conf	xfer >
< bxfer	park		

## Call on hold:

resume	endCall	newCall	redial >
< dir	cfwd	dnd	

# Ringing:

answer reject
---------------

## 2) Hard keys

- vm
- hold/unhold

# 3) Line keys

- BLF monitoring
- directed pickup

# 4) VSC

directed pickup

# SPA525G

1) Soft keys

Idle:

Redial	call Rtn	Directory	DND >
< Forward	Unpark		

### Idle with missed calls:

Call Rtn		Miss

Call:

Hold	End Call	Conf	Transfer >
BlindXfer	Park		

### Call on hold:

Resume	EndCall	EewCall	Redial >
< Directory	Forward	DND	

## **Ringing:**

Answer Igr	gnore		
------------	-------	--	--

### 2) Hard keys

• vm

### hold/unhold

## 3) Line keys

- BLF monitoring
- directed pickup

### 4) VSC

directed pickup

15.1.13.2 Yealink phones

T19P

1) Soft keys

Idle:

History DNE	) Menu
-------------	--------

## Idle with missed calls:

Exit		View

Call:

Tran Hold Conf Cancel	
-----------------------	--

### Call on hold:

Tran Resume	NewCall	Cancel
-------------	---------	--------

## **Ringing:**

Answer FWD	Silence	Reject
------------	---------	--------

## 2) Hard keys

- vm
- redial
- transfer

## 3) Line keys

Not available.

4) VSC

- transfer park
- · directed pick up
- park/unpark

## T20P

## 1) Soft keys

#### Idle:

History	DND	Menu
---------	-----	------

### Idle with missed calls:

Exit		View

### Call:

Tran	Hold	Conf	Cancel
------	------	------	--------

## Call on hold:

Tran Resume New	Call Cancel
-----------------	-------------

## **Ringing:**

		Silence	Reject
--	--	---------	--------

### 2) Hard keys

- vm
- redial
- transfer

### 3) Line keys

- BLF monitoring
- directed pickup

## 4) VSC

- transfer park
- park/unpark

# T21P

## 1) Soft keys

Idle:

|--|

Idle with missed calls:

Exit		View
		-

## Call:

Tran Hold	Conf	Cancel
-----------	------	--------

## Call on hold:

Tran Resume	NewCall	Cancel
-------------	---------	--------

### **Ringing:**

Answer FWD	Silence	Reject
------------	---------	--------

## 2) Hard keys

- vm
- redial
- transfer

### 3) Line keys

- BLF monitoring
- directed pickup

## 4) VSC

- transfer park
- park/unpark

### T22P

## 1) Soft keys

### Idle:

History DND Menu
------------------

## Idle with missed calls:

Exit View
-----------

## Call:

n Hold	Conf	Cancel	
--------	------	--------	--

## Call on hold:

Tran Resume NewCall Cancel
----------------------------

## **Ringing:**

wer FWD	Silence	Reject
---------	---------	--------

# 2) Hard keys

- vm
- redial
- transfer

# 3) Line keys

- BLF monitoring
- directed pickup

# 4) VSC

- park/unpark
- transfer park

### T23P

### 1) Soft keys

## Idle:

History	DND	Menu
---------	-----	------

## Idle with missed calls:

Exit View
-----------

### Call:

Tran Hold	Conf	Cancel
-----------	------	--------

## Call on hold:

Tran	Resume	NewCall	Cancel
------	--------	---------	--------

# Ringing:

Answer FWD Silence Reject
---------------------------

### 2) Hard keys

• vm

### redial

transfer

## 3) Line keys

- BLF monitoring
- directed pickup

## 4) VSC

- park/unpark
- transfer park

## T23G

### 1) Soft keys

## Idle:

History	Dir	DND	Menu

### Idle with missed calls:

Exit View			Exit			View
-----------	--	--	------	--	--	------

## Call:

Tran Hold	Conf	EndCall
-----------	------	---------

## Call on hold:

Tran         Resume         NewCall         EndCall
---

## **Ringing:**

Answer FWD		Reject
------------	--	--------

## 2) Hard keys

- vm
- redial
- transfer

## 3) Line keys

- BLF monitoring
- directed pickup

# 4) VSC

- unpark
- transfer park

## T26P

## 1) Soft keys

#### Idle:

History	D	DND	Menu

#### Idle with missed calls:

Exit	View
------	------

### Call:

Tran Hold	Conf	Cancel
-----------	------	--------

### Call on hold:

Tran Resume	NewCall	Cancel
-------------	---------	--------

## Ringing:

Answer FWD	Silence	Reject
------------	---------	--------

## 2) Hard keys

- vm
- redial
- transfer

### 3) Line keys

- BLF monitoring
- directed pickup

## 4) VSC

- unpark
- transfer park

### T28P

### 1) Soft keys

### Idle:

History DND Menu	
------------------	--

## Idle with missed calls:

Exit	Exit			View
------	------	--	--	------

### Call:

Tran Hold Conf	Cancel
----------------	--------

## Call on hold:

Tran Resume	NewCall	Cancel
-------------	---------	--------

# **Ringing:**

Answer	FWD	Silence	Reject

## 2) Hard keys

- vm
- redial
- transfer

# 3) Line keys

- BLF monitoring
- directed pickup

# 4) VSC

- park/unpark
- transfer park

# T32G

### 1) Soft keys

#### Idle:

History	DND	Menu
---------	-----	------

## Idle with missed calls:

Exit		View
	**************************************	

## Call:

Tran	Hold	Conf	Cancel
------	------	------	--------

# Call on hold:

Tran	1	Resume	NewCall	Cancel
------	---	--------	---------	--------

# **Ringing:**

Answer F	FWD	Silence	Reject
----------	-----	---------	--------

### 2) Hard keys

- vm
- redial
- transfer

## 3) Line keys

- BLF monitoring
- directed pickup

# 4) VSC

- unpark
- transfer park

### T38G

### 1) Soft keys

#### Idle:

History	DND	Menu
---------	-----	------

### Idle with missed calls:

Exit View
-----------

### Call:

Tran		Hold	Conf	Cancel
------	--	------	------	--------

## Call on hold:

Iran Resume NewCall Cancel
----------------------------

# **Ringing:**

Answer FWD	Silence	Reject
------------	---------	--------

# 2) Hard keys

- vm
- redial
- transfer

## 3) Line keys

- BLF monitoring
- · directed pickup

## 4) VSC

- unpark
- transfer park

### T41P

## 1) Soft keys

## Idle:

History	DN	ND	Menu

#### Idle with missed calls:

### Call:

Tran	Hold	Conf	Cancel

## Call on hold:

Tran	Resume	NewCall	Cancel
------	--------	---------	--------

# **Ringing:**

Answer FWD	) Silence	Reject	
------------	-----------	--------	--

# 2) Hard keys

- vm
- redial
- transfer

### 3) Line keys

- BLF monitoring
- · directed pickup

# 4) VSC

- park/unpark
- transfer park

# T42G

# 1) Soft keys

# Idle:

History DND Menu	
------------------	--

# Idle with missed calls:

Exit		View
------	--	------

## Call:

Tran	Hold	Conf	Cancel

# Call on hold:

n Resume	NewCall	Cancel
----------	---------	--------

### **Ringing:**

Answer FWD	Silence	Reject
------------	---------	--------

### 2) Hard keys

- vm
- redial

transfer

# 3) Line keys

- BLF monitoring
- directed pickup

# 4) VSC

- park/unpark
- transfer park

### T46G

### 1) Soft keys

#### Idle:

History	DND	Menu
---------	-----	------

#### Idle with missed calls:

Exit		View

#### Call:

Tran H	Hold	Conf	Cancel
--------	------	------	--------

# Call on hold:

Tran	Re	esume	NewCall	Cancel
------	----	-------	---------	--------

### **Ringing:**

Answer	FWD	Silence	Reject
--------	-----	---------	--------

# 2) Hard keys

- vm
- redial
- transfer

# 3) Line keys

- BLF monitoring
- directed pickup

# 4) VSC

- park/unpark
- transfer park

# T48G

# 1) Soft keys

#### Idle:

History	DND	Menu
---------	-----	------

### Idle with missed calls:

Exit		View

### Call:

Tran Hold Conf Cancel
-----------------------

### Call on hold:

Tran		Resume	NewCall	Cancel
------	--	--------	---------	--------

# **Ringing:**

Answer	FWD	Silence	Reject

# 2) Hard keys

- vm
- redial
- transfer

#### 3) Line keys

directed pickup

# 4) VSC

- park/unpark
- transfer park

W52P

#### 1) Soft keys

#### Idle:

History	Line
---------	------

#### Idle with missed calls:

Exit View
-----------

### Call:

Ext. Call	Options

#### Call on hold:

Resume	Line
--------	------

# **Ringing:**

Accort	
Accept	
•	

### 2) Hard keys

- vm
- redirect

# 3) VSC

- park/unpark
- transfer park

# 15.1.13.3 Panasonic phones

# KX-UT113

#### 1) Soft keys

### Idle:

Settings	Call Log	Phone book	
----------	----------	------------	--

### Call:

Blind	Pł	hone book	
-------	----	-----------	--

# Call on hold:

Call Log Phone book
---------------------

# Ringing:

Answer	Reject	
--------	--------	--

#### 2) Hard keys

- vm
- forward/dnd
- hold/unhold
- redial
- recall
- transfer
- conf

### 3) Line keys

Not available.

# 4) VSC

- park/unpark
- transfer park

# KX-UT123

## 1) Soft keys

#### Idle:

Settings	Call Log	Phone book	
----------	----------	------------	--

#### Call:

Blind Phone book
------------------

#### Call on hold:

Call Log	Phone book	
----------	------------	--

### **Ringing:**

Answer	Reject	
--------	--------	--

# 2) Hard keys

- vm
- forward/dnd
- hold/unhold
- redial
- recall
- transfer
- conf

# 3) Line keys

Not available.

# 4) VSC

- park/unpark
- transfer park

# KX-UT133

1) Soft keys

Settings Call Log	Phone book	
-------------------	------------	--

Blind		Phone book	
-------	--	------------	--

# Call on hold:

Call Log	Phone book	
----------	------------	--

#### **Ringing:**

	Γ	Answer		Reject	
--	---	--------	--	--------	--

# 2) Hard keys

- vm
- forward/dnd
- hold/unhold
- redial
- recall
- transfer
- conf

# 3) Line keys

- BLF monitoring
- · directed pickup

# 4) VSC

- unpark
- transfer park

# KX-UT136

```
1) Soft keys
```

	Settings	Call Log	Phone book	
--	----------	----------	------------	--

Blind		Phone book	
-------	--	------------	--

# Call on hold:

Call Log	Phone book	
----------	------------	--

#### **Ringing:**

Answer	Reject	

### 2) Hard keys

- vm
- forward/dnd
- hold/unhold
- redial
- recall
- transfer
- conf

# 3) Line keys

- BLF monitoring
- · directed pickup

# 4) VSC

- park/unpark
- transfer park

# KX-UT248

1) Soft keys

Settings	Call Log	Phone book	
----------	----------	------------	--

Blind		Phone book	
-------	--	------------	--

# Call on hold:

	Call Log	Phone book	
--	----------	------------	--

#### **Ringing:**

Answer	Reject	

# 2) Hard keys

- vm
- forward/dnd
- hold/unhold
- redial
- recall
- transfer
- conf

### 3) Line keys

- BLF monitoring
- directed pickup

# 4) VSC

- park/unpark
- transfer park

#### 15.1.13.4 Innovaphone

IP222

# 1) Soft keys

	All Calls	Home	Calls	My favorites	Phonebook
--	-----------	------	-------	--------------	-----------

Hold	Transfer	Park	Cancel
------	----------	------	--------

# Call on hold:

Resume         Transfer         Park         Cancel
---

# Ringing:

Answer Transfer	Silence	Reject
-----------------	---------	--------

# 2) Hard keys

- hold
- redial

# 3) Line keys

BLF monitoring

# 4) VSC

- unpark
- transfer park

#### IP232

#### 1) Soft keys

### Idle:

Setup All Calls Home	Calls	My favorites	Phonebook
----------------------	-------	--------------	-----------

## Call:

Hold	Transfer	Park	Cancel
------	----------	------	--------

# Call on hold:

Resume	Transfer	Park	Cancel
--------	----------	------	--------

## Ringing:

Answer		Transfer	Silence	Reject
--------	--	----------	---------	--------

# 2) Hard keys

- hold
- redial

# 3) Line keys

BLF monitoring

# 4) VSC

- unpark
- transfer park

# IP111

# 1) Soft keys

### Idle:

Setup All Calls Home Calls	My favorites Phonebook
----------------------------	------------------------

# Call:

Hold	Transfer	Park	Cancel

# Call on hold:

Resume Transfer	Park	Cancel
-----------------	------	--------

# Ringing:

Answer Transfer Silence Reject		Iranster		
--------------------------------	--	----------	--	--

### 2) Hard keys

- hold
- redial

### 3) Line keys

BLF monitoring

# 4) VSC

- unpark
- transfer park

#### IP240

#### 1) Soft keys

Not available.

#### 2) Hard keys

- hold
- redial
- conference
- dnd
- forward

### 3) Line keys

BLF monitoring

### 4) VSC

- transfer park
- unpark

# 15.2 Sipwise sip:phone App (SIP client)

You can order two commercial Unified Communication Clients for full end-to-end integration of voice, video, chat and presence features. There are two applications available:

• the sip:phone Desktop Client for Microsoft Windows, Apple OSX, and Linux;

• the sip:phone Mobile App for iOS and Android.

Both clients are fully brandable to the customer's corporate identity. The clients are not part of the standard delivery and need to be licensed separately. This handbook discusses the mobile client in details.

We continuously develop the mobile clients to provide new features, as they do not support the full range of features yet.

The sip:phone Mobile App is a mobile client for iOS and Android that supports voice calls via SIP, as well as presence and instant messaging via XMPP. The following sections describe the steps needed to integrate it into your sip:provider PRO.

#### 15.2.1 Zero Config Launcher

Part of the mobile apps is a mechanism to sign up to the service via a 3rd party website, which is initiated on the login screen and rendered within the app. During the sign-up process, the 3rd party service is supposed to create a new account and subscriber in the sip:provider PRO (e.g. automatically via the API) and provide the end user with the access credentials.

The mobile apps come with a zero config mechanism to simplify the end-customer log in using these credentials (especially ruling out the need to manually enter them). It makes it possible to deliver the access credentials via a side channel (e.g. Email, SMS) packed into a URL. The user just clicks the URL, and it automatically launches the app with the correct credentials. The following picture shows the overall workflow.

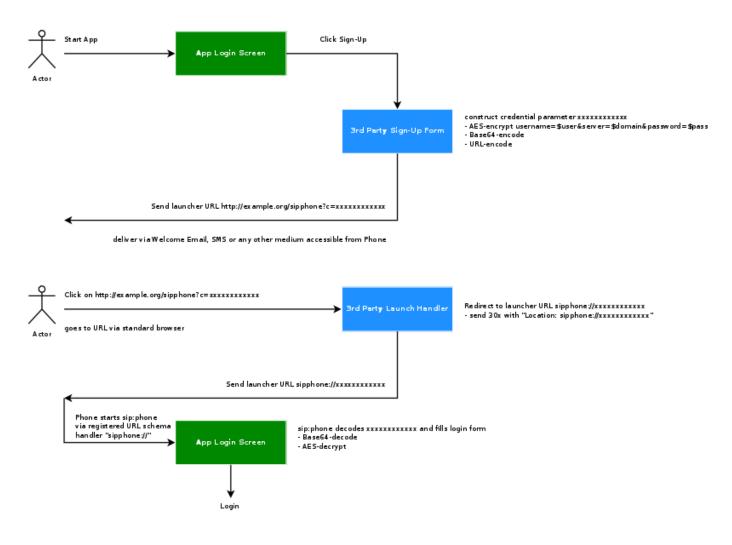


Figure 115: Provisioning Push Workflow

There are two components provided by a 3rd party system. One is the *3rd Party Sign-Up Form*, and the other is the *3rd Party Launch Handler*. The purpose of these components is to allow an end customer to open a link with the access credentials via the sip:phone app.

#### 15.2.1.1 3rd Party Sign-Up Form

The 3rd Party Sign-Up Form is a website the app shows to the end user when he taps the sign-up link on the app *Login Screen*. There, the end customer usually provides his contact details like name, address, phone number and email address, etc. After validation, the website creates an account and a subscriber in the sip:provider PRO via the API.

After successfully creating the account and the subscriber, this site needs to construct a specially crafted URL, which is sent back to the end customer via a side channel. Ideally, this channel would be an SMS if you want to verify the end customer's mobile number, or an email if you want to check the email address.

The sip:phone app registers a URL schema handler for URLs starting with sipphone://. If you start such a link, the app performs a Base64 decoding of the string right after the sipphone:// prefix and then decrypts the resulting binary string via AES using the keys defined during the branding step. The resulting string is supposed to be

username=\$user&server=\$domain&password=\$password.

Therefore, the *3rd Party Sign-Up Form* needs to construct this string using the credentials defined while creating the subscriber via the sip:provider PRO API, then encrypt it via AES, and finally perform a Base64 encoding of the result.

#### Note

Up until and including version mr5.4.2 of the sip:provider PRO, the SIP login credentials are used here. Future versions will connect to the REST interface of the sip:provider PRO using the web credentials first and fetch the SIP credentials along with other settings from there.

An example Perl code performs encoding of such a string. The AES key and initialization vector (\$key and \$iv) are the standard values of the sip:phone app and should work until you specified other values during the branding process.

```
#!/usr/bin/perl -w
use strict;
use Crypt::Rijndael;
use MIME::Base64;
use URI::Escape;
my $key = 'iBmTdavJ8joPW3HO';
my $iv = 'tww211Qe6cmywrp3';
my $plain = do { local $/; <> };
# pkcs#5 padding to 16 bytes blocksize
my $pad = 16 - (length $plain) % 16;
$plain .= pack('C', $pad) x $pad;
my $cipher = Crypt::Rijndael->new(
        $key,
        Crypt::Rijndael::MODE_CBC()
);
$cipher->set_iv($iv);
my $crypted = $cipher->encrypt($plain);
# store b64-encoded string and print to STDOUT
my $b64 = encode_base64($crypted, '');
print $b64, "\n";
# print to STDOUT using URL escaping also
print uri_escape($b64), "\n";
```

This snippet takes a string from STDIN, encrypts it via AES, encodes it via Base64 and sends the result to STDOUT. It also writes the second line with the same string, but this time, the URL is escaped. To test it, you would run it as follows on a shell, granted it's stored at /path/to/encrypt.pl.

This command would result in the output strings CI8VN8toaE40w8E4OH2rAuFj3Qev9QdLI/Wv/VaBCVK2yNkBZjxE9

eafXkkrQfmYdeu01PquS5P40zhUq8Mfjg== and CI8VN8toaE40w8E40H2rAuFj3Qev9QdLI%2FWv%2FVaBCVK 2yNkBZjxE9eafXkkrQfmYdeu01PquS5P40zhUq8Mfjg%3D%3D. The sip:phone can use the former string to automatically fill in the login form of the Login Screen if started via a Link like sipphone://CI8VN8toaE40w8E40H2rAuFj3Qev9 QdLI/Wv/VaBCVK2yNkBZjxE9eafXkkrQfmYdeu01PquS5P40zhUq8Mfjg==.

#### Here is the same code in PHP.

```
#!/usr/bin/php
<?php
$key = "iBmTdavJ8joPW3HO";
$iv = "tww211Qe6cmywrp3";
$clear = fgets(STDIN);
$cipher = fnEncrypt($clear, $key, $iv);
echo $cipher, "\n";
echo urlencode($cipher), "\n";
function fnEncrypt($clear, $key, $iv) {
        $pad = 16 - strlen($clear) % 16;
        $clear .= str_repeat(pack('C', $pad), $pad);
        return rtrim(base64_encode(mcrypt_encrypt(
                MCRYPT_RIJNDAEL_128, $key, $clear,
                MCRYPT_MODE_CBC, $iv)), "\0");
}
?>
```

Similar to the Perl code, you can call it like this:

However, a URL with the sipphone:// schema is not displayed as a link in an SMS or an Email client and thus can not be clicked by the end customer, so you need to make a detour via a regular http:// URL. To do so, you need a *3rd Party Launch Handler* to trick the phone to open such a link.

Therefore, that the *3rd Party Sign-Up Form* needs to return a link containing a URL pointing to the *3rd Party Launch Handler* and pass the URL escaped string gathered above to the client via an SMS or an Email. Since it is the regular http://link, it is clickable on the phone and can be launched from virtually any client (SMS, Email, etc.), which correctly renders an HTML link.

A possible SMS sent to the end customer (via the phone number entered in the sign-up from) could, therefore, look as follows (trying to stay below 140 chars).

```
http://example.org/p?c=CI8VN8toaE40w8E40H2rAuFj3Qev9QdLI
%2FWv%2FVaBCVK2yNkBZjxE9eafXkkrQfmYdeu01PquS5P40zhUq8Mfjg%3D%3D to launch sipphone
```

#### An HTML Email could look like this:

Welcome to Example.org,

```
<a href="http://www.example.org/sipphone?c=CI8VN8toaE40w8E40H2rAuFj3Qev9QdLI
%2FWv%2FVaBCVK2yNkBZjxE9eafXkkrQfmYdeu01PquS5P40zhUq8Mfjg%3D%3D">
click here
</a> to log in.
```

That way, you can do both: verify the contact details of the end customer, and send the end customer the login credentials in a secure manner.

#### 15.2.1.2 3rd Party Launch Handler

The URL http://www.example.org/sipphone mentioned above can be any simple script, and its sole purpose is to send back a 301 Moved Permanently or 302 Moved Temporarily with a Location:sipphone://xxxxxxx keader to tell the phone to open this link via the sip:phone app. The *xxxxxxxxxxxx* is the plain (non-URL-escaped) string generated by the above script.

An example CGI script performing this task follows.

```
#!/usr/bin/perl -w
use strict;
use CGI;
my $q = CGI->new;
my $c = $q->param('c');
print CGI::redirect("sipphone://$c");
```

The script simply takes the URL parameter c from the URL http://www.example.org/sipphone?c=CI8VN8toaE40 w8E40H2rAuFj3Qev9QdLI%2FWv%2FVaBCVK2yNkBZjxE9eafXkkrQfmYdeu01PquS5P40zhUq8Mfjg%3D%3D crafted above and puts its content into a Location header using the sipphone:// schema, and finally sends a 301 Moved Permanently back to the phone.

The phone follows the redirect by opening the URL using the sip:phone app, which in turn decrypts the content and fills in the login form.

#### Note

Future versions of the sip:provider PRO will be shipped with this launch handler integrated into the system. Up until and including the version mr5.4.2, this script needs to be installed on any webserver manually.

#### 15.2.2 Mobile Push Notification

The *mobile push* functionality provides the remote start of a mobile application on incoming calls via the Google GCM or the Apple APNS notification services. It enables you to offer your subscribers a modern and convenient service on mobile devices.

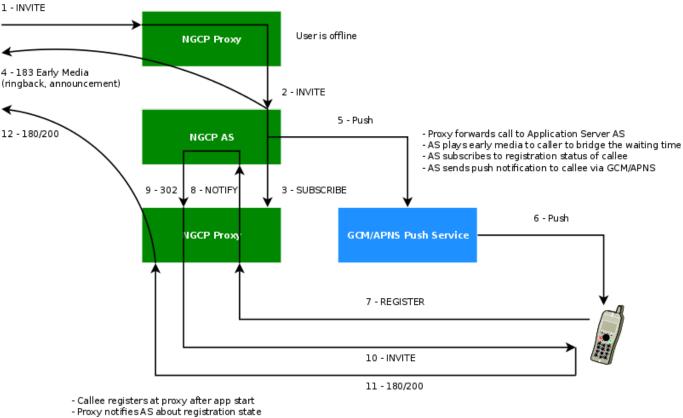


# Caution

Although suspending an application on a phone and waking it up via the mobile push notification service extends battery life, the whole mobile push notification concept is the best effort framework provided by Apple and Google for iOS and Android respectively, and therefore does not guarantee 100% reliability.

#### 15.2.2.1 Architecture

If the mobile push functionality is enabled and there are no devices registered for a subscriber, the call-flow looks as follows.



- AS deflects call to proxy

- Proxy completes call to callee

Figure 116: Mobile Push Workflow

- 1. The caller sends INVITE to proxy
- 2. The callee is offline, proxy forwards the call to AS (application server)
- 3. AS subscribes to the callee's registration events on proxy
- 4. AS sends early media to the caller as a feedback, as the call initiation process might take a while
- 5. AS sends the push request to GCM/APNS service
- 6. GCM/APNS service delivers the push request to the callee

- 8. Proxy sends registration notification to AS
- 9. AS deflects the call back to proxy
- 10. Proxy sends INVITE to the callee
- 11. The callee accepts the call
- 12. The response is sent back to the caller. Hence, the call setup is completed

In the case of a time-out (no registration notification within a particular time), the application server rejects the call request with an error.

#### 15.2.2.2 The Configuration Checklist

Follow this checklist to make sure you've completed all the steps. If you miss anything, the service may not work as expected.

Name	Description	Link
Obtain a trusted SSL certificate from a	Required for either application	Section 15.2.2.3
CA		
Create an Apple developer account	For iOS mobile application	Section 15.2.2.4
and enable the push notification		
service		
Obtain the Apple certificate for the app	For iOS mobile application	Section 15.2.2.5
Obtain the API key for the app from	For Android mobile application	Section 15.2.2.6
Google		
Provide the required information to	It is required to make beta builds and	Section 15.2.2.7
developers	publish the apps	
Adjust the configuration	Adjust the config.yml file and apply the	Section 15.2.2.8
	changes (usually performed by	
	Sipwise)	
Recheck your DNS Zone configuration	Check that the DNS Zone is correctly	Section 15.2.2.9
	configured	
Add DNS SRV records	Create specific DNS SRV records for	Section 15.2.2.10
	SIP and XMPP services	
Check NTP configuration	Ensure that all your servers show	Section 15.2.2.11
	exact time	
Enable Apple/Google Mobile Push in	It can be enabled for a domain or	Section 15.2.2.12
the Admin Panel	separate subscribers	
Configure a mobile application	Check that subscribers can easily	Section 15.2.2.13
	install and use your application	

#### 15.2.2.3 Obtain the Trusted SSL Certificate

A trusted SSL certificate is required, and we suggest obtaining it before starting the configuration.

The mobile application uses respective iOS/Android libraries to establish a secure TLS connection with certain sip:provider PRO services, such as SIP/XMPP/pushd(https). A *signed* SSL certificate is required to guarantee the security of this connection.

Any Certificate Authority (CA) such as Verisign and others can provide you with the required trusted SSL certificate (a certificate and the key files) which you will use in the configuration below.

#### 15.2.2.4 Create an Apple Account and Enable the Push Notification Service

Below is a brief instruction on how to create an Apple account and enable the Push Notification Service in it. You may need to perform additional steps depending on your project.

#### Note

You may only create an Apple account (step 1 below) and enroll into the Apple Developer Program (step 2 below) and Sipwise developers will do the rest. Still, you can perform all the steps by yourself.

- 1. Create an Apple developer account to get the Apple ID for your company. For this, go to developer.apple.com/account
- 2. Enrol in the Apple Developer Program. It is required to configure push notifications as you will need a push notification certificate for your App ID, which requires the Apple Developer Program membership. Go to developer.apple.com/programs for more details.

### 3. Register an App ID:

• Sign into developer.apple.com/account.

É Developer	Account	Demo Sipwise GmbH
Program Resources		
≔ Overview	Sipwise GmbH	
(i) Membership	Apple Developer Program	
People     Peo		
Certificates, IDs & Profiles		
(i) iTunes Connect	People	
CloudKit Dashboard	Send invitations to your development team so they can take advantage of membership resources.	
X Code-Level Support		
Additional Resources	Certificates, Identifiers & Profiles	
Documentation	Manage the certificates, identifiers, profiles, and devices you need to develop and distribute apps.	
♀ Forums		

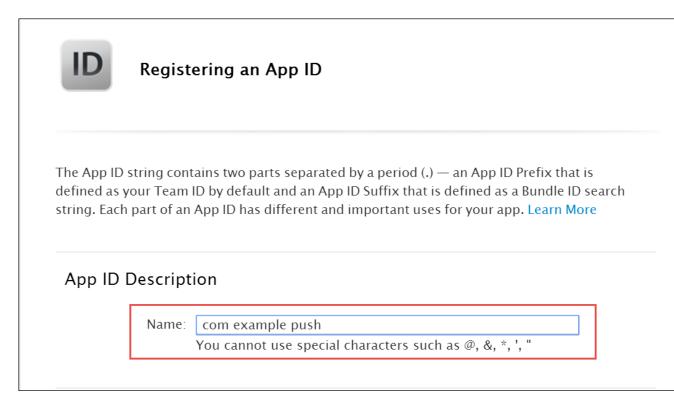
### • Click Certificates, IDs & Profiles.

🗴 Developer	Platforms	Resources	Program	Support	① Account	
Certificates, Identi	fiers & Profile	85				Demo 🔻
iOS, tvOS, watchOS	•		iOS Certi	ificates		+ 9
Certificates	3 Cer	ertificates Total				
= All	Name			Туре	Expires	
Pending						
Development						
Production						
D Identifiers						
App IDs						
Pass Type IDs						
Website Push IDs						
iCloud Containers						
App Groups						
Merchant IDs						

### • Under Identifiers, select App IDs.

🗯 Developer	Platforms	Resources	Program	Support	① Account	Q
Certificates, Identifie	ers & Profiles				[	Demo 🔻
iOS, tvOS, watchOS	•		iOS Ap	p IDs		+ 9
Certificates	3 App I	Ds total.				
All	Name		* ID			
Pending						
Development						
Production						
D Identifiers						
App IDs						
Pass Type IDs						
Website Push IDs						
iCloud Containers						
App Groups						
Merchant IDs						

• Click the Add button (+) in the upper-right corner.



• Enter a name for the App ID in the App ID Description block. This helps you identify the App ID later.

Apple TV	
Apple Watch	App ID Prefix
iPad	
iPhone	Value: XD7GAT4126 (Team ID)
iPod Touch	
Provisioning Profiles	App ID Suffix
All	
Development	Explicit App ID
Distribution	If you plan to incorporate app services such as Game Center, In-App Purchase, Data
	Protection, and iCloud, or want a provisioning profile unique to a single app, you must register an explicit App ID for your app.
	register an explicit App to for your app.
	To create an explicit App ID, enter a unique string in the Bundle ID field. This string should match the Bundle ID of your app.
	Bundle ID: com.example.push
	We recommend using a reverse-domain name style string (i.e.,
	com.domainname.appname). It cannot contain an asterisk (*).
	Wildcard App ID
	This allows you to use a single App ID to match multiple apps. To create a wildcard App
	ID, enter an asterisk (*) as the last digit in the Bundle ID field.

• Select *Explicit App ID* and enter the app's bundle ID in the *Bundle ID* field. Note that an explicit App ID exactly matches the bundle ID of an app you are building — for example, com.example.push. An explicit App ID can *not* contain an asterisk (\*).

Γ

App Services Select the services you would like to enable in App ID has been registered.	your app. You can edit your choices after this
Enable Services: App Groups	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
✓ In-App Purchase	/
Inter-App Audio	
<ul> <li>Wallet</li> <li>Push Notifications</li> <li>Personal VPN</li> </ul>	
	_
Cancel	Continue

• In the App Services section enable Push Notifications. Click Continue to submit the form

ID	Confirm your App ID.
	Comminyour App ID.

To complete the registration of this App ID, make sure your App ID information is correct, and click the submit button.

App ID Description:	com example push
Identifier:	XD7GAT4I26
Data Protection:	Disabled
Game Center:	Enabled
iCloud:	Disabled
In-App Purchase:	Enabled
Inter-App Audio:	Disabled
Passbook:	Disabled
Push Notifications:	Enabled

	Cancel	Back	Submit
--	--------	------	--------

• Click Submit to create the App ID.

#### 15.2.2.5 Obtain an Apple SSL Certificate and a Private Key

- 1. Create a CSR (Certificate Signing Request):
  - Sign into developer.apple.com/account/ios/certificate.

🗯 Developer	Platforms	Resources	Program	Support	① Account	С		
Certificates, Ident	ifiers & Profile	5				Demo 🔻		
iOS, tvOS, watchOS	•		iOS Certi	ificates		(+ Q		
I Certificates	0 Cert	0 Certificates Total						
■ All	Name			Туре	Expires			
<ul><li>Pending</li><li>Development</li><li>Production</li></ul>								
Production								

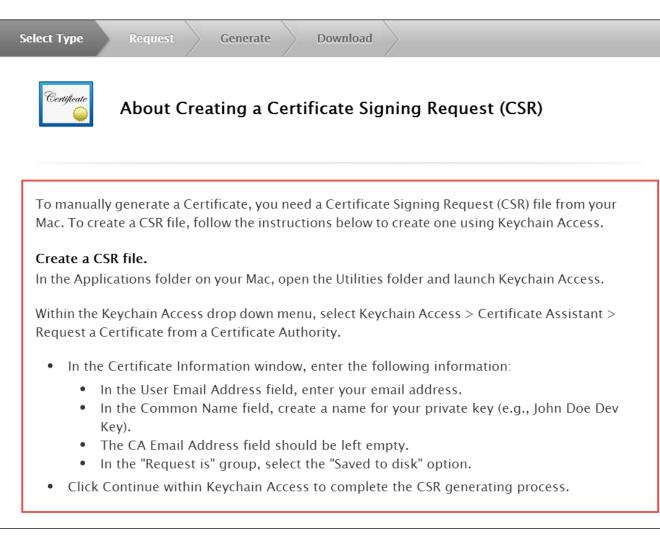
• Click the Add button (+) in the upper-right corner.

Select Type Request Generate Download	
Certificate What type of certificate do you need?	
******	
<ul> <li>Production</li> <li>App Store and Ad Hoc Sign your iOS app for submission to the App Store or for Ad Hoc distribution.</li> </ul>	
Apple Push Notification service SSL (Sandbox & Production)	
Establish connectivity between your notification server, the Apple Push Notification service sandbox, and production environments to deliver remote notifications to your app. When utilizing HTTP/2, the same certificate can be used to deliver app notifications, update ClockKit complication data, and alert background VoIP apps of incoming activity. A separate certificate is required for each app you distribute.	

• Select Apple Push Notification service SSL (Sandbox & Production) as the certificate type and click Continue.

Certificate O	Vhich App ID w	ould yo	ou like to	_			
				use?			
Notification se connect to the	t you want to enable rvice SSL certificate. Apple Push Notifica er can be used to cre	The App I tion servic	D-specific S ce. Note that	SL certificate only explicit	e allows your s App IDs with	erver to a specific	
Select an App	ID for your Apple F	Push Notif	fication serv	vice SSL Cert	ificate (Sandb	ox & Produ	ucti
,	App ID: XD7GAT	4126.com	.example.	push		T	

• Select your App ID and click Continue.



- Read the information about creating a CSR.
- Follow the instructions to create a CSR using Keychain Access in MAC.

#### Note

If you do not have access to a Mac, you can still create a CSR in Linux or Windows using OpenSSL, for example.

- 2. Get the Certificate and Private Key
  - When you have the CSR file return to the browser and click Continue.

elect Type	Request Generate Download
Certificate 🥥	Generate your certificate.
private ke and can b	rr CSR file is created, a public and private key pair is automatically generated. Your ey is stored on your computer. On a Mac, it is stored in the login Keychain by default be viewed in the Keychain Access app under the "Keys" category. Your requested e is the public half of your key pair.
<b>Upload C</b> Select .ce	<b>SR file.</b> rtSigningRequest file saved on your Mac.
Choo	ose File

• Click Choose File... in your browser.

Select Type Request Generate Download
Generate your certificate.
When your CSR file is created, a public and private key pair is automatically generated. Your private key is stored on your computer. On a Mac, it is stored in the login Keychain by default and can be viewed in the Keychain Access app under the "Keys" category. Your requested certificate is the public half of your key pair.
<b>Upload CSR file.</b> Select .certSigningRequest file saved on your Mac.
Choose File CertificateSigningRequest.certSigningRequest

• Select the CSR file you just created and saved and click Continue.

Certificate	ur certific	ate is ready.			
<b>Download, Install and Backup</b> Download your certificate to your Mac, then double click the .cer file to install in Keychain Access. Make sure to save a backup copy of your private and public keys somewhere secure.					
Access. Make sur					
Access. Make sur	e to save a ba	ckup copy of your private and public keys somewhere secure.			
Certificate	e to save a ba Name:	Apple Push Services: com.example.push			

- Click Download to download the certificate (give it the aps.cer name).
- Open the downloaded certificate file (it should automatically be opened in Keychain Access, otherwise open it manually in Keychain Access).
- · Find the certificate you just opened/imported in Keychain Access.
- · Expand the certificate to show the Private Key.
- Select only the Private Key portion of the certificate, right-click on it and select Export "Common Name"... from the menu.
- Choose a location (e.g. Desktop) and filename to export the .p12 file to and click Save.
- **Optionally** pick a password for the .p12 file to protect its private key contents and click *OK*. (You will then need to enter your log-in password to permit the export).
- 3. Generate a PEM file from the p12 file:
  - Open up your terminal and run the following commands to create a PEM file from the p12 file (If you input a password for the p12 file, you will need to enter it here):

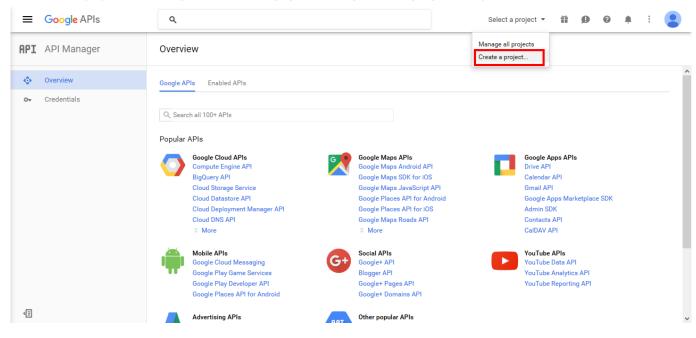
```
cd ~/Desktop
openssl x509 -in aps.cer -inform der -out PushChatCert.pem
openssl pkcs12 -in PushChatCert.p12 -out PushCertificate.pem -nodes -clcerts
openssl pkcs12 -nocerts -out PushChatKey.pem -in PushChatKey.p12
```

### 15.2.2.6 Obtain the API Key for the App from Google

You can use Google Cloud Messaging (GCM) to send push notifications to your subscribers with Android-based mobile devices. Google Cloud Messaging is a free service that acts as an intermediary between the NGCP and devices of your subscribers. Google's Cloud Connection Server (CCS), a part of GCP, manages the persistent connections with mobile devices to deliver your push notifications.

While communicating with CCS, the NGCP identifies itself using an API key. To get it, follow the steps below.

1. Create a new project in the Google APIs Console page. For this go to code.google.com/apis/console.



2. Click Create a Project ..

378 / 486

rview

APIs Enab		Project name 💿
		MyChat
	n all 100+ A	Your project ID will be mychat-1320 🕐 Edit
r A	Pls	Show advanced options
	Google Cl	Please email me updates regarding feature announcements, performance suggestions, feedback surveys and special offers.
	Compute	🔿 Yes 💿 No
	BigQuery	
	Cloud Sto	I agree that my use of any services and related APIs is subject to my compliance with the applicable Terms of Service.
	Cloud Dat	• Yes No
	Cloud Dep	
	Cloud DN	Create Cancel
	8 More	Cleate
	Mobile APIs	Social APIs
		Id Messaging G+ Google+ API

3. Input the project name, agree with the Terms of Service and click Create.

API API Manager	Overview		
🛟 Overview	Google APIs Enabled APIs (0)		
o⊷ Credentials	Q Search all 100+ APIs Popular APIs		
	Google Cloud APIs Compute Engine API BigQuery API Cloud Storage Service Cloud Datastore API Cloud Deployment Manager API Cloud DNS API © More	Google Maps APIs Google Maps Android API Google Maps SDK for iOS Google Maps JavaScript API Google Places API for Android Google Places API for iOS Google Maps Roads API V More	Google Apps APIs Drive API Calendar API Gmail API Google Apps Marketplace SDK Admin SDK Contacts API CalDAV API
	Mobile APIs Google Cloud Messaging Google Play Game Services Google Play Developer API Google Places API for Android	Get Social APIs Googlet API Blogger API Googlet Pages API Googlet Domains API	YouTube APIs YouTube Data API YouTube Analytics API YouTube Reporting API

4. Click Google Cloud Messaging on the Overview page.

=	Google APIs	٩		]	MyChat 💌	Ť	ø	0	0	:	
API	API Manager	Overview									
<del>ф</del> от	Overview Credentials	Enable     Coogle Cloud Messaging Google Cloud Messaging allows for push messaging to Android, iOS and Chrome users. Learn more Using credentials with this API Using an API key Main and API key. An API key identifies your project to check quotas and access. Go to the Credentials page to get an API key. You'll need a key for each platform, such as Web, Android, and iOS. Learn more	Your appl	ication	API key		Good	a gle serv	• • ice		
43											

5. Click Enable for the Google Cloud Messaging.

=	Google APIs	٩		MyChat 🔻	Ũ	ø	0	0	:	
API	API Manager	Overview								
	Overview	← Disable								
0*	Credentials	Google Cloud Messaging								
		This API is enabled, but you can't use it in your project until you create credentials. Click "Go to Credentials" to do this now (strongly recommended).				Go to	Creden	tials		
		Overview								
		Google Cloud Messaging allows for push messaging to Android, iOS and Chrome users. Learn more								
		Using credentials with this API								
		Using an API key To use this API you need an API key. An API key identifies your project to check quotas and access. Go to the Credentials page to get an API key. You'll need a key for each platform, such as Web, Android, and iOS. Learn more		0-11		•				
		Your app	plication	API key		Goog	le servi	ce		
Æ										

6. Click Go to Credentials.

	Google APIs	<b>्</b> MyChat 🕶 🏭 😰 🌲 🗄 😫
API	API Manager	Credentials
**	Overview	Add credentials to your project
ð	Credentials	<ul> <li>1 Find out what kind of credentials you need</li> <li>We'll help you set up the correct credentials If you wish you can skip this step and create an API key, client ID, or service account</li> <li>Which API are you using?</li> <li>Determines what kind of credentials you need.</li> <li>Google Cloud Messaging </li> <li>Where will you be calling the API from?</li> <li>Determines which settings you'll need to configure.</li> <li>Web server (e.g. node.js, Tomcat) </li> <li>What credentials do I need?</li> </ul> 2 Get your credentials

7. Select Google Cloud Messaging and Web Server from the corresponding lists and click What credentials do I need?

≡	Google APIs	Q. MyChat 🕶 🎬 😰 🏚 🗄 🤮
API	API Manager	Credentials
÷	Overview	Add credentials to your project
0+	Credentials	<ul> <li>Find out what kind of credentials you need Calling Google Cloud Messaging from a web server</li> <li>Create an API key</li> <li>This key should be kept secret on your server</li> <li>Every API request is generated by software running on a machine that you control. Per-user limits will be enforced using the address found in each request's userIp parameter, if specified. If the userIp parameter is missing, your machine's IP address will be used instead. Learn more</li> <li>Name MyChat Server key 1</li> <li>Accept requests from these server IP addresses (Optional) Examples: 192.168.0.1, 172.16.0.0/12, 2001:db8::1 or 2001:db8::/64</li> <li>1.2.3.4 ×</li> <li>5.6.7.8</li> </ul>
		Create API key

8. Adjust the API Key name and input the IP addresses of *all* your load balancers under *Accept requests from these server IP addresses*. Click *Create API key*.

#### Note

You may skip adding the IP addresses, otherwise list ALL your load balancers.

≡	Google APIs	9, MyChat 🕶 籠 😰 🖉 🌲 🗄 😫
API	API Manager	Credentials
<	Overview	Add credentials to your project
0-	Credentials	<ul> <li>Find out what kind of credentials you need Calling Google Cloud Messaging from a web server</li> <li>Create an API key Created API key 'MyChat Server key 1'</li> </ul>
		3 Get your credentials Here is your API key AIzaSyBLVAoIxr_o61C2n0iG5qUZ2uRdiD4R43c Copied Done Cancel

9. Copy your API key and click *Done*. Save the API key for future use.

	9, MyChat 🛩 🎬 🗩 🥝 🌲 🗄 🤮			
RPI API Manager	Credentials			
🔅 Overview	Credentials OAuth consent screen Domain verification			
o₊ Credentials	Create credentials 👻 Delete			
	Create credentials to access your enabled APIs. Refer to the API documentation for details.			
	API keys			
	Name         Creation date ∨         Type         Key			
	MyChat         May 30, 2016         Server         AlzaSyBLVAolxr_o6lC2n0iG5qUZ2uRdiD4R43c           Server         key 1			

### 15.2.2.7 Provide the Required Information to Developers

Please, provide Sipwise developers with the following files and information so that they can make beta builds and submit the application to the App Store:

- Access to your Apple developer account
- · The trusted SSL certificate and its private key
- The Apple SSL certificate and its private key

For the Android application, provide the following:

- · Access to your Google developer account
- · Google application API key

#### 15.2.2.8 Adjust the sip:provider PRO Configuration (Usually Performed by Sipwise)

- 1. Upload the Apple SSL certificate (PushChatCert.pem) and the private key (PushChatKey.pem) to /etc/ngcp-config/ssl/
- 2. Upload the trusted SSL certificate (CAsigned.crt) and the private key (CAsigned.key) to /etc/ngcp-config/ssl/
- 3. Specify the corresponding paths and names in the pushd section of the config.yml file:
  - apns: section (For iOS mobile application)
    - certificate: '/etc/ngcp-config/ssl/PushChatCert.pem'
    - enable: yes
    - key: '/etc/ngcp-config/ssl/PushChatKey.pem'
  - · enable: yes
  - · gcm: section (for Android mobile application)
    - enable: yes
    - key: 'google\_server\_api\_key\_here'
  - ssl: yes
  - · sslcertfile: /etc/ngcp-config/ssl/CAsigned.crt
  - sslcertkeyfile: /etc/ngcp-config/ssl/CAsigned.key
     You can find an example of /etc/ngcp-config/config.yml configuration in the config.yml overview section.
- 4. Apply your changes:

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

#### 15.2.2.9 Recheck Your DNS Zone Configuration

Check that your NS and A DNS records are correctly configured.

Let's consider the following example: \* the load-balancers have the lb01a.example.com and the lb01b.example.com names \* the shared name is lb01.example.com and the shared IP address is 1.1.1.1 \* the service name is voipservice.example.com

The following DNS records must be present:

Server Name	Record type	IP Address
lb01a.example.com	A	1.2.3.4
lb01b.example.com	A	5.6.7.8
lb01.example.com	A	1.1.1.1
voipservice.example.com	A	1.1.1.1

## 15.2.2.10 Add SRV Records to DNS

Add at least one record for each service: xmpp-server, xmpp-client, sips.

A regular SRV record has the following form:

\_service.\_proto.name. TTL class SRV priority weight port target

• service: the symbolic name of the service (xmpp-server, xmpp-client, sips).

- proto: the transport protocol of the desired service (TCP).
- name: the domain name (ending in a dot).
- · TTL: standard DNS time to live field.
- class: the standard DNS class field (this is always IN).
- priority: the priority of the target host (lower value means more preferred).
- weight: a relative weight for records with the same priority (the higher the value, the more requests will be sent).
- port: the TCP or UDP port of the service.
- target: the canonical hostname of the machine providing the service (ending in a dot).

#### Here are examples of the SRV records:

\_xmpp-server.\_tcp.voipservice.example.com. 18000 IN SRV 10 50 5269 voipservice.example.com. \_xmpp-client.\_tcp.voipservice.example.com. 18000 IN SRV 10 50 5222 voipservice.example.com. \_sips.\_tcp.voipservice.example.com. 18000 IN SRV 10 100 5061 voipservice.example.com.

You can always check whether the required SRV records are configured by executing the following commands:

dig SRV \_xmpp-client.\_tcp.voipservice.example.net dig SRV \_xmpp-server.\_tcp.voipservice.example.net dig SRV \_sips.\_tcp.voipservice.example.net

### 15.2.2.11 Check NTP Configuration

We strongly suggest that the clocks of all the nodes within the platform are synchronized. To ensure this, check that the NTP service is correctly configured on all your sip:provider PRO servers and works reliably. Execute the following command for quick test of time synchronization:

ntpq -p

If the current node synchronizes with an NTP server, this server will be marked by the star (\*) symbol.

## 15.2.2.12 Enable Apple/Google Mobile Push

It can be enabled for a domain or separate subscribers in the Admin Panel.

To enable the service for a domain:

- 1. Go to *Settings→Domains* and click on the *Preferences* button of the domain you want to enable Apple/Google Mobile Push for.
- 2. Go to the *Internals* group and enable the **mobile\_push\_enable** parameter.

nter	nternals				
	Attribute	Name	Value		
0	force_inbound_calls_to_peer	Force inbound calls to peer			
0	mobile_push_enable	Enable Apple/Google Mobile Push	×	🛛 🖾 Edit	
0	ip_header	IP Header Field			

## 15.2.2.13 Perform Tests

Perform tests when the application is available:

- 1. Download and install the application.
- 2. Open the application and input your registration username in the username@domain.name format and password.
- 3. Review the quality of application branding.
- 4. Make test calls.
- 5. Test the presence functionality.
- 6. Test the chat and group chat.
- 7. Test messaging.
- 8. Test the sharing functionality (e.g. pictures, video and voice messages and maps).
- 9. Check the application phone book integration with the phone's one

Make sure that the subscribers can start using your services in the easiest possible way.

## 15.3 Lawful Interception

## 15.3.1 Introduction

The Sipwise sip:provider PRO, as a communications platform carrying voice, fax and messaging data has to provide means for lawful interception of the content of communication by third party entities. Those Law Enforcement Agencies (LEAs) have to be able to connect to the Sipwise NGCP platform in a standardized way — ETSI, 3GPP and other organisations define the interface (and data exchange) between telecommunication operators and LEAs.

High level overview of lawful interception is shown in the following figure:

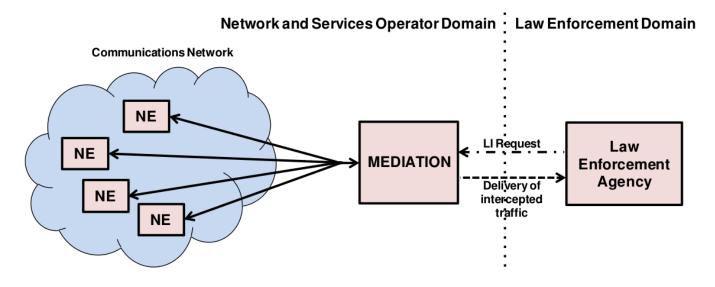
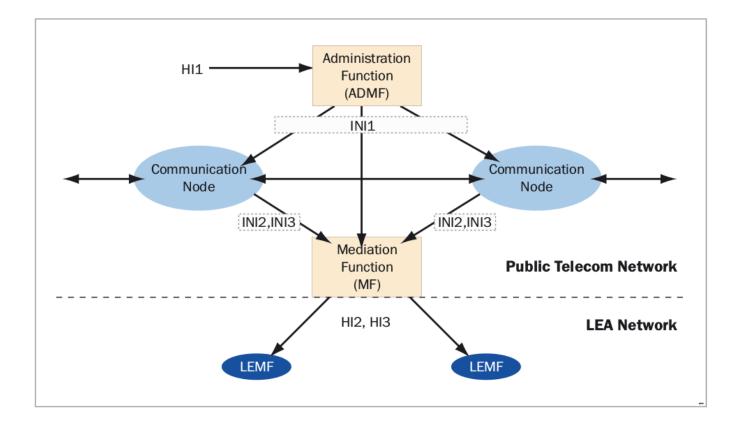


Figure 117: LI: High Level Overview

Main interfaces of lawful interception according to ETSI standard:



## Figure 118: LI: ETSI Interfaces

## 15.3.1.1 Terms and Abbreviations

## **Content of Communication (CC)**

Information exchanged between two or more users of a telecommunications service, excluding Intercept Related Information.

#### Note

This includes information which may, as part of some telecommunications service, be stored by one user for subsequent retrieval by another.

## **CC Internal Interception Function (CC-IIF)**

The CC-IIF shall cause the CC, specified by the CCTF, via the CCCI to be duplicated and passed to the MF.

## Content of Communication Control Interface (CCCI)

Carries controls information from the CCTF to the CC-IIF.

## **CC Trigger Function (CCTF)**

The purpose of the CCTF is to determine the location of the CC-IIF device associated to the target CC traffic, and to control the CC-IIF via the CCCI interface.

## Content of Communication Trigger Interface (CCTI)

Carries trigger information from the IRI-IIF to the CCTF.

## Handover Interface (HI)

Physical and logical interface across which the interception measures are requested from an operator, and the results of interception are delivered from an operator to an LEMF.

## **Intercept Related Information (IRI)**

Collection of information or data associated with telecommunication services involving the target identity, specifically call or service associated information or data (e.g. call identifier, unsuccessful call attempts) and location information.

## Intercept Related Information Internal Interception Function (IRI-IIF)

The purpose of the IRI-IIF is to generate IRI information associated with sessions, calls, connections and any other information involving interception targets identified by Law Enforcement Agency (LEA) sessions.

## **Internal Network Interface (INI)**

Network's internal interface between the Internal Intercepting Function and a mediation function.

## Law Enforcement Agency (LEA)

Organization authorized, by a lawful authorization based on a national law, to request interception measures and to receive the results of telecommunications interceptions.

### Law Enforcement Monitoring Facility (LEMF)

Law enforcement facility designated as the transmission destination for the results of interception relating to a particular interception subject.

## Lawful Interception Administration Function (AF)

The AF ensures that an intercept request from a LEA for IRI or CC or both is provisioned for collection from the network, and subsequent delivery to the LEMF.

## Lawful Interception Mediation Function (MF)

Mechanism which passes information between an access provider or network operator or service provider and a handover interface.

- 1. Firstly it receives information related to active intercepts from the IRI-IIF(s) and CC-IIF(s) within the service provider network.
- Secondly correlates and formats that IRI and CC information in real time for delivery to the LEMF over the HI2 and HI3 handover Interfaces.

## X1, X2 and X3 Interfaces

The 3GPP standard for Lawful Interception defines the handover interfaces with different names compared to the ETSI standard. The X*n* interface corresponds to the INI*n* interface and is functionally identical to the INI*n* interface.

## 15.3.2 Architecture and Configuration of LI Service

Sipwise sip:provider PRO platform implements the functions defined by LI requirements in a way that it relies on a third party provider for the Lawful Interception Mediation Function (MF).

Regarding other LI functions that are defined by ETSI / 3GPP standards there are 2 possible implementations:

- *Call loopback to NGCP*: the LI peer receives the call, extracts IRI and CC data and then routes the call back to NGCP. NGCP handles the looped back call as if that was initiated from NGCP and sets up the second call leg to the destination.
- *Call forwarded by peer* directly to destination: in this case NGCP will handle the call to LI peer as an ordinary second call leg to the destination.
- 2. Sipwise **NGCP itself provides** the required LI functions: AF and call data capturing; IRI and CC of intercepted calls are fowarded to the third party MF from NGCP.

This handbook will discuss the second setup in details in the following sections.

The below figure illustrates the logical connection of LI functions on Sipwise NGCP.

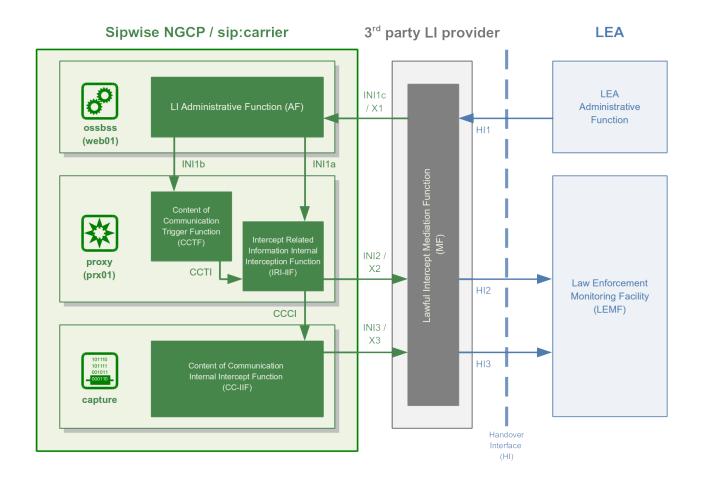


Figure 119: LI with 3rd Party Provider

## 15.3.2.1 Architecture Based on Captagent Module

### Note

This kind of LI implementation will be phased out in future NGCP releases. A short description is kept here for reference, as NGCP still (as of version mr4.5.2) supports LI services with captagent module.

The captagent based implementation of LI functions on Sipwise NGCP includes the following components:

- captagent: a software module provided by a third party; its operation can be summarized as follows:
  - 1. the captagent process gets LI requests through an API
  - 2. the process listens for and analyses SIP (*INVITE*) messages; based on the message headers it decides whether the SIP session must be intercepted
  - 3. in case the session must be intercepted, captagent sends IRI through X2 interface to the MF element
  - 4. based on the SDP data, the process captures session media and forwards that through X3 interface to the MF element
- third party MF: Group2000's LIMA system playes the role of Mediation Function (MF) and interacts with captagent module, using X1, X2 and X3 interfaces.

## 15.3.2.2 Architecture Based on Voisniff-NG Module

Although the implementation of LI services with captagent is still available and configurable on sip:provider PRO, Sipwise suggests deploying a revised solution with its voisniff-ng software module. This newer implementation also relies on a 3rd party LI provider representing the LI Mediation Function (MF), where Sipwise currently (as of NGCP version mr4.5.2) cooperates with Group2000, Pine and Utimaco.

## Sipwise NGCP components providing LI functions:

- ngcp-panel: this module is responsible for managing REST API for the whole NGCP in general
  - runs on: the active node (sp1 /sp2) on a sip:provider PRO platform
  - LI functions: AF; INI1 / X1 interface towards the MF
- · kamailio-proxy: this module serves as a generic call control function on the NGCP
  - runs on: the active node (sp1 /sp2) on a sip:provider PRO platform
  - LI functions: CCTF and IRI-IIF; INI2 / X2 interface towards the MF
- · voisniff-ng: this module is a generic element for capturing SIP and RTP traffic on the NGCP
  - runs on: the active node (sp1 /sp2) on a sip:provider PRO platform
  - LI functions: CC-IIF; INI3 / X3 interface towards the MF

#### Note

The voisniff-ng module is installed and activated by default on a Sipwise sip:provider PRO platform. It provides a possibility to get call statistics through Admin web interface, and is not readily configured for LI services.

#### Authentication and Confidentiality

It is required that the communication between the telecommunication operator's network element (that is: Sipwise NGCP) and the MF be authenticated and confidential, since the intercepted session related data and content of communication must not be disclosed to any 3rd party. For this purpose NGCP's LI service applies authentication and LI session data encryption based on public key cryptography mechanism (TLS).

Both Sipwise NGCP and the MF must authenticate themselves by certificates, for this reason the NGCP operator must ensure that valid certificates are deployed on the system. There is a need to contact the 3rd party LI provider, so that he can provide the necessary client certificates that NGCP will use to setup secured connection to the MF on X2 and X3 interfaces.

Similarly, the MF provider must contact the NGCP operator to offer him valid client certificates that the MF element will use to establish secured connection to the NGCP on X1 interface.

## 15.3.2.3 Configuration of LI Service

In order to enable LI services on sip:provider PRO the platform administrator has to explicitly enable lawful interception through the main configuration file (config.yml).

Here below is a sample configuration, which shows parameters of intercept and voisniff sections.

```
intercept:
  captagent:
   cin_max: '3000'
   cin_min: '0'
    country_code: '49'
    debug: '7'
   filter: 'port 5080'
   license: ''
    port: '18090'
   prefix_len: '3'
    schema: http
  enabled: no
  peer:
    acc: no
   inbound_prefix: LI_
    outbound_prefix: intercept_
  type: none
voisniff:
  admin_panel: yes
  daemon:
   bpf: 'port 5060 or 5062 or ip6 proto 44 or ip[6:2] & 0x1fff != 0'
```

```
external_interfaces: 'eth0 eth2'
  filter:
    exclude:
        active: '0'
        case_insensitive: '1'
        pattern: '\ncseq: *\d+ +(register|notify|options)'
    include: []
  internal_interfaces: lo
  li_x1x2x3:
    call_id:
      suffix:
        - _pbx-1
        - _b2b-1
    client_certificate: ''
    enabled: no
   local_name: sipwise
   x1:
     port: '18090'
 mysql_dump:
   enabled: yes
    num_threads: '4'
 mysql_dump_threads: '2'
  start: yes
  threads_per_interface: '2'
partitions:
  increment: '700000'
  keep: '10'
```

#### **Configuration Parameters**

#### intercept.enable

Set it to yes if you want to activate LI service. Default: no

#### intercept.peer.acc

Calls to be intercepted may be forwarded to LI peers. The LI peer may forward the call to the original destination, without looping the call back to NGCP. Set this parameter to yes if you want to enable billing for such calls. Default: no

## intercept.peer.inbound\_prefix

Calls to be intercepted may be forwarded to LI peers. This parameter specifies the prefix that is prepended to SIP usernames when the call is looped back to NGCP, in order to avoid sending the call again to any LI peer. Used by NGCP internally. Default: LI\_

## intercept.peer.outbound\_prefix

Calls to be intercepted may be forwarded to LI peers. This parameter specifies the prefix that is prepended to SIP usernames when the call is routed to an LI peer. It will be stripped off by rewrite rules of the peer, before sending the call effectively to the peer. Used by NGCP internally. Default: intercept\_

The LI service provider module; allowed values are:

- none: LI service is not activated
- · peer: LI service is activated and call data capturing is performed by SIP peers
- captagent: LI service is activated and call data capturing is performed by captagent module
- voisniff: LI service is activated and call data capturing is performed by voisniff module

Default: none

#### voisniff.admin\_panel,voisniff.daemon.mysql\_dump.\*,voisniff.partitions.\*

These parameters are not used in LI configuration, but only for call statistics which can be retrieved through the Admin web interface.

## voisniff.daemon.bpf

This sets the basic packet filter applied by <code>voisniff-ng</code> module when capturing packets on network interfaces. Default: "port 5060 or 5062 or ip6 proto 44 or ip[6:2] & 0x1fff !=0"

#### Note

The default value basically allows capturing SIP traffic only. It is usually necessary to modify the parameter in order to capture both SIP and RTP traffic. An example of such a value: "udp or ip6 proto 44 or ip[6:2] & 0x1fff !=0".

#### voisniff.daemon.external\_interfaces

This is a list of network interfaces (typically VLAN IDs) where voisniff-ng should listen for and capture packets.

#### voisniff.daemon.filter.exclude

Additional filter to determine packets that need to be excluded from capturing.

#### voisniff.daemon.filter.include

Additional filter to determine packets that need to be included in capturing.

#### voisniff.daemon.internal\_interfaces

A list of network interfaces which are considered only for internal communication between voisniff-ng and other NGCP components. Packets on these interfaces are not captured.

## voisniff.daemon.li\_x1x2x3.call\_id

Pattern that determines which SIP Call-IDs should voisniff-ng listen for and store as IRI (Intercept Related Information).

#### voisniff.daemon.li\_x1x2x3.client\_certificate

The client certificate that NGCP uses to connect over TLS to a 3rd party LI provider.

#### voisniff.daemon.li\_x1x2x3.enabled

Set it to yes to enable LI services via X1, X2 and X3 interfaces. Default: no

#### voisniff.daemon.li\_x1x2x3.local\_name

This parameter maps to the header.source field of the X2 protocol. It's an arbitrary string and can be used to identify the sending NGCP system. Default: sipwise

#### Note

As of NGCP version mr4.5.2, this is currently not used.

#### voisniff.daemon.li\_x1x2x3.private\_key

The private key used to encrypt data sent to a 3rd party LI provider.

#### voisniff.daemon.li\_x1x2x3.x1.port

The port number on which <code>voisniff-ng</code> listens for incoming X1 messages. Default: 18090

#### Note

You should leave the parameter set to the default value, unless there is a good reason to change it. The default value ensures backward compatibility with captagent LI module.

#### voisniff.daemon.start

Determines whether voisniff service must be started on the platform. Set it to yes if you'd like to activate voisniff that is needed for LI service too. Default: yes

## voisniff.daemon.threads\_per\_interface

This is a performance tuning option and controls how many threads per enabled sniffing interface should be launched. Example: if it's set to 10 and 3 interfaces are enabled for sniffing, a total of 30 threads will be launched. Default: 2

## Caution

Do not set it to a high number, or simply leave it at its default value, unless there is a performance problem with voisniff service. Please keep in mind that a high number of threads might also decrease the overall system performance of NGCP!

## 15.3.3 X1, X2 and X3 Interface Specification

Short description of Xn interfaces:

- The X1 interface is used by an LI provider to create, modify, delete and list interceptions on the Sipwise NGCP. It is designed as RESTful HTTP interface using JSON (with JSON-HAL in responses from the NGCP) as content type to provision interceptions.
- The X2 interface is a TLV based interface with JSON payload with a simple request/response mechanism over a secure TLS connection, used to pass intercepted signaling data towards an LI provider.
- The X3 interface is also a TLV based interface with a binary payload encapsulating the intercepted RTP data.

## 15.3.3.1 X1 Interface

The resource used to work with interceptions is always https://ngcp-ip:1443/api/interceptions/

#### Authentication

Authentication and authorization on the NGCP API is performed via HTTP Basic Auth or SSL Client certificates.

• HTTP Basic Auth: With cURL use --user username:password option to specify your access credentials.

curl -i - X GET -- user myuser:mypassword https://example.org:1443/api/interceptions/

Additionally use the --insecure option if you are testing against a self-signed server certificate.

• SSL Client Authentication: You can generate and download client certificates for administrators and resellers via the NGCP Panel in the Administrators view.

For the actual client authentication, you will need two files which you can download from the panel after creating the client certificates:

- 1. The client certificate generated via the NGCP Panel. This is usually labelled NGCP-API-client-certificate-xxxx.pem.
- 2. The CA certificate used to sign the server certificate, in case it as been self-signed or the CA is not recognized by the client host environment.

With *cURL* use --cert /path/to/NGCPAPIclientcertificatexxxxx.pem to specify the client certificate, and --cacert /path/to/cacert.pem to specify the CA certificate in case of a self-signed server certificate.

curl -i - X GET --cert /path/to/NGCPAPIclientcertificatexxxxx.pem \
-- cacert /path/to/cacert.pem https://example.org:1443/api/interceptions/

Additionally use the --insecure option if you are testing against a self-signed server certificate.

#### **API Description**

#### **Collection Actions**

Allowed methods for the collection as in METHOD /api/interceptions/

- OPTIONS
- POST
- GET
- HEAD

#### **Item Actions**

Allowed methods for a collection item as in METHOD /api/interceptions/id

- PATCH
- OPTIONS
- DELETE
- PUT
- GET
- HEAD

#### Properties

- liid (Number): The LI ID for this interception.
- number (String): The number to intercept.

- x2\_host (String): The IP address of the X2 interface.
- x2\_password (null, String): The password for authenticating on the X2 interface.
- x2\_port (Number): The port of the X2 interface.
- x2\_user (null, String): The username for authenticating on the X2 interface.
- x3\_host (null, String): The IP address of the X3 interface.
- x3\_port (null, Number): The port of the X3 interface.
- x3\_required (null, Boolean): Whether to also intercept call content via X3 interface (false by default).

#### **Query Parameters**

- liid: Filter for interceptions of a specific interception ID
- number: Filter for interceptions of a specific number (in E.164 format)
- order\_by: Order collection by a specific attribute. Possible values are: id, reseller\_id, liid, number, c c\_required, delivery\_host, delivery\_port, delivery\_user, delivery\_pass, modify\_t imestamp, create\_timestamp, deleted, uuid, sip\_username, sip\_domain, cc\_delivery \_host, cc\_delivery\_port
- order\_by\_direction: Direction which the collection should be ordered by. Possible values are: asc (default), desc

### **API Examples**

#### Get a specific interception

#### · Request:

```
curl - i -- insecure -- user administrator:administrator - X GET
https://localhost:1443/api/interceptions/528
```

```
HTTP/1.1 200 OK
Server: nginx
Date: Tue, 01 Dec 2015 09:43:41 GMT
ContentType: application/hal+json; profile="http://purl.org/sipwise/ngcpapi/";
    charset=utf8
ContentLength: 634
Connection: keepalive
Link: </api/interceptions/>; rel=collection
Link: </http://purl.org/sipwise/ngcpapi/>; rel=profile
Link: </api/interceptions/528>; rel="item self"
SetCookie: ngcp_panel_session=35b56d921c36c1fc6edb8fcd0a86dd9af6lec62a; path=/;
    expires=Tue, 01 Dec 2015 10:43:41 GMT; HttpOnly
StrictTransportSecurity: maxage=15768000
    {
        "_links" : {
            "collection" : {
```

```
"href" : "/api/interceptions/"
  },
  "curies" : {
    "href" : "http://purl.org/sipwise/ngcpapi/#rel {rel}",
    "name" : "ngcp",
    "templated" : true
  },
  "profile" : {
    "href" : "http://purl.org/sipwise/ngcpapi/"
  },
  "self" : {
   "href" : "/api/interceptions/528"
  }
},
"id" : 528,
"liid" : 918273,
"number" : "0014155550132",
"x2_host" : "192.168.42.42",
"x2_password" : null,
"x2_port" : 3002,
"x2_user" : null,
"x3_host" : "192.168.42.42",
"x3_port" : 3003,
"x3_required" : true
```

#### Get all interceptions for a number

## • Request:

```
curl - i -- insecure -- user administrator:administrator - X GET \
https://localhost:1443/api/interceptions/?number=0014155550132
```

```
HTTP/1.1 200 OK
Server: nginx
Date: Tue, 01 Dec 2015 09:47:36 GMT
ContentType: application/hal+json; profile="http://purl.org/sipwise/ngcpapi/";
    charset=utf8
ContentLength: 1283
Connection: keepalive
SetCookie: ngcp_panel_session=238550c5737058db619b183d925b5f9a61261cfe; path=/;
    expires=Tue, 01 Dec 2015 10:47:36 GMT; HttpOnly
StrictTransportSecurity: maxage=15768000
{
    "_embedded" : {
        "gep:interceptions" : {
```

```
"_links" : {
         "collection" : {
            "href" : "/api/interceptions/"
         },
         "curies" : {
            "href" : "http://purl.org/sipwise/ngcpapi/#rel {rel}",
            "name" : "ngcp",
            "templated" : true
         },
         "profile" : {
            "href" : "http://purl.org/sipwise/ngcpapi/"
         },
         "self" : {
            "href" : "/api/interceptions/520"
         }
      },
      "id" : 520,
      "liid" : 1,
      "number" : "0014155550132",
      "x2_host" : "192.168.42.42",
      "x2_password" : null,
      "x2_port" : 3002,
      "x2_user" : null,
      "x3_host" : "192.168.42.42",
      "x3_port" : 3003,
      "x3_required" : true
  }
},
"_links" : {
   "curies" : {
      "href" : "http://purl.org/sipwise/ngcpapi/#rel {rel}",
      "name" : "ngcp",
      "templated" : true
  },
   "ngcp:interceptions" : {
     "href" : "/api/interceptions/520"
   },
   "profile" : {
     "href" : "http://purl.org/sipwise/ngcpapi/"
  },
   "self" : {
      "href" : "/api/interceptions/?page=1&rows=10"
  }
},
"total_count" : 1
```

#### Get all interceptions for all numbers

#### • Request:

```
curl - i -- insecure -- user administrator:administrator - X GET \
https://localhost:1443/api/interceptions/
```

```
HTTP/1.1 200 OK
Server: nginx
Date: Tue, 01 Dec 2015 09:43:18 GMT
ContentType: application/hal+json; profile="http://purl.org/sipwise/ngcpapi/";
 charset=utf8
ContentLength: 2364
Connection: keepalive
SetCookie: ngcp_panel_session=68398eea5bdd3885ad0517e1f6d367ccc80111fa; path=/;
 expires=Tue, 01 Dec 2015 10:43:18 GMT; HttpOnly
StrictTransportSecurity: maxage=15768000
{
   "_embedded" : {
      "ngcp:interceptions" : [
        {
            "_links" : {
               "collection" : {
                  "href" : "/api/interceptions/"
               },
               "curies" : {
                  "href" : "http://purl.org/sipwise/ngcpapi/#rel {rel}",
                  "name" : "ngcp",
                  "templated" : true
               },
               "profile" : {
                  "href" : "http://purl.org/sipwise/ngcpapi/"
               },
               "self" : {
                  "href" : "/api/interceptions/520"
               }
            },
            "id" : 520,
            "liid" : 1,
            "number" : "0014155550132",
            "x2_host" : "192.168.42.42",
            "x2_password" : null,
            "x2_port" : 3002,
            "x2_user" : null,
            "x3_host" : "192.168.42.42",
            "x3_port" : 3003,
            "x3_required" : true
```

```
},
     {
         "_links" : {
            "collection" : {
              "href" : "/api/interceptions/"
           },
            "curies" : {
              "href" : "http://purl.org/sipwise/ngcpapi/#rel{rel}",
              "name" : "ngcp",
              "templated" : true
           },
            "profile" : {
              "href" : "http://purl.org/sipwise/ngcpapi/"
           },
            "self" : {
              "href" : "/api/interceptions/528"
           }
        },
         "id" : 528,
         "liid" : 918273,
        "number" : "0014155550132",
         "x2_host" : "192.168.42.42",
        "x2_password" : null,
                                    "x2_user" : null,
         "x2_port" : 3002,
        "x3_host" : "192.168.42.42",
         "x3_port" : 3003,
        "x3_required" : true
     }
  1
},
"_links" : {
  "curies" : {
     "href" : "http://purl.org/sipwise/ngcpapi/#rel {rel}",
     "name" : "ngcp",
     "templated" : true
  },
  "ngcp:interceptions" : [
    {
        "href" : "/api/interceptions/520"
     },
     {
        "href" : "/api/interceptions/528"
     }
  ],
  "profile" : {
     "href" : "http://purl.org/sipwise/ngcpapi/"
  },
   "self" : {
```

```
"href" : "/api/interceptions/?page=1&rows=10"
}
},
"total_count" : 2
```

#### Get interception for specific LIID

• Request:

```
curl - i -- insecure -- user administrator:administrator -X GET \
https://localhost:1443/api/interceptions/?liid=9876
```

```
HTTP/1.1 200 OK
Server: nginx
Date: Tue, 01 Dec 2015 09:50:41 GMT
ContentType: application/hal+json; profile="http://purl.org/sipwise/ngcpapi/";
 charset=utf8
ContentLength: 1283
Connection: keepalive
SetCookie: ngcp_panel_session=23960dde6bb90f0c5c84575890194c53cce120ce; path=/;
 expires=Tue, 01 Dec 2015 10:50:40 GMT; HttpOnly
StrictTransportSecurity: maxage=15768000
{
   "_embedded" : {
      "ngcp:interceptions" : {
         "_links" : {
            "collection" : {
               "href" : "/api/interceptions/"
            },
            "curies" : {
               "href" : "http://purl.org/sipwise/ngcpapi/#rel {rel}",
               "name" : "ngcp",
               "templated" : true
            },
            "profile" : {
               "href" : "http://purl.org/sipwise/ngcpapi/"
            },
            "self" : {
               "href" : "/api/interceptions/520"
            }
         },
         "id" : 520,
         "liid" : 1,
         "number" : "0014155550132",
         "x2_host" : "192.168.42.42",
```

```
"x2_password" : null,
      "x2_port" : 3002,
      "x2_user" : null,
      "x3_host" : "192.168.42.42",
      "x3_port" : 3003,
      "x3_required" : true
   }
},
"_links" : {
   "curies" : {
     "href" : "http://purl.org/sipwise/ngcpapi/#rel {rel}",
      "name" : "ngcp",
      "templated" : true
  },
   "ngcp:interceptions" : {
     "href" : "/api/interceptions/520"
  },
   "profile" : {
      "href" : "http://purl.org/sipwise/ngcpapi/"
   },
   "self" : {
      "href" : "/api/interceptions/?page=1&rows=10"
  }
},
"total_count" : 1
```

#### Create interception for a specific number

Request:

```
curl - i -- insecure -- user administrator:administrator -X POST \
- H "ContentType: application/json" -- data \
'{"liid":123, "number":"31032222203", "x2_host":"127.0.0.1", "x2_port":12345,
    "x3_required":true, "x3_host":"127.0.0.2", "x3_port":23456}' \
https://localhost:1443/api/interceptions/
```

Response:

```
HTTP/1.1 201 Created
TransferEncoding: chunked
Connection: close
Location: /api/interceptions/528
SetCookie: ngcp_panel_session=e7817079d121fae4d86448b10e1fa21d0201c526; path=/;
    expires=Tue, 01 Dec 2015 10:43:18 GMT; HttpOnly
StrictTransportSecurity: maxage=15768000
```

The path to the newly created interception is found in the Location header of the response.

#### Update specific interception

#### · Request:

```
curl - i -- insecure -- user administrator:administrator -X PUT \
- H "ContentType: application/json" - H 'Prefer: return=representation' -- data \
'{"liid":918273, "number":"0014155550132", "x2_host":"192.168.42.42", "x2_port":5000,
    "x3_required":false}' \
https://localhost:1443/api/interceptions/123
```

```
HTTP/1.1 200 OK
ContentType: application/hal+json; profile="http://purl.org/sipwise/ngcpapi/";
 charset=utf8
ContentLength: 621
Link: </api/interceptions/>; rel=collection
Link: <http://purl.org/sipwise/ngcpapi/>; rel=profile
Link: </api/interceptions/530>; rel=self
PreferenceApplied: return=representation
SetCookie: ngcp_panel_session=0b56e4a197b0e9f6e22a998e85473a0184770740; path=/;
  expires=Tue, 01 Dec 2015 10:56:17 GMT; HttpOnly
{
   "_links" : {
      "collection" : {
         "href" : "/api/interceptions/"
      },
      "curies" : {
         "href" : "http://purl.org/sipwise/ngcpapi/#rel {rel}",
         "name" : "ngcp",
         "templated" : true
      },
      "profile" : {
         "href" : "http://purl.org/sipwise/ngcpapi/"
      },
      "self" : {
         "href" : "/api/interceptions/530"
      }
   },
   "id" : 530,
   "liid" : 918273,
   "number" : "0014155550132",
   "x2_host" : "192.168.42.42",
   "x2_password" : null,
   "x2_port" : 5000,
   "x2_user" : null,
   "x3_host" : null,
   "x3_port" : null,
   "x3_required" : false
```

}

The Prefer: return=representation header forces the API to return the content, otherwise status 201 with no content is returned.

#### Update only certain items for a specific interception

#### · Request:

```
curl - i -- insecure -- user administrator:administrator - X PATCH \
- H "ContentType: application/jsonpatch+json" - H 'Prefer: return=representation' \
--data '[{"op":"replace", "path":"/x2_host", "value":"192.168.42.42"}, {"op":"replace",
    "path":"/x2_port", "value":4000}]' \
https://localhost:1443/api/interceptions/530
```

```
• Response:
```

```
HTTP/1.1 200 OK
Server: nginx
Date: Tue, 01 Dec 2015 10:06:06 GMT
ContentType: application/hal+json; profile="http://purl.org/sipwise/ngcpapi/";
 charset=utf8
ContentLength: 620
Connection: close
Link: </api/interceptions/>; rel=collection
Link: <http://purl.org/sipwise/ngcpapi/>; rel=profile
Link: </api/interceptions/530>; rel=self
PreferenceApplied: return=representation
SetCookie: ngcp_panel_session=0693129d63d543a85f96d464ff9a8f807cfc4d18; path=/;
 expires=Tue, 01 Dec 2015 11:06:06 GMT; HttpOnly
StrictTransportSecurity: maxage=15768000
{
   "_links" : {
      "collection" : {
         "href" : "/api/interceptions/"
      },
      "curies" : {
         "href" : "http://purl.org/sipwise/ngcpapi/#rel {rel}",
         "name" : "ngcp",
         "templated" : true
      },
      "profile" : {
         "href" : "http://purl.org/sipwise/ngcpapi/"
      },
      "self" : {
         "href" : "/api/interceptions/530"
      }
   },
   "id" : 530,
```

```
404 / 486
```

```
"liid" : 918273,
"number" : "0014155550132",
"x2_host" : "192.168.42.42",
"x2_password" : null,
"x2_port" : 4000,
"x2_user" : null,
"x3_host" : null,
"x3_port" : null,
"x3_required" : false
```

## **Delete specific interception**

### • Request:

```
curl - i -- insecure -- user administrator:administrator -X DELETE \
https://localhost:1443/api/interceptions/123
```

## Response:

```
HTTP/1.1 204 No Content
Server: nginx
Date: Tue, 01 Dec 2015 10:08:49 GMT
Connection: keepalive
SetCookie: ngcp_panel_session=570c66b66732629766f86b8ed9bd0d64902ae73e; path=/;
    expires=Tue, 01 Dec 2015 11:08:49 GMT; HttpOnly
XCatalyst: 5.90042
StrictTransportSecurity: maxage=15768000
```

## 15.3.3.2 X2 Interface

The communication via the X2 interface consists of request-response pairs.

#### Request

The request is formatted as: X2/<bodylength>/<body>

Body part has the following items:

Element	Туре	Length	Description
/x2/header/source	String	arbitrary	identifier of Sipwise node which captured the data
		length	
/x2/header/destination	String	arbitrary	identifier of LI mediation system
		length	

Table 23: X2 Message Body Items

Element	Туре	Length	Description
/x2/header/type	String	arbitrary	always "sip" (but later potentially "xmpp" and others too)
		length	
/x2/header/version	PosInteger	arbitrary	always "1"
		length	
/x2/header/timestamp	String	27 chars	format: YYYY-MM-DDThh:mm:ss.ffffffZ; timestamp in
			UTC when the X2 package is sent to mediation
/x2/body/dialogid	PosInteger	arbitrary	globally increasing counter for each new communication
		length	dialog (e.g. call)
/x2/body/messageid	PosInteger	arbitrary	increasing counter for each new x2 message within a
		length	dialog, starting from 0
/x2/body/timestamp	String	27 chars	format: YYYY-MM-DDThh:mm:ss.ffffffZ; timestamp in
			UTC when the package has been captured on the wire
/x2/body/interceptions			one or more elements containing the following
			information, one element per intercepted target:
/x2/body/interceptions/liid	PosInteger	arbitrary	interception id ("liid") as set via X1 interface
		length	
/x2/body/interceptions/direction	String	arbitrary	either "totarget" or "fromtarget" from the soft-switch
		length	perspective (if target is the called party, it is "totarget", if
			target is the calling party, it is "fromtarget").
/x2/body/data	Base64	arbitrary	content of full IP frame and up on the OSI layer; packets
	encoded		fragmented on the wire are provided in fully assembled
			format

## Table 23: (continued)

## Example of full message:

```
X2/418/
{
  "header": {
    "source": "prx01a.example.com",
    "destination": "x2destination.example.com",
    "type": "sip",
    "version": 1,
    "timestamp": "20150311T09:18:04.729803Z"
  },
  "body": {
    "dialogid": 4,
    "messageid": 0,
    "timestamp": "20150311T09:18:04.729123Z",
    "interceptions": [
      { "liid": 174, "direction": "fromtarget" },
      { "liid": 175, "direction": "totarget" }
```

```
],
   "data": "<base64 encoded ip,udp/tcp,sip frame>"
}
```

## Response

- Success: X2-ACK/0/
- Error: X2-ERR/<length>/<error string>

## Keep-Alive Mechanism

A regular keep-alive mechanism with a default value of 10s is used on the connection if it is re-used across multiple messages.

- Request: X2/0/
- Response: X2-ACK/0/

## 15.3.3.3 X3 Interface

On the X3 interface TLV based packets are sent via secured (TLS) connection on a pre-established stream. X3 messages do not need to be acknowledged, except for keep-alive messages.

## X3 Message Structure

Field	Length
Header	arbitrary
CCCID	4 bytes
Messageld	4 bytes
Timestamp	8 bytes
Payload	arbitrary

#### Table 24: X3 Message Structure

## Header Details

Table 25: X3:	Header Details
---------------	----------------

Field	Length	Content
type	2 bytes	always "X3"
delimiter	1 byte	always "/"
length	arbitrary	ASCII string

## Table 25: (continued)

Field	Length	Content
delimiter	1 byte	always "/"

## **CCCID** Details

dialogid (32 bit in network byte order, reset to 0 after 2<sup>32</sup>-1)

The dialogid is referencing the /x2/body/dialogid field in order to correlate an X3 packet to an X2 call.

### Messageld Details

messageid (32 bit in network byte order, reset to 0 after  $2^{32}$ -1)

The messageid is a counter within a dialog sequencing the X3 packets sent from the NGCP. This counter is not correlated in any way with X2, rather than starting at 0 with the first RTP packet captured within a dialog.

## **Timestamp Details**

- seconds (32 bit in network byte order)
- fraction (32 bit in network byte order)

The timestamp represents the Unix epoch starting from 1970-01-01.

## **Payload Details**

Field	Length
original ip header	20 bytes for v4, 40 bytes for v6
original udp header	8 bytes
original rtp header	variable, 12-72 bytes
original rtp payload	arbitrary

#### Keep-Alive Mechanism

A regular keep-alive mechanism with a default value of 10s is used on the connection if it is re-used across multiple messages.

- Request: X3/0/
- Response: X3-ACK/0/

# A Basic Call Flows

## A.1 General Call Setup

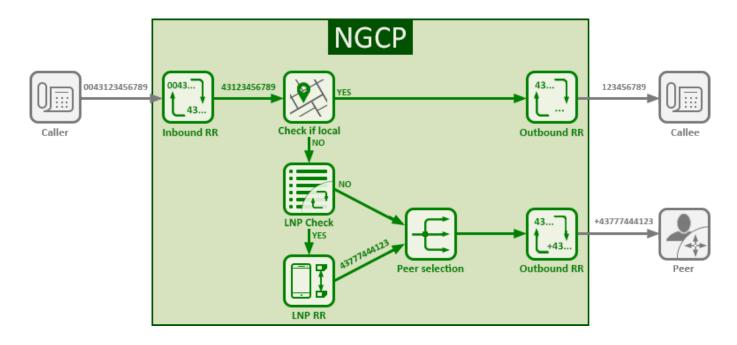


Figure 120: General Call Setup

NGCP performs the following checks when processing a call coming from a subscriber and terminated at a peer:

- Checks if the IP address where the request came from is in the list of trusted IP addresses. If yes, this IP address is taken as the identity for authentication. Otherwise, NGCP performs the digest authentication.
- When the subscriber is authorized to make the call, NGCP applies the Inbound Rewrite Rules for the caller and the callee assigned to the subscriber (if any). If there are no Rewrite Rules assigned to the subscriber, the ones assigned to the subscriber's domain are applied. On this stage the platform normalises the numbers from the subscriber's format to E.164.
- · Matches the callee (called number) with local subscribers.
  - If it finds a matching subscriber, the call is routed internally. In this case, NGCP applies the Outbound Rewrite Rules associated with the callee (if any). If there are no Rewrite Rules assigned to the callee, the ones assigned to the callee's domain are applied.
  - If it does not find a matching subscriber, the call goes to a peer as described below.
- Queries the LNP database to find out if the number was ported or not. For details of LNP queries refer to the Local Number Porting Section 5.4 chapter.
  - If it was ported, NGCP applies the LNP Rewrite Rules to the called number.
- Based on the priorities of peering groups and peering rules (see Section 4.5.2.3 for details), NGCP selects peering groups for call termination and defines their precedence.

- Within every peering group the weight of a peering server defines its probability to receive the call for termination. Thus, the bigger the weight of a server, the higher the probability that NGCP will send the call to it.
- Applies the Outbound Rewrite Rules for the caller and the callee assigned to a peering server when sending the call to it.

## A.2 Endpoint Registration

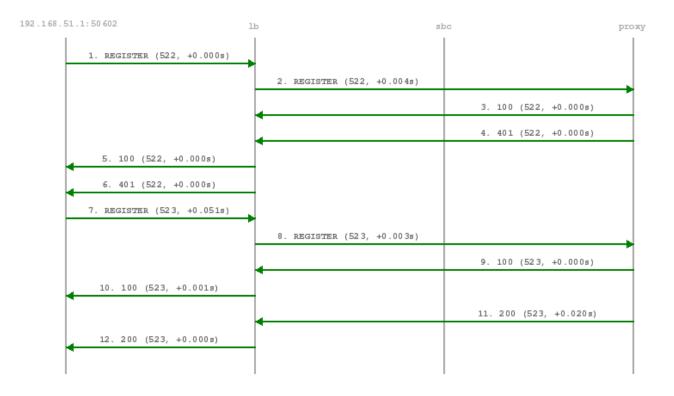


Figure 121: Registration Call-Flow

The subscriber endpoint starts sending a REGISTER request, which gets challenged by a 401. After calculating the response of the authentication challenge, it sends the REGISTER again, including the authentication response. The SIP proxy looks up the credentials of the subscriber in the database, does the same calculation, and if the result matches the one from the subscriber, the registration is granted.

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

If NAT is detected, the SIP proxy sends a OPTION message to the registered contact every 30 seconds, in order to keep the NAT binding on the NAT device open. Otherwise, for subsequent calls to this contact, the sip:provider PRO wouldn't be able to reach the endpoint behind NAT (NAT devices usually drop a UDP binding after not receiving any traffic for ~30-60 seconds).

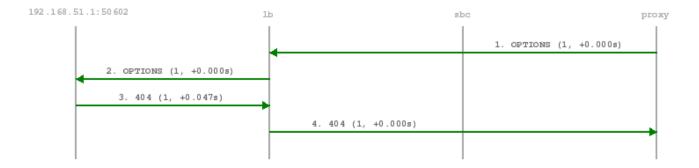
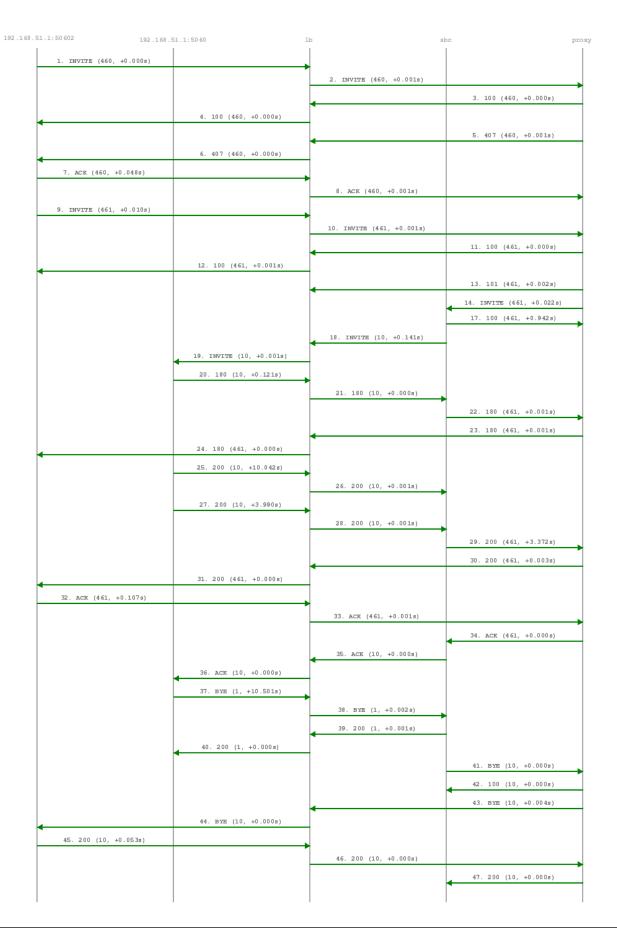


Figure 122: NAT-Ping Call-Flow

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



The calling party sends an INVITE (e.g. sip:someuser@example.org) via the SIP load-balancer to the SIP proxy. The proxy replies with an authorization challenge in the 407 response, and the calling party sends the INVITE again with authentication credentials. The SIP proxy checks if the called party is a local user. If it is, and if there is a registered contact found for this user, then (after various feature-related tasks for both the caller and the callee) the Request-URI is replaced by the URI of the registered contact (e.g. sip:me@1.2.3.4:1234;transport=UDP). If it's not a local user but a numeric user, a proper PSTN gateway is being selected by the SIP proxy, and the Request-URI is rewritten accordingly (e.g. sip:+43123456789@ 2.3.4.5:5060).

Once the proxy has finished working through the call features of both parties involved and has selected the final destination for the call, and - optionally - has invoked the Media Relay for this call, the INVITE is sent to the SIP B2BUA. The B2BUA creates a new INVITE message from scratch (using a new Call-ID and a new From-Tag), copies only various and explicitly allowed SIP headers from the old message to the new one, filters out unwanted media capabilities from the SDP body (e.g. to force audio calls to use G.711 as a codec) and then sends the new message via the SIP load-balancer to the called party.

SIP replies from the called party are passed through the elements back to the calling party (replacing various fields on the B2BUA to match the first call leg again). If a reply with an SDP body is received by the SIP proxy (e.g. a 183 or a 200), the Media Relay is invoked again to prepare the ports for the media stream.

Once the 200 is routed from the called party to the calling party, the media stream is fully negotiated, and the endpoints can start sending traffic to each outer (either end-to-end or via the Media Relay). Upon reception of the 200, the SIP proxy writes a start record for the accounting process. The 200 is also acknowledged with an ACK message from the calling party to the called party, according to the SIP 3-way handshake.

Either of the parties can tear down the media session at any time by sending a BYE, which is passed through to the other party. Once the BYE reaches the SIP proxy, it instructs the Media Relay to close the media ports, and it writes a stop record for accounting purposes. Both the start- and the stop-records are picked up by the *mediator* service in a regular interval and are converted into a Call Detail Record (CDR), which will be rated by the *rate-o-mat* process and can be billed to the calling party. For calls made by subscribers on a prepaid plan, rating occurs at call runtime and is actually done by the B2BUA (which is necessary to properly support multiple parallel calls by the same subscriber). The final rating data is then passed on to *rate-o-mat* which will update the CDRs accordingly.

## A.4 Session Keep-Alive

The SIP B2BUA acts as refresher for the Session-Timer mechanism as defined in RFC 4028. If the endpoints indicate support for the UPDATE method during call-setup, then the SIP B2BUA will use an UPDATE message if enabled per peer, domain or subscriber via Provisioning to check if the endpoints are still alive and responsive. Both endpoints can renegotiate the timer within a configurable range. All values can be tuned using the Admin Panel or the APIs using Peer-, Domain- and Subscriber-Preferences.

## Tip

Keep in mind that the values being used in the signaling are always half the value being configured. So if you want to send a keep-alive every 300 seconds, you need to provision *sst\_expires* to 600.

If one of the endpoints doesn't respond to the keep-alive messages or answers with 481 Call/Transaction Does Not Exist, then the call is torn down on both sides. This mechanism prevents excessive over-billing of calls if one of the endpoints

is not reachable anymore or "forgets" about the call. The BYE message sent by the B2BUA triggers a stop-record for accounting and also closes the media ports on the Media Relay to stop the call.

Beside the Session-Timer mechanism to prevent calls from being lost or kept open, there is a **maximum call length** of 21600 seconds per default defined in the B2BUA. This is a security/anti-fraud mechanism to prevent overly long calls causing excessive costs.

## A.5 Voicebox Calls

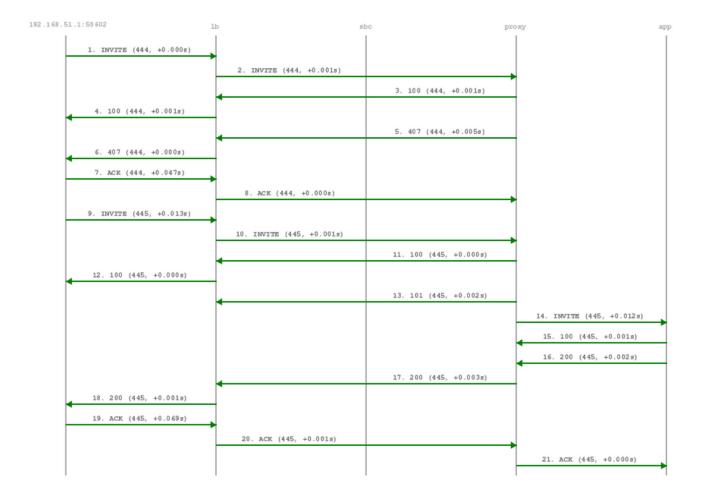


Figure 124: Voicebox Call-Flow

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

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

In case of an endpoint leaving a new message on the voicebox, the Message-Waiting-Indication (MWI) mechanism triggers the sending of a unsolicited NOTIFY message, passing the number of new messages in the body. As soon as the voicebox owner

dials into his voicebox (e.g. by calling sip:voicebox@example.org from his SIP account), another NOTIFY message is sent to his devices, resetting the number of new messages.



## Important

The sip:provider PRO does not require your device to subscribe to the MWI service by sending a SUBSCRIBE (it would rather reject it). On the other hand, the endpoints need to accept unsolicited NOTIFY messages (that is, a NOTIFY without a valid subscription), otherwise the MWI service will not work with these endpoints.

# **B** NGCP configs overview

## B.1 config.yml Overview

/etc/ngcp-config/config.yml is the main configuration YAML file used by Sipwise NGCP. After every changes it need to run the command ngcpcfg apply my commit message to apply changes (followed by ngcpcfg push in the PRO version to apply changes to sp2). The following is a brief description of the main variables contained into /etc/ngcp-config/ config.yml file.

## B.1.1 apps

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

```
apps:
    malicious_call: no
```

• malicious\_call: if set to yes, the Malicious Call Identification (MCID) application will be enabled

## B.1.2 asterisk

The following is the asterisk section:

```
asterisk:
 log:
   facility: local6
 rtp:
   maxport: 20000
   minport: 10000
 sip:
   bindport: 5070
   dtmfmode: rfc2833
 voicemail:
   enable: 'no'
   fromstring: 'Voicemail server'
   greeting:
     busy_custom_greeting: '/home/user/file_no_extension'
     busy_overwrite_default: 'no'
     busy_overwrite_subscriber: 'no'
     unavail_custom_greeting: '/home/user/file_no_extension'
     unavail_overwrite_default: 'no'
     unavail_overwrite_subscriber: 'no'
   mailbody: 'You have received a new message from {\rm MAILBOX} \leftrightarrow
       } on ${VM_DATE}.'
   mailsubject: '[Voicebox] New message ${VM_MSGNUM} in voicebox ${VM_MAILBOX}'
   max_msg_length: 180
```

```
maxgreet: 60
maxmsg: 30
maxsilence: 0
min_msg_length: 3
normalize_match: '^00|\+([1-9][0-9]+)$'
normalize_replace: '$1'
serveremail: voicebox@sip.sipwise.com
```

- · log.facility: rsyslog facility for asterisk log, defined in /etc/asterisk/logger.conf.
- rtp.maxport: RTP maximum port used by asterisk.
- · rtp.minport: RTP minimun 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 minimun length of a voicemail message, in seconds.
- voicemail.maxsilence: Maxsilence defines how long Asterisk will wait for a contiguous period of silence before terminating an incoming call to voice mail. The default value is 0, which means the silence detector is disabled and the wait time is infinite.
- · voicemail.serveremail: Provides the email address from which voicemail notifications should be sent.
- voicemail.normalize\_match: Regular expression to match the From number for calls to voicebox.
- · voicemail.normalize\_replace: Replacement string to return, in order to match an existing voicebox.

#### B.1.3 autoprov

The following is the autoprovisioning section:

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

```
server_certfile: '/etc/ngcp-config/ssl/myserver.crt'
server_keyfile: '/etc/ngcp-config/ssl/myserver.key'
ssl_enabled: 'yes'
softphone:
config_lockdown: 0
webauth: 0
```

• autoprov.skip\_vendor\_redirect: Skip phone vendor redirection to the vendor provisioning web site.

### B.1.4 backuptools

The following is the backup tools section:

```
backuptools:
  cdrexport_backup:
    enable: 'no'
  etc_backup:
    enable: 'no'
 mail:
    address: noc@company.org
    error_subject: '[ngcp-backup] Problems detected during daily backup'
    log_subject: '[ngcp-backup] Daily backup report'
    send_errors: 'no'
    send_log: 'no'
  mysql_backup:
    enable: 'no'
    exclude_dbs: 'syslog sipstats information_schema'
  rotate_days: 7
  storage_dir: '/var/backup/ngcp_backup'
  temp_backup_dir: '/tmp/ngcp_backup'
```

- · backuptools.cdrexport\_backup.enable: Enable backup of cdrexport (.csv) directory.
- backuptools.etc\_backup.enable: Enable backup of /etc/\* directory.
- backuptools.mail.address: Destination email address for backup emails.
- backuptools.mail.error\_subject: Subject for error emails.
- backuptools.mail.log\_subjetc: Subject for daily backup report.
- backuptools.mail.send\_error: Send daily backup error report.
- backuptools.mail.send\_log: Send daily backup log report.
- backuptools.mysql\_backup.enable: Enable daily mysql backup.
- backuptools.mysql\_backup.exclude\_dbs: exclude mysql databases from backup.

- backuptools.rotate\_days: Number of 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.temp\_backup\_dir: Temporary storage directory of backups.

# B.1.5 cdrexport

### The following is the cdr export section:

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

- cdrexport.daily\_folder:: Set yes if you want to create a daily folder for CDRs under the configured path.
- · cdrexport.export\_failed: Export CDR for failed calls.
- · cdrexport.export\_incoming: Export CDR for incoming calls.
- · cdrexport.exportpath: The path to store CDRs in .csv format.
- · cdrexport.full\_names: Use full namen for CDRs instead of short ones.
- cdrexport.monthly\_folder: Set yes if you want to create a monthly folder (ex. 201301 for January 2013) for CDRs under configured path.

### B.1.6 checktools

### The following is the check tools section:

```
checktools:
  collcheck:
   cpuidle: 0.1
   dfused: 0.9
   eximmaxqueue: 15
   loadlong: 2
   loadmedium: 2
   loadshort: 3
   maxage: 600
   memused: 0.7
   siptimeout: 15
   swapfree: 0.5
   active_check_enable: 1
```

```
420 / 486
```

```
asr_ner_statistics: 1
exim_check_enable: 0
force: 0
kamailio_check_concurrent_calls_enable: 0
kamailio_check_dialog_active_enable: 1
kamailio_check_dialog_early_enable: 1
kamailio_check_dialog_incoming_enable: 1
kamailio_check_dialog_local_enable: 1
kamailio_check_dialog_outgoing_enable: 1
kamailio_check_dialog_relay_enable: 1
kamailio_check_shmem_enable: 1
kamailio_check_usrloc_regdevices_enable: 1
kamailio_check_usrloc_regusers_enable: 1
mpt_check_enable: 1
mysql_check_enable: 1
mysql_check_replication: 1
oss_check_provisioned_subscribers_enable: 1
sip_check_enable: 1
sipstats_check_num_packets: 1
sipstats_check_num_packets_perday: 1
sipstats_check_partition_size: 1
snmpd:
  communities:
    public:
      - localhost
  trap_communities:
    public:
      - localhost
```

- checktools.collcheck.cpuidle: Sets the minimum value for CPU usage (0.1 means 10%).
- checktools.collcheck.dfused: Sets the maximun value for DISK usage (0.9 means 90%).
- checktools.collcheck.loadlong/loadlong/loadshort: Max values for load (long, short, medium term).
- · checktools.collcheck.maxage: Max age in seconds.
- · checktools.collcheck.memused: Sets the maximun value for MEM usage (0.7 means 70%).
- · checktools.collcheck.siptimeout: Max timeout for sip options.
- checktools.collcheck.swapfree: Sets the minimun value for SWAP free (0.5 means 50%).
- checktools.exim\_check\_enable: Exim queue check plugin for ngcp-witnessd.
- · checktools.active\_check\_enable: Active node check plugin for ngcp-witnessd.
- checktools.asr\_ner\_statistics: enable/Disable ASR/NER statistics.
- · checktools.force: Perform checks even if not active from ngcp-check-active command.

- · checktools.kamailio\_check\_\*: Enable/Disable SNMP collective check pluglin for Kamailio.
- checktools.mpt\_check\_enable: MPT raid SNMP check plugin.
- checktools.mysql\_check\_enable: MySQL SNMP check plugin.
- checktools.mysql\_check\_replication: MySQL replication check.
- checktools.oss\_check\_provisioned\_subscribers\_enable: OSS provisioned subscribers count plugin.
- checktools.sip\_check\_enable/sipstats\_check\_\*: Enable/Disable SIP check plugins.
- checktools.snmpd.communities: Sets the SNMP community and sources (separated by comma , ex. source: 127.0.0.1, 10.10.10.2, 10.10.10.3).
- checktools.snmpd.trap\_communities: Sets the SNMP TRAP community and destination sink (separated by comma , ex. sink: 127.0.0.1, 10.10.10.2, 10.10.10.3).

### B.1.7 cleanuptools

The following is the cleanup tools section:

```
cleanuptools:
  acc_cleanup_days: 90
  archive_targetdir: '/var/backups/cdr'
 binlog_days: 15
  cdr_archive_months: 2
  cdr_backup_months: 2
  cdr_backup_retro: 3
  compress: gzip
  delete_old_cdr_files:
    enabled: 'no'
   max_age_days: 30
   paths:
        max_age_days: ~
        path: '/home/jail/home/*/20[0-9][0-9][0-9][0-9][0-9][0-9]'
        remove_empty_directories: 'yes'
        wildcard: 'yes'
        max_age_days: ~
        path: '/home/jail/home/cdrexport/resellers/*/20[0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
        remove_empty_directories: 'yes'
        wildcard: 'yes'
       max_age_days: ~
        path: '/home/jail/home/cdrexport/system/20[0-9][0-9][0-9][0-9]/[0-9][0-9]'
        remove_empty_directories: 'yes'
        wildcard: 'yes'
  sql_batch: 10000
```

trash\_cleanup\_days: 30

- cleanuptools.acc\_cleanup\_days: CDR records in acc table in kamailio database will be deleted after this time
- · cleanuptools.binlog\_days: Time after MySQL binlogs will be deleted.
- cleanuptools.cdr\_archive\_months: How many months worth of records to keep in monthly CDR backup tables, instead of dumping them into archive files and dropping them from database.
- cleanuptools.cdr\_backup\_months: How many months worth of records to keep in the current cdr table, instead of moving them
  into the monthly CDR backup tables.
- cleanuptools.cdr\_backup\_retro: How many months to process for backups, going backwards in time and skipping cdr\_backup\_months months first, and store them in backup tables. Any older record will be left untouched.
- cleanuptools.delete\_old\_cdr\_files:
  - enabled: Enable (yes) or disable (no) exported CDR cleanup.
  - max\_age\_days: Gives the expiration time of the exported CDR files in days. There is a general value which may be overridden by a local value provided at a specific path. The local value is valid for the particular path only.
  - paths: an array of path definitions
    - \* path: a path where CDR files are to be found and deleted; this may contain wildcard characters
    - \* wildcard: Enable (yes) or disable (no) using wildcards in the path
    - \* remove\_empty\_directories: Enable (yes) or disable (no) removing empty directories if those are found in the given path
    - \* max\_age\_days: the local expiration time value for files in the particular path
- · cleanuptools.sql\_batch: How many records to process within a single SQL statement.
- cleanuptools.trash\_cleanup\_days: Time after CDRs from acc\_trash and acc\_backup tables in kamailio database will be deleted.

For the description of *cleanuptools* please visit Cleanuptools Description Section 12.5 section of the handbook.

### B.1.8 cluster\_sets

The following is the cluster sets section:

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

- cluster\_sets.<label>: an arbitrary label of the cluster set; in the above example we have default
- cluster\_sets.<label>.dispatcher\_id: a unique, numeric value that identifies a particular cluster set
- cluster\_sets.default\_set: selects the default cluster set
- cluster\_sets.type: the type of cluster set; can be central or distributed

### B.1.9 database

The following is the database section:

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

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

### B.1.10 faxserver

The following is the fax server section:

```
faxserver:
  enable: yes
  fail_attempts: '3'
  fail_retry_secs: '60'
  mail_from: 'Sipwise NGCP FaxServer <voipfax@ngcp.sipwise.local>'
```

- faxserver.enable: yes/no to enable or disable ngcp-faxserver on the platform respectively.
- faxserver.fail\_attempts: Amount of attempts to send a fax after which it is marked as failed.
- · faxserver.fail\_retry\_secs: Amount of seconds to wait between "fail\_attemts".
- · faxserver.mail\_from: Sets the e-mail From Header for incoming fax.

#### B.1.11 general

The following is the general section:

```
general:
   adminmail: adjust@example.org
   companyname: sipwise
   lang: en
   maintenance: no
   production: yes
   timezone: localtime
```

- · general.adminmail: Email address used by monit to send notifications to.
- general.companyname: Label used in SNMPd configuration.
- general.lang: Sets sounds language (e.g: de for German)
- · general.production: Label to hint self-check scripts about installation mode.
- general.maintenance: maintenance mode necessary for safe upgrades.
- · general.timezone: sip:provider PRO Timezone

### B.1.12 heartbeat

The following is the heartbeat section:

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

- · heartbeat.hb\_watchdog.enable: Enable heartbeat watchdog in order to prevent and fix split brain scenario.
- heartbeat.hb\_watchdog.action\_max: Max errors before taking any action.
- heartbeat.hb\_watchdog.interval: Interval in secs for the check.
- heartbeat.hb\_watchdog.transition\_max: Max checks in transition state.
- heartbeat.pingnodes: List of pingnodes for heartbeat. Minimun 2 entries, otherwise by default NGCP will set the default gateway and DNS servers as pingnodes.

### B.1.13 intercept

The following is the legal intercept section:

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

• intercept.captagent.enable: Enable captagent for Lawful Interception (addictional NGCP module).

### B.1.14 kamailio

The following is the kamailio section:

```
kamailio:
    lb:
    debug: 'no'
    extra_sockets: ~
    max_forwards: 70
    nattest_exception_ips:
```

```
- 1.2.3.4
    - 5.6.7.8
  pkg_mem: 16
  port: 5060
  security:
    dos_ban_enable: 'yes'
    dos_ban_time: 300
    dos_reqs_density_per_unit: 50
    dos_sampling_time_unit: 5
    dos_whitelisted_ips: ~
    dos_whitelisted_subnets: ~
    failed_auth_attempts: 3
    failed_auth_ban_enable: 'yes'
    failed_auth_ban_time: 3600
  shm_mem: 2012
  start: 'yes'
  strict_routing_safe: 'no'
  tcp_children: 8
  tcp_max_connections: 2048
  tls:
    enable: 'no'
    port: 5061
    sslcertfile: '/etc/kamailio/kamailio-selfsigned.pem'
    sslcertkeyfile: '/etc/kamailio/kamailio-selfsigned.key'
  udp_children: 8
  use_dns_cache: 'on'
proxy:
  allow_info_method: 'no'
  allow_peer_relay: 'no'
  allow_refer_method: 'no'
  authenticate_bye: 'no'
  cf_depth_limit: 10
  children: 8
  debug: 'no'
  default_expires: 3600
  enum_suffix: e164.arpa.
  filter_100rel_from_supported: 'yes'
  filter_failover_response: '408|500|503'
  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
```

```
426 / 486
```

```
-118[0-9]{2}
foreign_domain_via_peer: 'no'
ignore_auth_realm: 'no'
keep_original_to: 'no'
lnp:
  api:
    invalid_lnp_routing_codes:
      - ^EE00
      - ^DD00
    lnp_request_blacklist: []
    lnp_request_whitelist: []
    request_timeout: '1000'
  enabled: no
  type: api
max_expires: 43200
max_gw_lcr: 128
max_registrations_per_subscriber: 5
min_expires: 60
nathelper_dbro: 'no'
natping_interval: 30
natping_processes: 7
nonce_expire: 300
pbx:
  hunt_display_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: yes
perform_peer_lcr: 0
pkg_mem: 16
port: 5062
presence:
  enable: 'yes'
  max_expires: '3600'
  reginfo_domain: example.org
proxy_lookup: 'no'
set_ruri_to_peer_auth_realm: 'no'
shm_mem: 64
start: 'yes'
tcp_children: 4
use_enum: 'no'
usrloc_dbmode: 1
```

- kamailio.lb.debug: Enable intensive debug level.
- · kamailio.lb.extra\_sockets: Add here extra sockets for Load Balancer.

- · kamailio.lb.max\_forwards: Set the value for the Max Forwards SIP header for outgoing messages.
- kamailio.lb.nattest\_exception\_ips: List of IPs that don't need the NAT test.
- kamailio.lb.shm\_mem: Shared memory used by Kamailio Load Balancer.
- kamailio.lb.pkg\_mem: PKG memory used by Kamailio Load Balancer.
- kamailio.lb.security.dos\_ban\_enable: Enable/Disable DoS Ban.
- kamailio.lb.security.dos\_ban\_time: Sets the ban time.
- kamailio.lb.security.dos\_reqs\_density\_per\_unit:: Sets the requests density per unit (if we receive more then \* lb.dos\_reqs\_density\_per\_u within dos\_sampling\_time\_unit the user will be banned).
- kamailio.lb.security.dos\_sampling\_time\_unit: Sets the DoS unit time.
- kamailio.lb.security.dos\_whitelisted\_ips: Write here the whitelisted IPs.
- · kamailio.lb.security.failed\_auth\_attempts: Sets how many authentication attempts allowed before ban.
- kamailio.lb.security.failed\_auth\_ban\_enable: Enable/Disable authentication ban.
- kamailio.lb.security.failed\_auth\_ban\_time: Sets how long a user/IP has be banned.
- kamailio.lb.strict\_routing\_safe: Enable strict routing handle feature.
- kamailio.lb.tls.enable: Enable TLS socket.
- kamailio.lb.tls.port: Set TLS listening port.
- · kamailio.lb.tls.sslcertificate: Path for the SSL certificate.
- kamailio.lb.tls.sslcertkeyfile: Path for the SSL key file.
- kamailio.proxy.allow\_info\_method: Allow INFO method.
- kamailio.proxy.allow\_peer\_relay: Allow peer relay. Call coming from a peer that doesn't matcha a local subscriber will try to go out again, matching the peering rules.
- kamailio.proxy.allow\_refer\_method: Allow REFER method. Enable it with caution.
- kamailio.proxy.authenticate\_bye: Enable BYE authentication.
- · kamailio.proxy.cf\_depth\_limit: CF loop detector. How many CF loops are allowed before drop the call.
- kamailio.proxy.debug: Enable intensive debug level.
- · kamailio.proxy.default\_expires: Default expires value in seconds for REGISTER messages.
- kamailio.proxy.foreign\_domain\_via\_peer: Enable calls to foreign domains via peers.
- · kamailio.proxy.shm\_mem: Shared memory used by Kamailio Proxy.
- · kamailio.proxy.pkg\_mem: PKG memory used by Kamailio Proxy.
- kamailio.proxy.enum\_suffix: Sets ENUM suffix don't forget . (dot).

- kamailio.proxy.filter\_100rel\_from\_supported: Enable filtering of 100rel from Supported header, to disable PRACK.
- · kamailio.proxy.filter\_failover\_response: Response codes with no failover routing required.
- kamailio.proxy.fritzbox.enable: Enable detection for Fritzbox special numbers. Ex. Fritzbox add some prefix to emergency numbers.
- kamailio.proxy.fritzbox.prefixes: Fritybox prefixes to check. Ex. 0\$avp(caller\_ac)
- kamailio.proxy.fritzbox.special\_numbers: Specifies Fritzbox special number patterns. They will be checked with the prefixes defined. Ex. 112, so the performed check will be sip:0\$avp(caller\_ac)112@ if prefix is 0\$avp(caller\_ac)
- kamailio.proxy.ignore\_auth\_realm: Ignore SIP authentication realm.
- kamailio.proxy.keep\_original\_to: Not used now.
- kamailio.proxy.lnp.enabled: Enable/disable LNP (local number portability) lookup during call setup
- kamailio.proxy.lnp.type: method of LNP lookup; valid values are: local (local LNP database) and api (LNP lookup through external gateways). *PLEASE NOTE:* the api type of LNP lookup is only available for NGCP PRO / CARRIER installations.
- kamailio.proxy.lnp.api.invalid\_lnp\_routing\_codes [only for api type]: number matching pattern for routing numbers that represent invalid call destinations; an announcement is played in that case and the call is dropped
- kamailio.proxy.lnp.api.lnp\_request\_whitelist [only for api type]: list of matching patterns of called numbers for which LNP lookup must be done
- kamailio.proxy.lnp.api.lnp\_request\_blacklist [only for api type]: list of matching patterns of called numbers for which LNP lookup must not be done
- kamailio.proxy.lnp.api.request\_timeout [only for api type]: timeout in milliseconds while Proxy waits for the response of an LNP query from *Sipwise LNP daemon*
- kamailio.proxy.max\_expires: Sets the maximum expires in seconds for registration.
- · kamailio.proxy.max\_gw\_lcr: Defines the maximum number of gateways in lcr\_gw table
- kamailio.proxy.max\_registrations\_per\_subscriber: Sets the maximum registration per subscribers.
- kamailio.proxy.min\_expires: Sets the minimum expires in seconds for registration.
- kamailio.proxy.natping\_interval: Sets the NAT ping interval in seconds.
- kamailio.proxy.nathelper\_dbro: Default is "no". This will be "yes" on CARRIER in order to activate the use of a read-only connection using LOCAL\_URL
- kamailio.proxy.nonce\_expire: Nonce expire time in seconds.
- kamailio.proxy.pbx.hunt\_display\_fallback\_format: Default is [H %s]. Sets the format of the hunt group indicator that is sent as initial part of the From Display Name when subscriber is called as a member of PBX hunt group if the preferred format defined by the hunt\_display\_format and hunt\_display\_indicator can not be used (as in the case of not provisioned subscriber settings). The %s part is replaced with the value of the hunt\_display\_fallback\_indicator variable.
- kamailio.proxy.pbx.hunt\_display\_fallback\_indicator: The internal kamailio variable that sets the number or extension of the hunt group. Default is \$var(cloud\_pbx\_hg\_ext) which is populated during call routing with the extension of the hunt group.

- kamailio.proxy.pbx.hunt\_display\_format: Default is [H %s]. Sets the format of hunt group indicator that is sent as initial part of the From Display Name when subscriber is called as a member of PBX hunt group. This is the preferred (default) indicator format with Display Name, where the %s part is replaced with the value of the hunt\_display\_indicator variable.
- 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.perform\_peer\_lcr: Enable/Disable Least Cost Routing based on peering fees.
- kamailio.proxy.port: SIP listening port.
- · kamailio.proxy.presence.enable: Enable/disable presence feature
- kamailio.proxy.presence.max\_expires: Sets the maximum expires value for PUBLISH/SUBSCRIBE message. Defines expiration
  of the presentity record.
- · kamailio.proxy.presence.reginfo\_domain: Set FQDN of the NGCP domain used in callback for mobile push.
- · kamailio.proxy.set\_ruri\_to\_peer\_auth\_realm: Set R-URI using peer auth realm
- · kamailio.proxy.use\_enum: Enable/Disable ENUM feature.

## B.1.15 Inpd

The following section defines configuration of LNP daemon, that is used when LNP queries are served by external gateways  $\rightarrow$  the so called LNP API mode.

lnpd:

```
config:
  daemon:
    foreground: 'false'
    json-rpc:
        ports:
        - '8095'
    loglevel: '6'
    sip:
        port: '5095'
    threads: '4'
instances:
    default:
        module: sigtran
```

```
destination: 0.0.0.0
from-domain: voip.example.com
headers:
    - header: INAP-Service-Key
    value: '2'
reply:
    tcap: raw-tcap
enabled: no
```

• Inpd.enabled: Enable/disable LNP daemon

Inpd.config: details are shown in Configuration of LNP daemon Section 5.4.2.3

## B.1.16 mediator

The following is the mediator section:

```
mediator:
    interval: 10
```

• mediator.interval: Running interval of mediator.

# B.1.17 modules

### The following is the modules section:

modules:

```
- enable: no
name: dummy
options: numdummies=2
```

- modules: list of configs needed for load kernel modules on boot.
- · enable: Enable/disable loading of the specific module (yes/no)
- · name: kernel module name
- · options: kernel module options if needed

# B.1.18 nginx

The following is the nginx section:

```
nginx:
status_port: 8081
xcap_port: 1080
```

- · nginx.status\_port: Status port used by nginx server
- nginx.xcap\_port: XCAP port used by nginx server

### B.1.19 ntp

The following is the ntp server section:

#### ntp:

servers:

- 0.debian.pool.ntp.org
- 1.debian.pool.ntp.org
- 2.debian.pool.ntp.org
- 3.debian.pool.ntp.org

• ntp.servers: Define your NTP server list.

# B.1.20 ossbss

The following is the ossbss section:

```
ossbss:
  apache:
    port: 2443
    proxyluport: 1080
    restapi:
      sslcertfile: '/etc/ngcp-panel/api_ssl/api_ca.crt'
      sslcertkeyfile: '/etc/ngcp-panel/api_ssl/api_ca.key'
    serveradmin: support@sipwise.com
    servername: "\"myserver\""
    ssl_enable: 'yes'
    sslcertfile: '/etc/ngcp-config/ssl/myserver.crt'
    sslcertkeyfile: '/etc/ngcp-config/ssl/myserver.key'
  frontend: 'no'
  htpasswd:
      pass: '{SHA}w4zj3mxbmynIQ1jsUEjSkN2z2pk='
      user: ngcpsoap
  logging:
    apache:
      acc:
        facility: daemon
        identity: oss
        level: info
      err:
        facility: local7
        level: info
```

```
ossbss:
    facility: local0
   identity: provisioning
    level: DEBUG
  web:
    facility: local0
    level: DEBUG
provisioning:
  allow_ip_as_domain: 1
  allow_numeric_usernames: 0
  auto_allow_cli: 1
  carrier:
   account_distribution_function: roundrobin
   prov_distribution_function: roundrobin
  credit_warnings:
      domain: example.com
      recipients:
        - nobody@example.com
      threshold: 1000
  faxpw_min_char: 0
  log_passwords: 0
  no_logline_truncate: 0
  pw_min_char: 6
  routing:
    ac_regex: '[1-9]\d{0,4}'
   cc_regex: '[1-9]\d{0,3}'
    sn_regex: '[1-9]\d+'
  tmpdir: '/tmp'
```

- ossbss.frontend: Enable disable SOAP interface. Set value to fcgi to enable old SOAP interface.
- ossbss.htpasswd: Sets the username and SHA hashed password for SOAP access. You can generate the password using the following command: htpasswd -nbs myuser mypassword.
- ossbss.provisioning.allow\_ip\_as\_domain: Allow or not allow IP address as SIP domain (0 is not allowed).
- ossbss.provisioning.allow\_numeric\_usernames: Allow or not allow numeric SIP username (0 is not allowed).
- · ossbss.provisioning.faxpw\_min\_char: Minimum number of characters for fax passwords.
- ossbss.provisioning.pw\_min\_char: Minimum number of characters for sip passwords.
- ossbss.provisioning.log\_password: Enable logging of passwords.
- ossbss.provisioning.routing: Regexp for allowed AC (Area Code), CC (Country Code) and SN (Subscriber Number).

### B.1.21 pbx (only with additional cloud PBX module installed)

The following is the PBX section:

```
pbx:
bindport: 5085
enable: 'no'
highport: 55000
lowport: 50001
media_processor_threads: 10
session_processor_threads: 10
xmlrpcport: 8095
```

• pbx.enable: Enable Cloud PBX module.

# B.1.22 prosody

The following is the prosody section:

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

• prosody.ctrl\_port: XMPP server control port.

• prosody.log\_level: Prosody loglevel.

#### B.1.23 pushd

The following is the pushd section:

```
pushd:
    apns:
        certificate: '/etc/ngcp-config/ssl/PushChatCert.pem'
        enable: yes
        endpoint: gateway.push.apple.com
        feedback_endpoint: feedback.push.apple.com
        feedback_interval: 3600
        key: '/etc/ngcp-config/ssl/PushChatKey.pem'
        socket_timeout: 0
    enable: yes
    gcm:
        enable: yes
        key: 'google_api_key_here'
        priority:
        call: high
```

```
groupchat: normal
invite: normal
message: normal
muc:
  exclude: []
  force_persistent: 'true'
  owner_on_join: 'true'
  one_device_per_subscriber: no
  port: 45060
  processes: 4
  ssl: yes
  sslcertfile: /etc/ngcp-config/ssl/CAsigned.crt
  sslcertkeyfile: /etc/ngcp-config/ssl/CAsigned.key
  unique_device_ids: no
```

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

# B.1.24 qos

The following is the QOS section:

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

- qos.tos\_rtp: TOS value for RTP traffic.
- qos.tos\_sip: TOS value for SIP traffic.

### B.1.25 rate-o-mat

The following is the rate-o-mat section:

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

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

### B.1.26 redis

The following is the redis section:

```
redis:
   database_amount: 16
   port: 6379
   syslog_ident: redis
```

- redis.database\_amout: Set the number of databases in redis. The default database is DB 0.
- · redis.port: Accept connections on the specified port, default is 6379
- · redis.syslog\_ident: Specify the syslog identity.

# B.1.27 reminder

The following is the reminder section:

```
reminder:
retries: 2
retry_time: 60
sip_fromdomain: voicebox.sipwise.local
sip_fromuser: reminder
wait_time: 30
weekdays: '2, 3, 4, 5, 6, 7'
```

- · reminder.retries: How many times the reminder feature have to try to call you.
- reminder.retry\_time: Seconds between retries.
- reminder.wait\_time: Seconds to wait for an answer.

# B.1.28 rsyslog

The following is the rsyslog section:

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

- · rsyslog.elasticsearch.enable: Enable/Disable Elasticsearch web interface
- rsyslog.external\_address: Set the remote rsyslog server.
- rsyslog.ngcp\_logs\_preserve\_days: Specify how many days to preserve old rotated log files in /var/log/ngcp/old path.

# B.1.29 rtpproxy

### The following is the rtp proxy section:

```
rtpproxy:
   allow_userspace_only: yes
   enabled: yes
   log_level: '6'
   maxport: '40000'
   minport: '30000'
   prefer_bind_on_internal: no
   recording:
      enabled: no
      mp3_bitrate: '48000'
```

```
nfs_host: 192.168.1.1
nfs_remote_path: /var/recordings
output_dir: /var/lib/rtpengine-recording
output_format: wav
output_mixed: yes
output_single: yes
resample: no
resample_to: '16000'
spool_dir: /var/spool/rtpengine
rtp_timeout: '60'
rtp_timeout_onhold: '3600'
```

- rtpproxy.allow\_userspace\_only: Enable/Disable the user space failover for rtpengine (yes means enable). By default rtpengine works in kernel space.
- rtpproxy.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.
- rtpproxy.maxport: Maximum port used by rtpengine for RTP traffic.
- rtpproxy.minport: Minimum port used by rtpengine for RTP traffic.
- rtpproxy.recording.enabled: Enable support for call recording.
- rtpproxy.recording.mp3\_bitrate: If saving audio as MP3, bitrate of the output file.
- rtpproxy.recording.nfs\_host: Mount an NFS share from this host for storage.
- rtpproxy.recording.nfs\_remote\_path: Remote path of the NFS share to mount.
- rtpproxy.recording.output\_dir: Local mount point for the NFS share.
- rtpproxy.recording.output\_format: Either wav for PCM output or mp3.
- rtpproxy.recording.output\_mixed: Create output audio files with all contributing audio streams mixed together.
- rtpproxy.recording.output\_single: Create separate audio files for each contributing audio stream.
- rtpproxy.recording.resample: Resample all audio to a fixed bitrate (yes or no).
- rtpproxy.recording.resample\_to: If resampling is enabled, resample to this sample rate.
- rtpproxy.recording.spool\_dir: Local directory for temporary metadata file storage.
- rtpproxy.rtp\_timeout: Consider a call dead if no RTP is received for this long (60 seconds).
- rtpproxy.rtp\_timeout\_onhold: Maximum limit in seconds for an onhold (1h).

### B.1.30 security

The following is the security section. Usage of the firewall subsection is described in Section 13.2:

```
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 the NGCP firewall (default: yes)
- security.firewall.logging.file: File firewall log messages go to (default: /var/log/firewall.log)
- security.firewall.logging.tag: String prepended to all log messages (internally DROP is added to any tag indicating the action triggering the message, default: NGCPFW)
- security.firewall.nat\_rules4: Optional list of IPv4 firewall rules added to table nat using iptables-persistent syntax (default: undef)
- security.firewall.nat\_rules6: Optional list of IPv6 firewall rules added to table nat using iptables-persistent syntax (default: undef)
- security.firewall.policies.forward: Default policy for iptables FORWARD chain (default: DROP)
- security.firewall.policies.input: Default policy for iptables INPUT chain (default: DROP)
- security.firewall.policies.output: Default policy for iptables OUTPUT chain (default: ACCEPT)
- security.firewall.rules4: Optional list of IPv4 firewall rules added to table filter using iptables-persistent syntax (default: undef)
- security.firewall.rules6: Optional list of IPv6 firewall rules added to table filter using iptables-persistent syntax (default: undef)

### B.1.31 sems

The following is the SEMS section:

```
sems:
 bindport: 5080
 conference:
    enable: 'yes'
    max_participants: 10
  debug: 'no'
  highport: 50000
  lowport: 40001
  media_processor_threads: 10
 prepaid:
    enable: 'yes'
  sbc:
    calltimer_enable: 'yes'
    calltimer_max: 3600
    outbound_timeout: 6000
    sdp_filter:
      codecs: PCMA, PCMU, telephone-event
      enable: 'yes'
      mode: whitelist
    session_timer:
      enable: 'yes'
      max_timer: 7200
      min_timer: 90
      session_expires: 300
  session_processor_threads: 10
  vsc:
    block_override_code: 80
    cfb_code: 90
    cfna_code: 93
    cft_code: 92
    cfu_code: 72
    clir_code: 31
    directed_pickup_code: 99
    enable: 'yes'
    park_code: 97
    reminder_code: 55
    speedial_code: 50
    unpark_code: 98
    voicemail_number: 2000
  xmlrpcport: 8090
```

• sems.conference.enable: Enable/Disable conference feature.

· sems.conference.max\_participants: Sets the number of concurrent participant.

- sems.highport: Maximum ports used by sems for RTP traffic.
- sems.debug: Enable/Disable debug mode.
- · sems.lowport: Minimum ports used by sems for RTP traffic.
- sems.prepaid.enable: Enable/Disable prepaid feature.
- sems.sbc.calltimer\_max: Sets the maximum call duration for inter-domain calls.
- · sems.sbc.outbound\_timeout:: Sets the maximum call duration for outboud calls.
- sems.sbc.session\_timer.enable: Enable/Disable session timers (deprecated, use the web interface configuration).
- sems.vsc.\*: Define here the VSC codes.

### B.1.32 snmpagent

#### The following is the SNMP Agent section:

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

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

### B.1.33 sshd

The following is the sshd section:

```
sshd:
   listen_addresses:
        - 0.0.0.0
```

 sshd: specify interface where SSHD should run on. By default sshd listens on all IPs found in network.yml with type ssh\_ext. Unfortunately sshd can be limited to IPs only and not to interfaces. The current option makes it possible to specify allowed IPs (or all IPs with 0.0.0.0).

# B.1.34 voisniff

The following is the voice sniffer section:

```
voisniff:
  admin_panel: 'no'
  daemon:
    bpf: 'port 5060 or 5062 or ip6 proto 44 or ip[6:2] & 0x1fff != 0'
    external_interfaces: 'eth0 eth1'
    filter:
      exclude:
          active: 0
          case_insensitive: 1
          pattern: '\ncseq: *\d+ +(register|notify|options)'
      include: []
    internal_interfaces: lo
    mysql_dump_threads: 4
    start: 'no'
    threads_per_interface: 10
  partitions:
    increment: 700000
    keep: 10
```

- voisniff.admin\_panel: Enable/Disable SIP STATS on Admin interface. Default is no.
- voisniff.deamon.external\_interfaces: Define binding interfaces.
- voisniff.deamon.start: Change to yes if you want voisniff start at boot. Default is no.

### B.1.35 www\_admin

The following is the WEB Admin interface (www\_admin) section:

```
www_admin:
  ac_dial_prefix: 0
 apache:
   autoprov_port: 1444
 billing_features: 1
  callingcard_features: 0
 callthru_features: 0
 cc_dial_prefix: 00
  conference_features: 1
  contactmail: adjust@example.org
  dashboard:
   enabled: 1
 default_admin_settings:
   call_data: 0
   is_active: 1
   is_master: 0
   read_only: 0
```

show\_passwords: 1

```
442 / 486
```

```
domain:
  preference_features: 1
  rewrite_features: 1
  vsc_features: 0
fastcgi_workers: 2
fax_features: 1
fees_csv:
  element_order:
    - source
    - destination
    - direction
    - zone
    - zone_detail
    - onpeak_init_rate
    - onpeak_init_interval
    - onpeak_follow_rate
    - onpeak_follow_interval
    - offpeak_init_rate
    - offpeak_init_interval
    - offpeak_follow_rate
    - offpeak_follow_interval
    - use_free_time
http_admin:
  autoprov_port: 1444
  port: 1443
  serveradmin: support@sipwise.com
  servername: "\"myserver\""
  ssl_enable: 'yes'
  sslcertfile: '/etc/ngcp-config/ssl/myserver.crt'
  sslcertkeyfile: '/etc/ngcp-config/ssl/myserver.key'
http_csc:
  autoprov_bootstrap_port: 1445
  autoprov_port: 1444
  port: 443
  serveradmin: support@sipwise.com
  servername: "\"myserver\""
  ssl_enable: 'yes'
  sslcertfile: '/etc/ngcp-config/ssl/myserver.crt'
  sslcertkeyfile: '/etc/ngcp-config/ssl/myserver.key'
logging:
  apache:
    acc:
      facility: daemon
      identity: oss
      level: info
    err:
      facility: local7
```

```
level: info
peer:
  preference_features: 1
peering_features: 1
security:
  password_allow_recovery: 0
  password_max_length: 40
  password_min_length: 6
  password_musthave_digit: 0
  password_musthave_lowercase: 1
  password_musthave_specialchar: 0
  password_musthave_uppercase: 0
  password_sip_autogenerate: 0
  password_sip_expose_subadmin: 1
  password_web_autogenerate: 0
  password_web_expose_subadmin: 1
speed_dial_vsc_presets:
  vsc:
    - '*0'
    - '*1'
    - '*2'
    - '*3'
    - '*4'
    - '*5'
    - '*6'
    - '*7'
    - '*8'
    - '*9'
subscriber:
  auto_allow_cli: 0
  extension_features: 0
voicemail_features: 1
```

- www\_admin.http\_admin.\*: Define the Administration interface and certificates.
- www\_admin.http\_csc.\*: Define the Customers interface and certificates.
- www\_admin.contactmail: Email to show in the GUI's Error page.

### B.2 constants.yml Overview

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



# Caution

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

# B.3 network.yml Overview

/etc/ngcp-config/network.yml is one of the main configuration files that contains network-related configuration parameters, like IP addresses and roles of the node(s) in sip:provider PRO 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 10 section of the handbook.

# Sample host configuration for sip:provider PRO

```
sp1:
 dbnode: '1'
 eth0:
    dns_nameservers:
     - 192.168.51.30
      - 192.168.51.31
   gateway: 192.168.22.1
   hwaddr: 06:1e:bc:e2:ec:fb
   ip: 10.0.2.15
   netmask: 255.255.255.0
   shared_ip: ~
   shared_v6ip: ~
    type:
      - web_ext
      - ssh_ext
      - web_int
 eth1:
   hwaddr: 6e:7f:3a:f9:db:1f
    ip: 192.168.255.251
   netmask: 255.255.255.248
    shared_ip:
      - 192.168.255.250
    shared_v6ip: ~
    type:
      - ha_int
     - ssh_ext
 eth2:
    ip: 10.15.20.107
   netmask: 255.255.255.0
    shared_ip:
      - 10.15.20.151
    type:
```

```
- ssh_ext
    - web_ext
    - web_int
    - sip_ext
    - rtp_ext
   - mon_ext
interfaces:
 - lo
 - eth0
 - ethl
  - eth2
lo:
 advertised_ip: []
 cluster_sets:
    - default
 hwaddr: 00:00:00:00:00:00
 ip: 127.0.0.1
 netmask: 255.0.0.0
 shared_ip: []
 shared_v6ip: []
 type:
   - sip_int
   - web_ext
   - web_int
   - aux_ext
    - ssh_ext
   - api_int
 v6ip: '::1'
peer: sp2
role:
 - proxy
 - lb
 - mgmt
 - rtp
 – db
```

```
status: 'online'
```

# C NGCP-Faxserver Configuration

For an overview of Faxserver architecture and features, please see the Faxserver Section 5.9 chapter.

# C.1 Faxserver Components

Starting from mr4.3 release there is a completely reworked fax server in a form of standalone daemon that uses Asterisk as its transmission component. No other component—such as hylafax or iaxmodem—is necessary to send and receive faxes on sip:provider PRO platform.

# C.2 Enabling Faxserver

In order to configure functions of NGCP Faxserver one needs to update the main NGCP configuration file /etc/ngcp-config/ config.yml with the correct fax options:

```
faxserver:
  enable: yes
  fail_attempts: '3'
  fail_retry_secs: '60'
  keep_failed_fax: yes
  keep_failed_fax_days: '60'
  keep_received_fax: yes
  keep_received_fax: yes
  keep_sent_fax: yes
  keep_sent_fax: yes
  keep_sent_fax: Yes
  keep_sent_fax_days: '60'
  mail_from: 'Sipwise NGCP FaxServer <voipfax@ngcp.sipwise.local>'
```

### Parameters are:

- enable: must be yes to enable Faxserver
- fail\_...: the number and timeout of fax sending retrials
- keep\_...: fax retention definitions: enabling and length in days
- mail\_from: the From header in the e-mail that is sent by Fax2Mail feature when a fax is received

# C.3 Fax Templates Configuration

One needs to update /etc/ngcp-config/templates/etc/ngcp-faxserver/faxserver.conf.tt2 if he wants to use custom content in the fax and e-mail templates that are used by Faxserver to generate the actual fax or e-mail. This may be done under the "User templates" section in the file.

### Applying new Faxserver configuration

Once the above mentioned configuration files have been modified the new settings must be applied:

ngcpcfg apply 'Configured fax server'
ngcpcfg push

# C.4 Fax Services Configuration per Subscriber

Fax services must be explicitly activated for subscribers before they can send or receive faxes. This activation and the custom settings may be set on the NGCP Web panel in the following way (as an administrator):

- · Go to Subscribers and find the subscriber that you want to modify settings for
- Click on Preferences button
- Select FaxFeatures

In both sections *Fax2Mail and SendFax* and *Mail2Fax* there is a field: Active. This must be changed from no to yes if the particular fax service must be activated.

When fax services have been activated the user sees a summary of settings in FaxFeatures section on his Preferences page:

Voicemail and Voicebox				
Fax Features				
Fax2Mail and Sendfax				
Name	Value			
Name in Fax Header for Sendfax				
Active	yes			
Destinations	subscriber1@example.org as TIFF			

Mail2Fax

Name	Value	
Active	yes	
Secret Key (empty=disabled)		
Secret Key Renew	never	
Last Secret Key Modify Time		
Secret Key Renew Notify		
ACL	regex from_email subscriber1@example.org and received_from any to <u>^4399.+</u> destination	

Speed Dial

Figure 125: Fax Settings

Details of Fax2Mail, SendFax and Mail2Fax settings are described in subsequent paragraphs.

# C.5 Fax2Mail and SendFax Settings

- Name in Fax Header for SendFax: optional field that contains the subscribers name on faxes sent from the Web panel directly
- Destinations: e-mail addresses and selections of notification items that define about which event and where an e-mail is sent; this is a list of such definitions

Edit destinations	×
Destination Email	subscriber1@example.org
File Type	TIFF
Deliver Incoming Faxes	
Deliver Outgoing Faxes	
Receive Reports	
	Remove
	Add another destination
	Save

# Figure 126: Fax2Mail Destination

The parameters for a destination are as follows:

- Destination Email: the e-mail address where the notification must be sent
- File Type: file format of faxes attached to e-mails
- Deliver Incoming Faxes: select this in order to receive incoming faxes in e-mail
- Deliver Outgoing Faxes: select this in order to receive a report about sent faxes
- Receive Reports: select this in order to receive reports about success / failure of fax transmissions

# C.6 Mail2Fax Settings

A subscriber can restrict access to his Mail2Fax service with some methods, those can also be combined:

- using a secret key that is only known to him, and is inserted in every mail that he sends to NGCP to be forwarded as fax
- using an *access control list (ACL)* that determines from which endpoint and for which destination a mail-to-fax is accepted by NGCP platform
- Secret Key: the secret key used to validate the sender of an e-mail; not used if left empty
- Secret Key Renew: secret key renewal period; NGCP platform will enforce renewal of the secret key when the defined time has elapsed
- Last Secret Key Modify Time: information about the last secret key modification time
- Secret Key Renew Notify: an e-mail address where the notification about secret key modification is sent
- · ACL: access control list, see the details below; this is a list of access control rules

Edit acl	×
From email	subscriber1@example.org
Received from IP	
Destination	^4399.+
Use Regex	
	Remove
	Add another rule
	Save

Figure 127: Mail2Fax Access Control List

### The parameters for access control rules:

• From email: this sender is allowed to use Mail2Fax service

- Received from IP: this IP address or host name must be present in From e-mail header
- Destination: either a complete phone number in E.164 format, or a regular expression ("Use Regex" checkbox must be ticked) that may define a range of numbers. Examples: "4313334445" as a single number; "^4399.+" as a regular expression: all destinations starting with "4399"

# Caution

When neither Secret Key, nor ACL is defined then Mail2Fax service will deny accepting any e-mail for sending faxes!

# C.7 Sending Fax from Web Panel

A subscriber can log in to his *Customer Self Care* website and send faxes directly from there. In order to do this, one needs to do the following:

- Go to  $\textit{Settings} \rightarrow \textit{Web Fax}$  page

# Тір

The list of received faxes is also available here.

• Press Send Fax button to start entering data, such as recipient and content for the fax being sent:

	×	
43993003		
Normal	•	
Test fax		
This is the plain text content, but there will also be an attachment.		
Browse handbook-pro.pdf		
Sen	nd	
	Normal         Test fax         This is the plain text content, but there will also be an attachment.         Browse         handbook-pro.pdf	Normal Test fax This is the plain text content, but there will also be an attachment.

Figure 128: Sending Fax from Web Panel

Both plain text message and attached files can be sent in the fax. First page(s) will contain the plain text message and the content of attached files will follow that.

# C.8 Faxserver Mail2Fax Configuration

Using NGCP Faxserver's Mail2Fax service requires the configuration of sip:provider PRO's local mail server that is *Exim*. It has to be configured in a way that it can receive mails from outside of the server, because *Exim* by default listens only on the local interfaces for incoming mails.

# **Exim Configuration**

The NGCP platform administrator must reconfigure Exim in order to enable receiving e-mails for fax sending:

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

PLEASE NOTE: When entering configuration data the following points must be kept in mind:

```
• operation mode has to be set to "mail sent by smarthost; no local mail"
```

• "mail2fax.example.org" must be added to accepted domains, where "example.org" is the domain name of the NGCP platform operator

#### **DNS Configuration**

It is necessary to add a subdomain starting as mail2fax. to the list of domain names. That is where the faxes will be sent by users to trigger Mail2Fax service.

#### Tip

Alternatively, edit /etc/ngcp-config/templates/etc/exim4/conf.d/router/999\_mail2fax.tt2 file and adjust it to your personal preferences. Although this is not recommended and should only be done by Sipwise support engineers.

# C.9 Sending Fax Using E-mail Clients

When sending an e-mail that should be converted to a fax, there are some points to keep in mind so that Faxserver properly processes the e-mail.

### To header:

- must contain the subscriber's number who is sending the fax, as the username part of the mail address
- must contain the specific domain starting with  ${\tt maillfax}$  .
- · Subject header: must contain the fax destination number
- · Body should consist of plain text data
- · Adding attachments is possible, but only plain text and PDF formats are supported

### Secret Key

In order to use the "secret key" access control feature, it should be either put in the first row of the e-mail body followed by an empty line, or included as a plain text attachment. Once it has been validated, it will be removed from the email.

# ) Important

Either add the secret key to the body, or attach it. Never do both as only one will be recognized and removed, leaving the other one to be sent as part of the fax.

### Mail Example

Provided there is a subscriber on sip:provider PRO platform with the 43130111 number, the destination fax is 43130222 and the secret key is "MySecretKey":

```
From: User Name <username@example.org>
To: 43130111@mail2fax.example.org
```

# C.10 Managing Faxes via the REST API

It is possible to send and receive faxes and configure fax settings using the built-in REST API interface.

In subsequent sections you can find examples of using the API for sending, receiving faxes and changing fax settings.

# C.10.1 Configuring Fax Settings

# C.10.1.1 Retrieving Fax Settings

The following example retrieves the fax settings for the subscriber with ID 3.

```
Method: GET
Content-Type: application/hal+json
```

https://127.0.0.1:1443/api/faxserversettings/3

The output format is as follows (only the relevant output data is shown):

```
"active" : true,
 "destinations" : [
    {
       "destination" : "user@company.com",
       "filetype" : "PDF14",
       "incoming" : true,
       "outgoing" : true,
       "outgoing" : true
    }
],
"name" : null,
"password" : null
```

# C.10.1.2 Updating Fax Settings

The following example updates a specific parameter. Namely, it deactivates the fax feature for the subscriber with ID 3.

```
Method: PATCH
Content-Type: application/json-patch+json
```

https://127.0.0.1:1443/api/faxserversettings/3

```
--data-binary '[ { "op" : "replace", "path" : "/active", "value" : 0 } ]'
```

### C.10.2 Sending a Fax

The following request sends a PDF file located at /tmp/test\_fax.pdf as fax to 431110002 from the subscriber with ID 3.

```
Method: POST
Content-Type: multipart/form-data
https://127.0.0.1:1443/api/faxes/
--form 'json={"destination" : "431110002", "subscriber_id" : 3}' --form 'faxfile=@/tmp/ ↔
test_fax.pdf'
```

### C.10.3 Receiving a Fax

All received faxes are stored on the server and can be retrieved on demand. You can retrieve a stored fax by following these steps:

1. Firstly, obtain the internal ID of the fax:

```
Method: GET
Content-Type: application/json
https://127.0.0.1:1443/api/faxes/3
```

This request returns the list of stored faxes for the subscriber with ID 3. One of the available faxes is returned like this:

```
"callee" : "431110002",
"caller" : "431110001",
"direction" : "out",
"duration" : "0",
"filename" : "d9799276-b7d9-454f-98c3-714edf7e3072.tif",
"id" : 5,
"pages" : "1",
"quality" : "8031x7700",
"reason" : "Normal Clearing / SIP 200 OK [1/3]",
"signal_rate" : "14400",
"status" : "SUCCESS",
"subscriber_id" : 1,
"time" : "2016-07-30 09:49:59"
```

2. Now, to retrieve the fax with ID 5, use the following request:

Method: GET Content-Type: application/hal+json

https://127.0.0.1:1443/api/faxerecordings/5

By default, the fax is in the TIFF format. It is also possible to request it in a different format. To retrieve the same fax in PDF14, use the following request:

https://127.0.0.1:1443/api/faxerecordings/5?format=pdf14

#### C.10.4 Configuring Mail2Fax Settings

The configuration of Mail2Fax settings via the REST API is similar to the fax settings configuration.

## C.10.4.1 Retrieving Mail2Fax Configuration

To get the Mail2Fax configuration for the subscriber with ID 3, use the following request:

```
Method: GET
Content-Type: application/hal+json
https://127.0.0.1:1443/api/mailtofaxsettings/3
```

The output format is as follows (only the relevant output data is shown):

```
"acl" : [],
"active" : false,
"secret_key" : "secretkeypassword",
"secret_key_renew" : "daily",
"secret_renew_notify" : [
        {
            "destination" : "userl@company.com"
        }
]
```

## C.10.4.2 Updating Mail2Fax Configuration

The following set of requests changes the Mail2Fax configuration with new secret key settings.

```
· Secret key value:
```

```
Method: PATCH
Content-Type: application/json-patch+json
```

https://127.0.0.1:1443/api/faxserversettings/3

```
--data-binary '[ { "op" : "replace", "path" : "/secret_key", "value" : " ↔
newsecretkeypassword" } ]'
```

· Secret key renewal interval:

```
Method: PATCH
Content-Type: application/json-patch+json
--data-binary '[ { "op" : "replace", "path" : "/secret_key_renew", "value" : "monthly" } ↔
]'
```

· List of email addresses that receive the automatic secret key update notifications:

```
Method: PATCH
Content-Type: application/json-patch+json
--data-binary '[ { "op" : "replace", "path" : "/secret_renew_notify", "value" : [ { " ↔
    destination": "user2@company.com" }, { "destination": "user3@company.com" } ] } ]'
```

## C.10.5 Using Advanced Faxserver and Mail2Fax Settings via the REST API

On the NGCP REST API documentation web page you can find the complete list of available Faxserver and Mail2Fax configuration parameters: https://<ngcp\_ip\_address>:1443/api



## Important

The information on the web page is relevant for your platform version and may change in next releases.

After visiting the API documentation main page, you can find the following entries related to Faxserver operations:

- Faxes (https://<ngcp\_ip\_address>:1443/api/#faxes)
- FaxRecordings (https://<ngcp\_ip\_address>:1443/api/#faxrecordings)
- FaxserverSettings (https://<ngcp\_ip\_address>:1443/api/#faxserversettings)

## C.11 Troubleshooting

The following log file may be used to check Faxserver functionality: /var/log/ngcp/faxserver.log

## C.11.1 Session ID (SID)

Faxserver stores basic information about each processed fax in a session file. The most important element within this set of data is the *Session ID* (SID) that uniquely identifies a fax throughout its lifetime.

Session ID is a long hexadecimal string (a kind of UUID) that can be read from the above mentioned Faxserver logfile, and which itself is used also as the filename in files that belong to a specific sent / received fax. An example:

root@sp1:~# cat /var/spool/ngcp/faxserver/failed/1e480167-5de6-4cc2-948b-de58d1a0bb8c.err

```
created: 2016-09-06 04:41:32
caller: 11111111
callee: 222222222
file: 1e480167-5de6-4cc2-948b-de58d1a0bb8c.tif
sid: 1e480167-5de6-4cc2-948b-de58d1a0bb8c
dir: out
attempts: 0
fail_attempts: 3
fail_retry_secs: 60
quality: normal
status: FAILED
error: Internal error
modified: 2016-09-06 17:41:30
```

#### root@sp1:~#

The data element sid is the session ID. Other important elements are:

- caller and callee: these are probably searched for when trying to figure out what happened to a specific fax transmission, if you don't know the SID
- dir: direction of fax transmission: in'coming or 'out'going or 'mtf for mail-to-fax
- status: shows success or failure
- error: the error cause in case of failed faxes

## C.11.2 Fax Storage Location

Faxserver stores all of its processed faxes at the path: /var/spool/ngcp/faxserver/... Within that directory the most relevant subdirectories are failed and completed that store the SID file and the fax itself in TIFF format of those faxes that failed or were successful, respectively.

## D RTC:engine

## **D.1** Overview

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

## D.2 RTC:engine enabling

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

## D.2.1 Enabling services via CLI

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

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

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

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

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

The WebRTC subscriber is just a normal subscriber which has just a different configuration in his Preferences. You need to change the following preferences under Subscribers  $\rightarrow$  Details  $\rightarrow$  Preferences  $\rightarrow$  NAT and Media Flow Control:

- use\_rtpproxy: Always with rtpproxy as additional ICE candidate
- transport\_protocol: RTP/SAVPF (encrypted SRTP with RTCP feedback)

The transport\_protocol setting may change, depending on your WebRTC client/browser configuration. Supported protocols are the following:

- · Transparent (Pass through using the client's transport protocol)
- RTP/AVP (Plain RTP)
- RTP/SAVP (encrypted SRTP)
- RTP/AVPF (RTP with RTCP feedback)
- RTP/SAVPF (encrypted SRTP with RTCP feedback)
- UDP/TLS/RTP/SAVP (Encrypted SRTP using DTLS)
- UDP/TLS/RTP/SAVPF (Encrypted SRTP using DTLS with RTCP feedback)



## Warning

The below configuration is enough to handle a WebRTC client/browser. As mentioned, you may need to tune a little bit your transport\_protocol configuration, depending on your client/browser settings.

In order to have a bridge between normal SIP clients (using plain RTP for example) and WebRTC client, the normal SIP clients' preferences have to have the following configuration:

## transport\_protocol: RTP/AVP (Plain RTP)

This will teach Sip Provider to translate between Plain RTP and RTP/SAVPF when you have calls between normal SIP clients and WebRTC clients.

## D.2.3 Create RTC:engine session

## D.2.3.1 Create sessions

## **Request:**

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

#### **Response Header:**

Location: /api/rtcsessions/7

#### D.2.3.2 Receive sessions

#### **Request:**

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

**Response Header:** 

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

Tip Use rtc\_browser\_token in your cdk.Client.

## D.3 RTC:engine protocol details

## D.3.1 Terminology

## D.3.1.1 Connector

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

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

#### D.3.1.2 Router

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

## D.3.1.3 User

## D.3.1.4 App

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

## D.3.1.5 Network

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

## D.3.1.6 Session

## D.3.1.7 Account

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

## D.3.1.8 Browser SDK

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

## D.3.2 Messages

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

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

## D.3.2.1 Fields

## D.3.2.2 method

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

## D.3.2.3 from

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

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

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

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

## D.3.2.4 to

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

## D.3.2.5 session

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

## D.3.2.6 body

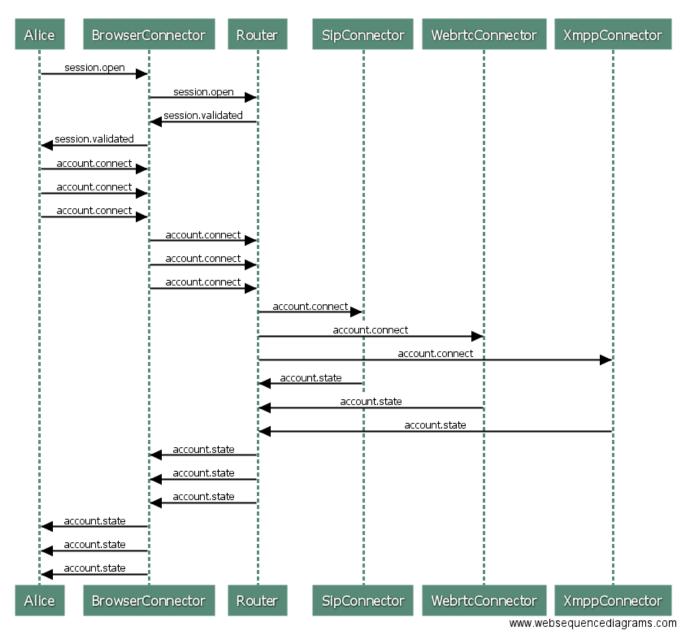
The body contains the payload of the message. Every message type has its own body schema.

## D.3.3 Account

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

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

## D.3.3.1 Flow



## D.3.3.2 Messages

## D.3.3.3 account.connect

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

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

## D.3.3.4 account.state

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

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

#### D.3.3.5 State reasons

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

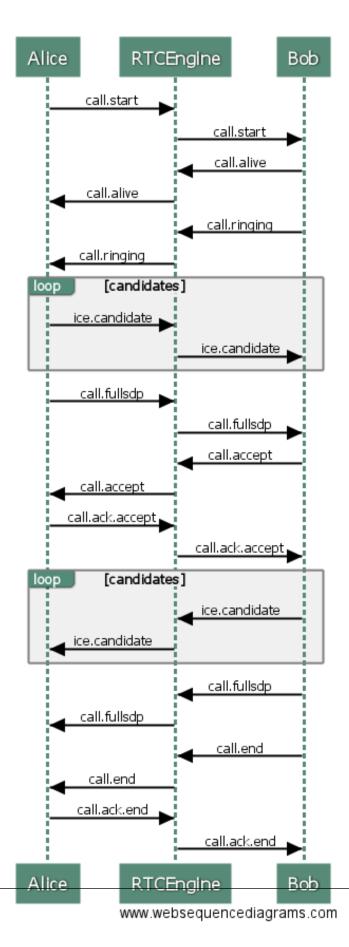
- WRONG\_CREDENTIALS
- CONNECTOR\_UNAVAILABLE
- CONNECTOR\_BUSY
- CONNECTOR\_ERROR
- ACCOUNT\_NOT\_FOUND

## D.3.3.6 States

- CONNECTED
- DISCONNECTED

## D.3.4 Call

## D.3.4.1 Flow



## D.3.4.2 call.start

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

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

## D.3.4.3 Body properties

#### D.3.4.4 id

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

#### D.3.4.5 gcid

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

## D.3.4.6 account

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

## D.3.4.7 replace

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

## D.3.4.8 trickle

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

## D.3.4.9 target

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

#### D.3.4.10 sdp

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

## D.3.4.11 call.alive

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

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

#### D.3.4.12 call.ringing

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

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

#### D.3.4.13 call.accept

The callee sends this message after accepting the call explicitly.

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

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

## D.3.4.14 call.ack.accept

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

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

## D.3.4.15 call.candidate

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

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

## D.3.4.16 call.fullsdp

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

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

## D.3.4.17 call.change....

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

- D.3.4.18 call.change.lock.resetD.3.4.19 call.change.lockD.3.4.20 call.change.lock.okD.3.4.21 call.change.offer
- D.3.4.22 call.change.answer

## D.3.4.23 call.dtmf

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

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

## D.3.4.24 call.end

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

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

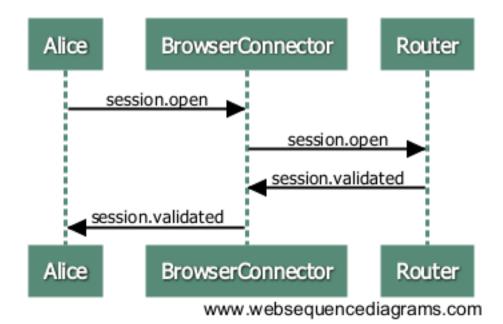
## D.3.4.25 call.ack.end

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

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

## D.3.5 Session

## D.3.5.1 Flow



## D.3.5.2 Messages

## D.3.5.3 session.open

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

## D.3.5.4 session.validated

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

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

```
"to": "browser:ws1",
"session": "session1"
"body": {
    "result": true,
    "accounts": {
        "account1": {
            "identifier": "sip:account1@foo.bar"
            "target": "sip-connector:b2bua-account1",
            "network": {
                "tag": "sip-network"
            }
        }
    }
}
```

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

```
{
   "method": "session.validated",
   "from": "core",
   "to": "browser:ws1",
   "session": "session1"
    "body": {
        "result": false,
        "reason": {
            "type": "invalidToken",
            "message": "Your token is not a valid user session token!"
        }
    }
},
```

## D.3.5.5 Reason types

invalidToken

}

- tokenExpired
- missingCredentials

## E comx-fileshare-service

## E.1 Overview

The *comx-fileshare-service* is a Node.js (4.4.0) based filesharing service and it is intended to be used via REST API. This service allows you to upload arbitrary files to the server and to download/share them with a generated link.

The API can be used with in 2 ways:

- · with simple identification, which means that only credentials of a user/subscriber are needed for authentication
- with session identification, which also provides for example the time-to-live (TTL) functionality besides authentication, and will be used in combination with the *RTC:engine*.

## E.2 Configuration and Usage

#### E.2.1 Change authentication method

To use NGCP subscribers as authentication against the API, you need to set it in the comx-fileshare-service config.js:

```
simpleUpload: {
   authentication: {
     enabled: true,
     subscriber: true,
     username: 'foo8',
     password: 'bar8'
  }
}
```

#### You can now authenticate like this with the API:

```
curl -i -X POST --insecure --form file=@/tmp/test.txt --form --user '43991002@domain.tld: ↔
    x43991002'
https://$NGCP_IP/rtc/fileshare/uploads
```

If you want to use the credentials from the config.js you need so set it to the following settings:

```
simpleUpload: {
   authentication: {
     enabled: true,
     subscriber: false,
     username: 'foo8',
     password: 'bar8'
   }
}
```

In this case, the login parameter would be this:

curl -i -X POST --insecure --form file=@/tmp/test.txt --form --user 'foo:bar'
https://\$NGCP\_IP/rtc/fileshare/uploads

## E.2.2 Database Structure

Table information for the *fileshare* database:

• downloads table:

Field Name	Field Type	Description
id	CHAR, PRIMARY KEY	Internal ID of the download action
state	ENUM	State of the download
uploaded_id	CHAR, FOREIGN KEY	External ID used for accessing the uploaded file in
		uploads table
created_at	DATETIME	Download action creation time
updated_at	DATETIME	Time of last download action modification

Table 27: Details of downloads Table in fileshare Database

• sessions table:

#### Table 28: Details of sessions Table in fileshare Database

Field Name	Field Type	Description
id	CHAR, PRIMARY KEY	Internal ID of the session
ttl	INT	Time-to-live value of the session (in seconds)
created_at	DATETIME	Session creation time
updated_at	DATETIME	Time of last session modification

• uploads table:

Field Name	Field Type	Description
id	CHAR, PRIMARY KEY	Internal ID of the file entry
data	LONGBLOB	The file data
original_name	VARCHAR	Original name of the file
mime_type	VARCHAR	MIME type of the file

## Table 29: (continued)

Field Name	Field Type	Description
size	INT	File size in bytes
ttl	INT	Time-to-live value of the file
state	ENUM	State of the file
session_id	CHAR, FOREIGN KEY	External ID used to access session data in
		sessions table
created_at	DATETIME	File creation / upload time
updated_at	DATETIME	Time of last file modification

## E.3 Activation of Filesharing Service on NGCP

The service is installed on every sip:provider PRO system, but is not activated by default. In order to activate the service, connect with SSH to your server, open /etc/ngcp-config/config.yml with your editor of choice and change the fileshare. enable property from no to yes:

fileshare:
 enable: yes

Apply the new configuration in the usual way:

```
ngcpcfg apply 'Enabled comx-fileshare-service'
ngcpcfg push
```

and check the status with monit summary. It should be now up and running.

## E.4 Message Sequence Chart

## E.4.1 Simple Message Sequence

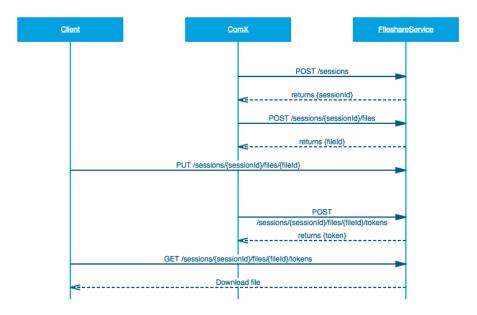


Figure 129: Sequence Simple

## E.4.2 Detailed Message Sequence

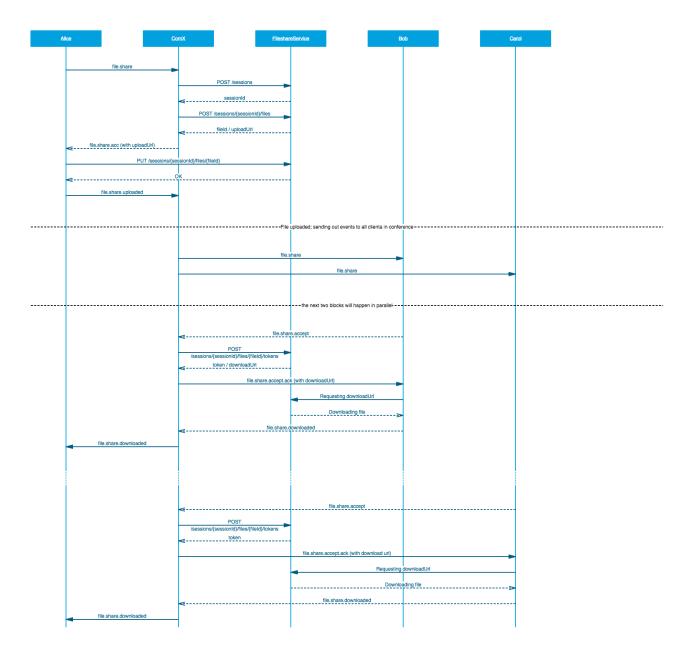


Figure 130: Sequence Detailed

## E.5 API of Filesharing Service

## E.5.1 HTTP Authentication

Type: Basic Auth username/password

#### E.5.2 Upload and Download with Simple Identification

The following HTTP methods can be used to perform file upload and download:

POST /uploads	// Simple upload
GET /uploads/{fileId}	// Simple download

## E.5.3 Upload and Download with Session Identification

The following HTTP methods can be used to perform file upload and download, and to manage sessions.

#### Session identification:

```
GET
       /sessions/{sessionId}/files
                                                               // Get all files of a session
       /sessions/{sessionId}/files/{fileId}/tokens/{tokenId}
GET
                                                              // Download a single file
POST
       /sessions
                                                               // Create a new session
POST
       /sessions/{sessionId}/files
                                                               // Create a new file entry
POST
       /sessions/{sessionId}/files/{fileId}/tokens
                                                               // Generate a download token
PUT
       /sessions/{sessionId}/files/{fileId}
                                                               // Upload and store a file
```

#### Simple identification:

GET	/uploads/{fileId}	// Get uploaded file
POST	/uploads	// Upload file

#### E.5.4 Curl Example for Simple Upload Request

curl -i -X POST --insecure --form file=@/tmp/test.txt --form --user 'foo:bar'
https://\$NGCP\_IP/rtc/fileshare/uploads

#### E.5.5 Upload Parameters

#### E.5.5.1 file

The parameter *file* defines the path to the desired file that should be uploaded.



This upload parameter is mandatory!

Curl example:

curl -i -X POST --insecure --form file=@/tmp/test.txt https://\$NGCP\_IP/rtc/fileshare/ ↔
uploads

#### E.5.5.2 user

The parameter user defines the user to authenticate with the fileshare service.

Caution This upload parameter is mandatory!

curl -i -X POST --insecure --form --user 'foo:bar' https://\$NGCP\_IP/rtc/fileshare/uploads

#### E.5.5.3 TTL

The parameter *ttl* defines the time-to-live (in seconds), that is how long the uploaded file will be available for download. The default values for this parameter are defined in the configuration file:

```
models: {
    session: {
        ttl: 86400 * 7
    },
    upload: {
        ttl: 3600
    }
}
```

#### Curl example:

{

```
curl -i -X POST --insecure --form file=@/tmp/test.txt --form ttl=3600
--user 'foo:bar' https://$NGCP_IP/rtc/fileshare/uploads
```

Response from *curl* when TTL is expired:

```
"message": "upload expired"
}
```

Response in the log file when TTL is expired:

Error at /uploads/88e5905d-5d96-4750-ab3d-77a1ed26f569: message=upload expired, status=410

## E.5.6 Number of Possible Downloads

There is a significant difference in the usage of the filesharing service between the approach within the *RTC:engine* and the simple upload/download one:

- If you are using the **simple upload and download** approach, the generated download link you get for your file can be used as many times as required, as long as the TTL is not expired.
- The approach with the Session ID, which will be used with the *RTC:engine* implementation, limits the download to one-time only. This means that the generated download link can be used only once. If you plan to share the URL with multiple persons, you have to generate one link for each recipient.

# F NGCP Internals

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

## F.1 Pending reboot marker

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

## F.2 Redis id constants

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

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

## F.2.1 Redis monitoring keys

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

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

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

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

The node hashes contain the following keys:

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

The fsys hashes contain the following keys:

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

The proc hashes contain the following keys:

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

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

The mysql hashes contain the following keys:

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

## F.3 Enum preferences

All tables are in database "provisioning".

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

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

voip\_preferences\_enum is:

id

boring pkey

preference\_id

Reference to table voip\_preferences.

label

A label to be displayed in frontends.

value

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

usr\_pref

```
dom_pref
peer_pref
    Flag if this is to be used for [usr|dom|peer] preferences.
default_val
    Flag indicating if this should be used as a default value when
    creating new entities or introducing new enum preferences (both done
    via triggers). (Note: For this to work, value must also be set.)
Relevant triggers:
enum_update
    Propagates changes of voip_preferences_enum.value to
    voip_[usr|dom|peer]_preferences.value
enum_set_default
    Will create entries for default values when adding a new enum
    preference. The default value is the tuple from voip_preferences_enum
    WHERE default_val=1 AND value NOT NULL.
trigger voip_dom_crepl_trig
trigger voip_phost_crepl_trig
trigger voip_sub_crepl_trig
```

```
These three triggers will set possible default values (same condition as for enum_set_default) when creating new subscribers/domains/peers.
```

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